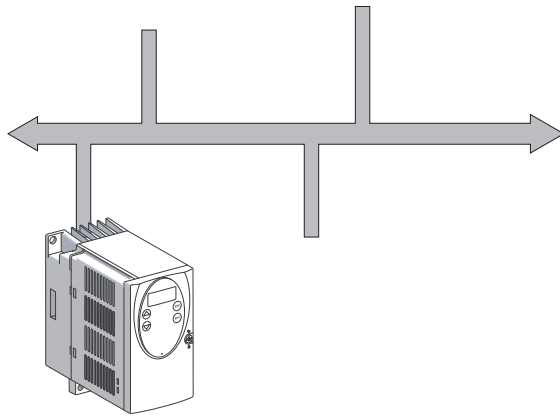


Technical Documentation



Field bus

CPD CANopen

Order no.: ACC1MDACA01EN

Edition: V1.01, 01.2005

Berger Lahr GmbH & Co. KG
Breslauer Str. 7
D-77933 Lahr

Important information

The drive systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or unbraked movements can never be totally excluded without additional safety equipment. For this reason personnel must never be in the danger zone of the drives unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on drives and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

For more information see the chapter on safety.

Not all product types are available in all countries. Please see the current catalogue for the availability of products.

We reserve the right to make technical changes.

All information refers to specifications and not to assured properties.

Most product designations are registered trademarks of their proprietors, even when not specifically noted.

Table of Contents

| | |
|---|-----------|
| Important information | -2 |
| Table of Contents | -3 |
| Writing conventions and symbols | -7 |
| 1 Introduction | |
| 1.1 CAN bus | 1-1 |
| 1.2 CANopen technology | 1-2 |
| 1.2.1 CANopen description language | 1-2 |
| 1.2.2 Communications layers | 1-2 |
| 1.2.3 Objects | 1-3 |
| 1.2.4 CANopen profiles | 1-4 |
| 1.3 Documentation and literature references | 1-5 |
| 2 Safety | |
| 2.1 Qualification of personnel | 2-1 |
| 2.2 Intended use | 2-1 |
| 2.3 Hazard categories | 2-2 |
| 2.4 General safety instructions | 2-2 |
| 3 Basics | |
| 3.1 Communications profile | 3-1 |
| 3.1.1 Object directory | 3-1 |
| 3.1.2 Communications objects | 3-2 |
| 3.1.3 Communications relationships | 3-5 |
| 3.2 Service data communication | 3-7 |
| 3.2.1 Overview | 3-7 |
| 3.2.2 SDO data exchange | 3-7 |
| 3.2.3 SDO message | 3-8 |
| 3.2.4 Read and write data | 3-9 |
| 3.3 Process data communication | 3-11 |
| 3.3.1 Overview | 3-11 |
| 3.3.2 PDO data exchange | 3-11 |
| 3.3.3 PDO message | 3-12 |
| 3.3.4 PDO mapping | 3-15 |
| 3.4 Synchronisation | 3-17 |
| 3.5 Emergency service | 3-19 |
| 3.5.1 Error evaluation and handling | 3-19 |
| 3.6 Network management services | 3-21 |
| 3.6.1 NMT services for device control | 3-21 |
| 3.6.2 services for connection monitoring | 3-23 |

4 Installation

5 Commissioning

| | | |
|-------|---|-----|
| 5.1 | Setting up the unit. | 5-1 |
| 5.2 | Setting the address and the baud rate | 5-1 |
| 5.3 | SYCON CANopen Configuration Software. | 5-1 |
| 5.3.1 | Creating a new network | 5-1 |
| 5.3.2 | Selecting the CANopen Master | 5-2 |
| 5.3.3 | Setting the bus parameters | 5-2 |
| 5.3.4 | Selecting and inserting the nodes | 5-3 |
| 5.3.5 | Configuring the network node | 5-4 |
| 5.3.6 | Setting the PDO4 mapping | 5-4 |

6 Operation

| | | |
|-------|--|------|
| 6.1 | Operating modes | 6-1 |
| 6.2 | Standardised operating modes. | 6-2 |
| 6.2.1 | Point-to-point operating mode | 6-2 |
| 6.2.2 | Speed profile operating mode | 6-4 |
| 6.2.3 | Referencing operating mode | 6-5 |
| 6.3 | Manufacturer-specific operating modes | 6-6 |
| 6.3.1 | Current regulation mode. | 6-6 |
| 6.3.2 | Speed regulation operating mode | 6-8 |
| 6.3.3 | Electronic gear mode. | 6-10 |
| 6.3.4 | Manual movement mode | 6-12 |
| 6.4 | Functions. | 6-13 |
| 6.4.1 | ramp function | 6-13 |
| 6.4.2 | Quick-Stop function | 6-13 |
| 6.4.3 | Motor stop | 6-13 |
| 6.4.4 | Standstill window | 6-13 |
| 6.4.5 | Reversal of direction of rotation | 6-14 |
| 6.4.6 | Monitoring functions. | 6-15 |
| 6.4.7 | Monitoring inputs and outputs of the unit. | 6-16 |
| 6.4.8 | Backing up and restoring object data. | 6-16 |

7 Diagnostics and troubleshooting

| | | |
|-------|---|-----|
| 7.1 | Field bus communication error diagnosis | 7-1 |
| 7.2 | Error diagnosis over field bus | 7-2 |
| 7.2.1 | Message objects | 7-2 |
| 7.2.2 | Messages: on the unit status | 7-2 |
| 7.3 | CANopen error messages | 7-3 |
| 7.3.1 | error register | 7-3 |
| 7.3.2 | Error code table | 7-3 |
| 7.3.3 | SDO error message ABORT | 7-3 |

8 Service, maintenance and disposal

| | | |
|-----|---------------------------|-----|
| 8.1 | Service address | 8-1 |
|-----|---------------------------|-----|

9 Object directory

| | | |
|--------|---|------|
| 9.1 | Specifications for the objects. | 9-1 |
| 9.2 | Overview object group 1000h | 9-2 |
| 9.3 | Assignment object group 6000h | 9-6 |
| 9.4 | Details object group 1000h | 9-7 |
| 9.4.1 | 1000h Device type | 9-7 |
| 9.4.2 | 1001h Error register | 9-7 |
| 9.4.3 | 1003h Predefined error field | 9-8 |
| 9.4.4 | 1005h COB-Id SYNC message | 9-9 |
| 9.4.5 | 1008h Manufacturer device name | 9-9 |
| 9.4.6 | 1009h Manufacturer hardware version | 9-10 |
| 9.4.7 | 100Ah Manufacturer software version | 9-10 |
| 9.4.8 | 100Ch Guard time | 9-10 |
| 9.4.9 | 100Dh Life time factor. | 9-11 |
| 9.4.10 | 1010h Save Parameters | 9-12 |
| 9.4.11 | 1011h Restore Default Parameters | 9-13 |
| 9.4.12 | 1014h COB-ID emergency message | 9-14 |
| 9.4.13 | 1015h Inhibit time emergency message | 9-14 |
| 9.4.14 | 1016h Consumer Heartbeat Time | 9-15 |
| 9.4.15 | 1017h Producer Heartbeat Time | 9-15 |
| 9.4.16 | 1018h Identity Object | 9-16 |
| 9.4.17 | 1020h data for configuration | 9-17 |
| 9.4.18 | 1029h Error Behaviour | 9-19 |
| 9.4.19 | 1200h 1st server SDO parameter. | 9-19 |
| 9.4.20 | 1201h 2nd server SDO parameter | 9-20 |
| 9.4.21 | 1400h 1st receive PDO parameter | 9-21 |
| 9.4.22 | 1401h 2nd receive PDO_parameter | 9-23 |
| 9.4.23 | 1402h 3rd receive PDO parameter | 9-24 |
| 9.4.24 | 1403h 4th receive PDO parameter | 9-25 |
| 9.4.25 | 1600h 1st receive PDO mapping | 9-26 |
| 9.4.26 | 1601h 2nd receive PDO mapping | 9-27 |
| 9.4.27 | 1602h 3rd receive PDO mapping | 9-28 |
| 9.4.28 | 1603h 4th receive PDO mapping | 9-30 |
| 9.4.29 | 1800h 1st transmit PDO parameter | 9-30 |
| 9.4.30 | 1801h 2nd transmit PDO parameter | 9-32 |
| 9.4.31 | 1802h 3rd transmit PDO parameter | 9-33 |
| 9.4.32 | 1803h 4th transmit PDO parameter | 9-35 |
| 9.4.33 | 1A00h 1st transmit PDO mapping | 9-36 |
| 9.4.34 | 1A01h 2nd transmit PDO mapping | 9-37 |
| 9.4.35 | 1A02h 3rd transmit PDO mapping | 9-38 |
| 9.4.36 | 1A03h 4th transmit PDO mapping | 9-40 |

10 Glossaries

| | | |
|------|-------------------------------|------|
| 10.1 | Terms and Abbreviations. | 10-1 |
|------|-------------------------------|------|

11 Index

Writing conventions and symbols

Work steps If work steps must be carried out in sequence, they are shown as follows:

- Special prerequisites for the following work steps
- Step 1
- ◁ Important response to this work step
- Step 2

If a response to a work step is specified, this will inform you that the step has been carried out correctly.

Unless otherwise stated, the individual instruction steps must be carried in the given sequence.

Lists Lists can be sorted alphanumerically or by priority. Lists are structured as follows:

- Point 1
- Point 2
 - Subpoint to 2
 - Subpoint to 2
- Point 3

Making work easier Information on making work easier can be found at this symbol:



*This offers supplementary information on making work easier.
See the chapter on safety for an explanation of the safety instructions.*

1 Introduction

1.1 CAN bus

The CAN bus (CAN:**Controller Area Network**) was originally developed for fast, economical data transmission in automotive engineering. In the meantime the CAN bus is also used in industrial automation technology and has been further developed for communication at field bus level.

Features of the CAN bus

The CAN bus is a standardised open bus, through which devices, sensors and actuators from different manufacturers communicate with each other. The features of the CAN bus are

- Multimaster capacity

Every device in the field bus can send and receive data independently without being assigned to an "ordering" master function.

- Message-oriented communication

Devices can be linked into an existing network without requiring reconfiguration of the entire system. The address of a new device does not need to be specified on the network.

- Prioritisation of messages

Messages with higher priority are sent first for time-critical applications.

- Residual error probability

Various backup processes in the network reduce the probability of an undetected, faulty data transfer to less than 10^{-11} . In practice, 100%-secure transmission can be assumed.

Transmission technology

In the CAN bus multiple devices are connected via a bus cable. Every network device can send and receive messages. Data between network devices are transmitted serially.

Network devices

Examples of CAN bus devices are

- automation devices, e.g. PLCs
- PCs
- input/output modules
- drive controllers
- analysis devices
- Sensors and actuators

1.2 CANopen technology

1.2.1 CANopen description language

CANopen is a device and manufacturer-independent description language for communication on the CAN bus. CANopen offers a unified base for exchanging commands and data between CAN bus devices.

1.2.2 Communications layers

CANopen uses the CAN bus technology for data communications.

CANopen is based on the ISO-OSI layer model on the data communications basic network service. 3 layers secure data communications in the CAN bus.

- CAN Physical Layer
- CAN Data Link Layer
- CANopen Application Layer

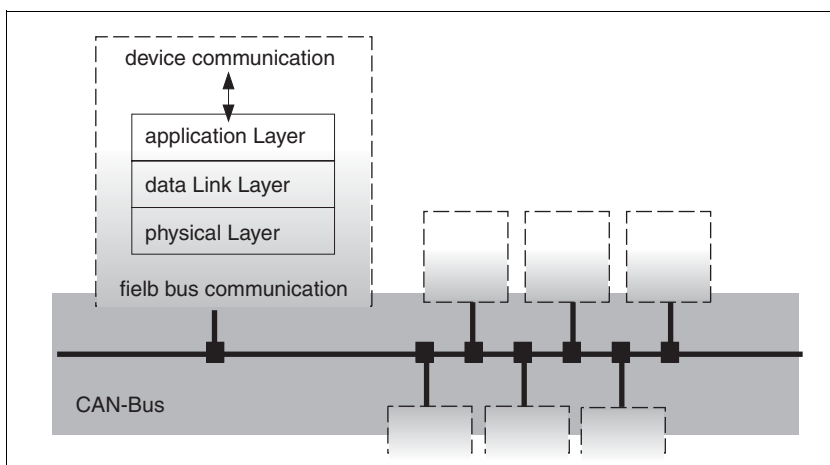


Figure 1.1 CANopen layer model

CAN Physical Layer The physical layer defines the electrical properties of the CAN bus such as plug connectors, cable length and cable properties such as bit-coding and bit-timing.

CAN Data Link Layer The data link layer connects the network devices. It sets the priorities of individual data packets and monitors and corrects errors.

CANopen Application Layer The application layer uses communications objects (COB) to exchange data between the various devices. Communication objects are elementary components for creating a CANopen application.

1.2.3 Objects

All processes under CANopen are executed via objects. Objects carry out different tasks; they act as communications objects for data transport to the field bus, control the process of establishing a connection or monitor the network devices. If objects are directly connected to the device (device-specific objects), the device functions can be used and changed with device-specific objects.



The product includes parameters for the CANopen object groups 3000_h and 6000_h. The names of the parameters and the data type of the parameters for the object group 6000_h may be different from the DSP 402 definition. For a detailed description of all parameters see the operator's manual for the product in the chapter on parameters.

Object directory

The central controller connection for all objects is the object directory of every network device. Other devices find all objects here with which they can establish a connection with the device.

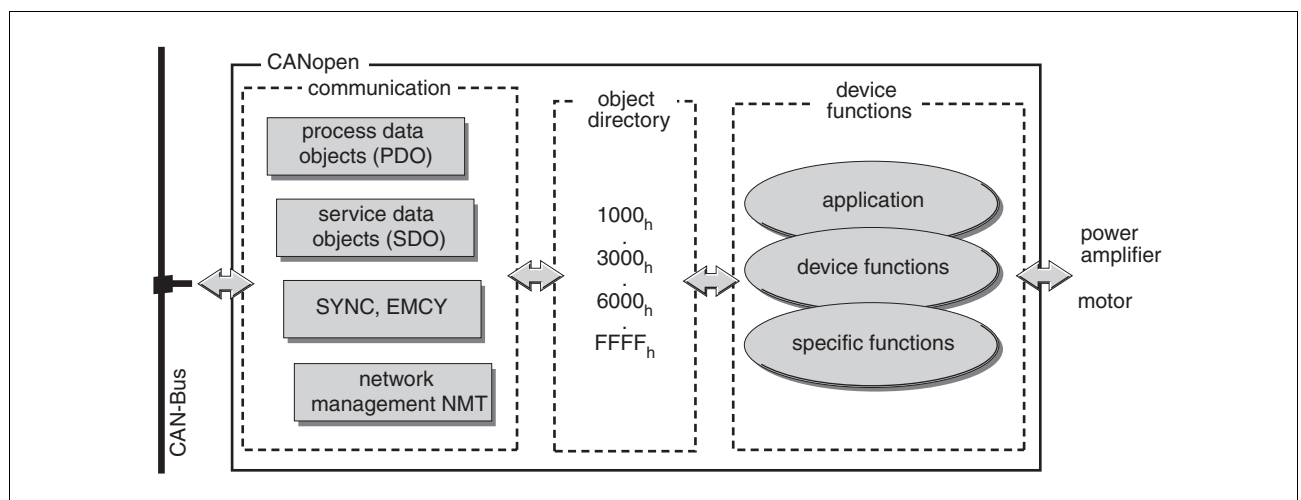


Figure 1.2 Device model with object directory

Objects for describing the data types and executing the communications tasks and device functions under CANopen are registered.

Object index

Every object is addressed over a 16-bit index, which is displayed as a four-character hexadecimal number. The objects are arranged in groups in the object directory. The following table shows an overview of the object directory as per the CANopen agreement.

| Index range (hex) | Object groups |
|--------------------------------------|-------------------------------|
| 1000 _h -2FFF _h | Communications profile |
| 3000 _h -5FFF _h | Manufacturer-specific objects |
| 6000 _h -9FFF _h | Standardised device profiles |
| A000 _h -FFFF _h | reserved |

A list of the CANopen objects can be found in chapter 9 “Object directory”.

1.2.4 CANopen profiles

Standardised profiles Standardised profiles describe objects that can be applied to various devices without additional configuration. The association for CAN in Automation e. V. (CiA) has standardised different profiles. They include:

- the communications profile DS 301
- the device profile DSP 402

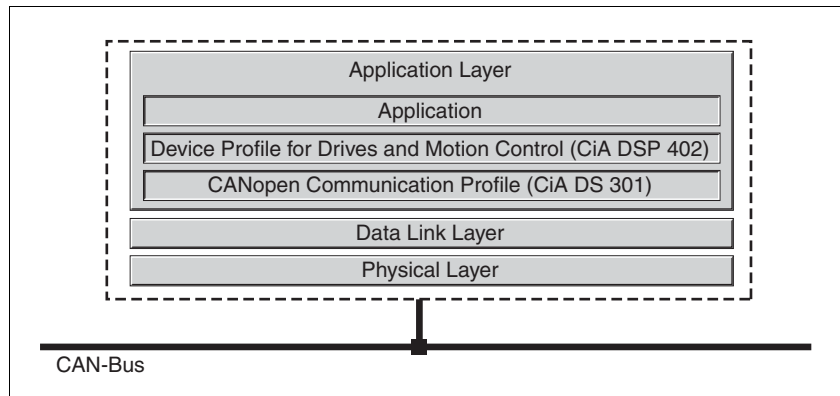


Figure 1.3 CANopen reference model

DS301 communications profile The DS 301 communications profile forms the interface between device profiles and CAN bus. It was specified in 1995 under the name DS 301 and defines unified standards for common data exchange between different device types under CANopen.

The communications profile objects in the device carry out the tasks of data and parameter exchange with other network devices and initialise, control and monitor the device in the network.

DSP 402 device profile The DSP 402 device profile describes standardised objects for positioning, monitoring and settings of drives. The tasks of the objects are:

- Device control and status monitoring (Device Control)
- Standardised parameter setting
- Switching, verification and execution of operating modes

Manufacturer-specific profiles The basic functions of a device can be used with device profiles standardised with objects. Only manufacturer-specific device profiles offer the complete range of functions. The objects with which the special functions of a device can be used under CANopen are defined in them.

1.3 Documentation and literature references

CAN interest group CiA - CAN in Automation
Am Weichselgarten 26
D-91058 Erlangen
<http://www.can-cia.org/>

- CANopen standards*
- CiA Draft Standard 301 (DS 301)
CANopen application layer and communication profile
V4.02, February 2002
 - CiA Draft Standard Proposal 402 (DSP 402)
Device profile for drives and motion control
V2.0, July 2002
 - ISO/DIS 11898: Controller Area Network (CAN) for high speed
communication; 1993
 - EN 50325-4: Industrial communications subsystem based on
ISO 11898 for controller device interfaces (CANopen); 2002

Literature Controller Area Network,
Konrad Etschberger, Carl Hanser Verlag
ISBN 3-446-19431-2

- Documentation*
- Device manual for the AC servo drive system CPD
 - CANopen field bus manual

2 Safety

2.1 Qualification of personnel

Only technicians who are familiar with and understand the contents of this manual and the other relevant manuals are authorised to work on and with this drive system. The technicians must be able to detect potential dangers that may be caused by setting parameters, changing parameter values and generally by the mechanical, electrical and electronic equipment.

The technicians must have sufficient technical training, knowledge and experience to recognise and avoid dangers.

The technicians must be familiar with the relevant standards, regulations and safety regulations that must be observed when working on the drive system.

2.2 Intended use

The drive systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or unbraked movements can never be totally excluded without additional safety equipment. For this reason personnel must never be in the danger zone of the drives unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on drives and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

In the system configuration described the drive systems must be used in industrial applications only and must have a fixed connection only.

In all cases the applicable safety regulations and the specified operating conditions, such as environmental conditions and specified technical data, must be observed.

The drive systems may be commissioned and operated only after installation in accordance with EMC requirements and the product-specific specifications.

To prevent personal injury and damage to property damaged drive systems must not be installed or operated.

Changes and modifications of the drive systems are not permitted and if made all no warranty and liability will be accepted.

The drive system must be operated only with the specified wiring and approved accessories. In general, use only original accessories and spare parts.

The drive systems must not be operated in an environment subject to explosion hazard (ex area).

2.3 Hazard categories

Safety notes and general information are indicated by hazard messages in the manual. In addition there are symbols and instructions affixed to the product that warn of possible hazards and help to operate the product safely.

Depending on the seriousness of the hazard, the messages are divided into three hazard categories.



DANGER!

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death, serious injury, or equipment damage.



WARNING!

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.



CAUTION!

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

2.4 General safety instructions



DANGER!

Danger of injury by complex system!

When starting field bus operation the attached controllers are generally out of view of the operator and cannot be directly monitored.

- Start the system only if there are no persons within the actuation zone of the moving system components and the system can be operated safely.

**WARNING!**

Danger of injury and damage to system components by loss of control!

- Observe the accident prevention regulations.
- The system manufacturer must consider the possible errors that could occur with the signals and in particular the critical functions to ensure a safe status during and after errors. Examples for these are: emergency stop, final position limit, power failure and restart.
- Consideration of possible errors must also include unexpected delay and failure of signals or functions.
- Separate redundant controller paths must be provided for dangerous functions.
- Verify the effectiveness of the measures.

3 Basics

3.1 Communications profile

CANopen manages communications between the network devices with object directories and objects. A network device can use process data objects (PDO) and service data objects (SDO) to request the object data from the object directory of another device and, if permissible, write back modified values.

The following can be done with the access to the objects of the network devices

- exchange parameter values
- start movement functions of individual CAN bus devices
- query status information

3.1.1 Object directory

Every CANopen device administers an object directory, in which all objects for communications are listed.

Index, subindex

The objects are addressed in the object directory with a 16-bit long index. One or more 8-bit-long subindex entries to every object point to individual data fields in the object. Index and subindex are shown in hexadecimal characters, recognisable by the attached "h".

Example

The following table shows index and subindex entries with the example of the object `software position limit (607Dh)` for identifying the position of the software limit switch.

| Index | Subindex | Name | Meaning |
|-------------------|-----------------|---------------------|---------------------------|
| 607D _h | 00 _h | - | Number of data fields |
| 607D _h | 01 _h | min. position limit | Bottom limit value switch |
| 607D _h | 02 _h | max. position limit | Top limit value switch |

Table 3.1 Example for index and subindex entries

Object descriptions in the manual

The objects of the following object groups are described to distinguish them for the CANopen programming of a device:

- 1xxx_h objects: Communications objects in this chapter
- 3xxx_h objects: Manufacturer-specific objects required for the control of the device, in chapter 6 "Operation".
- 6xxx_h objects: Standardised objects of the device profile in chapter 6 "Operation"

Standardised objects

Standardised objects form the basis of applying the same applications for the various network devices of a device type. This requires the devices to list the objects in their directory. Standardised objects are defined in the DS 301 communications profile and the DSP 402 device profile.

3.1.2 Communications objects

Overview The communications objects are standardised with the DS301 CANopen communications profile. The objects can be classified into four groups according to their tasks.

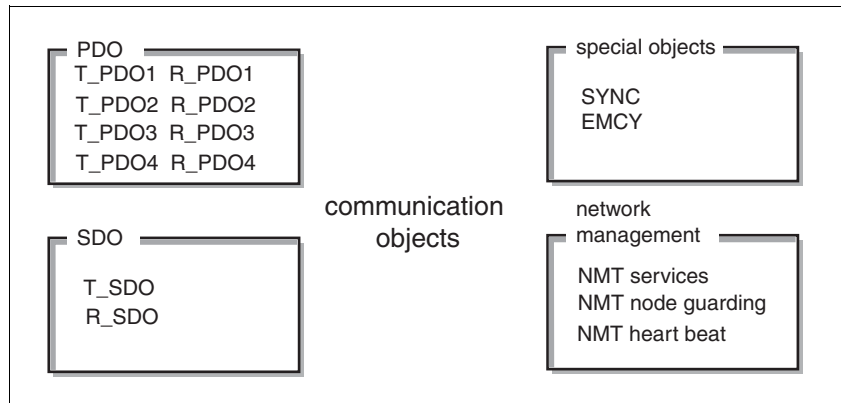


Figure 3.1 The following are considered communications objects from the point of view of the device:
T_...: "Transmit", R_...: "Receive"

- PDO (process data object) for real-time transmission of process data
- SDO (service data object) for read and write access to the object directory
- Objects for controlling CAN messages:
 - SYNC object (synchronisation object) for synchronisation of network devices
 - EMCY object (emergency object) for the error display of a device or its peripheral.
- Network management services:
 - NMT services for initialisation and network control (NMT: network management)
 - NMT Node Guarding for monitoring the network devices
 - NMT heartbeat for monitoring the network devices

CAN message Data are exchanged on the CAN bus as CAN messages. A CAN message sends the communications object and a variety of administration and control information to ensure data transmission without loss and errors.

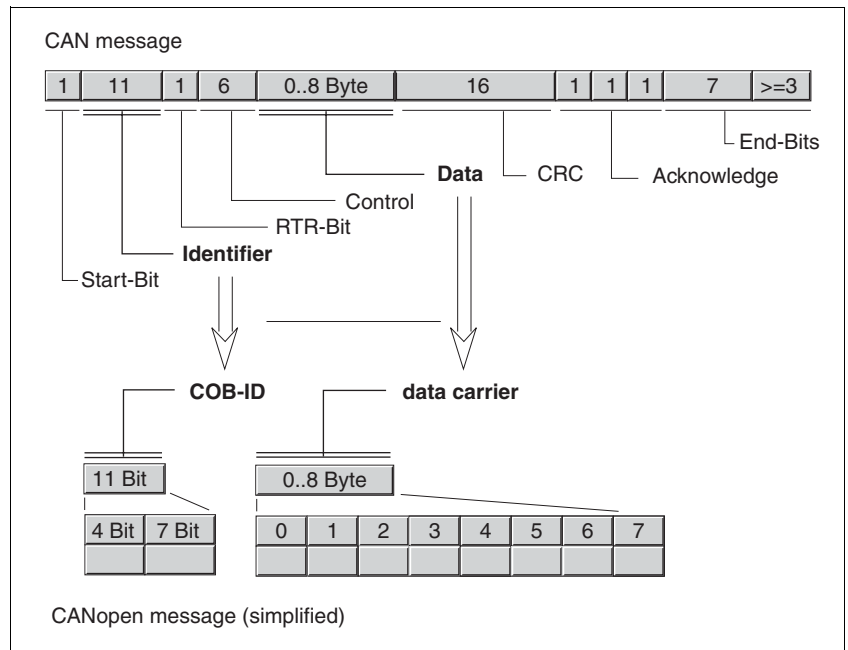


Figure 3.2 CAN message and simplified display of CANopen message

CANopen message

The CAN message can be displayed in simplified form for work with CANopen objects and for data exchange, because most of the bits are used to ensure error-free data transmission. These bits are automatically removed from the received message by the data security layer, the data link layer of the OSI layer model, and added to a message before transmission.

The two bit fields "identifier" and "data" form the simplified CANopen message. The "identifier" corresponds to the "COB Id" and the "data" field to the maximum 8-byte data frame of a CANopen message.

COB-ID

The COB Id (**C**ommunication **O**bject **I**dentifier) has two tasks in the control of communications objects:

- Bus arbitration: specification of transmission priorities
- identification of communications objects

An 11-bit COB identifier as per the CAN 3.0A specification is defined for CAN communications. It comprises two parts:

- Function code, 4 bit size
- Node-ID, 7 bit size.

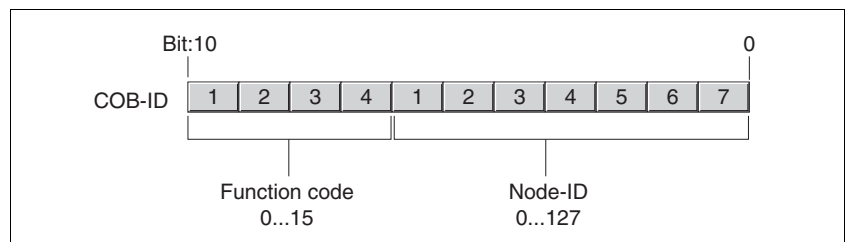


Figure 3.3 COB Id with function code and node address

COB-IDs of the communications objects

The following table shows the COB-IDs of all communications objects in the factory setting. The column "Index of object parameters" shows the index of special objects with which the settings of the communications objects can be read or modified by SDO.

| Communications object | Function code | Node address node-Id [1...127] | COB-ID decimal (hexadecimal) | Index of object parameters |
|------------------------------------|---------------|--------------------------------|--|--|
| NMT Start/Stop Service | 0 0 0 0 | 0 0 0 0 0 0 0 | 0 (0 _h) | - |
| SYNC object | 0 0 0 1 | 0 0 0 0 0 0 0 | 128 (80 _h) | 1005 _h ...1007 _h |
| EMCY object | 0 0 0 1 | x x x x x x x | 128 (80 _h) + node-Id | 1014 _h , 1015 _h |
| T_PDO1 | 0 0 1 1 | x x x x x x x | 384 (180 _h) + node-Id | 1800 _h |
| R_PDO1 | 0 1 0 0 | x x x x x x x | 512 (200 _h) + node-Id | 1400 _h |
| T_PDO2 | 0 1 0 1 | x x x x x x x | 640 (280 _h) + node-Id | 1801 _h |
| R_PDO2 | 0 1 1 0 | x x x x x x x | 768 (300 _h) + node-Id | 1401 _h |
| T_PDO3 | 0 1 1 1 | x x x x x x x | 896 (380 _h) + node-Id | 1802 _h |
| R_PDO3 | 1 0 0 0 | x x x x x x x | 1024 (400 _h) + node-Id | 1402 _h |
| T_PDO4 | 1 0 0 1 | x x x x x x x | 1152 (480 _h) + node-Id | 1803 _h |
| R_PDO4 | 1 0 1 0 | x x x x x x x | 1280 (500 _h) + node-Id | 1403 _h |
| T_SDO | 1 0 1 1 | x x x x x x x | 1408 (580 _h) + node-Id | - |
| NMT error control | 1 1 1 0 | x x x x x x x | 1792 (700 _h) + node-Id | 100E _h |
| R_SDO | 1 1 0 0 | x x x x x x x | 1536 (600 _h) + node-Id | - |
| LMT Services ¹⁾ | 1 1 1 1 | 1 1 0 0 1 0 x | 2020 (7E4 _h), 2021 (7E5 _h) | |
| NMT Identify Service ¹⁾ | 1 1 1 1 | 1 1 0 0 1 1 0 | 2022 (7E6 _h) | |
| DBT Services ¹⁾ | 1 1 1 1 | 1 1 0 0 x x x | 2023 (7E7 _h), 2024 (7F8 _h) | |
| NMT Services ¹⁾ | 1 1 1 1 | 1 1 0 1 0 0 x | 2025 (7E9 _h), 2026 (7EA _h) | |

1) not supported by the device

Table 3.2 COB Ids of all communications objects



COB Ids of PDOs can be changed as required. The assignment scheme for COB Ids specifies only one basic setting.

Function code

The function code classifies the communications objects. Because the bits of the function code in the COB Id are significantly higher, the function code simultaneously controls the transmission priorities: Objects with a small function code are sent at high priority. For example, with simultaneous bus access an object with the function code "1" is sent before an object with the function code "3".

Node address

Every network device is configured before network operation. It is given a unique, 7-bit-long node address (node-Id) between 1 (01_h) and 127 (7F_h). The device address "0" is reserved for "broadcast" transmissions, which are used to send the messages to all devices simultaneously.

Example Selection of a COB-Id

For a device with the node address 5 is the COB-Id of the communications object T_PDO1:

$$384 + \text{node-Id} = 384 (180_{\text{h}}) + 5 = 389 (185_{\text{h}}).$$

Data frame The data frame of the CANopen message can hold up to 8 bytes of data. In addition to the data frame for SDOs and PDOs special frame types are specified in the CANopen profile:

- Error data frame
- Remote data frame for requesting a message

The data frames are described with the relevant communications objects.

3.1.3 Communications relationships

CANopen uses three relationships for communications between network devices:

- Master-slave relationship
- Client-server relationship
- Producer-consumer relationship

Master-slave relationship

A "master" in the network controls the message traffic. A "slave" only responds when addressed by the master.

The master-slave relationship is used with network management objects to guarantee a controlled network start and to monitor the connection of devices.

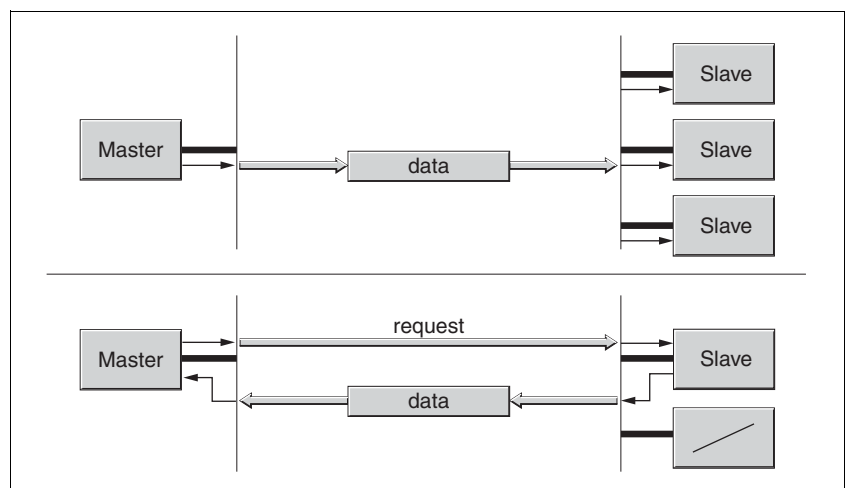


Figure 3.4 Master-slave relationships

The exchange of messages can be executed unconfirmed and confirmed. If the master sends an unconfirmed CAN message, it can be received by multiple or no slaves.

To confirm the message, the master requests a message from a specific slave, which then responds with the desired data.

Client-server relationship

A client-server relationship is always established between two devices. The server is the device whose object list is used during the data exchange. The client addresses and starts the exchange of messages and waits for a response from the server.

A client-server relationship is implemented with SDOs to send configuration data and long messages.

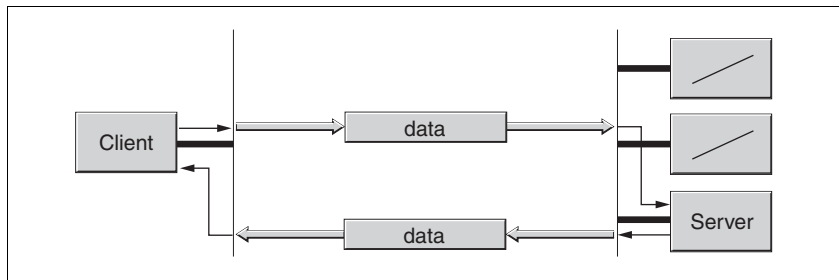


Figure 3.5 Client-server relationship

The client addresses and sends a CAN message to a server. The server evaluates the message and sends the answer data as response.

Producer-consumer relationship

The producer-consumer relationship is used for exchanging messages with process data, because this relationship enables fast data exchange without administration data.

A producer sends data, a consumer receives data.

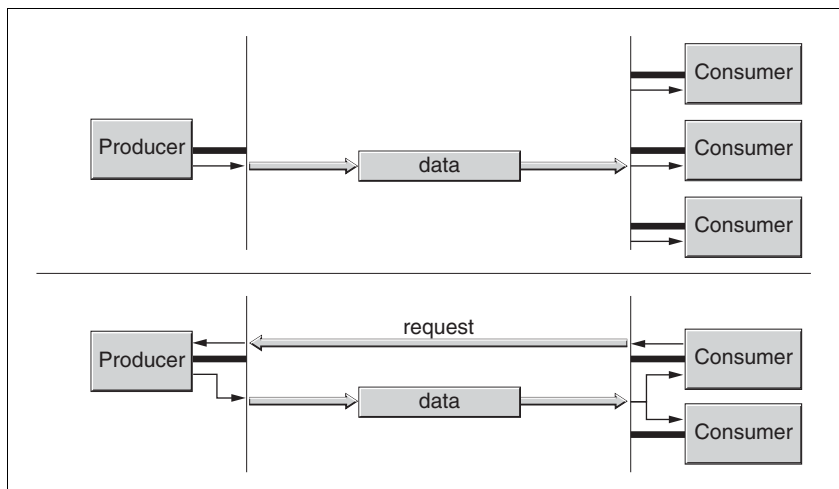


Figure 3.6 Producer-consumer relationships

The producer sends a message that can be received by one or more network devices. The producer does not receive a receipt response. The message transmission can be triggered

- by an internal event, e.g. "target position reached"
- by the synchronisation object SYNC
- by request of a consumer

For details on the function of the producer-consumer relationship and the request of messages see chapter 3.3 "Process data communication".

3.2 Service data communication

3.2.1 Overview

Service data objects (SDO: **S**ervice **D**ata **O**bject) can be used to access the entries of an object directory via index and subindex. The values of the objects can be read and - if permissible - also be changed.

Every network device has at least one server SDO to be able to respond to read and write requests from a different device. A client SDO is only required to request SDO messages from the object directory of a different device or to change them there.

The T_SDO of a SDO client can be used to send the request for data exchange and to receive with the R_SDO. The data frame of a SDO is always 8 bytes.

SDOs have a higher COB-Id than PDOs and therefore are send over the CAN bus at a lower priority.

3.2.2 SDO data exchange

A service data object (SDO) sends parameter data between two devices. The data exchange conforms to the client-server relationship. The server is the device to whose object directory a SDO message refers.

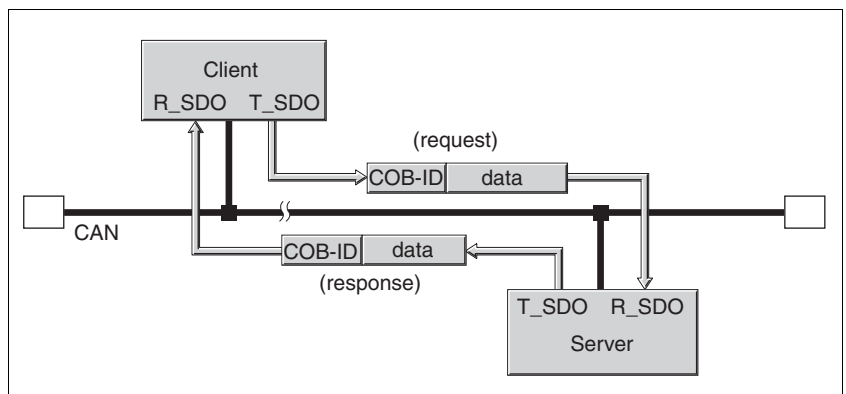


Figure 3.7 SDO message exchange with request and response

Message types

Client-server communications are triggered by the client to send parameter values to the server or to obtain them from the server. In both cases the client starts the communication with a request and receives a response from the server.

3.2.3 SDO message

A SDO message in simplified form consists of the COB-Id and the SDO data frame, in which up to four bytes of data can be sent. Longer data strings are distributed over multiple SDO messages with a special protocol.

The device sends SDOs of up to 4 bytes data length (data). Larger quantities of data such as 8.byte values of the "Visible String 8" data type can be distributed over multiple SDOs and are sent successively in 7-byte blocks.

Example The following diagram shows an example of a SDO message.

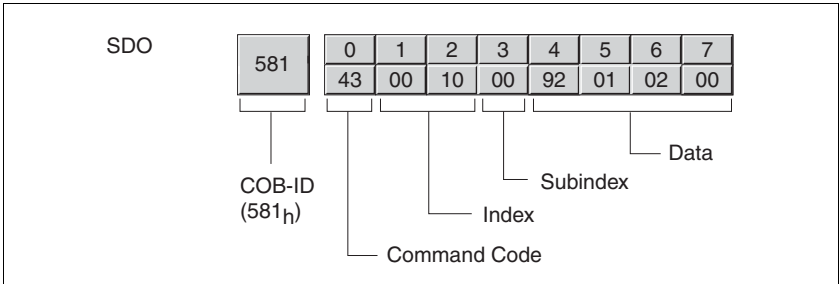


Figure 3.8 SDO message, example

COB-ID and data frame R_SDO and T_SDO have different COB-Ids.

The data frame of a SDO messages includes the following:

- Command code in which the SDO message type and the data length of the transmitted value are encrypted
- Index and subindex, which point to the object whose data are transported with the SDO message
- Data that comprise up to 4 bytes

Evaluation of numeric values

Index and data are transmitted left-aligned in Intel format. If the SDO contains numerical values over one byte in length, the data must be converted bit-by-bit before and after a transmission.

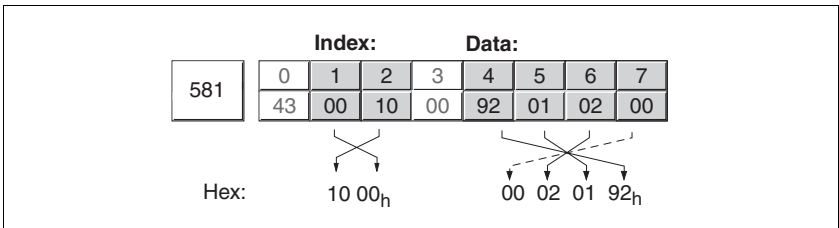


Figure 3.9 Repositioning numeric values greater than 1 byte

3.2.4 Read and write data

Write data The client starts a write request by sending index, subindex, data length and value.

The server sends a response indicating whether the data were correctly processed. The response contains the same index and subindex, but no data.

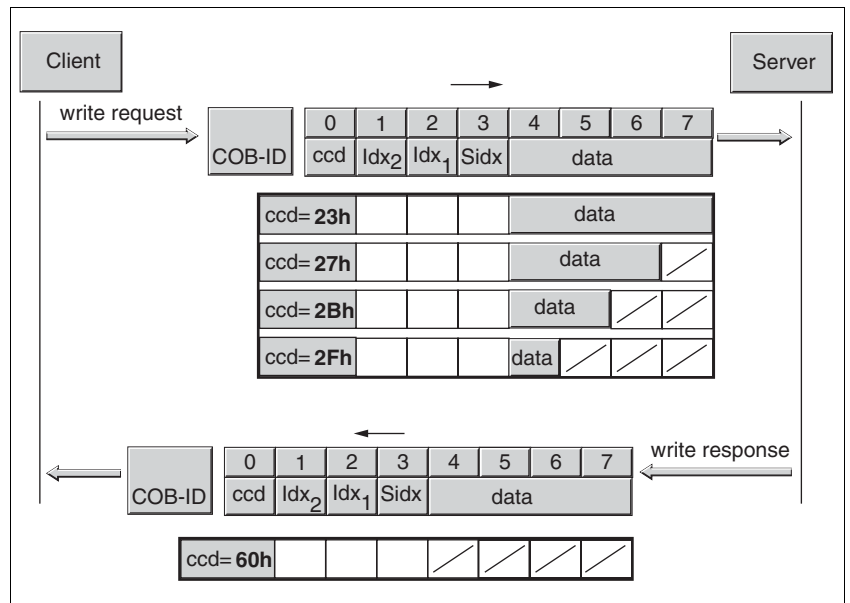


Figure 3.10 Writing parameter values

Unused bytes in the data field are shown with a slash in the graphic. The content is not defined.

ccd-coding The table below shows the command code for writing parameter values. It depends on the message type and the transmitted data length.

| Message type | Data length used | | | | |
|----------------|------------------|-----------------|-----------------|-----------------|-----------------|
| | 4 byte | 3 byte | 2 byte | 1 byte | |
| write request | 23 _h | 27 _h | 2B _h | 2F _h | Send parameters |
| write response | 60 _h | 60 _h | 60 _h | 60 _h | response |
| error response | 80 _h | 80 _h | 80 _h | 80 _h | Error |

Table 3.3 Command codes for writing parameter values

Read data The client starts a read request by sending index and subindex that point to the object or the object value whose value it wants to read out.

The server responds to the query with the desired data. The SDO response contains the same index and subindex. The length of the answer data is specified in the command code "ccd".

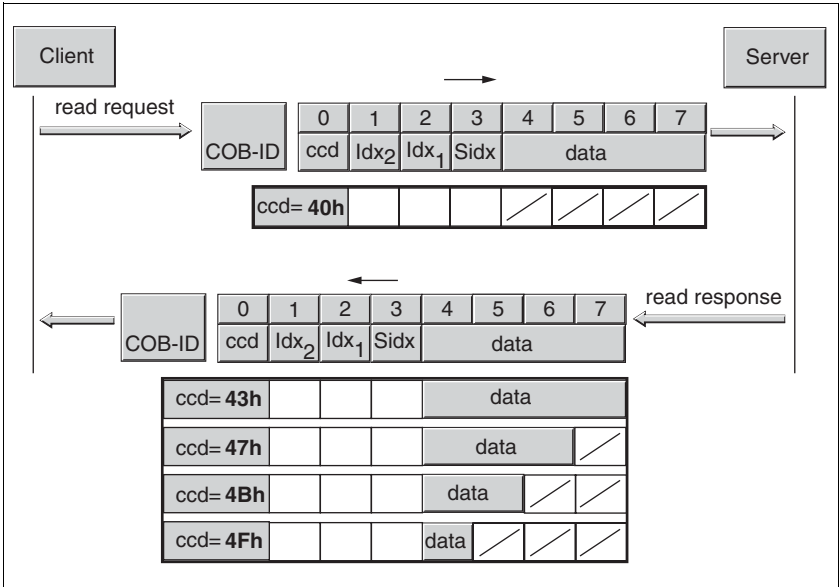


Figure 3.11 Reading parameter value

Unused bytes in the data field are shown with a slash in the graphic. The content is not defined.

ccd-coding The table below shows the command code for sending a read value. It depends on the message type and the transmitted data length.

| Message type | Data length used | | | | |
|----------------|------------------|-----------------|-----------------|-----------------|--------------------|
| | 4 byte | 3 byte | 2 byte | 1 byte | |
| read request | 40 _h | 40 _h | 40 _h | 40 _h | Request read value |
| read response | 43 _h | 47 _h | 4B _h | 4F _h | Return read value |
| error response | 80 _h | 80 _h | 80 _h | 80 _h | Error |

Table 3.4 Command code for sending a read value

Error response If a message could not be evaluated without errors, the server sends an error message. For details on the evaluation of the error message see chapter 7.3.3 "SDO error message ABORT".

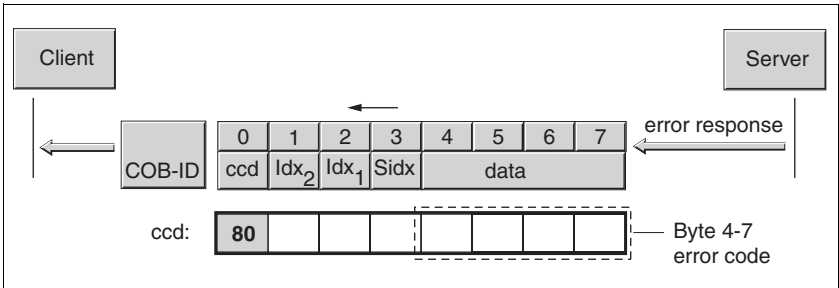


Figure 3.12 Response with error message (error response)

3.3 Process data communication

3.3.1 Overview

Process Data Objects (PDO) (Process Data Object) are used for real-time data exchange of process data such as actual and setpoint position or operating status of the device. The transmission can be executed very fast, because it is sent without additional administration data and does not require a response from the recipient.

The flexible data length of a PDO message also increases the data throughput. A PDO message can send up to 8 bytes of data. If only 2 bytes are occupied, only 2 data bytes are sent.

The length of a PDO message and the allocation of the data fields is specified by PDO mapping. For more information see chapter 3.3.4 “PDO mapping”.

PDO messages can be exchanged between devices that generate or process process data.

3.3.2 PDO data exchange

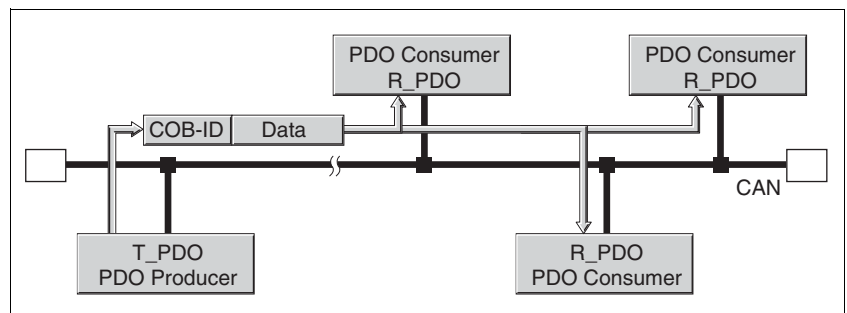


Figure 3.13 PDO data exchange

Data exchange with PDOs conforms to the producer-consumer relationship and can be triggered by three methods

- synchronised
- event-driven, asynchronous
- by request of a consumer, asynchronous

The synchronised data processing is controlled by the SYNC object. Synchronous PDO messages are sent immediately like the standard PDO messages, but are only evaluated on the next SYNC. For example, multiple drives can be started simultaneously by synchronised data exchange.

The device evaluates PDO messages that are called on request or are event-controlled immediately.

The transmission type can be specified separately for every PDO with subindex 02_h (transmission type) of the PDO communications parameter. The objects are shown in Table 3.5.

3.3.3 PDO message

T_PDO, R_PDO A PDO always is available for sending and receiving a PDO message:

- The T_PDO for sending PDO messages (T: Transmit),
- The R_PDO for receiving PDO messages (R: Receive).



The following settings for PDOs correspond to the standard defaults for the device, unless otherwise specified. They can be read and set via objects of the communications profile.

The device uses 8 PDOs, 4 receive PDOs and 4 send PDOs. All PDOs are evaluated or transmitted event-controlled in the default setting.

PDO settings The settings for PDOs can be read and changed with 8 communications objects:

| Object | Meaning |
|---|---------------------|
| 1st receive PDO parameter (1400 _h) | Settings for R_PDO1 |
| 2nd receive PDO parameter (1401 _h) | Settings for R_PDO2 |
| 3rd receive PDO parameter (1402 _h) | Settings for R_PDO3 |
| 4th receive PDO parameter (1403 _h) | Settings for R_PDO4 |
| 1st transmit PDO parameter (1800 _h) | Settings for T_PDO1 |
| 2nd transmit PDO parameter (1801 _h) | Settings for T_PDO2 |
| 3rd transmit PDO parameter (1802 _h) | Settings for T_PDO3 |
| 4th transmit PDO parameter (1803 _h) | Settings for T_PDO4 |

Table 3.5 Communications objects for PDO

Enable PDO In the default setting of the PDOs R_PDO1 and T_PDO1 are enabled. The other PDOs must be enabled first.

A PDO is enabled with bit 31 (valid bit) in subindex 01_h of that communications object:

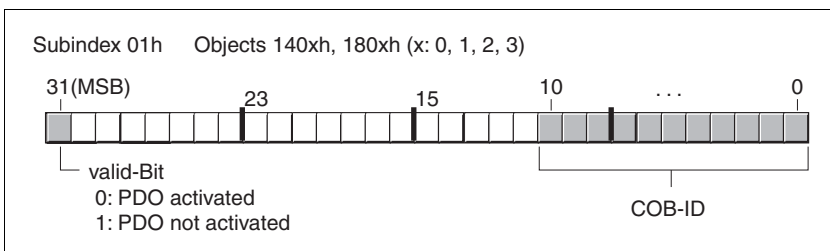


Figure 3.14 Enable PDOs with subindex 01_h, enable bit 31

Example Setting for R_PDO3 in object 1402_h

- Subindex 01_h = 8000 04xx_h: R_PDO3 not enabled
- Subindex 01_h = 0000 04xx_h: R_PDO3 enabled.

Values for "x" in the example depend on the setting of the COB-Id.

PDO time intervals The time intervals "inhibit time" and "event timer" can be set for every send PDO.

- The time interval "inhibit time" can be used to reduce the load on the CAN bus, which can be the result of continuous transmission of

T_PDOs. If an interval time that is not equal to zero is entered, a sent PDO will only be sent again when the interval time expires. The time is set with subindex 03_h.

- The time interval "event timer" triggers an event message periodically. After the interval time has expired the device transmits the event-controlled T_PDO. The time is set with subindex 05_h.

Receive PDOs

The objects for R_PDO1, R_PDO2 and R_PDO3 are permanently specified. The object that is represented in the PDO R_PDO4 can be modified by PDO mapping.

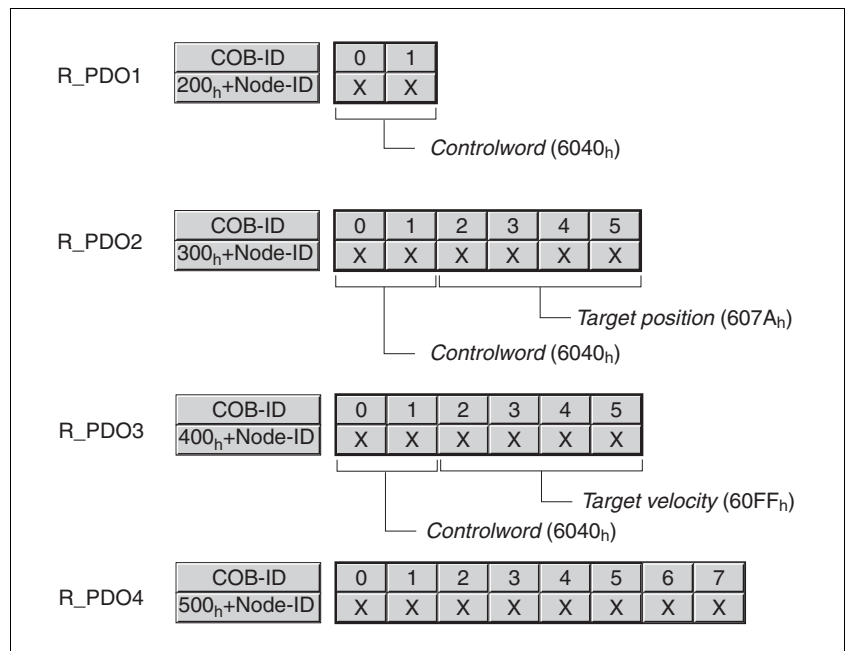


Figure 3.15 Receive PDOs

- R_PDO1** In the first receive PDO the control word, object `controlword` (6040_h), of the status machine is represented, which can be used to set the operating status of the device.
- R_PDO1 is evaluated asynchronously, i.e. is event-controlled. R_PDO1 is permanently set.
- R_PDO2** With the second PDO the control word and the target position of a movement command, object `target position` (607A_h), is received for a point-to-point positioning in the "profile position mode".
- R_PDO2 is evaluated asynchronously, i.e. is event-controlled. R_PDO2 is permanently set.
- For details on the SYNC object see chapter 3.4 "Synchronisation".
- R_PDO3** In the third receive PDO the control word and the setpoint speed, object `Target velocity` (60FF_h), is mapped for the speed mode in the "profile velocity mode".
- R_PDO3 is evaluated asynchronously, i.e. is event-controlled. R_PDO3 is permanently set.
- R_PDO4** Manufacturer-specific object values are transmitted with the fourth receive PDO. R_PDO4 is empty by default.

R_PDO4 is evaluated asynchronously, i.e. is event-controlled. R_PDO4 can be used to map various manufacturer-specific objects with PDO mapping.

Send PDOs The objects for T_PDO1, T_PDO2 and T_PDO3 are permanently specified. The object that is represented in the PDO T_PDO4 can be modified by PDO mapping.

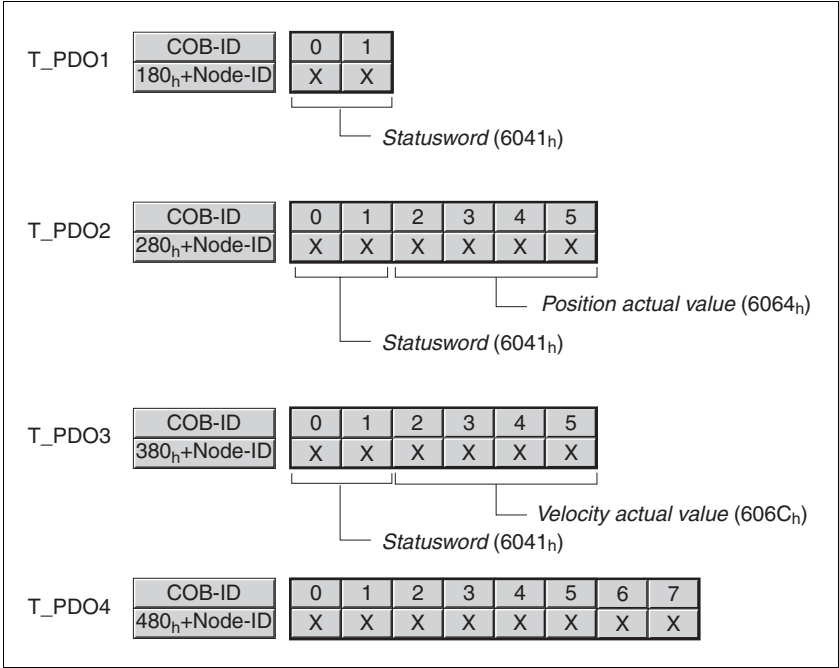


Figure 3.16 Send PDOs

- T_PDO1* In the first send PDO the status word, object `statusword (6041h)`, of the status machine is mapped.
- T_PDO1 is sent asynchronously and event-controlled at every change of the status information. No other objects can be mapped with T_PDO1.
- T_PDO2* In the second send PDO the status word and the current position of the motor, object `Position actual value (6064h)`, is mapped to monitor a point-to-point positioning in the "profile position mode".
- T_PDO2 is sent after receipt of a SYNC object and event-controlled. T_PDO4 can also be used to map other objects with PDO mapping.
- T_PDO3* In the third send PDO the status word and the current speed, object `Velocity actual value (606Ch)`, is mapped for monitoring the speed mode in "profile velocity mode".
- T_PDO3 is sent asynchronously and event-controlled at every change of the status information. No other objects can be mapped with T_PDO3.
- T_PDO4* Manufacturer-specific object values (for monitoring) are sent with the fourth send PDO. T_PDO4 is empty by default.
- T_PDO4 is sent asynchronously and event-controlled at every change of the status information. T_PDO4 can be used to map various manufacturer-specific objects with PDO mapping.

3.3.4 PDO mapping

Up to 8 bytes of data from different areas of the object directory can be sent with a PDO message. The mapping of data in a PDO message is referred to as PDO mapping.

Figure 3.17 shows data exchange between PDOs and object directory with two examples of objects in T_PDO4 and R_PDO4 of the PDOs.

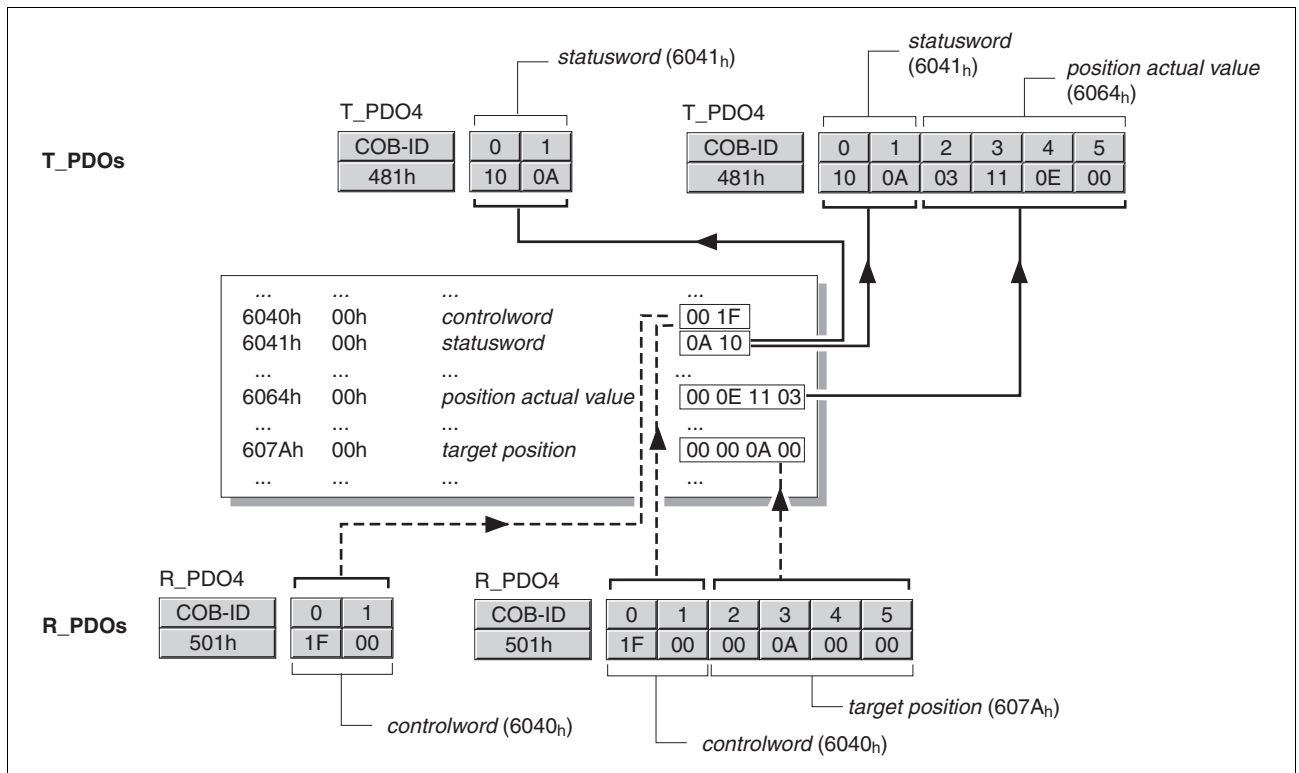


Figure 3.17 PDO mapping, in this case for a device with node address 1

Static PDO mapping

The device uses static and dynamic PDO mapping. In static PDO mapping all objects are mapped in accordance with a fixed, non-modifiable setting in the relevant PDO.

The settings for PDO mapping are defined in an assigned communications object for every PDO.

| Object | PDO mapping for | type |
|---|-----------------|---------|
| 1st receive PDO mapping (1600 _h) | R_PDO1 | static |
| 2nd receive PDO mapping (1601 _h) | R_PDO2 | static |
| 3rd receive PDO mapping (1602 _h) | R_PDO3 | static |
| 4th receive PDO mapping (1603 _h) | R_PDO4 | dynamic |
| 1st transmit PDO mapping (1A00 _h) | T_PDO1 | static |
| 2nd transmit PDO mapping (1A01 _h) | T_PDO2 | static |
| 3rd transmit PDO mapping (1A02 _h) | T_PDO3 | static |
| 4th transmit PDO mapping (1A03 _h) | T_PDO4 | dynamic |

Structure of entries Up to 8 bytes of 8 different objects can be mapped in a PDO. Every communications object for setting the PDO mapping provides 4 subindex entries. A subindex entry contains 3 pieces of information on the object: the index, the subindex and the number of bits that the object occupies in the PDO.

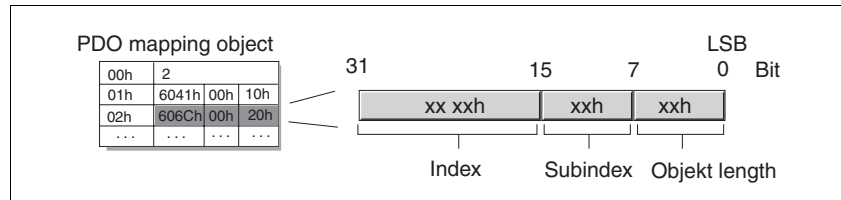


Figure 3.18 Structure of entries for the PDO mapping

The number of valid subindex entries is contained in subindex 00_h of the communications object.

PDO-Mapping objects

| Object (Index:Subindex) | PDO | Data type |
|--|-------|-----------|
| _IO_act (3008:1 _h) | T_PDO | UINT16 |
| ANA1_act (3009:1 _h) | T_PDO | INT16 |
| ANA2_act (3009:5 _h) | T_PDO | INT16 |
| JOGactivate (301B:9 _h) | R_PDO | UINT16 |
| _actionStatus (301C:4 _h) | T_PDO | UINT16 |
| _p_actRAMPusr (301F:2 _h) | T_PDO | INT32 |
| CUR_I_target (3020:4 _h) | R_PDO | INT16 |
| SPEEDn_target (3021:4 _h) | R_PDO | INT16 |
| GEARdenom (3026:3 _h) | R_PDO | INT32 |
| GEARnum (3026:4 _h) | R_PDO | INT32 |
| controlword (6040 _h) | R_PDO | UINT16 |
| statusword (6041 _h) | T_PDO | UINT16 |
| modes of operation (6061 _h) | T_PDO | INT8 |
| position actual value (6064 _h) | T_PDO | INT32 |
| velocity actual value (606C _h) | T_PDO | INT32 |
| target position (607A _h) | R_PDO | INT32 |
| profile velocity (6081 _h) | R_PDO | UINT32 |
| target velocity (60FF _h) | R_PDO | INT32 |

3.4 Synchronisation

The synchronisation object SYNC controls the synchronous exchange of messages between network devices for purposes such as the simultaneous start of multiple drives.

The data exchange conforms to the producer-consumer relationship. The SYNC object is sent to all devices by a network device and can be evaluated by all devices that support synchronous PDOs.

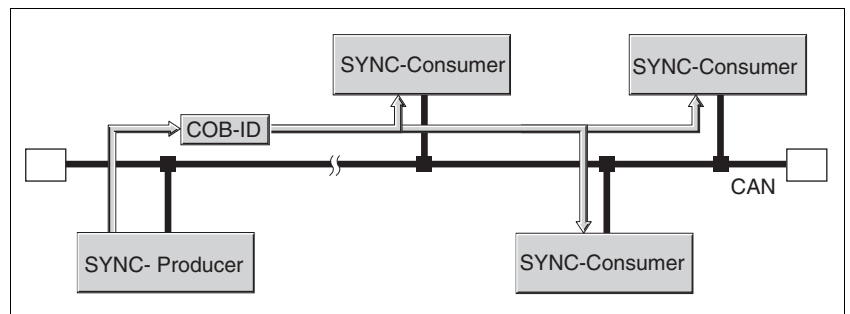


Figure 3.19 SYNC message

Time values for synchronisation

Two time values define the behaviour of synchronous data transfer:

- The cycle time specifies the time intervals between 2 SYNC messages. It is set with the object `Communication cycle period(1006h)`.
- The synchronous time window specifies the time interval in which the synchronous PDO messages must be received and sent. The time window is defined with the object `Synchronous window length (1007h)`.

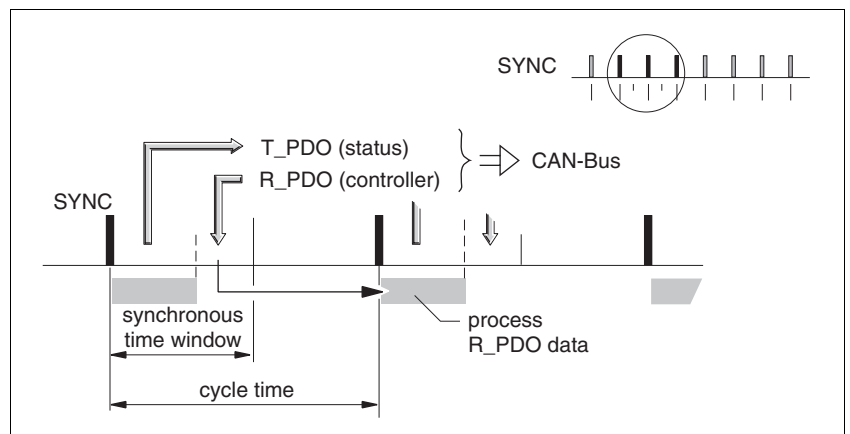


Figure 3.20 Synchronisation periods

Synchronous data transmission

From the point of view of a SYNC receiver, in one time window the status data are sent first in a T_PDO, then new control data are received via an R_PDO. However, the control data are only processed when the next SYNC message is received. The SYNC object itself does not transmit data.

Cyclic and acyclic data transfer

Synchronous exchange of messages can be executed cyclically or acyclically.

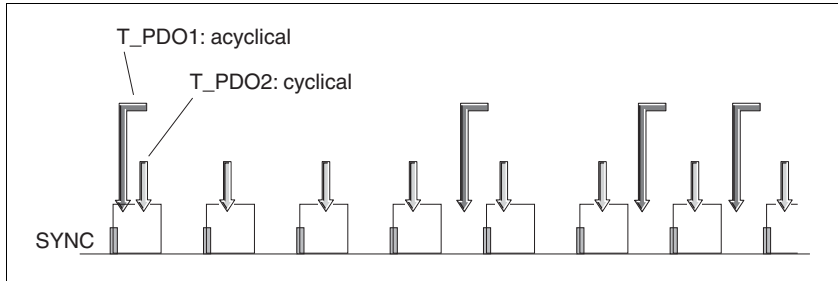


Figure 3.21 Cyclic and acyclic transmission

In cyclic transmission PDO messages are exchanged continuously in a specified cycle, e.g. with every SYNC message.

If a synchronous PDO message is sent acyclically, it can be sent or received at any time, but will only be valid with the next SYNC message.

The cyclic or acyclic behaviour of a PDOs is stored in subindex `transmission type (02h)` of the corresponding PDO parameter, e.g. for `R_PDO1` in the object `1st receive PDO parameter (1400h:02h)`.

COB-Id, SYNC object

For fast transmission the SYNC object is transmitted unconfirmed and with high priority.

The COB-Id of the SYNC object is set to the value 128 (80_h) by default. The value can be changed after initialising the network with the object `COB-ID SYNC Message (1005h)`.

"Start" PDO

In the default setting of the PDOs `R_PDO2/T_PDO2` and `R_PDO3/T_PDO3` are received and transmitted synchronously. Both PDOs are used for starting and monitoring operating modes. The synchronisation allows an operating mode to be started simultaneously on multiple devices and, for example, synchronisation of the feed of a multi-motor portal drive.

3.5 Emergency service

The emergency service reports internal device error over the CAN bus. The error message is sent to all devices with an EMCY object in accordance with the consumer-producer relationship.

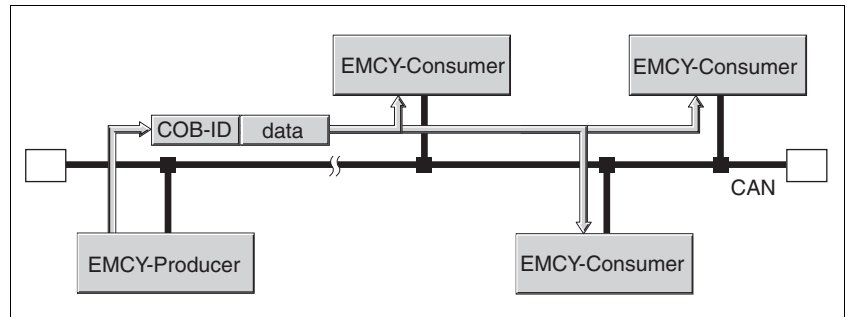


Figure 3.22 Error message via EMCY objects

Boot Up message

The communications profile DS 301, version 3.0, defines an additional task for the EMCY object: sending a boot-up message. A boot-up message informs all network devices that the device that sent the message is ready for operation in the CAN network.

The boot-up message is sent with the COB-ID 700h + Node-ID and one data byte (00h).

3.5.1 Error evaluation and handling

EMCY message

If an internal device error occurs, the device switches to error status as per the CANopen status machine. At the same time it sends an EMCY message with error register and error code.

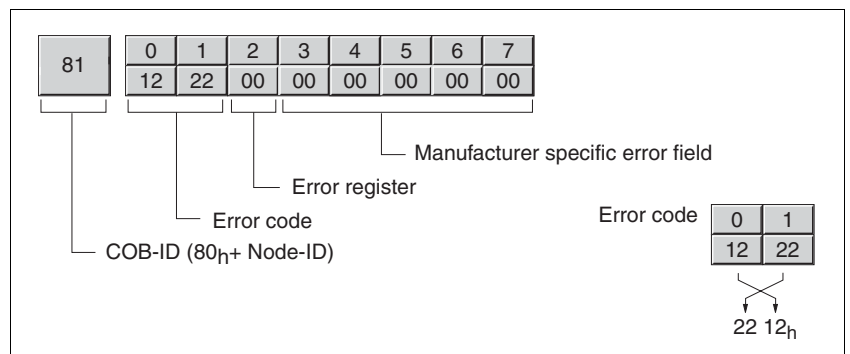


Figure 3.23 EMCY message

Byte 0, 1 - error code: Error code, value is also saved in the object Error code (603F_h)

Byte 2 - error register: Error register, value is also saved in the object Error register (1001_h), see 7.3.1 "error register".

Byte 3, 4 - Manufacturer-specific error code of mapped object

Byte 5, 6 - Index of mapped object

Byte 7 - Subindex of mapped object

| | |
|--------------------------------------|---|
| <i>COB-ID</i> | <p>The COB-Id is calculated from the node address for every device in the network that supports an EMCY object:</p> <p>$\text{COB-Id} = \text{function code EMCY object (80}_{\text{h}}) + \text{node-Id}$</p> <p>The function code of the COB-Id can be changed with the object COB-ID emergency(1014_h).</p> |
| <i>Error register and error code</i> | <p>The error register reports the error status of the device in bit-coded form. Bit 0 remains set so long as an error is pending. The remaining bits identify the error type. The precise cause of error can be found with the error code. The error code is sent in Intel format as a 2-byte value and must be reversed by bytes for evaluation.</p> <p>A list of all error messages and responses by the device and remedies can be found in chapter 7 "Diagnostics and troubleshooting".</p> |
| <i>Error memory</i> | <p>The device saves the error register in the object Error register (1001_h) and the last error that occurred in the object Error code (603F_h). The last 20 error messages are backed up in sequence of occurrence in the object FLT_err_num (303C:1_h). FLT_MemReset (303B:5_h) resets the read flag of the error memory to the oldest error.</p> |

3.6 Network management services

Network management (NMT) is a component of the CANopen communications profile and is used to initialise the network and start, stop and monitor the network devices in network mode.

NMT services are executed in a master-slave relationship. The NMT master addresses individual NMT slaves through their node address. A message with node address "0" is directed to all NMT slaves simultaneously.

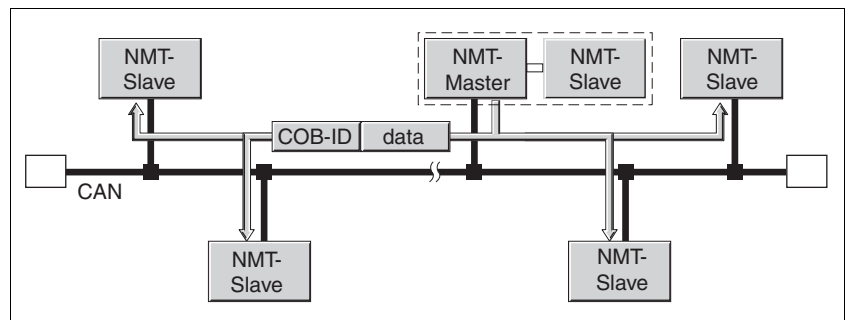


Figure 3.24 NMT services over the master-slave relationship

The device can only take on the function of a NMT slave.

NMT services NMT services can be divided into two groups:

- Services for device control, to initialise devices for CANopen communications and to control the behaviour of devices in network operation
- Services for connection monitoring, to ensure error-free network operation

3.6.1 NMT services for device control

NMT status machine The NMT status machine describes the initialising and status of a NMT slave in network operation.

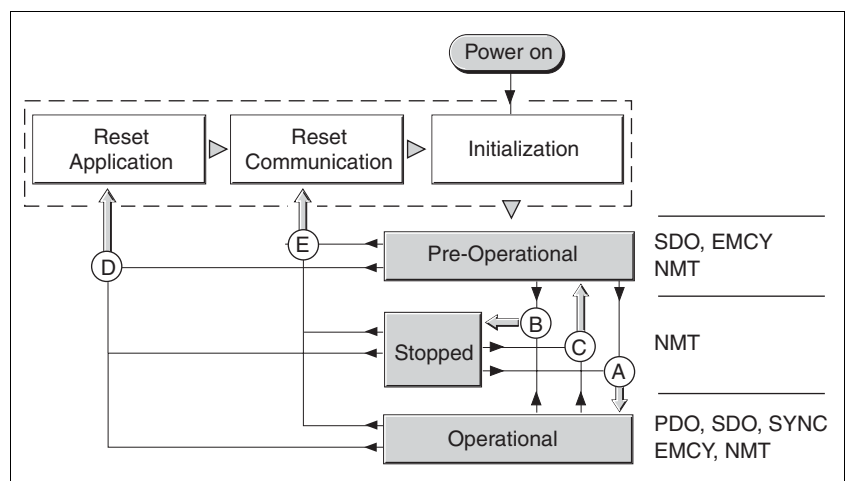


Figure 3.25 NMT status machine and available communications objects

The graphic shows on the right side all communications objects that can be used in the specific network status.

Initialisation A NMT slave automatically runs through an initialisation phase after the supply voltage is switched on (power on) to prepare it for CAN bus operation. On completion of the initialising process the slave switches to the "pre-operational" status and sends a boot-up message. Now a NMT master can control the operational behaviour of a NMT slave in the network with 5 NMT services, shown in the above graph with the letters A to E.

| NMT service | Transition | Meaning |
|--|------------|---|
| Start remote node (Start network nodes) | A | Switch to "Operational" status Start normal network operation to all devices |
| Stop remote node (Stop network nodes) | B | Switch to "Stopped" status Stop communications of the device in the network. If connection monitoring is enabled, it remains switched on |
| Enter pre-operational (Switch to "pre-operational") | C | Switch to "Pre-Operational" status All communications objects except for PDOs can be used. The "pre-operational" status can be used for configuration by SDOs: - PDO mapping - start of synchronisation - start of connection monitoring |
| Reset node (Reset nodes) | D | Switch to "Reset application" status Load saved data of the device profiles and switch automatically to "pre-operational" via "Reset communication" status |
| Reset communication (Reset communications data) | E | Switch to "Reset communication" status Load saved data of the communications profiles and switch automatically to "pre-operational" status |

Non-volatile saved data If the supply voltage is switched on (power on), the device loads the non-volatile saved object data from the EEPROM to the RAM.

NMT message The NMT services for device control are sent as unconfirmed message with the COB-Id = 0. By default they receive top priority on the CAN bus.

The data frame of the NMT device service consists of 2 bytes.

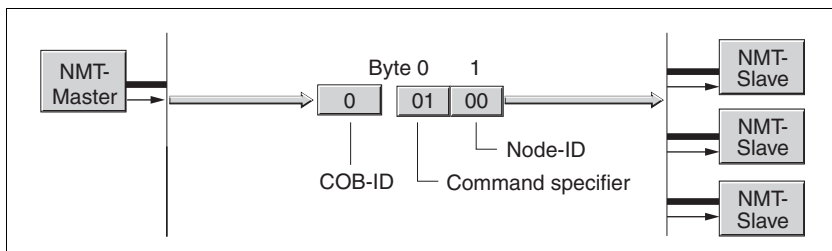


Figure 3.26 NMT message

The first byte, the "command specifier" identifies the NMT service in use.

| Command Specifier | NMT service | Transition |
|------------------------|-----------------------|------------|
| 1 (01 _h) | Start remote node | A |
| 2 (02 _h) | Stop remote node | B |
| 128 (80 _h) | Enter Pre-Operational | C |
| 129 (81 _h) | Reset node | D |
| 130 (82 _h) | Reset communication | E |

The second byte addresses the receiver of a NMT message with a node address between 1 and 127 (7F_h). A message with the node address "0" is directed to all NMT slaves.

3.6.2 services for connection monitoring

Connection monitoring monitors the communications status of network devices, so a response to the failure of a device or an interruption in the network is possible.

Three NMT services for connection monitoring are available:

- "Node guarding" for monitoring the connection of a NMT slave
- "Life guarding" for monitoring the connection of a NMT master
- "Heartbeat" for the unconfirmed connection message from network devices.

3.6.2.1 Node/Life guarding

COB-ID Connection monitoring is executed with the communications object NMT error control (700_h+node-Id). The COB-Id for every NMT slave is calculated from the node address:

COB-Id = function code NMT error control (700_h) + node-Id .

Structure of the NMT message On request of the NMT master the NMT slave responds with one data byte.

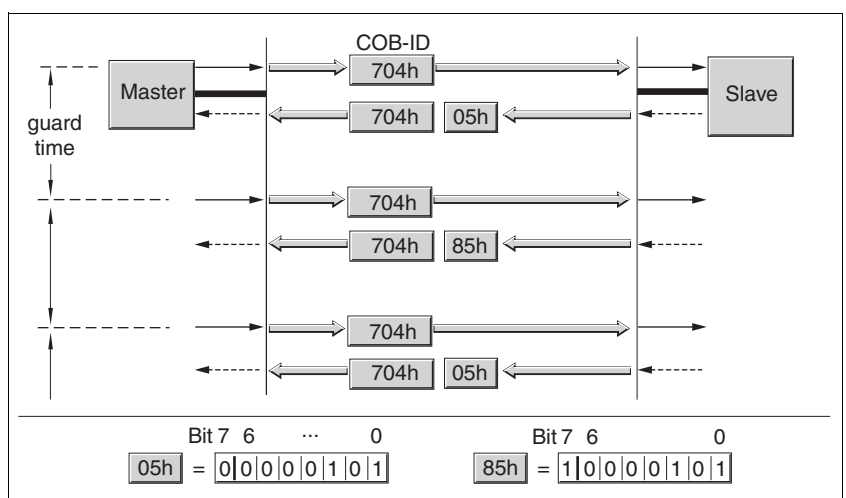


Figure 3.27 Acknowledgement of the NMT slave

Bit 0 to 6 identify the NMT status of the slave:

- 4 (04_h): "Stopped"
- 5 (05_h): "Operational"
- 127 (7F_h) : "Pre-Operational"

After every guard-time interval bit 7 switches its status between "0" and "1", so the NMT master can detect and ignore a second acknowledgement within the guard-time interval. The first request when starting connection monitoring begins with bit 7 = 0.

Connection monitoring must not be enabled during the initialisation phase of a device. The status of bit 7 is reset as soon as the device runs though the NMT status "Reset communication".

Connection monitoring continues in the NMT status "Stopped".

Configuration Node/life guarding is configured by:

- guard time (100C_h)
- life time factor (100D_h)

Connection error The NMT master reports a connection error to the higher level master program if:

- the slave does not acknowledge within the "guard time" period
- the NMT status of the slave has changed without the initiation of the NMT master.

Figure 3.28 shows an error message after the end of the third cycle because of a missing answer of a NMT slave.

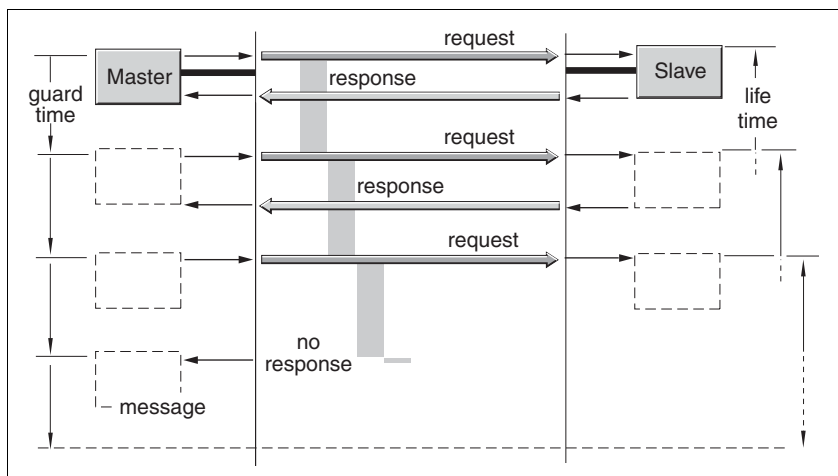


Figure 3.28 "Node guarding" and "life guarding" with time intervals

3.6.2.2 Heartbeat

The optional heartbeat protocol replaces the node/life guarding protocol. It is recommended for new device versions.

A heartbeat producer transmits a heartbeat message cyclically at the frequency defined in the object `Producer heartbeat time` (1017_h). One or more consumers can receive this message. `Producer heartbeat time` (1016_h) = 0 disables heartbeat monitoring.

The relationship between producer and consumer can be configured with objects. If a consumer does not receive a signal within the time intervals defined in the heartbeat time consumer object (1016_h), it generates an error message (heartbeat event). `Consumer heartbeat time` (1016_h) = 0 disables monitoring by a consumer.

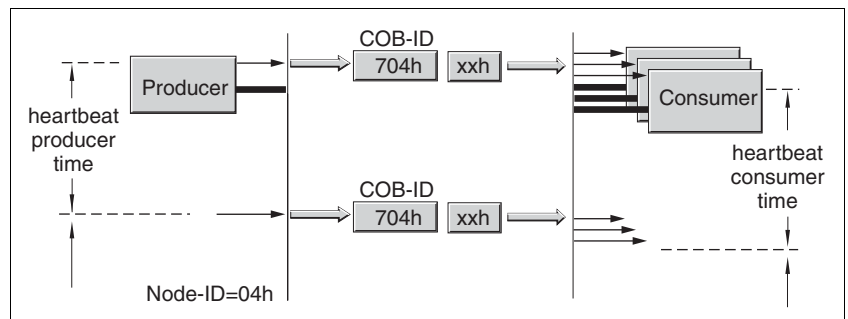


Figure 3.29 "Heartbeat" monitoring

Data byte for NMT status evaluation of the heartbeat producer:

- 0 (00_h): "Boot Up"
- 4 (04_h): "Stopped"
- 5 (05_h): "Operational"
- 127 (7F_h): "Pre-Operational"

Time intervals

The time intervals are set in 1-ms steps and must not be set smaller for the consumer than for the producer. Whenever the heartbeat message is received the time interval of the producer is restarted.

Start of monitoring

Heartbeat monitoring starts as soon as the time interval of the producer is greater than zero. If heartbeat monitoring is active during the NMT status change to "Pre-Operational", the heartbeat monitoring starts by sending the boot up message. The boot up message is a heartbeat message with one data byte 00_h.

Devices can monitor each other by heartbeat message. They have consumer and producer function simultaneously.

4 Installation



WARNING!

Danger of injury and damage to system components by loss of control!

- Observe the accident prevention regulations.
- The system manufacturer must consider the possible errors that could occur with the signals and in particular the critical functions to ensure a safe status during and after errors. Examples for these are: emergency stop, final position limit, power failure and restart.
- Consideration of possible errors must also include unexpected delay and failure of signals or functions.
- Separate redundant controller paths must be provided for dangerous functions.
- Verify the effectiveness of the measures.



WARNING!

Interference with signals and devices may cause injury

Distorted signals can cause unexpected device responses.

- Install the wiring in accordance with the EMC requirements.
- Check compliance with the EMC requirements, particularly in an environment subject to strong interference.

For information on device installation and connecting the device to the field bus see the device manual.

5 Commissioning

5.1 Setting up the unit

For installation in the network the unit must be mechanically and electrically installed correctly and the unit must be successfully commissioned.

Set up the unit following the manual. This prepares the unit for operation in the network.

5.2 Setting the address and the baud rate

Up to 32 devices can be addressed in one CAN bus network branch and up to 127 devices in the extended network. Every unit is identified by a unique address. The default node address for a unit is 127.

The baud rate is preset to 125 kbaud.



Every unit must be assigned its own node address, i.e. every node address must be assigned only once in the network.

Setting address and baud rate

During the address and baud rate setting the address is set locally at the unit in the parameter `canAddr` and the baud rate in the parameter `canBaud`.

The baud rate must be the same for all units in the field bus.

5.3 SYCON CANopen Configuration Software

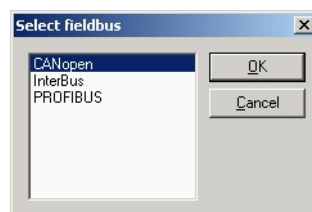
The CANopen network can be configured using the "SyCon" configuration software.

► In order to do so please follow the steps indicated below:

5.3.1 Creating a new network

A new network is created using the menu option "File - New".

- Select CANopen as the fieldbus network.
- Confirm your selection by clicking on "OK".

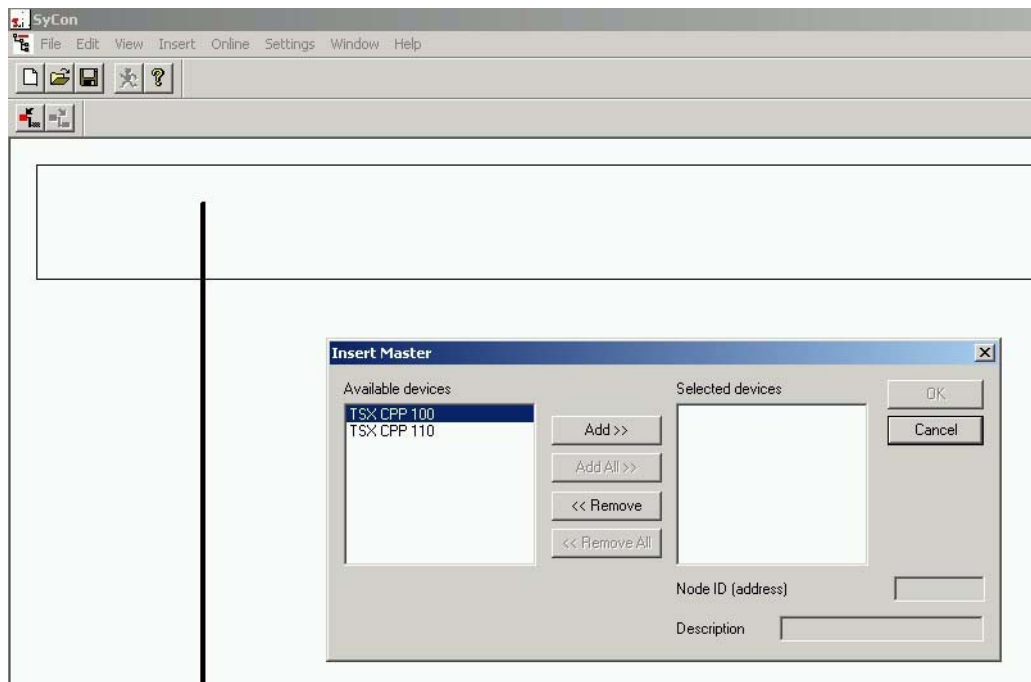


5.3.2 Selecting the CANopen Master

The network master can be selected using the menu option "Insert - Master". In the example the Premium PLC TSX CPP 110 card is used.

The node address and a brief description can be entered directly.

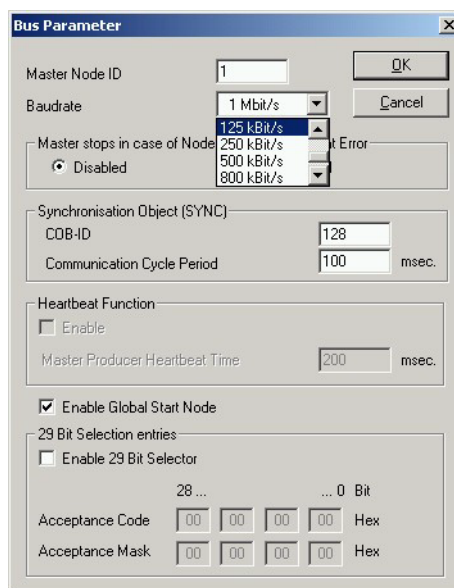
- Confirm your selection by clicking on "OK".



5.3.3 Setting the bus parameters

The CANopen communication parameters are set using the menu option "Settings - Bus Parameter...". Please also consult the Operation Instructions for the SyCon configuration software.

- Confirm your selection by clicking on "OK".



Bus Parameter

Master Node ID: 1

Baudrate: 1 Mbit/s

Master stops in case of Node Error: ☒ Disabled

Synchronisation Object (SYNC): COB-ID: 128, Communication Cycle Period: 100 msec.

Heartbeat Function: ☐ Enable, Master Producer Heartbeat Time: 200 msec.

☒ Enable Global Start Node

29 Bit Selection entries: ☐ Enable 29 Bit Selector

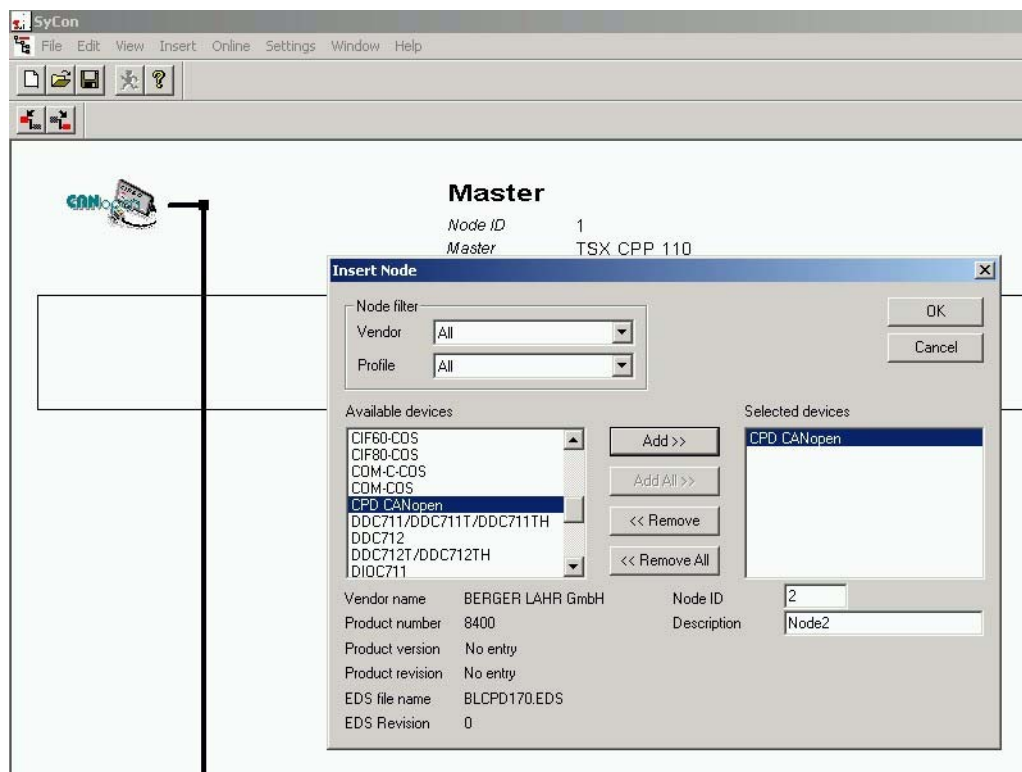
Acceptance Code: 00 00 00 00 Hex

Acceptance Mask: 00 00 00 00 Hex

5.3.4 Selecting and inserting the nodes

The network nodes are selected using the menu option "Insert - Node". In the example a CPD170 servo drive is used.

- Confirm your selection by clicking on "OK".



SyCon

File Edit View Insert Online Settings Window Help

Master

Node ID: 1

Master: TSX CPP 110

Insert Node

Node filter:

Vendor: All

Profile: All

Available devices:

- CIF60-COS
- CIF80-COS
- COM-C-COS
- COM-COS
- CPD CANopen**
- DDC711/DDC711T/DDC711TH
- DDC712
- DDC712T/DDC712TH
- DI0C711

Selected devices:

- CPD CANopen**

Buttons: Add >>, Add All >>, << Remove, << Remove All

Vendor name: BERGER LAHR GmbH

Product number: 8400

Product version: No entry

Product revision: No entry

EDS file name: BLCPD170.EDS

EDS Revision: 0

Node ID: 2

Description: Node2

5.3.5 Configuring the network node

By double clicking on the network node the user accesses the node configuration. This allows setting the communication characteristics for the selected node.

These are essentially the PDO characteristics for the freely parameterizable PDO4.

Node Configuration

Node: CPD CANopen Node ID (address): 2

Description: CPD_Drive

File name: BLCPD170.EDS

☒ Activate node in actual configuration

☒ Automatic COB-ID allocation in accordance with Profile 301

Configuration Error Control Protocol

Emergency COB-ID: 130

Nodeguard COB-ID: 1794

Device Profile: 402 Device type: Servo Drive

Predefined Process Data Objects (PDOs) from EDS file:

| Obj.Idx. | PDO name | COB-ID |
|----------|--------------------------------|--------|
| 1801 | 2nd transmit PDO communication | |
| 1802 | 3rd transmit PDO communication | |
| 1803 | 4th transmit PDO communication | |

Actual node: 2 / CPD CANopen

PDO mapping method: DS301 V4

Add to configured PDOs

Configured PDOs:

| PDO name | Symbolic Name | COB-ID | I Type | I Addr. | I Len. | O Type | O Addr. | O Len. |
|------------------|---------------|--------|--------|---------|--------|--------|---------|--------|
| 4th receive PDO | PDO_1403 | 1282 | | | | QB | 0 | 0 |
| 4th transmit PDO | PDO_1803 | 1154 | IB | 0 | 0 | | | |

PDO Contents Mapping...

PDO Characteristics...

Define new Receive PDO...

Define new Transmit PDO...

Delete configured PDO

Symbolic Names

The "Configuration Error Control Protocol" button can be used to set the node's malfunctioning behaviour. Here the selection is made as to whether the node is monitored with the Nodeguarding Protokoll or with the Heartbeat Protokoll.

► Confirm your selection by clicking on "OK".

By double clicking on object 1403 "4th receive PDO communication" or object 1803 "4th transmit PDO communication" the dialogue field opens where the transmitting characteristics of the respective PDO can be set. The previously set values can be accepted without modification.

► Confirm by clicking on "OK".

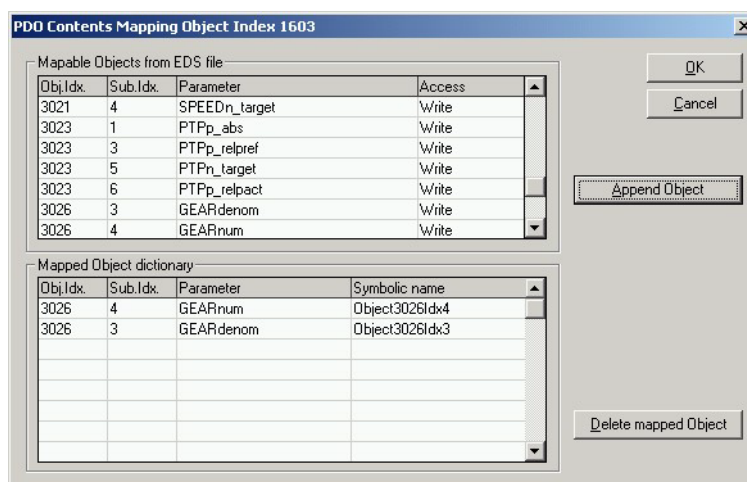
The PDO4 mapping can be freely set using the "PDO Contents Mapping" button.

5.3.6 Setting the PDO4 mapping

Both in the receiving PDO4 and the transmitting PDO4 up to 4 objects can be parameterised for each using the "Append Object" button. It must be ensured that the total number of 8 bytes is not exceeded during this procedure.

In the example, the receiving PDO4 gear numerator and denominator (32bit respectively) are allocated.

- Confirm your selection by clicking on "OK".



Save the configuration using the menu option "File - Save as...".

6 Operation

6.1 Operating modes

| | |
|--|--|
| <i>Local controller and field bus controller</i> | <p>With a local controller the movement is preset with analogue signals ($\pm 10V$) or with RS422 signals (e.g. pulse/direction).</p> <p>When the controller is connected over the field bus the setpoint values are preset via field bus commands.</p> <p>For detailed information on setting the type of controller see the device manual under commissioning.</p> |
| <i>Standardised operating modes</i> | <p>The unit operates in 3 standardised operating modes. They can be started and monitored with the objects of the CANopen device profile DSP 402.</p> <ul style="list-style-type: none">• Point-to-point mode via the objects of the object group <code>Profile position mode</code>• Speed profile via the objects of the object group <code>Profile velocity mode</code>• Referencing via the objects of the object group <code>Homing mode</code> |
| <i>Manufacturer-specific operating modes</i> | <p>The unit operates in 4 manufacturer-specific operating modes. They use all the functions of the unit. The operating modes are started and monitored with manufacturer-specific objects.</p> <ul style="list-style-type: none">• Current control• Speed control• Electronic gear• Manual movement |

6.2 Standardised operating modes

6.2.1 Point-to-point operating mode

The PDO2 must be enabled for the point-to-point operating mode (profile position mode). It can be used to specify motion parameters such as ramps and speeds.

Example Node address 1.

| Work step COB-ID / data | Object value |
|--|---|
| ► Enable R_PDO2 601 / 23 01 14 01 01 03 00 04 | 1401:1 _h 0400 0301 _h |
| ◁ 581 / 60 01 14 01 00 00 00 00 | |
| ► Enable T_PDO2 601 / 23 01 18 01 81 02 00 04 | 1801:1 _h 0400 0281 _h |
| ◁ 581 / 60 01 18 01 00 00 00 00 | |
| ► Set acceleration ramp to 2000 rpm*s 601 / 23 83 60 00 D0 07 00 00 | 6083 _h 0000 07D0 _h |
| ◁ 581 / 60 83 60 00 00 00 00 00 | |
| ► Set deceleration ramp to 4000 rpm*s 601 / 23 84 60 00 A0 0F 00 00 | 6084 _h 0000 0FA0 _h |
| ◁ 581 / 60 84 60 00 00 00 00 00 | |
| ► Restrict setpoint speed to 6000 rpm 601 / 23 7F 60 00 70 17 00 00 | 607F _h 0000 1770 _h |
| ◁ 581 / 60 7F 60 00 00 00 00 00 | |
| ► Set setpoint speed to 4000 rpm 601 / 23 81 60 00 A0 0F 00 00 | 6081 _h 0000 0FA0 _h |
| ◁ 581 / 60 81 60 00 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |
| ► Start operating mode 601 / 2F 60 60 00 01 00 00 00 | 6060 _h 01 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ► Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 01 00 01 00 | 01 _h |

| Work step COB-ID / data | Object value |
|--|-----------------|
| ► PDO2: set relative position with NewSetpoint=1 301 / 5F 00 30 75 00 00 | |
| ◁ T_PDO2 with status word and position actual value 281 / 37 56 00 00 00 00 | |
| ◁ Position reached 281 / 37 56 30 75 00 00 | |
| ► PDO2: NewSetpoint=0 301 / 4F 00 30 75 00 00 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.2.2 Speed profile operating mode

The PDO3 must be enabled for the speed profile operating mode (profile velocity mode).

Example Node address 1.

| Work step COB-ID / data | Object value |
|--|---|
| ► Enable R_PDO3 601 / 23 02 14 01 01 04 00 04 | 1402:1 _h 0400 0401 _h |
| ◁ 581 / 60 02 14 01 00 00 00 00 | |
| ► Enable T_PDO3 601 / 23 02 18 01 81 03 00 04 | 1802:1 _h 0400 0381 _h |
| ◁ 581 / 60 02 18 01 00 00 00 00 | |
| ► Set acceleration ramp to 2000 rpm*s 601 / 23 83 60 00 D0 07 00 00 | 6083 _h 0000 07D0 _h |
| ◁ 581 / 60 83 60 00 00 00 00 00 | |
| ► Set deceleration ramp to 10000 rpm*s 601 / 23 84 60 00 10 27 00 00 | 6084 _h 0000 2710 _h |
| ◁ 581 / 60 84 60 00 00 00 00 00 | |
| ► Restrict setpoint speed to 10000 rpm 601 / 23 7F 60 00 10 27 00 00 | 607F _h 0000 2710 _h |
| ◁ 581 / 60 7F 60 00 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |
| ► Start operating mode 601 / 2F 60 60 00 03 00 00 00 | 6060 _h 03 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ► Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 03 00 01 00 | 03 _h |
| ► PDO3: send setpoint speed 1000 rpm 401 / 0F 00 E8 03 00 00 | |
| ◁ T_PDO2 with status word and position actual value 381 / 37 02 00 00 00 00 00 | |
| ◁ Setpoint speed reached 381 / 37 06 E8 03 00 00 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.2.3 Referencing operating mode

The parameters for the referencing operating mode (Homing Mode) are set with SDOs and enabled with PDO1.

Example Node address 1.

| Work step COB-ID / data | Object value |
|--|---|
| ► Setpoint speed for movement to limit switch 100 rpm 601 / 23 99 60 01 64 00 00 00 | 6099:1 _h 0000 0064 _h |
| ◁ 581 / 60 99 60 01 00 00 00 00 | |
| ► Setpoint speed for free movement 10 rpm 601 / 23 99 60 02 0A 00 00 00 | 6099:2 _h 0000 000A _h |
| ◁ 581 / 60 99 60 02 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |
| ► Start operating mode 601 / 2F 60 60 00 06 00 00 00 | 6060 _h 06 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ► Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 06 00 01 00 | 06 _h |
| ► Select reference movement method, LimN (17) 601 / 2F 98 60 00 11 00 00 00 | 6098 _h 11 _h |
| ◁ 581 / 60 98 60 00 00 00 00 00 | |
| ► Reference movement with PDO1 (homing operation start) 201 / 1F 00 | |
| ◁ TPDO1 reference movement active 181 / 37 02 | |
| ◁ TPDO1 reference movement finished 181 / 37 D6 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.3 Manufacturer-specific operating modes

6.3.1 Current regulation mode.

Example Node address 1.

| Work step COB-ID / data | Object value |
|---|---|
| ► Mapping R_PDO4: number of mapped objects = 0 601 / 2F 03 16 00 00 00 00 00 | 1603:0 _h 00 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► R_PDO4 first parameter = CUR_I_target (3020:4 _h) 601 / 23 03 16 01 10 04 20 30 | 1603:1 _h 3020 0410 _h |
| ◁ 581 / 60 03 16 01 00 00 00 00 | |
| ► R_PDO4 Number of mapped objects = 1 601 / 2F 03 16 00 01 00 00 00 | 1603:0 _h 01 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► Mapping T_PDO4: number of mapped objects = 0 601 / 2F 03 1A 00 00 00 00 00 | 1A03:0 _h 00 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► T_PDO4 first parameter = _p_actusr (6064:0) 601 / 23 03 1A 01 20 00 64 60 | 1A03:1 _h 6064 0020 _h |
| ◁ 581 / 60 03 1A 01 00 00 00 00 | |
| ► T_PDO4 Number of mapped objects = 1 601 / 2F 03 1A 00 01 00 00 00 | 1A03:0 _h 01 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► Enable R_PDO4 (COB-ID) 601 / 23 03 14 01 01 05 00 04 | 1403:1 _h 0400 0501 _h |
| ◁ 581 / 60 03 14 01 00 00 00 00 | |
| ► Enable T_PDO4 (COB-ID) 601 / 23 03 18 01 81 04 00 04 | 1803:1 _h 0400 0481 _h |
| ◁ 581 / 60 03 18 01 00 00 00 00 | |
| ► Setpoint specification with parameter 601 / 2B 1B 30 10 02 00 00 00 | 301B:10 _h 02 _h |
| ◁ 581 / 60 1B 30 10 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |

| Work step COB-ID / data | Object value |
|---|---------------------------------------|
| ▶ Start operating mode 601 / 2F 60 60 00 FD 00 00 00 | 6060 _h -03 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ▶ Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 FD 00 01 00 | -03 _h |
| ▶ PDO4 send setpoint current 1000 (10A) 501 / E8 03 | |
| ◁ T_PDO4 with current position 481 / 00 CE 09 00 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.3.2 Speed regulation operating mode

Example Node address 1.

| Work step COB-ID / data | Object value |
|--|---|
| ► Mapping R_PDO4: number of mapped objects = 0 601 / 2F 03 16 00 00 00 00 00 | 1603:0 _h 00 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► R_PDO4 first parameter = SPEEDn_target (3021:4 _h) 601 / 23 03 16 01 10 04 21 30 | 1603:1 _h 3021 0410 _h |
| ◁ 581 / 60 03 16 01 00 00 00 00 | |
| ► R_PDO4 Number of mapped objects = 1 601 / 2F 03 16 00 01 00 00 00 | 1603:0 _h 01 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► Mapping T_PDO4: number of mapped objects = 0 601 / 2F 03 1A 00 00 00 00 00 | 1A03:0 _h 00 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► T_PDO4 first parameter = _p_actusr (6064:0) 601 / 23 03 1A 01 20 00 64 60 | 1A03:1 _h 6064 0020 _h |
| ◁ 581 / 60 03 1A 01 00 00 00 00 | |
| ► T_PDO4 Number of mapped objects = 1 601 / 2F 03 1A 00 01 00 00 00 | 1A03:0 _h 01 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► Enable R_PDO4 (COB-ID) 601 / 23 03 14 01 01 05 00 04 | 1403:1 _h 0400 0501 _h |
| ◁ 581 / 60 03 14 01 00 00 00 00 | |
| ► Enable T_PDO4 (COB-ID) 601 / 23 03 18 01 81 04 00 04 | 1803:1 _h 0400 0481 _h |
| ◁ 581 / 60 03 18 01 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |
| ► Start operating mode 601 / 2F 60 60 00 FC 00 00 00 | 6060 _h -04 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ► Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 FC 00 01 00 | -04 _h |

| Work step COB-ID / data | Object value |
|--|---|
| ► Setpoint specification with parameter 601 / 2B 1B 30 11 02 00 00 00 | 301B:11 _h 02 _h |
| ◁ 581 / 60 1B 30 11 00 00 00 00 | |
| ► PDO4 send setpoint speed 1000 rpm 501 / E8 03 | |
| ◁ T_PDO4 with current position 481 / 6E 97 04 00 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.3.3 Electronic gear mode

Example Node address 1.

| Work step COB-ID / data | Object value |
|--|---|
| ► Mapping R_PDO4: number of mapped objects = 0 601 / 2F 03 16 00 00 00 00 00 | 1603:0 _h 00 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► R_PDO4 first parameter = GEARnum (3026:4 _h) 601 / 23 03 16 01 20 04 26 30 | 1603:1 _h 3026 0420 _h |
| ◁ 581 / 60 03 16 01 00 00 00 00 | |
| ► R_PDO4 first parameter = GEARdenom (3026:3 _h) 601 / 23 03 16 02 20 03 26 30 | 1603:2 _h 3026 0320 _h |
| ◁ 581 / 60 03 16 02 00 00 00 00 | |
| ► R_PDO4 Number of mapped objects = 2 601 / 2F 03 16 00 02 00 00 00 | 1603:0 _h 02 _h |
| ◁ 581 / 60 03 16 00 00 00 00 00 | |
| ► Mapping T_PDO4: number of mapped objects = 0 601 / 2F 03 1A 00 00 00 00 00 | 1A03:0 _h 00 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► T_PDO4 first parameter = _p_actusr (6064:0) 601 / 23 03 1A 01 20 00 64 60 | 1A03:1 _h 6064 0020 _h |
| ◁ 581 / 60 03 1A 01 00 00 00 00 | |
| ► T_PDO4 Number of mapped objects = 1 601 / 2F 03 1A 00 01 00 00 00 | 1A03:0 _h 01 _h |
| ◁ 581 / 60 03 1A 00 00 00 00 00 | |
| ► Enable R_PDO4 (COB-ID) 601 / 23 03 14 01 01 05 00 04 | 1403:1 _h 0400 0501 _h |
| ◁ 581 / 60 03 14 01 00 00 00 00 | |
| ► Enable T_PDO4 (COB-ID) 601 / 23 03 18 01 81 04 00 04 | 1803:1 _h 0400 0481 _h |
| ◁ 581 / 30 03 18 01 00 00 00 00 | |
| ► Signal selection position interface 601 / 2B 05 30 02 01 00 00 00 | 3005:2 _h 01 _h |
| ◁ 581 / 60 05 30 02 00 00 00 00 | |
| ► Enable gearbox with immediate synchronisation 601 / 2B 1B 30 12 01 00 00 00 | 301B:12 _h 01 _h |
| ◁ 581 / 60 1B 30 12 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |

| Work step COB-ID / data | Object value |
|---|---|
| <ul style="list-style-type: none"> ▶ Activation of power circuit with PDO1 <ul style="list-style-type: none"> 201 / 00 00 201 / 06 00 201 / 0F 00 ◁ T_PDO1 (status: operation enabled) <ul style="list-style-type: none"> 181 / 37 46 | |
| <ul style="list-style-type: none"> ▶ Start operating mode <ul style="list-style-type: none"> 601 / 2F 60 60 00 FE 00 00 00 ◁ 581 / 60 60 60 00 00 00 00 00 | 6060 _h -02 _h |
| <ul style="list-style-type: none"> ▶ Check operating status ¹⁾ <ul style="list-style-type: none"> 601 / 40 61 60 00 00 00 00 00 ◁ Operating mode active <ul style="list-style-type: none"> 581 / 4F 61 60 00 FE 00 01 00 | 6061 _h -02 _h |
| <ul style="list-style-type: none"> ▶ PDO4 send gear factor 2/3 <ul style="list-style-type: none"> 501 / 02 00 00 00 03 00 00 00 ◁ T_PDO4 with current position <ul style="list-style-type: none"> 481 / 53 76 01 00 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.3.4 Manual movement mode

Example Node address 1.

| Work step COB-ID / data | Object value |
|---|--|
| ► Speed slow movement to 100 rpm 601 / 2B 29 30 04 64 00 00 00 | 3029:4 _h 0064 _h |
| ◁ 581 / 60 29 30 04 00 00 00 00 | |
| ► Speed fast movement to 250 rpm 601 / 2B 29 30 05 FA 00 00 00 | 3029:5 _h 00FA _h |
| ◁ 581 / 60 29 30 05 00 00 00 00 | |
| ► NMT Start Remote Node 0 / 01 00 | |
| ◁ T_PDO1 with status word 181 / 31 66 | |
| ► Activation of power circuit with PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 | |
| ◁ T_PDO1 (status: operation enabled) 181 / 37 46 | |
| ► Start operating mode 601 / 2F 60 60 00 FF 00 00 00 | 6060 _h -01 _h |
| ◁ 581 / 60 60 60 00 00 00 00 00 | |
| ► Check operating status ¹⁾ 601 / 40 61 60 00 00 00 00 00 | 6061 _h |
| ◁ Operating mode active 581 / 4F 61 60 00 FF 00 01 00 | -01 _h |
| ► Manual movement (pos. direction of rotation, slow) 601 / 2B 1B 30 09 01 00 00 00 | 301B:9 _h 01 _h |
| ◁ 581 / 60 1B 30 09 00 00 00 00 | |
| ◁ T_PDO1 with status word 181 / 37 02 | |
| ► Manual movement (pos. direction of rotation, fast) 601 / 2B 1B 30 09 05 00 00 00 | 301B:9 _h 05 _h |
| ◁ 581 / 60 1B 30 09 00 00 00 00 | |
| ◁ T_PDO1 with status word 181 / 37 42 | |

1) The operating status must be checked until the unit has enabled the specified operating mode.

6.4 Functions

6.4.1 ramp function

The unit controls the acceleration and deceleration behaviour of the motor with ramp functions. The gradient and shape of the ramp describe the ramp function. The ramp gradient shows the motor's change of speed, and the shape of the ramp the acceleration over time.

For details of the ramp function see the device manual in the chapter on the functions.

| Object (Index:Subindex) | Meaning |
|---|------------------------------|
| Profile acceleration (6083 _h) | Acceleration [usr] |
| Profile deceleration (6084 _h) | Deceleration [usr] |
| Max profile velocity (607F _h) | Limitation of setpoint speed |

6.4.2 Quick-Stop function

Quick-Stop is an emergency braking function, which stops the motor, e.g. if a fault occurs.

For details of the Quick-Stop function see the device manual in the chapter on the functions.

The motor can be decelerated with the Quick-Stop current, object LIM_I_maxQSTP (3011:5_h), with the stop current, object LIM_I_maxHalt (3011:6_h) or the deceleration ramp of the travel profile, object Profile deceleration (6084_h).

| Object (Index:Subindex) | Meaning |
|---|---|
| LIM_I_maxQSTP (3011:5 _h) | Current limiting for Quick-Stop [0.01A] |
| LIM_I_maxHalt (3011:6 _h) | Current limiting for stop [0.01A] |
| Profile deceleration (6084 _h) | Deceleration ramp of the travel profile |

6.4.3 Motor stop

The drive can be stopped during a movement command over the field bus. If bit 8 in object Controlword (6040_h) switches to "1", the unit brakes the motor at the deceleration ramp that was specified for the movement command. The movement and position data are retained.

The movement command is continued as soon as bit 8 is switched to "0" again.

6.4.4 Standstill window

If the motor is retained at zero speed with closed-loop control enabled, minimum speed variations prevent detection of the motor standstill. If the motor remains in the standstill window for an adjustable period, the closed-loop control reports motor standstill. Bit 10 in the status word, object Statusword (6041_h) is set.

For details of the standstill window function see the device manual in the chapter on the functions.

| Object (Index:Subindex) | Meaning |
|---|---|
| Position window (6067 _h) | Standstill window, permissible control deviation |
| Position window time (6068 _h) | Time for which control deviations must apply in the standstill window for standstill to be signalled [ms] |

6.4.5 Reversal of direction of rotation

The direction of rotation of the drive can be reversed with the object `POSdirOfRotat` (3006:12_h). The limit switch connections must be reversed at the same time.

For details of reversing the direction of rotation see the device manual in the chapter on the functions.

| Object (Index:Subindex) | Meaning |
|---------------------------------------|--------------------------------|
| POSdirOfRotat (3006:12 _h) | Inversion of sense of rotation |

6.4.6 Monitoring functions

Positioning limits The motor can be moved to any point on the axis within the axis positioning range by specifying an absolute positioning process.

The axis travel range is specified in internal units in the range -2^{28} to $+2^{28}$ increments. The resolution of the motor encoder in increments is specified as the internal unit.

At a position overrun bit 15 (REF_OK) of the object *Statusword* (6041_h) is set to 0.

Software limit switch The software limit switch position is specified with the object *Max position limit* (607D_h) of the DSP 402 device profile.

The determining factor for position monitoring of the software limit switch range is the setpoint position of the position controller. Depending on the controller setting, therefore, the motor can stop before it reaches the limit switch position. Bit 2 of the object *_SigLatched* (301C:8_h) reports the tripping of the limit switch position.

| Object (Index:Subindex) | Meaning |
|---|---|
| Min position limit (607D:1 _h) | negative position limit for software limit switch |
| Max position limit (607D:2 _h) | positive position limit for software limit switch |
| _SigLatched (301C:8 _h) | monitoring signal bit 2 SW_LimP / SW_LimN |

Limit switch signal During motion, both limit switches are monitored by input signals LIMN and LIMP. When the limit switch is tripped the motor is stopped. Bit 1 of the object *_SigLatched* (301C:8_h) report the tripping of the limit switch.

| Object (Index:Subindex) | Meaning |
|------------------------------------|--|
| IOsigLimN (3006:F _h) | 0: inactive 1: break contact 2: make contact |
| IOsigLimN (3006:10 _h) | 0: inactive 1: break contact 2: make contact |
| _SigLatched (301C:8 _h) | Bit 1: LimP / LimN |

Following error monitoring Following error monitoring checks for positional discrepancies between the actual position of the motor and its setpoint. If the difference exceeds a following error threshold value, the unit reports an error. The threshold for the following error deviation can be set

| Object (Index:Subindex) | Meaning |
|---|--|
| Following error window (6065 _h) | Maximum permitted following error of the position controller [Inc] |

Monitoring parameters The unit and operating status can be monitored with various objects.

| Object (Index:Subindex) | Meaning |
|-----------------------------------|--------------------------------------|
| _SigActive (301C:7 _h) | Current status of monitoring signals |

| Object (Index:Subindex) | Meaning |
|--------------------------------------|------------------------------------|
| _SigLatched (301C:8 _h) | Saved status of monitoring signals |
| _WarnActive (301C:B _h) | Active warnings bit-coded |
| _WarnLatched (301C:C _h) | Saved warnings bit-coded |
| _actionStatus (301C:4 _h) | Action word |
| error code (603F _h) | cause of last interruption |

6.4.7 Monitoring inputs and outputs of the unit

The analogue signal and the digital signals of the unit can be monitored over the field bus.

The analogue inputs ANA1 and ANA2 are monitored with the objects ANA1_act (3009:1_h) and ANA2_act (3009:5_h). The digital inputs are monitored with the object _IO_act (3008:1_h). For example, this enables the start of a manual movement to be monitored by field bus with the interface signals.

| Object (Index:Subindex) | Meaning |
|---------------------------------|--|
| ANA1_act (3009:1 _h) | Monitoring analogue input ANA1 |
| ANA2_act (3009:5 _h) | Monitoring analogue input ANA2 |
| _IO_act (3008:1 _h) | Monitoring digital inputs and outputs [mV] |

6.4.8 Backing up and restoring object data

The unit copies non-volatile saved object data to the RAM memory after the unit is switched on. The unit works with the data in RAM during operation.

The data must be transferred to the non-volatile memory with the object PAReeprSave (3004:1_h) to back up user-specific object settings in case of power failure.

User-specific object settings can be reset with the object PARusrReset (3004:4_h).

| Object (Index:Subindex) | Meaning |
|------------------------------------|--------------------------------------|
| PAReeprSave (3004:1 _h) | Backing up object settings in EEPROM |
| PARusrReset (3004:4 _h) | Restoring object settings |

7 Diagnostics and troubleshooting

7.1 Field bus communication error diagnosis

Field-mode mode must be functioning to be able to evaluate operational and error messages.

Connections to field-bus mode

If the unit cannot be addressed over the field bus, first check the connections. The device manual contains the specifications of the unit and information on network and unit installation. Check the following:

- 24V_{DC} power supply
- Power connections to the unit
- Field-bus cable and field-bus wiring
- Network connection to the unit

The commissioning software can also be used for error diagnosis. For details see the software manual.

Baud rate and address

If a connection to a device cannot be made, check the baud rate and node address.

- The baud rate of all network devices must be set to the same value
- The node address of every device must be between 1 and 127 and must be different for device

To set the baud rate and node address see Chapter 5.2 “Setting the address and the baud rate”.

Function test on the field bus

After correct configuration of the transmission data test the field bus operation. This requires installation of a CAN configuration tool that displays CAN messages. The acknowledgement of the unit is captured by a boot-up message:

- Switch the power supply off and on again.
- Observe the network messages after switching it on. After initialisation of the bus the unit sends a boot-up message (COB-Id 700_h + node ID and 1 data byte with the content 00_h).
- The boot-up message is sent over the bus with the factory setting of the node address at 127 (7F_h). The unit can then be put into operation via NMT services.



If network operation cannot be started, the network function of the unit must be checked by your local representative. Contact your local representative.

7.2 Error diagnosis over field bus

7.2.1 Message objects

A number of objects provide information on the operating and error status:

- Object `Statusword` (`6041h`)
operating status, see device manual
- Object `EMCY` (`80h + node-ID`)
error message of a device with error status and error code, see Chapter 3.5 "Emergency service"
- Object `Error register` (`1001h`)
error status
- Object `Error code` (`603Fh`)
error code of the last occurring error
- Devices use the special SDO ABORT error message to report the failed message exchange via SDO (cancel)

7.2.2 Messages: on the unit status

A distinction is made between synchronous and asynchronous errors when evaluating and handling errors.

Synchronous error

The unit reports a synchronous error directly as a response to a message that cannot be evaluated. Possible causes can be faulty transmission or illegal data. For a list of synchronous errors see chapter 7.3.1 "error register".

Asynchronous errors

Asynchronous errors are reported by the monitoring devices of the unit as soon as a unit error occurs. An asynchronous error is reported via bit 3, "Fault", of the object `statusword` (`6041h`). For errors that cause a movement interruption the unit sends an EMCY message.

Asynchronous errors are also reported via bits 5..7 of the object `driveStat` (`2041h`).

7.3 CANopen error messages

CANopen error messages are displayed as an EMCY message. They are evaluated via the object `Error register (1001h)` and Error code (603F_h). For information on the object EMCY see chapter 3.5 “Emergency service”.

CANopen reports errors that occur during data exchange by SDO with the special SDO error message ABORT.

7.3.1 error register

The object `Error register(1001h)` shows the error status of a device in bit-coded form. The exact cause of error must be determined with the error code table. Bit 0 is set as soon as an error occurs.

| Bit | Message | Meaning |
|-----|-------------------------|--------------------------------------|
| 0 | generic error | An error has occurred |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | Communication | Error in network communications |
| 5 | Device profile-specific | Error in execution by device profile |
| 6 | reserved | reserved |
| 7 | Manufacturer-specific | Manufacturer-dependent error message |

7.3.2 Error code table

The error code is evaluated with the object `error code (603Fh)`, an object of the DSP 402 device profile, and output as a four-character hexadecimal number. The error code shows the cause of the last movement interruption. The meaning of the error code can be found in the device manual in the section on error diagnosis and troubleshooting.

7.3.3 SDO error message ABORT

A SDO error message is output as a response to an error in a SDO transmission. The cause of error is shown in `error code`, byte 4 to byte 7.

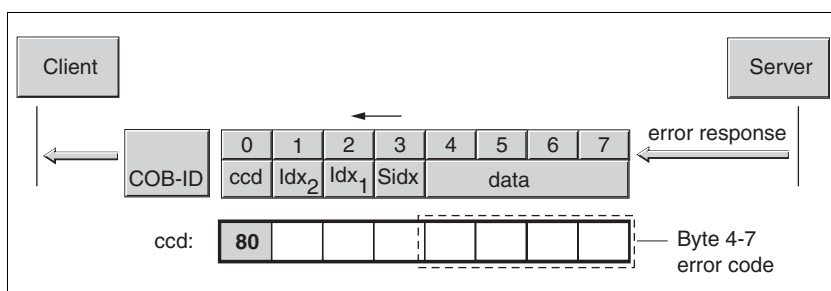


Figure 7.1 SDO error message as answer to a SDO message

The table below shows all error messages that may occur with the unit during data exchange.

| Error code | Meaning |
|--|---|
| 0503 0000 _h | Inverse bit not inverted |
| 0504 0000 _h | Time-out during SDO transfer |
| 0504 0001 _h | Command specifier CS incorrect or unknown |
| 0504 0002 _h 0504 0003 _h 0504 0004 _h | - |
| 0504 0005 _h | No free memory |
| 0601 0000 _h | No access to object possible |
| 0601 0001 _h | No read access, because write-only object (wo) |
| 0601 0002 _h | No write access, because read object (ro) |
| 0602 0000 _h | Object does not exist in object directory |
| 0604 0041 _h | Object does not support PDO mapping |
| 0604 0042 _h | PDO mapping: number or length of objects exceed the byte length of the PDO |
| 0604 0043 _h | Parameters are not compatible |
| 0604 0047 _h | Device detects internal incompatibility |
| 0606 0000 _h | Hardware error, access denied |
| 0607 0010 _h | Data type and parameter length do not match |
| 0607 0012 _h | Data type does not match, parameter too long |
| 0607 0013 _h | Data type does not match, parameter too short |
| 0609 0011 _h | Subindex not supported |
| 0609 0030 _h | Value range of parameter too large (relevant only for write access) |
| 0609 0031 _h | Parameter values too great |
| 0609 0032 _h | Parameter values too small |
| 0609 0036 _h | Top value is less than bottom value |
| 0800 0000 _h | General error |
| 0800 0020 _h | Data cannot be uploaded or saved to the application. |
| 0800 0021 _h | Device control is executed locally, data cannot be uploaded or saved. |
| 0800 0022 _h | Device status prevent uploading and saving data. |
| 0800 0023 _h | Object directory either not present or cannot be generated, e.g. if data error occurs when generating from file. |
| 0800 xxxx _h | Manufacturer-specific error, xxxx corresponds to the error number of the device. It is listed in the error code table of the device manual. |

8 Service, maintenance and disposal



CAUTION!

Destruction of system components and loss of control monitoring!

Interrupting the negative connection of the control supply can cause excessive voltages at the signal connections.

- Never switch the negative connection of the control supply.
- Check for correct connection before switching on.
- Never connect the control supply or change its wiring while there is supply voltage present.

8.1 Service address



*If you have any questions please contact your local dealer.
Your dealer will be happy to give you the name of a
customer service outlet in your area.*

9 Object directory

9.1 Specifications for the objects

Index This index shows the position of the object in the object directory. The index value is shown in hexadecimal.

Object code The object code shows the data structure of the object.

| Object code | Meaning | Coding |
|--------------|--|--------|
| VAR | A single value, for example of the type Integer8, Unsigned32 or Visible String8. | 7 |
| ARR (ARRAY) | A data field in which every entry is of the same data type. | 8 |
| REC (RECORD) | A data field that contains entries that are a combination of single data types. | 9 |

| Data type | Value range | Data length | DS 301 coding |
|------------------|----------------------------|-------------|---------------|
| Boolean | 0 = false, 1 = true | 1 byte | 0001 |
| Integer8 | -128 .. +127 | 1 byte | 0002 |
| Integer16 | -32768 .. +32767 | 2 byte | 0003 |
| Integer32 | -2147483648 .. +2147483647 | 4 byte | 0004 |
| Unsigned8 | 0 .. 255 | 1 byte | 0005 |
| Unsigned16 | 0 .. 65535 | 2 byte | 0006 |
| Unsigned32 | 0 .. 4294967295 | 4 byte | 0007 |
| Visible String8 | ASCII characters | 8 byte | 0009 |
| Visible String16 | ASCII characters | 16 byte | 0010 |

RO/RW Note on reading and writing the values
 RO: values are read-only
 RW: values are read and write.

PDO R_PDO: Mapping for R_PDO possible
 T_PDO: Mapping for T_PDO possible
 no entry: PDP mapping not possible with the object

Min/max values Show the permissible range in which the object value is defined and valid.

Default values Factory settings.

per. Designation of whether the value of the parameter is persistent, i.e. after switching off the unit it is retained in the memory. When entering via HMI the unit stores the value of the parameter automatically at each change. When changing a value via commissioning software or field bus, the user must explicitly store the value change in the persistent memory.

9.2 Overview object group 1000_h

| Index | Subindex | Name | Obj. code | Data type | Access | PDOs | Description | Page |
|-------------------|-----------------|---|-----------|-----------------|--------|------|---|------|
| 1000 _h | | device type | VAR | Unsigned32 | ro | | Device type and profile | 9-7 |
| 1001 _h | | error register | VAR | Unsigned8 | ro | | error register | 9-7 |
| 1003 _h | | predefined error field | ARR | | rw | | Error history, memory for error messages | 9-8 |
| 1003 _h | 00 _h | number of errors | VAR | Unsigned8 | rw | | Number of error entries | 9-8 |
| 1003 _h | 01 _h | error field | VAR | Unsigned32 | ro | | Error number | 9-8 |
| 1005 _h | | COB-ID SYNC | VAR | Unsigned32 | rw | | Identifier of the synchronisation object | 9-9 |
| 1008 _h | | manufacturer device name | VAR | Visible String8 | ro | | User device name | 9-9 |
| 1009 _h | | manufacturer hardware version | VAR | Visible String8 | ro | | Hardware status | 9-10 |
| 100A _h | | manufacturer software version | VAR | Visible String8 | ro | | Software version | 9-10 |
| 100C _h | | guard time | VAR | Unsigned16 | rw | | Time span for node guarding [ms] | 9-10 |
| 100D _h | | life time factor | VAR | Unsigned8 | rw | | Repeat factor for the node guarding protocol | 9-11 |
| 1010 _h | | save parameters | ARR | Unsigned32 | rw | | Saves parameters: | 9-12 |
| 1010 _h | 01 _h | save all parameters | VAR | Unsigned32 | rw | | Saves all parameters | 9-12 |
| 1010 _h | 02 _h | save communication parameters | VAR | Unsigned32 | rw | | Saves parameters for communications | 9-12 |
| 1011 _h | | restore default of parameters | ARR | Unsigned32 | rw | | Resets parameter values to the default setting | 9-13 |
| 1011 _h | 01 _h | restore default of all parameters | VAR | Unsigned32 | rw | | Resets all parameter values to the default setting | 9-13 |
| 1011 _h | 02 _h | restore default of communication parameters | VAR | Unsigned32 | rw | | Restores parameter settings for communications to default | 9-13 |
| 1014 _h | | COB-ID EMCY | VAR | Unsigned32 | rw | | Unsigned16 | 9-14 |
| 1015 _h | | inhibit time EMCY | VAR | Unsigned16 | rw | | Unsigned16 | 9-14 |
| 1016 _h | | Consumer Heartbeat Time | ARR | Unsigned32 | rw | | Unsigned16 | 9-15 |
| 1016 _h | 01 _h | Consumer Heartbeat Time | VAR | Unsigned32 | rw | | Time interval and Node-Id of the "Heartbeat"Receiver | 9-15 |
| 1017 _h | | Producer Heartbeat Time | VAR | Unsigned16 | rw | | Time interval for producer-"Heartbeat" | 9-15 |
| 1018 _h | | Identity Object | REC | Identity | ro | | Identification object: | 9-16 |
| 1018 _h | 01 _h | Vendor ID | VAR | Unsigned32 | ro | | Vendor ID | 9-16 |
| 1018 _h | 02 _h | Product code | VAR | Unsigned32 | ro | | Product code | 9-16 |
| 1018 _h | 03 _h | Revision number | VAR | Unsigned32 | ro | | Revision number | 9-16 |
| 1018 _h | 04 _h | Serial number | VAR | Unsigned32 | ro | | Serial number | 9-16 |
| 1020 _h | | Verify configuration | ARR | Unsigned32 | rw | | Retains data for configuration | 9-17 |
| 1020 _h | 01 _h | Configuration date | VAR | Unsigned32 | rw | | Date of configuration | 9-17 |
| 1020 _h | 02 _h | Configuration time | VAR | Unsigned32 | rw | | Time of configuration | 9-17 |

| Index | Subindex | Name | Obj. code | Data type | Access | PDO | Description | Page |
|-------------------|-----------------|---------------------------|-----------|-------------------|--------|-----|---|------|
| 1200 _h | | 1st server SDO parameter | REC | SDO server param. | ro | | First server SDO, settings | 9-19 |
| 1200 _h | 01 _h | COB-ID Client -> Server | VAR | Unsigned32 | ro | | Identifier Client -> Server | 9-19 |
| 1200 _h | 02 _h | COB-ID Server -> Client | VAR | Unsigned32 | ro | | Identifier Server -> Client | 9-19 |
| 1201 _h | | 2nd server SDO parameter | REC | SDO server param. | rw | | Second server SDO, settings | 9-20 |
| 1201 _h | 01 _h | COB-ID Client -> Server | VAR | Unsigned32 | rw | | Identifier Client -> Server | 9-20 |
| 1201 _h | 02 _h | COB-ID Server -> Client | VAR | Unsigned32 | rw | | Identifier Server -> Client | 9-20 |
| 1201 _h | 03 _h | Node-ID SDO Client | VAR | Unsigned32 | rw | | Node-ID SDO Client | 9-20 |
| 1400 _h | | 1st receive PDO parameter | REC | PDO comm. param. | rw | | First receive PDO (R_PDO1), settings | 9-21 |
| 1400 _h | 01 _h | COB-ID R_PDO1 | VAR | Unsigned32 | rw | | Identifier of the R_PDO1 | 9-21 |
| 1400 _h | 02 _h | transmission type R_PDO1 | VAR | Unsigned8 | rw | | Transmission type | 9-21 |
| 1401 _h | | 2nd receive PDO parameter | REC | PDO comm. param. | rw | | Second receive PDO (R_PDO2), settings | 9-23 |
| 1401 _h | 01 _h | COB-ID R_PDO2 | VAR | Unsigned32 | rw | | Identifier of the R_PDO2 | 9-23 |
| 1401 _h | 02 _h | transmission type R_PDO2 | VAR | Unsigned8 | rw | | Transmission type | 9-23 |
| 1402 _h | | 3rd receive PDO parameter | REC | PDO comm. param. | rw | | Third receive PDO (R_PDO3), settings | 9-24 |
| 1402 _h | 01 _h | COB-ID R_PDO3 | VAR | Unsigned32 | rw | | Identifier of the R_PDO3 | 9-24 |
| 1402 _h | 02 _h | transmission type R_PDO3 | VAR | Unsigned8 | rw | | Transmission type | 9-24 |
| 1403 _h | | 4th receive PDO parameter | REC | PDO comm. param. | ro | | Fourth receive PDO (R_PDO4), settings | 9-25 |
| 1403 _h | 01 _h | COB-ID R_PDO4 | VAR | Unsigned32 | rw | | Identifier of the R_PDO4 | 9-25 |
| 1403 _h | 02 _h | transmission type R_PDO4 | VAR | Unsigned8 | ro | | Transmission type | 9-25 |
| 1600 _h | | 1st receive PDO mapping | REC | PDO mapping | rw | | PDO mapping for R_PDO1, settings | 9-26 |
| 1600 _h | 01 _h | 1st mapped object R_PDO1 | VAR | Unsigned32 | rw | | First object for the mapping in R_PDO1 | 9-26 |
| 1601 _h | | 2nd receive PDO mapping | REC | PDO mapping | rw | | PDO mapping for R_PDO2, settings | 9-27 |
| 1601 _h | 01 _h | 1st mapped object R_PDO2 | VAR | Unsigned32 | rw | | First object for the mapping in R_PDO2 | 9-27 |
| 1601 _h | 02 _h | 2nd mapped object R_PDO2 | VAR | Unsigned32 | rw | | Second object for the mapping in R_PDO2 | 9-27 |
| 1602 _h | | 3rd receive PDO mapping | REC | PDO mapping | rw | | PDO mapping for R_PDO3, settings | 9-28 |
| 1602 _h | 01 _h | 1st mapped object R_PDO3 | VAR | Unsigned32 | rw | | First object for the mapping in R_PDO3 | 9-28 |

| Index | Subindex | Name | Obj. code | Data type | Access | Description | Page |
|-------------------|-----------------|----------------------------|-----------|------------------|--------|---|------|
| 1602 _h | 02 _h | 2nd mapped object R_PDO3 | VAR | Unsigned32 | rw | Second object for the mapping in R_PDO3 | 9-28 |
| 1603 _h | | 4th receive PDO mapping | REC | PDO mapping | ro | PDO mapping for R_PDO3, settings | 9-30 |
| 1603 _h | 01 _h | 1st mapped object R_PDO4 | VAR | Unsigned32 | ro | First object for the mapping in R_PDO4 | 9-30 |
| 1603 _h | 02 _h | 2nd mapped object R_PDO4 | VAR | Unsigned32 | ro | Second object for the mapping in R_PDO4 | 9-30 |
| 1603 _h | 03 _h | 3rd mapped object R_PDO4 | VAR | Unsigned32 | ro | Third object for the mapping in R_PDO4 | 9-30 |
| 1800 _h | | 1st transmit PDO parameter | REC | PDO comm. param. | rw | First send PDO (T_PDO1), settings | 9-30 |
| 1800 _h | 01 _h | COB-ID T_PDO1 | VAR | Unsigned32 | rw | Identifier of the T_PDO1 | 9-30 |
| 1800 _h | 02 _h | transmission type T_PDO1 | VAR | Unsigned8 | rw | Transmission type | 9-30 |
| 1800 _h | 03 _h | inhibit time T_PDO1 | VAR | Unsigned16 | rw | Blocking period for bus access (1=100µs) | 9-30 |
| 1800 _h | 04 _h | reserved T_PDO1 | VAR | Unsigned8 | rw | Priority for CAN bus arbitration ([0-7]). | 9-30 |
| 1800 _h | 05 _h | event timer T_PDO1 | VAR | Unsigned16 | rw | Period for event triggering (1=1 ms) | 9-30 |
| 1801 _h | | 2nd transmit PDO parameter | REC | PDO comm. param. | rw | Second send PDO (T_PDO2), settings | 9-32 |
| 1801 _h | 01 _h | COB-ID T_PDO2 | VAR | Unsigned32 | rw | Identifier of the T_PDO2 | 9-32 |
| 1801 _h | 02 _h | transmission type T_PDO2 | VAR | Unsigned8 | rw | Transmission type | 9-32 |
| 1801 _h | 03 _h | inhibit time T_PDO2 | VAR | Unsigned16 | rw | Blocking period for bus access (1=100µs) | 9-32 |
| 1801 _h | 04 _h | reserved T_PDO2 | VAR | Unsigned8 | rw | reserved | 9-32 |
| 1801 _h | 05 _h | event timer T_PDO2 | VAR | Unsigned16 | rw | Period for event triggering (1=1 ms) | 9-32 |
| 1802 _h | | 3rd transmit PDO parameter | REC | PDO comm. param. | rw | Third send PDO (T_PDO3), settings | 9-33 |
| 1802 _h | 01 _h | COB-ID T_PDO3 | VAR | Unsigned32 | rw | Identifier of the T_PDO3 | 9-33 |
| 1802 _h | 02 _h | transmission type T_PDO3 | VAR | Unsigned8 | rw | Transmission type | 9-33 |
| 1802 _h | 03 _h | inhibit time T_PDO3 | VAR | Unsigned16 | rw | Blocking period for bus access (1=100µs) | 9-33 |
| 1802 _h | 04 _h | reserved T_PDO3 | VAR | Unsigned8 | rw | reserved | 9-33 |
| 1802 _h | 05 _h | event timer T_PDO3 | VAR | Unsigned16 | rw | Period for event triggering (1=1 ms) | 9-33 |
| 1803 _h | | 4th transmit PDO parameter | REC | PDO comm. param. | ro | Fourth send PDO (T_PDO4), settings | 9-35 |
| 1803 _h | 01 _h | COB-ID T_PDO4 | VAR | Unsigned32 | rw | Identifier of the T_PDO4 | 9-35 |
| 1803 _h | 02 _h | transmission type T_PDO4 | VAR | Unsigned8 | ro | Transmission type | 9-35 |
| 1803 _h | 03 _h | inhibit time T_PDO4 | VAR | Unsigned16 | ro | Blocking period for bus access (1=100µs) | 9-35 |
| 1803 _h | 04 _h | reserved T_PDO4 | VAR | Unsigned8 | ro | reserved | 9-35 |
| 1803 _h | 05 _h | event timer T_PDO4 | VAR | Unsigned16 | rw | Period for event triggering (1=1 ms) | 9-35 |

| Index | Subindex | Name | Obj. code | Data type | Access | PDO | Description | Page |
|-------------------|-----------------|--------------------------|-----------|-------------|--------|-----|---|------|
| 1A00h | | 1st transmit PDO mapping | REC | PDO mapping | rw | | PDO mapping for T_PDO1, settings | 9-36 |
| 1A00h | 01 _h | 1st mapped object T_PDO1 | VAR | Unsigned32 | rw | | First object for the mapping in T_PDO1 | 9-36 |
| 1A01 _h | | 2nd transmit PDO mapping | REC | PDO mapping | rw | | PDO mapping for T_PDO2, settings | 9-37 |
| 1A01 _h | 01 _h | 1st mapped object T_PDO2 | VAR | Unsigned32 | rw | | First object for the mapping in T_PDO2 | 9-37 |
| 1A01 _h | 02 _h | 2nd mapped object T_PDO2 | VAR | Unsigned32 | rw | | Second object for the mapping in T_PDO2 | 9-37 |
| 1A02 _h | | 3rd transmit PDO mapping | REC | PDO mapping | rw | | PDO mapping for T_PDO3, settings | 9-38 |
| 1A02 _h | 01 _h | 1st mapped object T_PDO3 | VAR | Unsigned32 | rw | | First object for the mapping in T_PDO3 | 9-38 |
| 1A02 _h | 02 _h | 2nd mapped object T_PDO3 | VAR | Unsigned32 | rw | | Second object for the mapping in T_PDO3 | 9-38 |
| 1A03 _h | | 4th transmit PDO mapping | REC | PDO mapping | rw | | PDO mapping for T_PDO4, settings | 9-40 |
| 1A03 _h | 01 _h | 1st mapped object T_PDO4 | VAR | Unsigned32 | ro | | First object for the mapping in T_PDO4 | 9-40 |
| 1A03 _h | 02 _h | 2nd mapped object T_PDO4 | VAR | Unsigned32 | ro | | Second object for the mapping in T_PDO4 | 9-40 |
| 1A03 _h | 03 _h | 3rd mapped object T_PDO4 | VAR | Unsigned32 | ro | | Third object for the mapping in T_PDO4 | 9-40 |

9.3 Assignment object group 6000_h



The product includes parameters for the CANopen object groups 3000_h and 6000_h. The names of the parameters and the data type of the parameters for the object group 6000_h may be different from the DSP 402 definition. For a detailed description of all parameters see the operator's manual for the product in the chapter on parameters.

| Index | DSP 402 object name | DSP 402 data type | internal object name |
|---------------------|---------------------------------------|-------------------|----------------------|
| 603F:0 _h | Error code | UINT16 | _StopFault |
| 6040:0 _h | Control word | UINT16 | DCOMcontrol |
| 6041:0 _h | Status word | UINT16 | DCOMstatus |
| 6060:0 _h | Modes of operation | INT8 | DCOMopmode |
| 6061:0 _h | Modes of operation display | INT8 | _DCOMopmd_act |
| 6064:0 _h | Position actual value | INT32 | _p_actusr |
| 6065:0 _h | Following error window | UINT32 | SPV_p_maxDiff |
| 6067:0 _h | Position window | UINT32 | STANDp_win |
| 6068:0 _h | Position window time | UINT16 | STANDpwinTime |
| 606B:0 _h | Velocity demand value | INT32 | _n_actRAMP |
| 606C:0 _h | Velocity actual value | INT32 | _n_act |
| 607A:0 _h | Target position | INT32 | PPp_targetusr |
| 607D:1 _h | Min position limit | INT32 | SPVswLimNusr |
| 607D:2 _h | Max position limit | INT32 | SPVswLimPusr |
| 607F:0 _h | Max profile velocity | UINT32 | RAMPn_max |
| 6081:0 _h | Profile velocity | UINT32 | PPn_target |
| 6083:0 _h | Profile acceleration | UINT32 | RAMPacc |
| 6084:0 _h | Profile deceleration | UINT32 | RAMPdecel |
| 6086:0 _h | Motion profile type | INT16 | ProfileType |
| 6098:0 _h | Homing method | INT8 | HMmethod |
| 6099:1 _h | Homing speed during search for switch | UINT32 | HMn |
| 6099:2 _h | Homing speed during search for zero | UINT32 | HMn_out |
| 60F4:0 _h | Following error actual value | INT32 | _p_dif |
| 60FF:0 _h | Target velocity | INT32 | PVn_target |
| 6502:0 _h | Supported drive modes | UINT32 | SuppDriveModes |

9.4 Details object group 1000_h

9.4.1 1000_h Device type

The object shows the implemented device profile and the device type.

Object description

| | |
|-------------|-------------------|
| Index | 1000 _h |
| Object name | device type |
| Object code | VAR |
| Data type | Unsigned32 |

Values description

| | |
|---------------|-------------------------------|
| Subindex | 00 _h , device type |
| Meaning | Device type and profile |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 0042 0192 _h |
| can be saved | – |

Bit coding, subindex 00_h

| Bit | Access | Value | Meaning |
|-------|--------|-------------------|---|
| 31-24 | ro | 00 _h | not used |
| 23-16 | ro | 02 _h | Bit17=1: servo drive |
| 15-0 | ro | 0192 _h | Device profile DS-402 (192 _h) |

9.4.2 1001_h Error register

The object shows the error status of the unit. The detailed cause of error can be found with the object `predefined error field` (1003_h) and - for reasons of compatibility to units with different field bus profiles - the object `error code` (603F_h).

Errors are signalled by an EMCY message as soon as they occur.

Object description

| | |
|-------------|-------------------|
| Index | 1001 _h |
| Object name | error register |
| Object code | VAR |
| Data type | Unsigned8 |

Values description

| | |
|---------------|----------------------------------|
| Subindex | 00 _h , error register |
| Meaning | error register |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | – |
| can be saved | – |

Bit coding, subindex 00_h

| Bit | Access | Value | Meaning |
|-----|--------|-------|---|
| 0 | ro | – | Error! (generic error) |
| 1 | ro | – | reserved |
| 2 | ro | – | reserved |
| 3 | ro | – | reserved |
| 4 | ro | – | Communication profile (communication error) |
| 5 | ro | – | Device profile (device profile error) |
| 6 | ro | – | reserved |
| 7 | ro | – | manufacturer specific |

9.4.3 1003_h Predefined error field

The object saves the latest error messages that were shown as EMCY messages.

- The entry under subindex 00_h contains the number of saved error messages.
- The current error message is stored under subindex 01_h, older messages are moved to high subindex entries.
- Write a '0 to subindex 00_h resets the error list.

Object description

| | |
|-------------|------------------------|
| Index | 1003 _h |
| Object name | predefined error field |
| Object code | ARRAY |
| Data type | Unsigned32 |

Values description

| | |
|---------------|------------------------------------|
| Subindex | 00 _h , number of errors |
| Meaning | Number of error entries |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...1 |
| Default value | 1 |
| can be saved | – |
| Subindex | 01 _h , error field |
| Meaning | Error number |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |

Bit-coding, Subindex 00_h..05_h

Bytes 0..15: error code.byte 16..31 additional error information, not assigned in the unit.

9.4.4 1005_h COB-Id SYNC message

The object shows the COB-Id of the SYNC object and specifies whether a unit sends or receives SYNC messages.

The unit can only receive SYNC messages.

For synchronisation a unit in the network must send SYNC objects.

The COB-Id can be changed in the NMT "Pre-Operational" status.

Object description

| | |
|-------------|-------------------|
| Index | 1005 _h |
| Object name | COB-ID SYNC |
| Object code | VAR |
| Data type | Unsigned32 |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , COB-ID SYNC |
| Meaning | Identifier of the synchronisation object |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 8000 0080 _h |
| can be saved | – |

Bit coding, subindex 00_h

| Bit | Access | Value | Meaning |
|-------|--------|-------------------|---|
| 31 | ro | 0 _b | 1: Unit can receive SYNC messages (SYNC consumer) |
| 30 | ro | 1 _b | 1: Unit can send SYNC messages (SYNC producer) |
| 29 | ro | 0 _b | 0: 11-bit identifier (CAN 3.0A) 1: 29-bit identifier (CAN 3.0B) |
| 28-11 | ro | 0000 _h | Only relevant if bit 29=1 is not used by the unit. |
| 10-7 | rw | 0001 _b | Function code, bit 10..7 of the COB-ID |
| 6-0 | ro | 7F _h | Node address, bit 6..0 of the COB-ID |

9.4.5 1008_h Manufacturer device name

The object shows the device name of the manufacturer.

Object description

| | |
|-------------|--------------------------|
| Index | 1008 _h |
| Object name | manufacturer device name |
| Object code | VAR |
| Data type | Visible String8 |

Values description

| | |
|-------------|--|
| Subindex | 00 _h , manufacturer device name |
| Meaning | User device name |
| Access | read-only |
| PDO mapping | – |

| | |
|---------------|---|
| Value range | – |
| Default value | – |
| can be saved | – |

The following objects contain additional information on the device:
 - objects 6404_h, 6410_h: Motor data

9.4.6 1009_h Manufacturer hardware version

The object shows the version of the device hardware.

Object description

| | |
|-------------|-------------------------------|
| Index | 1009 _h |
| Object name | manufacturer hardware version |
| Object code | VAR |
| Data type | Visible String8 |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , manufacturer hardware version |
| Meaning | Hardware status |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | – |
| can be saved | – |

9.4.7 100A_h Manufacturer software version

The object shows the version of the device software.

Object description

| | |
|-------------|-------------------------------|
| Index | 100A _h |
| Object name | manufacturer software version |
| Object code | VAR |
| Data type | Visible String8 |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , manufacturer software version |
| Meaning | Software version |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | – |
| can be saved | – |

9.4.8 100C_h Guard time

The object shows the time span for node guarding of an NMT slave.

The time span for the connection monitoring of a NMT master is derived from the time span "guard time" multiplied by the factor "life time", object `Life time factor(100Dh)`.

The time span can be changed in the NMT "Pre-Operational" status.

Object description

| | |
|-------------|------------|
| Index | 100Ch |
| Object name | guard time |
| Object code | VAR |
| Data type | Unsigned16 |

Values description

| | |
|---------------|----------------------------------|
| Subindex | 00 _h , guard time |
| Meaning | Time span for node guarding [ms] |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

9.4.9 100D_h Life time factor

The object shows the factor that together with the time span "guard time" gives the time interval for the connection monitoring of a NMT master. Within this period the NMT slave device waits for a monitoring request by node guarding by the NMT master device.

$\text{life time} = \text{guard time} * \text{life time factor}$

The value "0" disables the monitoring of the NMT master.

If the connection monitoring by the NMT master remains disabled during the time interval "life time", the unit reports an error and switches to error status.

The time factor can be changed in the NMT "Pre-Operational" status.

The time span "guard time" is set with the object `Guard time (100Ch)`.

Object description

| | |
|-------------|------------------|
| Index | 100Dh |
| Object name | life time factor |
| Object code | VAR |
| Data type | Unsigned8 |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , life time factor |
| Meaning | Repeat factor for the node guarding protocol |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 0 |
| can be saved | – |

9.4.10 1010_h Save Parameters

The object is used to save parameters.

- Subindex 01_h, all parameters
- Subindex 02_h, parameter for communications
- Subindex 03_h, parameter for application

Object description

| | |
|-------------|-------------------|
| Index | 1010 _h |
| Object name | save parameters |
| Object code | ARRAY |
| Data type | Unsigned32 |

Values description

| | |
|---------------|--------------------------------------|
| Subindex | 00 _h , number of elements |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 3 |
| can be saved | – |

| | |
|---------------|---------------------------------------|
| Subindex | 01 _h , save all parameters |
| Meaning | saves all parameters |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 02 _h , save communication parameters |
| Meaning | saves parameters for communications |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 03 _h , save application parameters |
| Meaning | saves parameters for application |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

9.4.11 1011_h Restore Default Parameters

The object is used to restore the default parameters.

- Subindex 01_h, all parameters
- Subindex 02_h, parameters for communications
- Subindex 03_h, parameter for application

Object description

| | |
|-------------|----------------------------|
| Index | 1011 _h |
| Object name | restore default parameters |
| Object code | ARRAY |
| Data type | Unsigned32 |

Values description

| | |
|---------------|--------------------------------------|
| Subindex | 00 _h , number of elements |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 3 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 01 _h , restore default of all parameters |
| Meaning | Resets all parameter values to the default setting |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 02 _h , restore default of communication parameters |
| Meaning | Restores parameter settings for communications to default |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 03 _h , restore default of application parameters |
| Meaning | Restores parameter settings for the application to default |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

9.4.12 1014_h COB-ID emergency message

The object shows the COB-Id of the emergency object "EMCY".

Object description

| | |
|-------------|-------------------|
| Index | 1014 _h |
| Object name | COB-ID EMCY |
| Object code | VAR |
| Data type | Unsigned32 |

Values description

| | |
|---------------|------------------------------------|
| Subindex | 00 _h , COB-ID EMCY |
| Meaning | Identifier of the emergency object |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 4000 0080 _h + Node-ID |
| can be saved | – |

Bit coding, subindex 00_h

| Bit | Access | Value | Meaning |
|--------|--------|-------------------|---|
| 31, 30 | ro | 0 _b | reserved |
| 29 | ro | 0 _b | 0: 11-bit identifier (CAN 3.0A) 1: 29-bit identifier (CAN 3.0B) |
| 28-11 | ro | 0000 _h | Only relevant if bit 29=1 is not used by the unit. |
| 10-7 | rw | 0001 _b | Function code, bit 10-7 of the COB-ID |
| 6-0 | ro | – | Node address, bit 6-0 of the COB-ID |

The COB-Id can be changed in the NMT "Pre-Operational" status.

9.4.13 1015_h Inhibit time emergency message

The object specifies the waiting period for the repeated transmission of EMCY messages as a multiple of 100µs.

Object description

| | |
|-------------|-------------------|
| Index | 1015 _h |
| Object name | inhibit time EMCY |
| Object code | VAR |
| Data type | Unsigned16 |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , inhibit time EMCY |
| Meaning | Waiting time for repeated transmission of an EMCY |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

9.4.14 1016_h Consumer Heartbeat Time

The object saves the settings of the "heartbeat" consumer for NMT monitoring by "heartbeat" connection message.

Object description

| | |
|-------------|-------------------------|
| Index | 1016 _h |
| Object name | Consumer Heartbeat Time |
| Object code | ARRAY |
| Data type | Unsigned32 |

Values description

| | |
|---------------|--------------------------------------|
| Subindex | 00 _h , number of elements |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 3 |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 01 _h , Consumer Heartbeat Time |
| Meaning | Time interval and Node-Id of the "Heartbeat"Receiver |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 0 |
| can be saved | – |

Bit-coding subindex 01_h..03_h

| Bit | Meaning |
|--------|---------------------------------------|
| 31..24 | reserved |
| 23..16 | Node-ID |
| 15..0 | Time interval for "heartbeat" message |

The time interval is given as a multiple of 1 ms and must be greater than the producer "heartbeat" time, object `Producer Heartbeat Time` (1017_h). If the time interval is zero, the unit specified by Node-ID is not monitored.

9.4.15 1017_h Producer Heartbeat Time

The object saves the time interval of the "heartbeat" producer for NMT monitoring by "heartbeat" connection message as a multiple of 1 ms.

The producer "heartbeat" time must be less than the time interval of the "heartbeat" consumer, object `Consumer Heartbeat Time` (1016_h). A time interval of zero disables the monitoring.

Object description

| | |
|-------------|-------------------------|
| Index | 1017 _h |
| Object name | Producer Heartbeat Time |

| | | |
|---------------------------|---------------|---|
| <i>Values description</i> | Object code | VAR |
| | Data type | Unsigned16 |
| | Subindex | 00 _h , Producer Heartbeat Time |
| | Meaning | Time interval for producer-"Heartbeat" |
| | Access | read-write |
| | PDO mapping | – |
| | Value range | 0...65535 |
| | Default value | 0 |
| | can be saved | – |
| | | |

9.4.16 1018_h Identity Object

The object shows information on the device.

- Subindex 01_h (vendor Id) contains the identification identifier of the manufacturer,
- Subindex 02_h (product Id) shows the manufacturer-specific product code and
- Subindex 03_h (revision number) identifies special CANopen properties for the unit.
- Subindex 04_h (serial number) contains the unit serial number.

| | | |
|---------------------------|---------------|--------------------------------------|
| <i>Object description</i> | Index | 1018 _h |
| | Object name | Identity Object |
| | Object code | RECORD |
| | Data type | Identity |
| <i>Values description</i> | Subindex | 00 _h , number of elements |
| | Meaning | Number of values for the object |
| | Access | read-only |
| | PDO mapping | – |
| | Value range | – |
| | Default value | 4 |
| | can be saved | – |
| | Subindex | 01 _h , Vendor ID |
| | Meaning | Vendor ID |
| | Access | read-only |
| | PDO mapping | – |
| | Value range | – |
| | Default value | 0100 002E _h |
| | can be saved | – |
| | Subindex | 02 _h , Product code |

| | |
|---------------|-----------------------------------|
| Meaning | Product code |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |
| Subindex | 03 _h , Revision number |
| Meaning | Revision number |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |
| Subindex | 04 _h , Serial number |
| Meaning | Serial number |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |

9.4.17 1020_h data for configuration

The object is used to verify the configuration.

- Subindex 01_h, date of configuration
- Subindex 02_h, time of configuration

Object description

| | |
|-------------|-------------------|
| Index | 1020 _h |
| Object name | |
| Object code | RECORD |
| Data type | Identity |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , verify configuration |
| Meaning | Retains data for configuration |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 2 |
| can be saved | – |
| Subindex | 01 _h , configuration date |

| | |
|---------------|--------------------------------------|
| Meaning | Date of configuration |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | |
| can be saved | – |
| Subindex | 02 _h , configuration time |
| Meaning | Time of configuration |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | |
| can be saved | – |

9.4.18 1029_h Error Behaviour

The object shows the reaction of the NMT status machine, if an communication error occur.

Object description

| | |
|-------------|-------------------|
| Index | 1029 _h |
| Object name | Error Behaviour |
| Object code | ARRAY |
| Data type | Unsigned8 |

Values description

| | |
|---------------|-------------------------------------|
| Subindex | 00 _h , number of entries |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|---------------------------------------|
| Subindex | 01 _h , Communication Error |
| Meaning | Communication Error |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 0 |
| can be saved | – |

Settings, subindex 01_h

| Value | Description |
|-------|--|
| 0 | pre-operational (only if current state is operational) |
| 1 | no state change |
| 2 | stopped |

9.4.19 1200_h 1st server SDO parameter

The object saves the settings for the first server SDO.

Object description

| | |
|-------------|--------------------------|
| Index | 1200 _h |
| Object name | 1st server SDO parameter |
| Object code | RECORD |
| Data type | SDO server parameter |

Values description

| | |
|-------------|-------------------------------------|
| Subindex | 00 _h , number of entries |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |

| | |
|---------------|---|
| Value range | – |
| Default value | 2 |
| can be saved | – |
| Subindex | 01 _h , COB-ID Client -> Server |
| Meaning | Identifier Client -> Server |
| Access | read-only |
| PDO mapping | – |
| Value range | 0...4294967296 |
| Default value | 1536 + Node-ID |
| can be saved | – |
| Subindex | 02 _h , COB-ID Server -> Client |
| Meaning | Identifier Server -> Client |
| Access | read-only |
| PDO mapping | – |
| Value range | 0...4294967296 |
| Default value | 1408 + Node-ID |
| can be saved | – |

9.4.20 1201_h 2nd server SDO parameter

The object saves the settings for the second server SDO.

Object description

| | |
|-------------|--------------------------|
| Index | 1201 _h |
| Object name | 2nd server SDO parameter |
| Object code | RECORD |
| Data type | SDO server parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , number of entries |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 3 |
| can be saved | – |
| Subindex | 01 _h , COB-ID Client -> Server |
| Meaning | Identifier Client -> Server |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967296 |
| Default value | – |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 02 _h , COB-ID Server -> Client |
| Meaning | Identifier Server -> Client |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967296 |
| Default value | – |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 03 _h , Node-ID SDO Client |
| Meaning | Node-ID SDO Client |
| Access | read-write |
| PDO mapping | – |
| Value range | 1...127 |
| Default value | – |
| can be saved | – |

9.4.21 1400_h 1st receive PDO parameter

The object saves the settings for the first receive PDO R_PDO1.

Object description

| | |
|-------------|-----------------------------|
| Index | 1400 _h |
| Object name | 1st receive PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|-------------------------------------|
| Subindex | 00 _h , number of entries |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 2 |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the R_PDO1 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967296 |
| Default value | 512 + Node-ID |
| can be saved | – |

| | |
|----------|--|
| Subindex | 02 _h , transmission type = asynchronous |
| Meaning | Transmission type |
| Access | read-write |

| | |
|---------------|---------|
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 255 |
| can be saved | – |

Bit assignment subindex 01_h

| Bit | Access | Value | Meaning |
|-------|--------|-------------------|---|
| 31 | rw | 0 _b | 0: PDO is active 1: PDO is inactive |
| 30 | ro | 0 _b | 0: RTR (see below) is possible 1: RTR not allowed |
| 29 | ro | 0 _b | 0: 11-bit identifier (CAN 3.0A) 1: 29-bit identifier (CAN 3.0B) |
| 28-11 | ro | 0000 _h | Only relevant if bit 29=1 is not used by the unit. |
| 10-7 | rw | 0100 _b | Function code, bit 10-7 of the COB-ID |
| 6-0 | ro | – | Node address, bit 6-0 of the COB-ID |

Bit 31 A R_PDO can only be used if bit 31="0".

Bit 30: RTR bit Supports a device R_PDOs with RTR (remote transmission request), it can therefore request a PDO corresponding to the producer-consumer relationship from a PDO producer with RTR = "0".

The unit cannot request PDOs, but it can respond to the request for a PDO, see RTR bit for T_PDO1 settings (1800_h).

Bit coding, subindex 02_h The controller for evaluating R_PDO data is specified via subindex 02h. The values 241..251 are reserved.

| Transmission type | cyclic | acyclic | synchronous | asynchronous | RTR-controlled |
|-------------------|--------|---------|-------------|--------------|----------------|
| 0 | – | X | X | – | – |
| 1-240 | X | – | X | – | – |
| 252 | – | – | X | – | X |
| 253 | – | – | – | X | X |
| 254 | – | – | – | X | – |
| 255 | – | – | – | X | – |

If an R_PDO is transmitted synchronously (transmission type=0..252), the device evaluates the received data in accordance with the SYNC object.

- With acyclic transmission (transmission type=0) the evaluation is linked to the SYNC object, but not the transmission of the PDO. A received PDO message is evaluated with the following SYNC.

A value between 1 and 240 shows the number of SYNC cycles after which a received PDO is evaluated.

The values 252 to 254 are relevant for updating T_PDOs but not for sending them.

- 252: Updating send data with receipt of the next SYNC
- 253 updating send data with receipt of a request from a PDO consumer

- 254: Data update event-controlled, the triggering event is manufacturer-specific specified

R_PDOs with the value 255 are updated immediately with receipt of the PDOs. The triggering event is the data that are sent corresponding to the definition of the DS 402 device profile in the PDO.

Settings R_PDO1 is processed asynchronously and event-controlled.

The byte allocate of the R_PDO1 is specified via PDO mapping with the object `1st receive PDO mapping (1600h)`. The following allocate is the default for R_PDO1:

- Bytes 0..1: control word `controlword (6040h)`.

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

9.4.22 1401_h 2nd receive PDO_parameter

The object saves settings for the second receive PDO R_PDO2.

Object description

| | |
|-------------|-----------------------------|
| Index | 1401 _h |
| Object name | 2nd receive PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 2 |
| can be saved | – |

| | |
|---------------|---------------------------------|
| Subindex | 01 _h , COB-ID R_PDO2 |
| Meaning | Identifier of the R_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 768 + Node-ID |
| can be saved | – |

| | |
|---------------|-------------------------------------|
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 255 |

can be saved –

The meaning of the bit states and subindex values is described with the object 1st receive PDO parameters (1400_h).

Settings

R_PDO2 is processed synchronously, acyclically and event-controlled and must be enabled with bit 31=1 in subindex 01_h.

The byte allocate of R_PDO2 is specified via PDO mapping with the object 2nd Receive PDO mapping (1601_h). The following allocate is preset for the point-to-point mode in the "profile position mode":

- Bytes 0..1: control word `controlword` (6040_h)
- Bytes 2..5: target position of the movement command `target position` (607A_h)

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

The transmission type for the receive PDO can have three value ranges:

| | |
|----------|---|
| 0 | for an asynchronous cycle |
| 1 to 240 | assigns the receive PDO, only becomes active if a SYNC object is received |
| 255 | shows that the PDO is executed on receipt |

9.4.23 1402_h 3rd receive PDO parameter

The object saves settings for the third receive PDO R_PDO3.

Object description

| | |
|-------------|-----------------------------|
| Index | 1402 _h |
| Object name | 3rd receive PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 2 |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the R_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 1024 + Node-ID |
| can be saved | – |

| | |
|---------------|-------------------------------------|
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 255 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object 1st receive PDO-parameters (1400_h).

Settings

R_PDO3 is processed synchronously, acyclically and event-controlled and must be enabled with bit 31=1 in subindex 01_h.

The byte allocation of the R_PDO3 is specified via PDO mapping with the object 3rd Receive PDO mapping (1602_h). The following allocate is preset for speed mode in the "profile velocity mode":

- Bytes 0..1: control word `controlword` (6040_h)
- Bytes 2..5: setpoint speed of the movement command `Target velocity` (60FF_h)

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

The transmission type for the receive PDO can have three value ranges:

| | |
|----------|---|
| 0 | for an asynchronous cycle |
| 1 to 240 | assigns the receive PDO, only becomes active if a SYNC object is received |
| 255 | shows that the PDO is executed on receipt |

9.4.24 1403_h 4th receive PDO parameter

The object saves settings for the fourth receive PDO R_PDO4.

Object description

| | |
|-------------|-----------------------------|
| Index | 1403 _h |
| Object name | 4th receive PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 2 |
| can be saved | – |

| | |
|----------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
|----------|--------------------------------------|

| | |
|---------------|--------------------------|
| Meaning | Identifier of the R_PDO4 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 1280 + Node-ID |
| can be saved | – |

| | |
|---------------|-------------------------------------|
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 254 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object 1st receive PDO-parameters (1400_h).

PDO settings

R_PDO4 is processed synchronously and event-controlled and must be enabled with bit 31=1 in subindex 01_h.

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

9.4.25 1600_h 1st receive PDO mapping

The object shows which objects are mapped in R_PDO1 and transmitted with the PDO. When reading the object subindex 00_h the number of mapped objects is given.

Object description

| | |
|-------------|-------------------------|
| Index | 1600 _h |
| Object name | 1st receive PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , number of mapped objects |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 1 |
| can be saved | – |
| Subindex | 01 _h , CMD: Control word |
| Meaning | First object for the mapping in R_PDO1 |
| Access | read-only |
| PDO mapping | – |

| | |
|---------------|------------------------|
| Value range | 0...4294967295 |
| Default value | 6040 0010 _h |
| can be saved | – |

Bit coding from subindex 01_h

Every subindex entry from subindex 01_h gives the object and the byte length of the object. The object is identified via index and subindex, which refer to the object directory of the unit.

| Bit | Meaning |
|--------|------------------------|
| 31..16 | Index |
| 15..8 | Subindex |
| 7..0 | Object length in bytes |

Settings The PDO assignment for R_PDO1 can be modified. The following assignment is the default:

- Subindex 01_h: PDO mapping of the control word, object controlword (6040_h).

9.4.26 1601_h 2nd receive PDO mapping

The object shows which objects are mapped in R_PDO2 and transmitted with the PDO. When reading the object subindex 00h the number of mapped objects is given.

Object description

| | |
|-------------|-------------------------|
| Index | 1601 _h |
| Object name | 2nd receive PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , number of mapped application objects in PDO |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 2 |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 01 _h , PDO mapping for the first application object to be mapped (control word) |
| Meaning | First object for the mapping in R_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 6040 0010 _h |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 02 _h , PDO mapping for the second application object to be mapped (target position) |
| Meaning | Second object for the mapping in R_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 607A 0010 _h |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 03 _h , PDO mapping for the third application object to be mapped |
| Meaning | Third object for the mapping in R_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 04 _h , PDO mapping for the forth application object to be mapped |
| Meaning | Fourth object for the mapping in R_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

The meaning of the bit states is described with the object 1st receive PDO-mapping (1600_h).

Settings

The PDO assignment for R_PDO2 cannot be modified. The following allocate is preset for the point-to-point mode in the "profile position mode":

- Subindex 01_h: PDO mapping of the control word, object controlword (6040_h).
- Subindex 02_h: target position of the movement command, object target position (607A_h).

9.4.27 1602_h 3rd receive PDO mapping

The object shows which objects are mapped in R_PDO3 and transmitted with the PDO. When reading the object subindex 00_h the number of mapped objects is given.

Object description

| | |
|-------------|-------------------------|
| Index | 1602 _h |
| Object name | 3rd receive PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , number of mapped application objects in PDO |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 2 |
| can be saved | – |
| Subindex | 01 _h , PDO mapping for the first application object to be mapped (control word) |
| Meaning | First object for the mapping in R_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 6040 0010 _h |
| can be saved | – |
| Subindex | 02 _h , PDO mapping for the second application object to be mapped (target velocity) |
| Meaning | Second object for the mapping in R_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 60FF 0020 _h |
| can be saved | – |
| Subindex | 03 _h , PDO mapping for the third application object to be mapped |
| Meaning | Third object for the mapping in R_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |
| Subindex | 04 _h , PDO mapping for the forth application object to be mapped |
| Meaning | Fourth object for the mapping in R_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

The meaning of the bit states is described with the object `1st receive PDO-mapping (1600h)`.

Settings The PDO assignment for R_PDO3 cannot be modified. The following allocate is preset for speed mode in the "profile velocity mode":

- Subindex 01_h: PDO mapping of the control word, object `controlword (6040h)`.
- Bytes 2..5: setpoint speed of the movement command `Target velocity (60FFh)`

9.4.28 1603_h 4th receive PDO mapping

The object shows which objects are mapped in R_PDO4 and transmitted with the PDO. When reading the object subindex 00h the number of mapped objects is given.

Object description

| | |
|-------------|-------------------------|
| Index | 1603 _h |
| Object name | 4th receive PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|---------------------------------|
| Subindex | 00h, number of elements |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |

The meaning of the bit states is described at the object `1st receive PDO mapping (1600h)`.

Settings The PDO assignment for R_PDO4 can be modified.

9.4.29 1800_h 1st transmit PDO parameter

The object saves settings for the first send PDO T_PDO1.

Object description

| | |
|-------------|-----------------------------|
| Index | 1800 _h |
| Object name | 1st transmit PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|-------------------------------------|
| Subindex | 00 _h , number of entries |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 5 |

| | |
|---------------|--|
| can be saved | – |
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the T_PDO1 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 384 + Node-ID |
| can be saved | – |
| Subindex | 02 _h , transmission type = asynchronous |
| Meaning | Transmission type |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 255 |
| can be saved | – |
| Subindex | 03 _h , inhibit time |
| Meaning | Blocking period for bus access (1=100µs) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |
| Subindex | 05 _h , event timer |
| Meaning | Period for event triggering (1=1 ms) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object `1st receive PDO-parameters` (1400_h).

Settings

T_PDO1 is sent asynchronously and event-controlled at every change of the PDO data.

The byte allocate of the T_PDO1 is specified via PDO mapping with the object `1st transmit PDO mapping` (1A00_h). The following assignment is the default:

- Bytes 0..1: status word `statusword` (6041_h).

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

9.4.30 1801_h 2nd transmit PDO parameter

The object saves settings for the second send PDO T_PDO2.

Object description

| | |
|-------------|-----------------------------|
| Index | 1801 _h |
| Object name | 2nd transmit PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 5 |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the T_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | C000 0280 _h + Node-ID |
| can be saved | – |

| | |
|---------------|-------------------------------------|
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 1 |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 03 _h , inhibit time |
| Meaning | Blocking period for bus access (1=100µs) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

| | |
|----------|----------------------------|
| Subindex | 04 _h , reserved |
| Meaning | reserved |
| Access | – |

| | |
|---------------|---------|
| PDO mapping | – |
| Value range | 0...255 |
| Default value | – |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 05 _h , event timer |
| Meaning | Period for event triggering (1=1 ms) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 100 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object 1st receive PDO-parameters (1400_h).

Settings T_PDO2 is sent synchronously and acyclically.

The byte allocate of the T_PDO2 is specified via PDO mapping with the object 2nd transmit PDO mapping (1A01_h). The following allocate is preset for the point-to-point mode in the "profile position mode":

- Bytes 0..1: Status word `statusword` (6041_h)
- Bytes 2..5: current position `position actual value` (6064_h).

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

9.4.31 1802_h 3rd transmit PDO parameter

The object saves settings for the third send PDO T_PDO3.

Object description

| | |
|-------------|-----------------------------|
| Index | 1802 _h |
| Object name | 3rd transmit PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 5 |
| can be saved | – |

| | |
|-------------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the T_PDO3 |
| Access | read-write |
| PDO mapping | – |

| | |
|---------------|--|
| Value range | 0...4294967295 |
| Default value | C000 0380 _h + Node-ID |
| can be saved | – |
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 1 |
| can be saved | – |
| Subindex | 03 _h , inhibit time |
| Meaning | Blocking period for bus access (1=100µs) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |
| Subindex | 04 _h , reserved |
| Meaning | reserved |
| Access | – |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | – |
| can be saved | – |
| Subindex | 05 _h , event timer |
| Meaning | Period for event triggering (1=1 ms) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 100 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object 1st receive PDO-parameters (1400_h).

Settings

T_PDO3 is sent synchronously and acyclically.

The byte allocate of the T_PDO3 is specified via PDO mapping with the object 3rd transmit PDO mapping (1A02_h). The following allocate is preset for speed mode in the "profile velocity mode":

- Bytes 0..1: Status word `statusword` (6041_h)
- Bytes 2..5: current speed `velocity actual value` (606C_h).

The COB-ID of the object can be changed in the NMT "Pre-Operational" status.

9.4.32 1803_h 4th transmit PDO parameter

The object saves settings for the fourth send PDO T_PDO4.

Object description

| | |
|-------------|-----------------------------|
| Index | 1803 _h |
| Object name | 4th transmit PDO parameter |
| Object code | RECORD |
| Data type | PDO Communication Parameter |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , Largest sub-index supported |
| Meaning | Maximum supported subindex |
| Access | read-only |
| PDO mapping | – |
| Value range | – |
| Default value | 5 |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 01 _h , COB-ID used by PDO |
| Meaning | Identifier of the T_PDO4 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | C000 0480 _h + Node-ID |
| can be saved | – |

| | |
|---------------|-------------------------------------|
| Subindex | 02 _h , transmission type |
| Meaning | Transmission type |
| Access | read-only |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | 255 |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 03 _h , inhibit time |
| Meaning | Blocking period for bus access (1=100µs) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

| | |
|----------|----------------------------|
| Subindex | 04 _h , reserved |
|----------|----------------------------|

| | |
|---------------|----------|
| Meaning | reserved |
| Access | – |
| PDO mapping | – |
| Value range | 0...255 |
| Default value | – |
| can be saved | – |

| | |
|---------------|--------------------------------------|
| Subindex | 05 _h , event timer |
| Meaning | Period for event triggering (1=1 ms) |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...65535 |
| Default value | 0 |
| can be saved | – |

The meaning of the bit states and subindex values is described with the object `1st receive PDO-parameters` (1400_h).

Settings R_PDO4 is sent asynchronously and event-driven.

The COB-Id of the object can be changed in the NMT "Pre-Operational" status.

9.4.33 1A00_h 1st transmit PDO mapping

The object shows which objects are mapped in T_PDO1 and transmitted with the PDO. When reading the object subindex 00_h the number of mapped objects is given.

Object description

| | |
|-------------|--------------------------|
| Index | 1A00 _h |
| Object name | 1st transmit PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , number of mapped objects |
| Meaning | Number of values for the object |
| Access | read-only |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 2 |
| can be saved | – |
| Subindex | 01 _h , ETA: status word |
| Meaning | First object for the mapping in T_PDO1 |
| Access | read-only |
| PDO mapping | – |
| Value range | 0...4294967295 |

| | |
|---------------|------------------------|
| Default value | 6041 0010 _h |
| can be saved | – |

The meaning of the bit states is described with the object `1st receive PDO mapping` (1600_h).

Settings The PDO assignment for T_PDO1 cannot be modified. The following assignment is the default:

- Subindex 1: PDO mapping of the status word, object `statusword` (6041_h).

9.4.34 1A01_h 2nd transmit PDO mapping

The object shows which objects are mapped in T_PDO2 and transmitted with the PDO. When reading the object subindex 00_h the number of mapped objects is given.

Object description

| | |
|-------------|--------------------------|
| Index | 1A01 _h |
| Object name | 2nd transmit PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|--|
| Subindex | 00 _h , number of mapped application objects in PDO |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 2 |
| can be saved | – |
| Subindex | 01 _h , PDO mapping for the first application object to be mapped (status word) |
| Meaning | First object for the mapping in T_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 6041 0010 _h |
| can be saved | – |
| Subindex | 02 _h , PDO mapping for the second application object to be mapped (actual position) |
| Meaning | Second object for the mapping in T_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 6064 0020 _h |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 03 _h , PDO mapping for the third application object to be mapped |
| Meaning | Third object for the mapping in T_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 04 _h , PDO mapping for the fourth application object to be mapped |
| Meaning | Fourth object for the mapping in T_PDO2 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

The meaning of the bit states is described with the object `1st receive PDO-mapping` (1600_h).

Settings

The PDO assignment for T_PDO2 can be modified. The following allocate is preset for the point-to-point mode in the "profile position mode":

- Subindex 1: PDO mapping of the status word, object `statusword` (6041_h).
- Subindex 2: PDO mapping of the current position, object `position actual value` (6064_h).

9.4.35 1A02_h 3rd transmit PDO mapping

The object shows which objects are mapped in T_PDO3 and transmitted with the PDO. When reading the object subindex 00_h the number of mapped objects is given.

Object description

| | |
|-------------|--------------------------|
| Index | 1A02 _h |
| Object name | 3rd transmit PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|---|
| Subindex | 00 _h , number of mapped application objects in PDO |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | 1...8 |
| Default value | 2 |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 01 _h , PDO mapping for the first application object to be mapped (status word) |
| Meaning | First object for the mapping in T_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 6041 0010 _h |
| can be saved | – |

| | |
|---------------|--|
| Subindex | 02 _h , PDO mapping for the second application object to be mapped (actual velocity) |
| Meaning | Second object for the mapping in T_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | 606C 0020 _h |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 03 _h , PDO mapping for the third application object to be mapped |
| Meaning | Third object for the mapping in T_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

| | |
|---------------|---|
| Subindex | 04 _h , PDO mapping for the forth application object to be mapped |
| Meaning | Fourth object for the mapping in T_PDO3 |
| Access | read-write |
| PDO mapping | – |
| Value range | 0...4294967295 |
| Default value | – |
| can be saved | – |

The meaning of the bit states is described with the object 1st receive PDO-mapping (1600_h).

Settings

The PDO assignment for T_PDO3 cannot be modified. The following allocate is preset for speed mode in the "profile velocity mode":

- Bytes 0..1: Status word `statusword` (6041_h)
- Bytes 2..5: current speed velocity actual value (606C_h).

9.4.36 1A03_h 4th transmit PDO mapping

The object shows which objects are mapped in T_PDO4 and transmitted with the PDO. When reading the object subindex 00h the number of mapped objects is given.

Object description

| | |
|-------------|--------------------------|
| Index | 1A03h |
| Object name | 4th transmit PDO mapping |
| Object code | RECORD |
| Data type | PDO mapping |

Values description

| | |
|---------------|---------------------------------|
| Subindex | 00h, number of elements |
| Meaning | Number of values for the object |
| Access | read-write |
| PDO mapping | – |
| Value range | – |
| Default value | 0 |
| can be saved | – |

The meaning of the bit states is described at the object 1st receive PDO mapping (1600_h) .

Settings The PDO allocate for T_PDO4 can be modified.

10 Glossaries

10.1 Terms and Abbreviations

| | |
|------------------------|---|
| <i>AC</i> | Alternating Current |
| <i>CAN</i> | (C ontroller A rea N etwork), standardized open Fieldbus over which the drives and other devices from different manufacturers communicate with one another. |
| <i>CANopen</i> | Device and manufacturer-independent description language for communication in the CAN bus |
| <i>CiA</i> | CAN in Automation , CAN interest group, sets standards for CAN and CANopen. |
| <i>COB</i> | (C ommunication O bject) communication object, transport unit in a CAN network. |
| <i>COB-ID</i> | (C ommunication O bject- I dentifier) uniquely identifies every communications object in a CAN network |
| <i>DC</i> | Direct current |
| <i>Default values</i> | Factory settings. |
| <i>DriveCom</i> | specification of the DSP 402 status machine was created in accordance with the DriveCom specification. |
| <i>DS 301</i> | standardises the CANopen communications profile |
| <i>DSP 402</i> | standardises the CANopen device profile for drives and positioning controls |
| <i>E</i> | Encoder |
| <i>I/O</i> | Inputs/Outputs |
| <i>EDS</i> | (E lectronic D ata S heet) electronic data sheet |
| <i>Input device</i> | A device that can be connected to the RS232 interface for commissioning, either the HMI handheld operating unit or a PC with the commissioning software.. |
| <i>Electronic gear</i> | An input speed is recalculated by the drive system using the values of an adjustable gear factor to derive a new output speed for the motor movement. |
| <i>EMCY object</i> | Emergency Object |
| <i>EMC</i> | Electromagnetic compatibility |
| <i>Encoder</i> | Sensor for recording the angular position of a rotating element. The encoder is mounted on the motor and signals the angular position of the rotor. |
| <i>Limit switch</i> | Switch that signals an overrun of the permissible travel range. |
| <i>Power amplifier</i> | A device that generates current for controlling the motor in accordance with the positioning signals from the controller. |
| <i>Error class</i> | Classification of possible operating faults of the drive system that result in an error status. |
| <i>Heartbeat</i> | used for unconfirmed connection message from network devices. |

| | |
|-------------------------|--|
| <i>HMI</i> | Human Machine Interface, handheld operating unit. |
| <i>Power circuit</i> | see power amplifier |
| <i>Life-Guarding</i> | (monitoring for signs of life) for monitoring the connection of a NMT master |
| <i>Mapping</i> | assignment of object directory entries to PDOs |
| <i>node ID</i> | Node address assigned to a device on the network. |
| <i>NMT</i> | network management (NMT), component of the CANopen communications profile, tasks: initialising network and devices, starting, stopping, monitoring devices |
| <i>Node-Guarding</i> | Monitoring function with slave at an interface for cyclic communication. |
| <i>Object directory</i> | List of all parameters, values and functions available in the unit. Every entry is uniquely references via index (16 bit) and subindex (8 bit). |
| <i>Parameter</i> | Device functions and values that can be set and called by the user. |
| <i>PDO</i> | Process Data Object |
| <i>per.</i> | Designation of whether the value of the parameter is persistent, i.e. after switching off the unit it is retained in the memory. When entering via HMI the unit stores the value of the parameter automatically at each change. When changing a value via commissioning software or field bus, the user must explicitly store the value change in the persistent memory. |
| <i>Quick Stop</i> | Quick stop, function used to provide quick braking of the motor via a command or in the event of a fault. |
| <i>R_PDO</i> | Receive PDO |
| <i>SDO</i> | Service Data Object |
| <i>SYNC object</i> | Synchronisation object |
| <i>T_PDO</i> | Transmit PDO |

11 Index

A

- Abbreviations 10-1
- ABORT 7-2, 7-3
- Absolute point-to-point positioning 3-13, 3-14
- Acyclic data transfer 3-18
- Address
 - check 7-1
- Address setting 5-1
 - using parameters 5-1
- Analogue input 6-16
- Asynchronous error 7-2

B

- Baud rate
 - check 7-1
 - in the field bus 5-1
- Baud rate setting 5-1
 - using parameters 5-1
- Bit field data 3-3
- Bit field identifier 3-3
- Bit fields
 - data 3-3
 - identifier 3-3
- Boot Up
 - message 3-22, 3-25
- Boot-up
 - message 3-19
- Bus arbitration 3-3

C

- CAN
 - message 3-2
- CAN 3.0A 3-3
- CANopen
 - communications profile, NMT 3-21
 - error messages 7-3
 - message 3-3
 - standards 1-5
 - status machine 3-19
- ccd
 - see command code
- Check
 - address 7-1
 - baud rate 7-1
- Client-Server 3-6
- Client-server
 - SDO data exchange 3-7
- COB Id
 - bus arbitration 3-3
 - identification of communications objects 3-3
 - of communications objects 3-4

- tasks 3-3
- COB-ID 3-3
- COB-Id
 - EMCY object 3-20
 - for node guarding 3-23
 - SDO 3-8
 - SYNC object 3-18
- Coding
 - command code 3-9, 3-10
- Command code
 - read value 3-10
 - SDO 3-8
 - see command code
 - write value 3-9
- Command specifier 3-23
- Commissioning 5-1
- Communications objects
 - COB Ids 3-4
 - control of 3-3
 - for PDO 3-12
 - identification 3-3
 - overview 3-2
- Communications profile
 - DS 301 1-4
- Communications relationship
 - client-server 3-5
 - master-slave 3-5
 - producer-consumer 3-5
- Connection error
 - node guarding 3-24
- Connection monitoring
 - NMT services 3-23
- connection monitoring
 - heartbeat 3-25
- Controller
 - field bus 6-1
 - local 6-1
- Current regulation
 - example 6-6
- Cyclic data transfer 3-18

D

- Danger classes 2-2
- Data
 - Intel format 3-8
 - non-volatile saved 3-22
 - read 3-10
 - SDO 3-8
 - write 3-9
- Data frame 3-5
 - of the NMT device service 3-22
 - SDO 3-8
- Data length
 - flexible 3-11
- Data transfer

- acyclic 3-18
 - cyclic 3-18
 - synchronous 3-17
- Device error
 - internal 3-19
- Device profile
 - DSP 402 1-4
- Diagnostics 7-1
- Disposal 8-1
- DS 301
 - communications profile 1-4
- DSP 402
 - device profile 1-4

E

- Electronic gear
 - example 6-10
- EMCY
 - COB-Id of the object 3-20
 - message 3-19
 - Object 3-2
 - object 3-19
- Emergency braking function
 - see Quick-Stop
- Emergency object
 - see EMCY object
- Emergency service 3-19
- Enable
 - PDO 3-12
- Error
 - evaluation 3-19
 - handling 3-19
 - messages to CANopen 7-3
 - response with SDO 3-10
- Error code 3-20
 - table 7-3
- Error diagnosis
 - connections to field-bus mode 7-1
 - function test on field bus 7-1
- Error memory 3-20
- Error register 3-20
- error register 7-3
- Example
 - current regulation 6-6
 - electronic gear 6-10
 - index and subindex entries 3-1
 - manual movement 6-12
 - point-to-point mode 6-2
 - referencing 6-5
 - SDO message 3-8
 - selection of a COB-Id 3-5
 - setting for R_PDO3 3-12
 - speed profile 6-4
 - speed regulation 6-8

F

Following error monitoring 6-15
Function code 3-3, 3-4
Function test
 on field bus 7-1

H

Heartbeat 3-23, 3-25
 mutual monitoring 3-25
 NMT status evaluation 3-25
 start of monitoring 3-25

I

Identification
 of communications objects 3-3
Index
 Intel format 3-8
 SDO 3-8
Inputs and outputs
 monitor 6-16
Intel format
 data 3-8
 error code 3-20
 index 3-8
Intended use 2-1

L

Layer model
 CAN Data Link Layer 1-2
 CAN Physical Layer 1-2
 CANopen Application Layer 1-2
Life guarding 3-23

M

Maintenance 8-1
Manual movement
 example 6-12
Manufacturer-specific
 object values
 R_PDO4 3-13
 T_PDO4 3-14
 operating modes 6-1, 6-6
 profiles 1-4
Master-Slave 3-5
Message 3-2
 boot-up 3-19
 CANopen 3-3
 EMCY 3-19
 NMT 3-22, 3-23
 PDO 3-12
 SDO 3-8
Message objects 7-2
 EMCY(80h+ node-Id) 7-2
 error code (603Fh) 7-2

- error register (1001h) 7-2
 - status word (6041h) 7-2
- Message-oriented communication 1-1
- Messages
 - asynchronous error 7-2
 - error code (603Fh) 7-3
 - error register (1001h) 7-3
 - on the unit status 7-2
 - synchronous error 7-2
- Monitor
 - inputs and outputs 6-16
- Monitoring
 - following error 6-15
 - parameters 6-15
 - positioning limits 6-15
 - software limit switch 6-15
- Monitoring functions 6-15
- Motor standstill 6-13
- Motor stop 6-13
- Movement interrupted
 - cause 7-3
- Multimaster capacity 1-1

N

- Network management
 - see NMT
- NMT
 - message 3-22
 - network services 3-21
 - receiver of a message 3-23
 - services 3-2, 3-21
 - for connection monitoring 3-23
 - for device control 3-21
 - initialising 3-22
 - status machine 3-21
 - status of slave 3-23
 - structure of a message 3-23
- Node address 3-4, 3-23
- Node guarding 3-23
 - COB-Id 3-23
 - connection error 3-24
- Node-ID 3-3

O

- Object data
 - back-up 6-16
 - restore 6-16
- Object groups
 - overview 1-3
- Objects
 - standardised 3-1
- Operating modes
 - manufacturer-specific 6-6
 - standardised 6-1
 - start and monitoring 3-18

- Operating status
 - of the device 3-13
- Operation 6-1
- Overview
 - communications objects 3-2
 - object groups 1-3

P

- PDO 3-2, 3-11
 - communications objects 3-12
 - enable 3-12
 - message 3-12
 - producer-consumer 3-11
 - R_PDO
 - see R_PDO
 - receive PDOs 3-13
 - send PDOs 3-14
 - settings 3-12
 - Start PDO 3-18
 - T_PDO
 - see T_PDO
 - time intervals 3-12
- PDO mapping 3-15
 - static 3-15
 - structure of entries 3-16
- Point-to-point mode
 - example 6-2
- Point-to-point positioning 3-13, 3-14
- Position controller
 - setpoint position 6-15
- Position overrun 6-15
- Positioning limits
 - monitor 6-15
- Prioritisation of messages 1-1
- Process Data Object
 - see PDO
- Process data objects
 - see PDO
- Producer-Consumer 3-6
- Producer-consumer
 - EMCY 3-19
 - heartbeat 3-25
 - PDO 3-11
 - SYNC 3-17
- Profiles
 - manufacturer-specific 1-4
 - standardised 1-4

Q

- Qualifications, personnel 2-1
- Quick Stop 6-13
- Quick-Stop 6-13
 - current 6-13

R

- R_PDO
 - R_PDO1 3-13
 - R_PDO2 3-13
 - R_PDO3 3-13
 - R_PDO4 3-13
- ramp function 6-13
- Ramp shape 6-13
- Ramp steepness 6-13
- Real-time data exchange 3-11
- Receive PDOs 3-13
- Receiver
 - of an NMT message 3-23
- Referencing
 - example 6-5
- Residual error probability 1-1
- response
 - to SDO error 3-10
- Reversal of direction of rotation 6-14

S

- SDO 3-2, 3-7
 - answer 3-10
 - COB-Id 3-8
 - command code 3-8
 - data 3-8
 - data frame 3-8
 - error message 7-2, 7-3
 - error response 3-10
 - Index, Subindex 3-8
 - message 3-8
 - message types 3-7
 - transmission error 7-3
- Send PDOs 3-14
- Service 8-1
- Service address 8-1
- Service Data Object
 - see SDO
- Service data objects 3-2
 - see SDO
- Services
 - EMCY 3-19
 - for connection monitoring 3-21
 - for device control 3-21
 - NMT 3-2, 3-21
- Setpoint position
 - of the position controller 6-15
- Setting up the unit 5-1
- Software limit switch
 - monitor 6-15
- Specification
 - CAN 3.0A 3-3
- Speed mode 3-13, 3-14
- Speed profile
 - example 6-4

- Speed regulation
 - example 6-8
- Speed variations
 - minimum 6-13
- Standardised operating modes 6-1
- Standstill window 6-13
- Status machine
 - CANopen 3-19
 - NMT 3-21
- Status word
 - of the status machine 3-14
- Subindex
 - SDO 3-8
- SYNC object 3-2, 3-17
 - COB-Id 3-18
 - with PDO 3-11
- Synchronisation 3-17
 - time values 3-17
- Synchronisation object
 - see SYNC object
- Synchronous
 - data transfer 3-17
 - error 7-2

T

- T_PDO
 - T_PDO1 3-14
 - T_PDO2 3-14
 - T_PDO3 3-14
 - T_PDO4 3-14
- Tasks
 - of the COB Id 3-3
- Terms 10-1
- Time interval
 - event timer 3-13
 - heartbeat 3-25
 - inhibit time 3-12
 - PDO 3-12
- Time values
 - for synchronisation 3-17
- Troubleshooting 7-1