LXM23A CANopen

Fieldbus protocol for servo drive

Fieldbus manual

V2.00, 10.2011

0198441113938, V2.00, 10.2011

www.schneider-electric.com

Schneider Electric

Important information

This manual is part of the product.

Carefully read this manual and observe all instructions.

Keep this manual for future reference.

Hand this manual and all other pertinent product documentation over to all users of the product.

Carefully read and observe all safety instructions and the chapter "Before you begin - safety information".

Some products are not available in all countries.
For information on the availability of products, please consult the catalog.

Subject to technical modifications without notice.

All details provided are technical data which do not constitute warranted qualities.

Most of the product designations are registered trademarks of their respective owners, even if this is not explicitly indicated.

0198441113938, V2.00, 10.2011

Table of contents

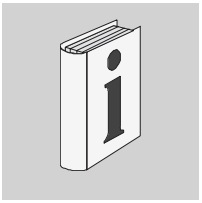
0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

About this manual



The information provided in this manual supplements the product manual.

Source manuals

The latest versions of the manuals can be downloaded from the Internet at:

http://www.schneider-electric.com

Corrections and suggestions

We always try to further optimize our manuals. We welcome your suggestions and corrections.

Please get in touch with us by e-mail:
techcomm@schneider-electric.com.

Work steps

If work steps must be performed consecutively, this sequence of steps is represented as follows:

- Special prerequisites for the following work steps
- ▸ Step 1
- ◃ Specific response to this work step
- ▸ Step 2

If a response to a work step is indicated, this allows you to verify that the work step has been performed correctly.

Unless otherwise stated, the individual steps must be performed in the specified sequence.

Making work easier

Information on making work easier is highlighted by this symbol:

Sections highlighted this way provide supplementary information on making work easier.

SI units

SI units are the original values. Converted units are shown in brackets behind the original value; they may be rounded.

Example:
Minimum conductor cross section: 1.5 mm^2 (AWG 14)

Glossary

Explanations of special technical terms and abbreviations.

Index

List of keywords with references to the corresponding page numbers.

Further reading

Recommended literature for further reading

CAN users and manufacturers organization

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

0198441113938, V2.00, 10.2011

CANopen standards

- CiA Standard 301 (DS301)
  CANopen application layer and communication profile
- CiA Standard 402 (DSP402)
  Device profile for drives and motion control
- ISO 11898: Controller Area Network (CAN) for high speed communication
- EN 50325-4: Industrial communications subsystem based on ISO 11898 for controller device interfaces (CANopen)

Literature

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

0198441113938, V2.00, 10.2011

1 Introduction

1

1.1 CAN bus

The CAN bus (**C**ontroller **A**rea **N**etwork) was originally developed for fast, economical data transmission in the automotive industry. Today, the CAN bus is also used in industrial automation technology and has been further developed for communication at fieldbus level.

Features of the CAN bus

The CAN bus is a standardized, open bus enabling communication between devices, sensors and actuators from different manufacturers. The features of the CAN bus comprise

- Multimaster capability

  Each device in the fieldbus can transmit and receive data independently without depending on an "ordering" master functionality.
- Message-oriented communication

  Devices can be integrated into a running network without reconfiguration of the entire system. The address of a new device does not need to be specified on the network.
- Prioritization of messages

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

  Various security features in the network reduce the probability of undetected incorrect data transmission to less than 10^{-11}.

Transmission technology

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

Network devices

Examples of CAN bus devices are

- Automation devices, for example, PLCs
- PCs
- Input/output modules
- Drives
- Analysis devices
- Sensors and actuators

0198441113938, V2.00, 10.2011

1.2 CANopen technology

1.2.1 CANopen description language

CANopen is a device- and manufacturer-independent description language for communication via the CAN bus. CANopen provides a common basis for interchanging commands and data between CAN bus devices.

1.2.2 Communication layers

CANopen uses the CAN bus technology for data communication.

CANopen is based on the basic network services for data communication as per the ISO-OSI model model. 3 layers enable data communication via the CAN bus.

- Physical Layer
- Data Link Layer
- Application Layer

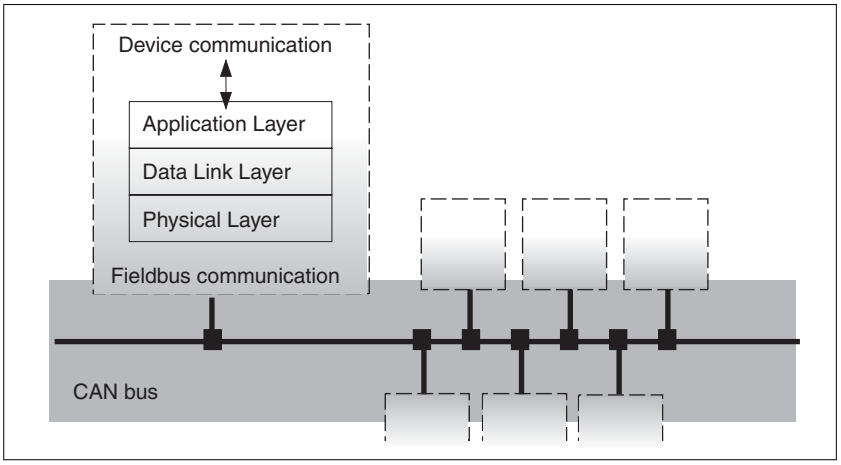


Figure 1: CANopen layer model

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

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

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

0198441113938, V2.00, 10.2011

1.2.3 Objects

Processes under CANopen are executed via objects. Objects carry out different tasks; they act as communication objects for data transport to the fieldbus, control the process of establishing a connection or monitor the network devices. If objects are directly linked to the device (device-specific objects), the device functions can be used and changed via these objects.

The product provides corresponding parameters for CANopen object groups 2000_h and 6000_h.
The names of the parameters and the data type of the parameters may be different from the DSP402 definition for object group 6000_h. In this case, enter the data type according to the DS402.
A detailed description of the parameters can be found in the product manual in the Parameters chapter.

Object dictionary

The object dictionary of each network device allows for communication between the devices. Other devices find the objects with which they can communicate in this dictionary.

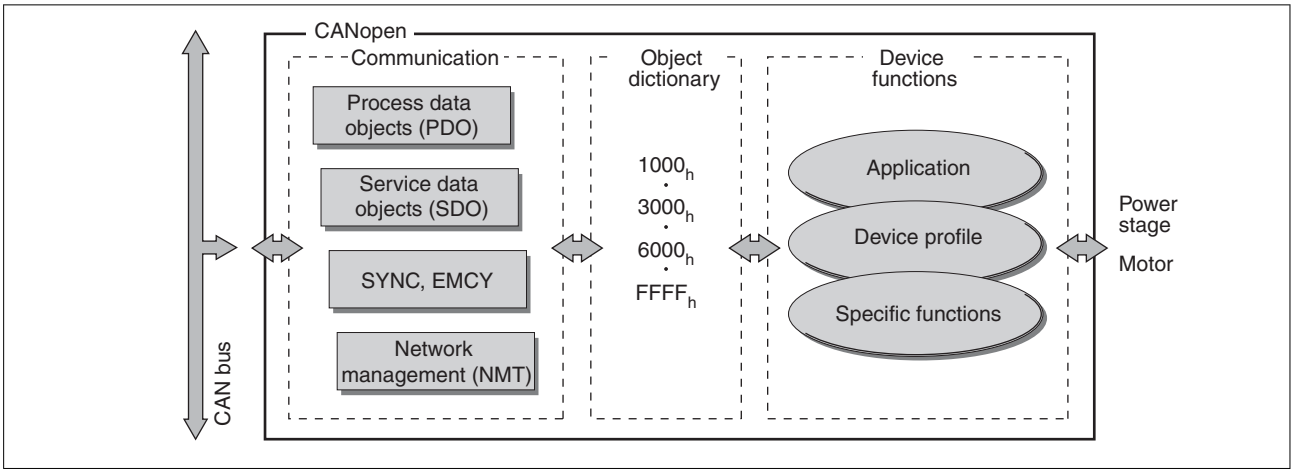


Figure 2: Device model with object dictionary

The object dictionary contains objects for describing the data types and executing the communication tasks and device functions under CANopen.

Object index

Each object is addressed by means of a 16 bit index, which is represented as a four-digit hexadecimal number. The objects are arranged in groups in the object dictionary. The following table shows an overview of the object dictionary as per the CANopen specifications.

Index range (hex)	Object groups
1000_h-$2FFF_h$	Communication profile
2000_h-$5FFF_h$	Vendor-specific objects
6000_h-$9FFF_h$	Standardized device profiles
$A000_h$-$AFFF_h$	Standardized network variable
$B000_h$-$BFFF_h$	Standardized system variable
$C000_h$-$FFFF_h$	Reserved.

See chapter "8 Object dictionary" for a list of the CANopen objects.

0198441113938, V2.00, 10.2011

1.2.4 CANopen profiles

Standardized profiles

Standardized profiles describe objects that are used with different devices without additional configuration. The users and manufacturers organization CAN in Automation has standardized various profiles. These include:

- DS301 communication profile
- DSP402 device profile

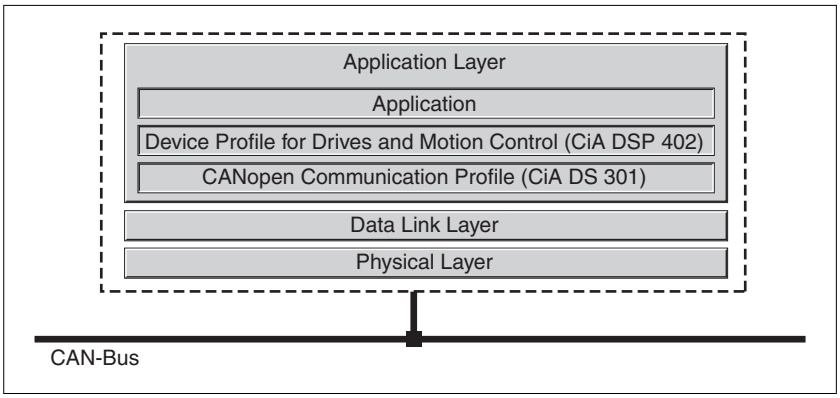


Figure 3: CANopen reference model

DS301 communication profile

The DS301 communication profile is the interface between device profiles and CAN bus. It was specified in 1995 under the name DS301 and defines uniform standards for common data exchange between different device types under CANopen.

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

DSP402 device profile

The DSP402 device profile describes standardized objects for positioning, monitoring and settings of drives. The tasks of the objects include:

- Device monitoring and status monitoring (Device Control)
- Standardized parameterization
- Changing, monitoring and execution of operating modes

Vendor-specific profiles

The basic functions of a device can be used with objects of standardized device profiles. Only vendor-specific device profiles offer the full range of functions. The objects with which the special functions of a device can be used under CANopen are defined in these vendor-specific device profiles.

0198441113938, V2.00, 10.2011

2 Before you begin - safety information

2

The information provided in this manual supplements the product manual. Carefully read the product manual before using the product.

2.1 Qualification of personnel

Only appropriately trained persons who are familiar with and understand the contents of this manual and all other pertinent product documentation are authorized to work on and with this product. In addition, these persons must have received safety training to recognize and avoid hazards involved. These persons must have sufficient technical training, knowledge and experience and be able to foresee and detect potential hazards that may be caused by using the product, by changing the settings and by the mechanical, electrical and electronic equipment of the entire system in which the product is used.

All persons working on and with the product must be fully familiar with all applicable standards, directives, and accident prevention regulations when performing such work.

2.2 Intended use

The functions described in this manual are only intended for use with the basic product; you must read and understand the appropriate product manual.

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements and the technical data.

Prior to using the product, you must perform a risk assessment in view of the planned application. Based on the results, the appropriate safety measures must be implemented.

Since the product is used as a component in an entire system, you must ensure the safety of persons by means of the design of this entire system (for example, machine design).

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts.

Any use other than the use explicitly permitted is prohibited and can result in hazards.

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel.

The product must NEVER be operated in explosive atmospheres (hazardous locations, Ex areas).

0198441113938, V2.00, 10.2011

2.3 Hazard categories

Safety instructions to the user are highlighted by safety alert symbols in the manual. In addition, labels with symbols and/or instructions are attached to the product that alert you to potential hazards.

Depending on the seriousness of the hazard, the safety instructions are divided into 4 hazard categories.

⚠ DANGER

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

⚠ 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.

CAUTION

CAUTION used without the safety alert symbol, is used to address practices not related to personal injury (e.g. **can result** in equipment damage).

0198441113938, V2.00, 10.2011

2.4 Basic information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop, overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical functions.
- System control paths may include communication links. Consideration must be given to the implication of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines. [1)]
- Each implementation of the product must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death or serious injury.

1) For USA: Additional information, refer to NEMA ICS 1.1 (latest edition), “Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control” and to NEMA ICS 7.1 (latest edition), “Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems”.

0198441113938, V2.00, 10.2011

2.5 Standards and terminology

Technical terms, terminology and the corresponding descriptions in this manual are intended to use the terms or definitions of the pertinent standards.

In the area of drive systems, this includes, but is not limited to, terms such as "safety function", "safe state", "fault", "fault reset", "failure", "error", "error message", "warning", "warning message", etc.

Among others, these standards include:

- IEC 61800 series: "Adjustable speed electrical power drive systems"
- IEC 61158 series: "Industrial communication networks - Fieldbus specifications"
- IEC 61784 series: "Industrial communication networks - Profiles"
- IEC 61508 series: "Functional safety of electrical/electronic/ programmable electronic safety-related systems"

Also see the glossary at the end of this manual.

0198441113938, V2.00, 10.2011

3 Basics

3

3.1 Communication profile

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

The following can be done by accessing the objects of the network devices

- Exchange parameter values
- Start motion functions of individual CAN bus devices
- Request status information

3.1.1 Object dictionary

Each CANopen device manages an object dictionary which contains the objects for communication.

Index, subindex

The objects are addressed in the object dictionary via a 16 bit index. One or more 8 bit subindex entries for each object specify individual data fields in the object. Index and subindex are shown in hexadecimal notation with a subscript "$_h$".

Example

The following table shows index and subindex entries using the example of the object `homing speeds(6099h)` for specifying the positions of software limit switches.

Index	Subindex	Meaning
6099_h	00_h	Number of data fields
6099_h	01_h	Search velocity for search for limit switch
6099_h	02_h	Search velocity for search for index pulse

Table 1: Example of index and subindex entries

Object descriptions in the manual

For CANopen programming of a device, the objects of the following object groups are described in detail:

- $1xxx_h$ objects: Communication objects in this chapter
- $2xxx_h$ objects: Vendor-specific objects required to control the device in chapter "6 Operation".
- $6xxx_h$ objects: Standardized objects of the device profile in chapter "6 Operation"

0198441113938, V2.00, 10.2011

Standardized objects Standardized objects allow you to use the same application program for different network devices of the same device type. This requires these objects to be contained in the object dictionary of the network devices. Standardized objects are defined in the DS301 communication profile and the DSP402 device profile.

3.1.2 Communication objects

Overview The communication objects are standardized with the DS301 CANopen communication profile. The objects can be classified into 4 groups according to their tasks.

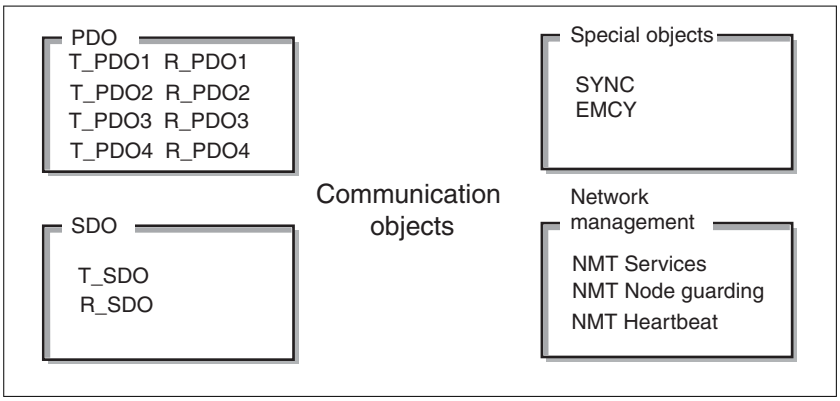


Figure 4: Communication objects; the following applies to the perspective of the network device: T_..: "Transmit", R_..: "Receive"

- PDOs (process data objects) for real-time transmission of process data
- SDOs (service data object) for read and write access to the object dictionary
- Objects for controlling CAN messages:
  - SYNC object (synchronization object) for synchronization of network devices
  - EMCY object (emergency object), for signaling errors of a device or its peripherals.
- Network management services:
  - NMT services for initialization and network control (NMT: network management)
  - NMT Node Guarding for monitoring the network devices
  - NMT Heartbeat for monitoring the network devices

CAN message Data is exchanged via the CAN bus in the form of CAN messages. A CAN message transmits the communication object as well as numerous administration and control data.

0198441113938, V2.00, 10.2011

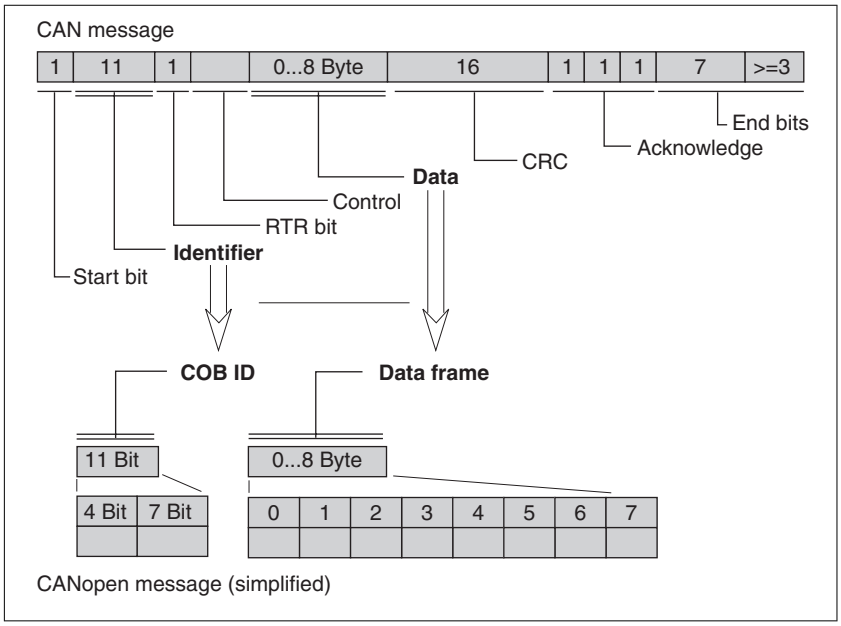


Figure 5: CAN message and simplified representation of CANopen message

CANopen message

For work with CANopen objects and for data exchange, the CAN message can be represented in simplified form because most of the bits are used for error correction. These bits are automatically removed from the receive message by the data link layer of the OSI model, and added to a message before it is transmitted.

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 data frame (maximum length 8 bytes) of a CANopen message.

COB ID

The COB ID (**C**ommunication **OB**ject **Id**entifier) has 2 tasks as far as controlling communication objects is concerned:

- Bus arbitration: Specification of transmission priorities
- Identification of communication objects

An 11 bit COB identifier as per the CAN 3.0A specification is defined for CAN communication; it comprises 2 parts

- Function code, 4 bits
- Node address (node ID), 7 bits.

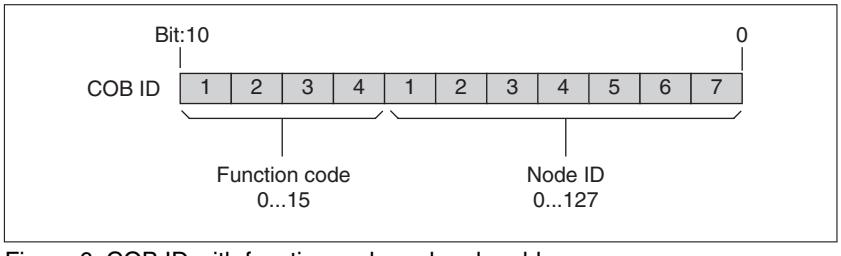


Figure 6: COB ID with function code and node address

COB IDs of the communication objects

The following table shows the COB IDs of the communication objects with the factory settings. The column "Index of object parameters" shows the index of special objects with which the settings of the communication objects can be read or modified via an SDO.

0198441113938, V2.00, 10.2011

Communication 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 1)	0 0 1 1	x x x x x x x	384 (180_h) + node ID	1800_h
R_PDO1 1)	0 1 0 0	x x x x x x x	512 (200_h) + node ID	1400_h
T_PDO2 1)	0 1 0 1	x x x x x x x	640 (280_h) + node ID	1801_h
R_PDO2 1)	0 1 1 0	x x x x x x x	768 (300_h) + node ID	1401_h
T_PDO3 1)	0 1 1 1	x x x x x x x	896 (380_h) + node ID	1802_h
R_PDO3 1)	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	-
R_SDO	1 1 0 0	x x x x x x x	1536 (600_h) + node ID	-
NMT error control	1 1 1 0	x x x x x x x	1792 (700_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 2: COB IDs of the communication objects

COB IDs of PDOs can be changed if required. The assignment pattern for COB IDs only specifies a basic setting.

Function code The function code classifies the communication objects. Since the bits of the function code in the COB ID are more significant, the function code also controls the transmission priorities: Objects with a lower function code are transmitted with higher priority. For example, an object with function code "1" is transmitted prior to an object with function code "3" in the case of simultaneous bus access.

Node address Each network device is configured before it can be operated on the network. The device is assigned a unique 7 bit 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 messages to all reachable devices simultaneously.

Example Selection of a COB ID

For a device with the node address 5, the COB ID of the communication object T_PDO1 is:

384+node ID = 384 (180_h) + 5 = 389 (185_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

0198441113938, V2.00, 10.2011

The data frames contain the respective communication objects.

3.1.3 Communication relationships

CANopen uses 3 relationships for communication between network devices:

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

Master-slave relationship

A network master controls the message traffic. A slave only responds when it is addressed by the master.

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

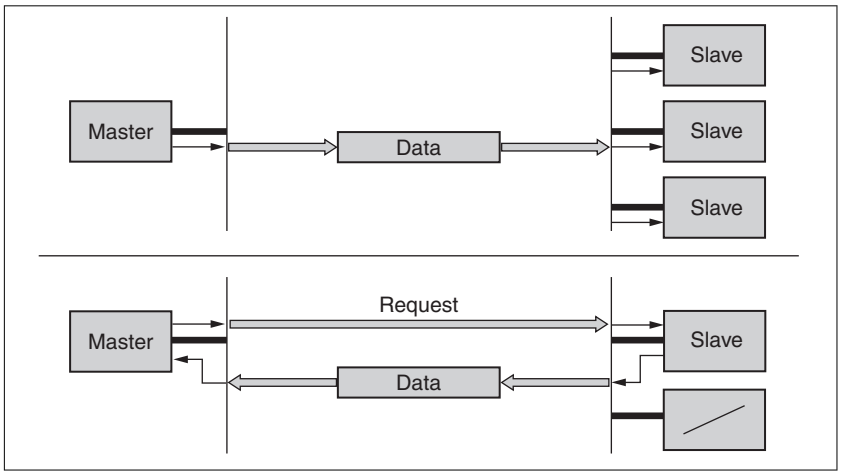


Figure 7: Master - slave relationships

Messages can be interchanged with and without confirmation. If the master sends an unconfirmed CAN message, it can be received by a single slave or by all reachable slaves or by no slave.

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

0198441113938, V2.00, 10.2011

Client-server relationship

A client-server relationship is established between 2 devices. The "server" is the device whose object dictionary is used during data exchange. The "client" addresses and starts the exchange of messages and waits for a confirmation from the server.

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

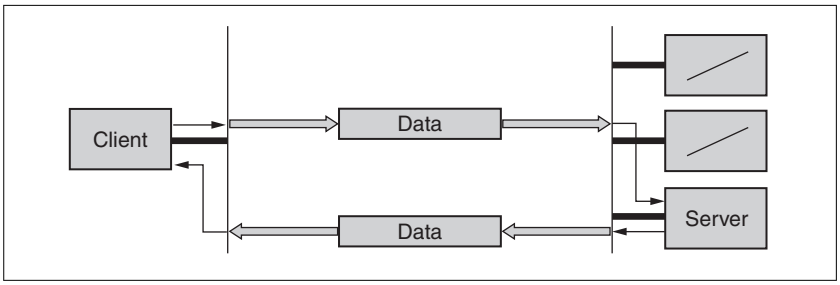


Figure 8: Client-server relationship

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

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.

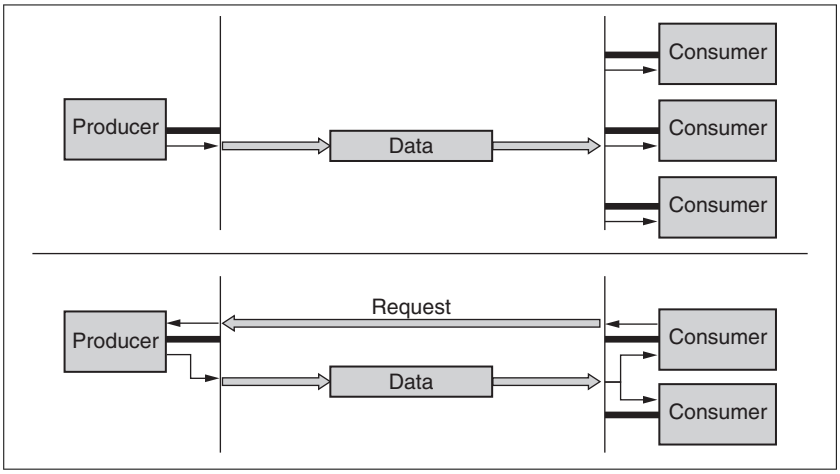


Figure 9: Producer-consumer relationships

The producer sends a message that can be received by one or more network devices. The producer does not receive an acknowledgement to the effect that the message was received. The message transmission can be triggered by

- An internal event, for example, "target position reached"
- The synchronization object SYNC
- A request of a consumer

See chapter "3.3 Process data communication" for details on the function of the producer-consumer relationship and on requesting messages.

0198441113938, V2.00, 10.2011

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 dictionary 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 dictionary of a different device or to change them in the dictionary.

The T_SDO of an SDO client is used to send the request for data exchange; the R_SDO is used to receive. The data frame of an SDO consist of 8 bytes.

SDOs have a higher COB ID than PDOs; therefore, they are transmitted over the CAN bus at a lower priority.

3.2.2 SDO data exchange

A service data object (SDO) transmits parameter data between 2 devices. The data exchange conforms to the client-server relationship. The server is the device to whose object dictionary an SDO message refers.

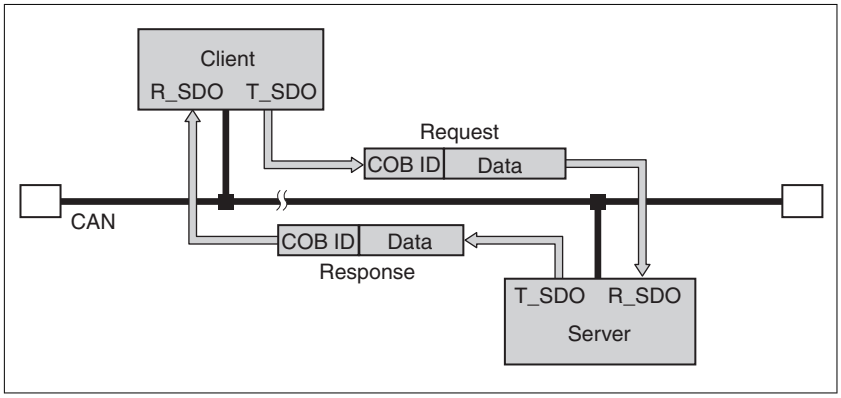


Figure 10: SDO message exchange with request and response

Message types Client-server communication is triggered by the client to send parameter values to the server or to get them from the server. In both cases, the client starts the communication with a request and receives a response from the server.

0198441113938, V2.00, 10.2011

3.2.3 SDO message

Put simply, an SDO message consists of the COB ID and the SDO data frame, in which up to 4 bytes of data can be sent. Longer data sequences are distributed over multiple SDO messages with a special protocol.

The device transmits SDOs with a data length of up to 4 bytes. Greater amounts of data such as 8 byte values of the data type "Visible String 8" can be distributed over multiple SDOs and are transmitted successively in blocks of 7 bytes.

Example

The following illustration shows an example of an SDO message.

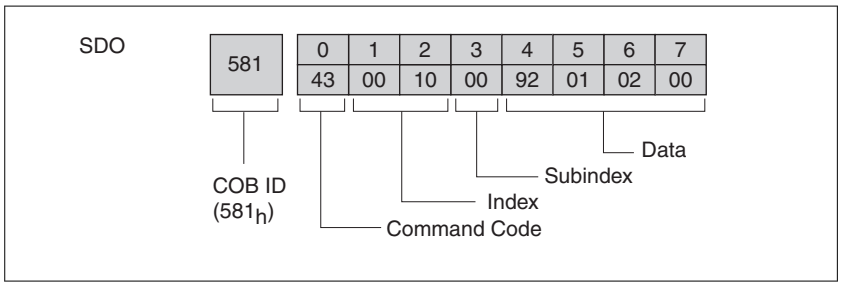


Figure 11: SDO message, example

COB ID and data frame

R_SDO and T_SDO have different COB IDs.
The data frame of an SDO messages consists of:

- Command code (ccd) which contains the SDO message type and the data length of the transmitted value
- Index and subindex which point to the object whose data is transported with the SDO message
- Data of up to 4 bytes

Evaluation of numeric values

Index and data are transmitted left-aligned in Intel format. If the SDO contains numerical values of more than 1 byte in length, the data must be rearranged byte-by-byte before and after a transmission.

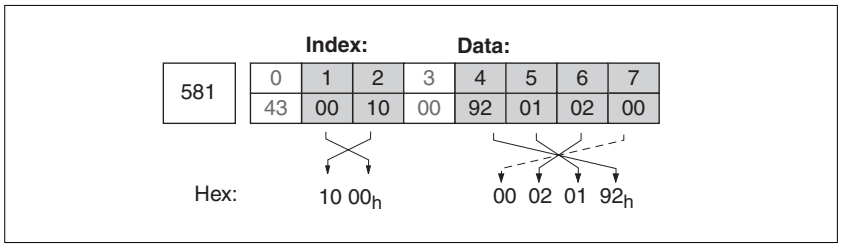


Figure 12: Rearranging numeric values greater than 1 byte

0198441113938, V2.00, 10.2011

3.2.4 Reading and writing data

Writing data

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

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

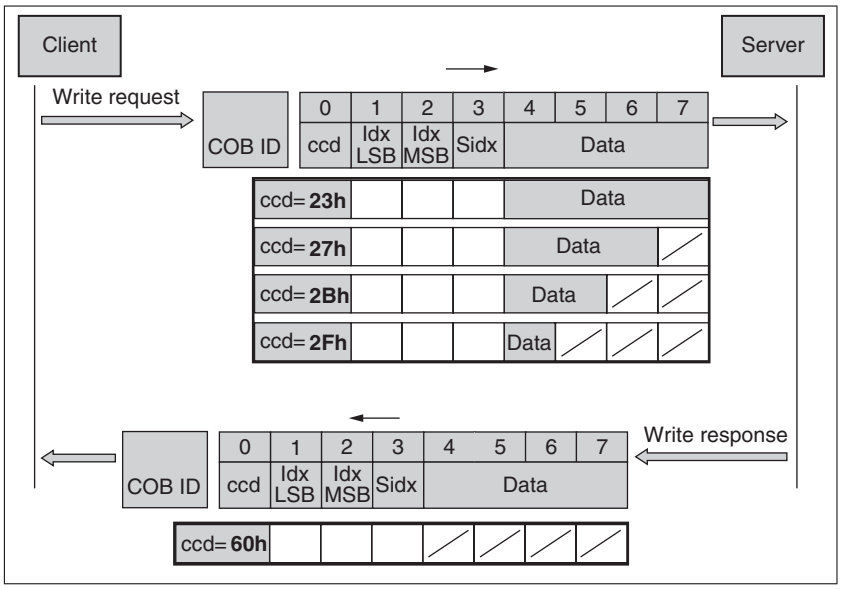


Figure 13: Writing parameter values

Unused bytes in the data field are shown with a slash in the graphic. The content of these data fields 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	$2B_h$	$2F_h$	Transmitting parameters
Write response	60_h	60_h	60_h	60_h	Confirmation
Error response	80_h	80_h	80_h	80_h	Error

Table 3: Command code for writing parameter values

0198441113938, V2.00, 10.2011

Reading data

The client starts a read request by transmitting the index and subindex that point to the object or part of the object whose value it wants to read.

The server confirms the request by sending the desired data. The SDO response contains the same index and subindex. The length of the response data is specified in the command code "ccd".

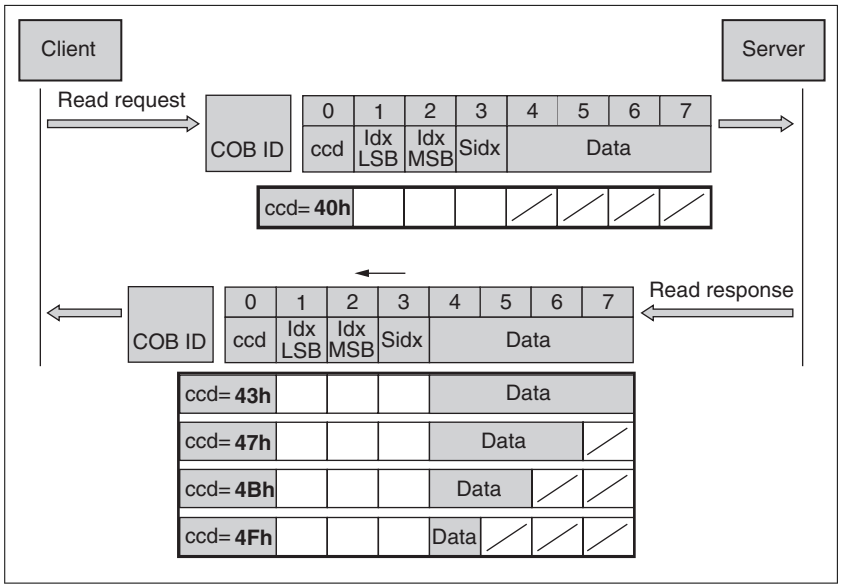


Figure 14: Reading a parameter value

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

ccd coding

The table below shows the command code for transmitting 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 4: Command code for transmitting a read value

Error response

If a message could not be evaluated, the server sends an error message. See chapter "7.3.2 SDO Abort Codes" for details on the evaluation of the error message.

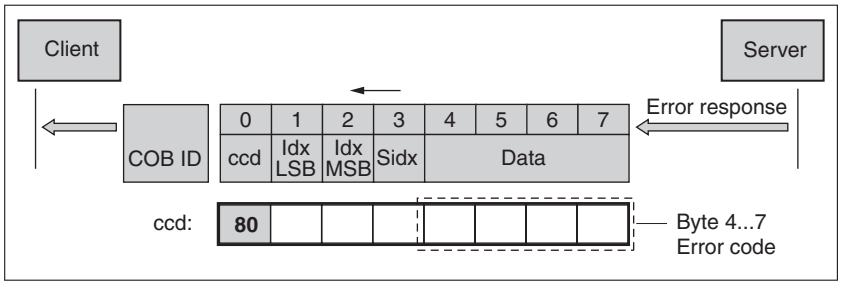


Figure 15: Response with error message (error response)

0198441113938, V2.00, 10.2011

3.2.5 Reading data longer than 4 bytes

If values of more than 4 bytes are to be transmitted with an SDO message, the message must be divided into several frames. Each frame consists of 2 parts.

- Request by the SDO client,
- Confirmation by the SDO server.

The request by the SDO client contains the command code "ccd" with the toggle bit and a data segment. The confirmation frame also contains a toggle bit in the "ccd" segment. In the first frame, the toggle bit has the value "0", in the subsequent frames it toggles between 1 and 0.

Reading data

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

The server confirms the request by transmitting index, subindex, data length and the first 4 bytes of the requested data. The command code specifies that data of more than 4 bytes are transmitted. The command code of the read response from the server to the first message is 41_h.

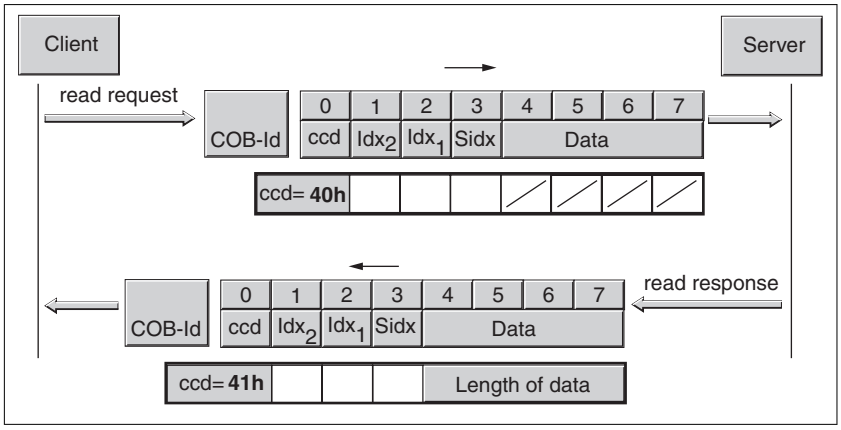


Figure 16: Transmitting the first message

In the next frames, the remaining data is requested and transmitted in packets of 7 bytes from the server.

ccd coding

The table below shows the command code for transmitting a read value. It depends on the message type, the value of the toggle bit, the transmitted data length and the value of the bit that indicates the end of the entire SDO message.

0198441113938, V2.00, 10.2011

Message type	Data length used							Meaning
	7 byte	6 byte	5 byte	4 byte	3 byte	2 byte	1 byte	
Read requestToggle Bit = 0	–	60h	60h	–	60h	60h	60h	Confirmation with Toggle Bit = 0
Read requestToggle Bit = 1	70h	70h	70h	70h	70h	70h	70h	Confirmation with Toggle Bit = 1
Read responseToggle Bit = 0	00h	–	–	–	–	–	–	Send parameter with Toggle Bit = 0
Read responseToggle Bit = 1	10h	–	–	–	–	–	–	Send parameter with Toggle Bit = 1
Read response last message-Toggle Bit = 0	01h	03h	05h	07h	09h	0Bh	0Dh	Transmit parameter with last message andToggle Bit = 0
Read response last message-Toggle Bit = 1	11h	13h	15h	17h	19h	1Bh	1Dh	Transmit parameter with last message andToggle Bit = 1
Error response	80h	80h	80h	80h	80h	80h	80h	Error

Refer to the DS301 of the CiA for additional information on this procedure.

0198441113938, V2.00, 10.2011

3.3 Process data communication

3.3.1 Overview

Process data objects (PDO: **P**rocess **D**ata **O**bject) are used for real-time data exchange of process data such as actual and reference values or the operating state of the device. Transmission is very fast because the data is sent without additional administration data and data transmission acknowledgement from the recipient is not required.

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

The length of a PDO message and the assignment of the data fields are specified by PDO mapping. See chapter "3.3.4 PDO mapping" for additional information.

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

3.3.2 PDO data exchange

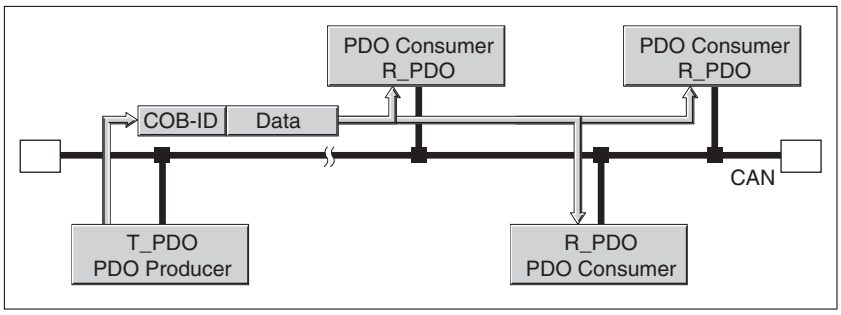


Figure 17: PDO data exchange

Data exchange with PDOs follows to the producer-consumer relationship and can be triggered in 3 ways

- Synchronized
- Event-driven, asynchronous

The SYNC object controls synchronized data processing. Synchronous PDO messages are transmitted immediately like the standard PDO messages, but are only evaluated on the next SYNC. For example, several drives can be started simultaneously via synchronized data exchange.

The device immediately evaluates PDO messages that are called on request or in an event-driven way.

The transmission type can be specified separately for each PDO with subindex 02_h (transmission type) of the PDO communication parameter. The objects are listed in Table 5.

0198441113938, V2.00, 10.2011

3.3.3 PDO message

T_PDO, R_PDO

One PDO each is available for sending and receiving a PDO message:

- T_PDO to transmit the PDO message (T: Transmit),
- R_PDO to receive PDO messages (R: Receive).

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

The device uses 8 PDOs, 4 receive PDOs and 4 transmit PDOs. By default, the PDOs are evaluated or transmitted in an event-driven way.

PDO settings

The PDO settings can be read and changed with 8 communication 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 5: Communication objects for PDO

Activating PDOs

With the default PDO settings, R_PDO1 and T_PDO1 are activated. The other PDOs must be activated first.

A PDO is activated with bit 31 (valid bit) in subindex 01_h of the respective communication object:

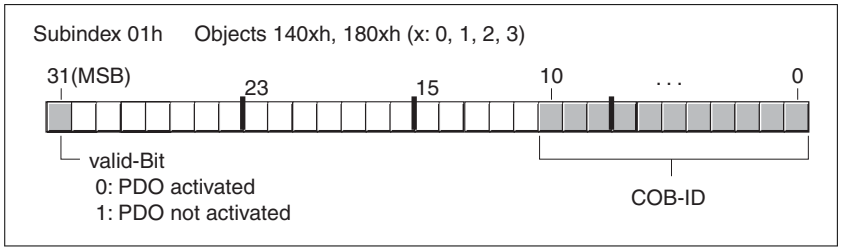


Figure 18: Activating PDOs via subindex 01_h, bit 31

Example

Setting for R_PDO3 in object 1402_h

- Subindex 01_h = 8000 04xx$_h$: R_PDO3 not activated
- Subindex 01_h = 0000 04xx$_h$: R_PDO3 activated.

Values for "x" in the example depend on the COB ID setting.

PDO time intervals

The time intervals "inhibit time" and "event timer" can be set for each transmit PDO.

0198441113938, V2.00, 10.2011

- The time interval "inhibit time" can be used to reduce the CAN bus load, which can be the result of continuous transmission of T_PDOs. If an inhibit time not equal to zero is entered, a transmitted PDO will only be re-transmitted after the inhibit time has elapsed. The time is set with subindex 03_h.
- The time interval "event timer" cyclically triggers an event message. After the time intervals has elapsed, 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, R_PDO3 and R_PDO4 are preset.

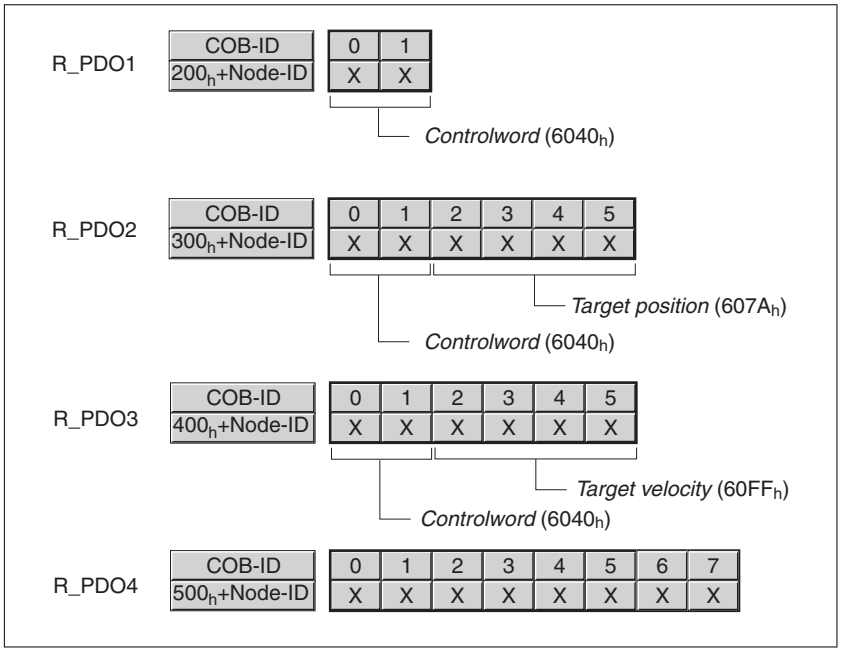


Figure 19: Receive PDOs

R_PDO1

R_PDO1 contains the control word, object `controlword (6040h)`, of the state machine which can be used to set the operating state of the device.

R_PDO1 is evaluated asynchronously, i.e. it is event-driven. R_PDO1 is preset.

R_PDO2

With R_PDO2, the control word and the target position of a motion command, object `target position (607Ah)`, are received for a movement in the operating mode "Profile Position".

R_PDO2 is evaluated asynchronously, i.e. it is event-driven. R_PDO2 is preset.

For details on the SYNC object see chapter "3.4 Synchronization".

R_PDO3

R_PDO3 contains the control word and the target velocity, object `Target velocity (60FFh)`, for the operating mode "Profile Velocity".

R_PDO3 is evaluated asynchronously, i.e. it is event-driven. R_PDO3 is preset.

R_PDO4

R_PDO4 is used to transmit vendor-specific object values. By default, R_PDO4 is empty.

R_PDO4 is evaluated asynchronously, i.e. it is event-driven.

0198441113938, V2.00, 10.2011

The R_PDOs can be used to map various vendor-specific objects by means of PDO mapping.

Transmit PDOs

The objects for T_PDO1, T_PDO2, T_PDO3 and T_PDO4 can be changed by means of PDO mapping.

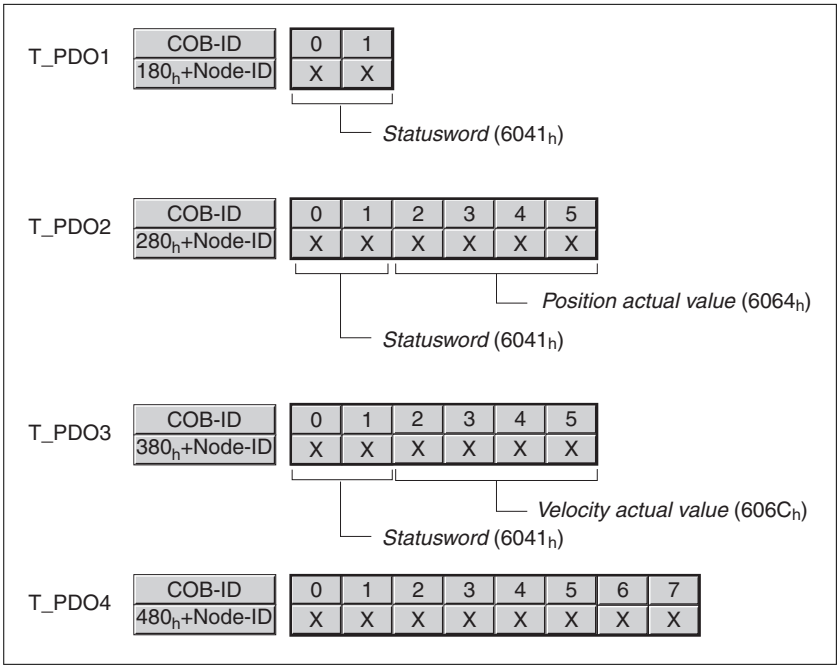


Figure 20: Transmit PDOs

T_PDO1

T_PDO1 contains the status word, object `statusword (6041h)`, of the state machine.

T_PDO1 is transmitted asynchronously and in an event-driven way whenever the status information changes.

T_PDO2

T_PDO2 contains the status word and the actual position of the motor, object `Position actual value (6064h)`, to monitor movements in the operating mode "Profile Position".

T_PDO2 is transmitted after receipt of a SYNC object and in an event-driven way.

T_PDO3

T_PDO3 contains the status word and the actual velocity, object `Velocity actual value (606Ch)`, for monitoring the velocity profile in the operating mode "Profile Velocity".

T_PDO3 is transmitted asynchronously and in an event-driven way whenever the status information changes.

T_PDO4

Vendor-specific object values (for monitoring) are transmitted with T_PDO4. By default, T_PDO4 is empty.

T_PDO4 is transmitted asynchronously and in an event-driven way whenever the data changes.

The T_PDOs can be used to map various vendor-specific objects via PDO mapping.

3.3.3.1 Event masks

The parameters `P3-18` ... `P3-21` are used to specify the objects which are to trigger an event.

0198441113938, V2.00, 10.2011

Example: If P3-18 = 1 only a change to the first PDO object triggers an event. If P3-18 = 15, each change to a PDO object triggers an event.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via fieldbus
P3-18	PDO 1 event mask Changes of values in the object trigger an event: Bit 0: First PDO object Bit 1: Second PDO object Bit 2: Third PDO object Bit 3: Fourth PDO object Changed settings become active immediately.	- 0 1 15	UINT16 R/W - -	CANopen 2312:0h
P3-19	PDO 2 event mask Changes of values in the object trigger an event: Bit 0: First PDO object Bit 1: Second PDO object Bit 2: Third PDO object Bit 3: Fourth PDO object Changed settings become active immediately.	- 0 1 15	UINT16 R/W - -	CANopen 2313:0h
P3-20	PDO 3 event mask Changes of values in the object trigger an event: Bit 0: First PDO object Bit 1: Second PDO object Bit 2: Third PDO object Bit 3: Fourth PDO object Changed settings become active immediately.	- 0 1 15	UINT16 R/W - -	CANopen 2314:0h
P3-21	PDO 4 event mask Changes of values in the object trigger an event: Bit 0: First PDO object Bit 1: Second PDO object Bit 2: Third PDO object Bit 3: Fourth PDO object Changed settings become active immediately.	- 0 15 15	UINT16 R/W - -	CANopen 2315:0h

3.3.4 PDO mapping

Up to 8 bytes of data from different areas of the object dictionary can be transmitted with a PDO message. Mapping of data to a PDO message is referred to as PDO mapping.

Chapter "8 Object dictionary" contains a list of vendor-specific objects that are available for PDO mapping.

The picture below shows the data exchange between PDOs and object dictionary on the basis of two examples of objects in T_PDO4 and R_PDO4 of the PDOs.

0198441113938, V2.00, 10.2011

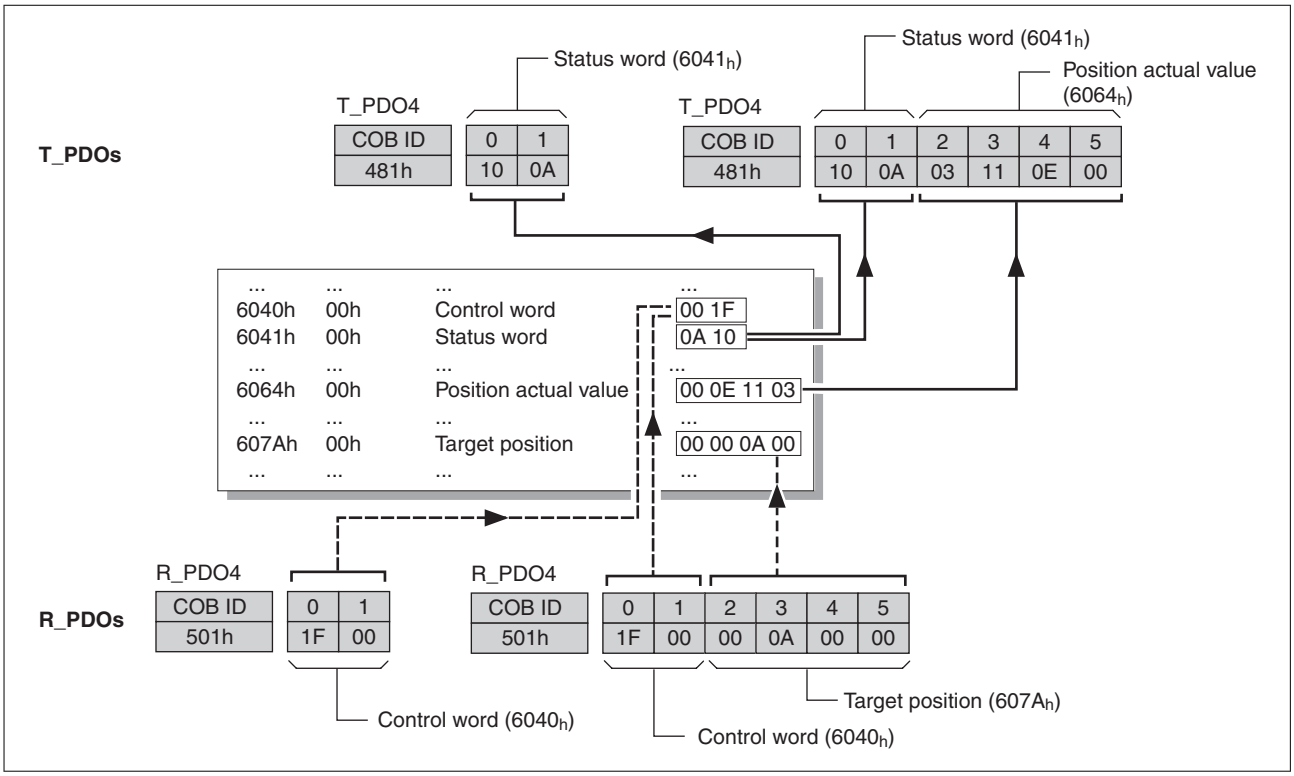


Figure 21: PDO mapping, in this case for a device with node address 1

Dynamic PDO mapping

The device uses dynamic PDO mapping. Dynamic PDO mapping means that objects can be mapped to the corresponding PDO using adjustable settings.

The settings for PDO mapping are defined in an assigned communication object for each PDO.

Object	PDO mapping for	Type
1st receive PDO mapping (1600_h)	R_PDO1	Dynamic
2nd receive PDO mapping (1601_h)	R_PDO2	Dynamic
3rd receive PDO mapping (1602_h)	R_PDO3	Dynamic
4th receive PDO mapping (1603_h)	R_PDO4	Dynamic
1st transmit PDO mapping ($1A00_h$)	T_PDO1	Dynamic
2nd transmit PDO mapping ($1A01_h$)	T_PDO2	Dynamic
3rd transmit PDO mapping ($1A02_h$)	T_PDO3	Dynamic
4th transmit PDO mapping ($1A03_h$)	T_PDO4	Dynamic

Structure of the entries

Up to 8 bytes of 8 different objects can be mapped in a PDO. Each communication 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 uses in the PDO.

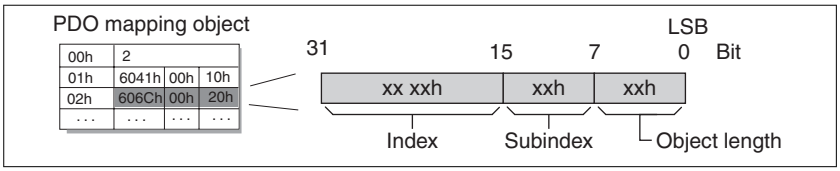


Figure 22: Structure of entries for PDO mapping

0198441113938, V2.00, 10.2011

Subindex 00_h of the communication object contains the number of valid subindex entries.

Object length	Bit value
08_h	8 bits
10_h	16 bits
20_h	32 bits

List of associated objects

0198441113938, V2.00, 10.2011

Index	Subindex	Object	PDO	Data type	Takes effect
1001_h	0	Error Register	T_PDO	UINT8	-
$603F_h$	0	Error Code	T_PDO	UINT16	-
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
6062_h	0	Position demand value	T_PDO	INT32	-
6063_h	0	Position actual internal value	T_PDO	INT32	-
6064_h	0	Position actual Value	T_PDO	INT32	-
6065_h	0	Following error window	R_PDO	UINT32	-
6067_h	0	Position window	R_PDO	UINT32	-
6068_h	0	Position window time	R_PDO	UINT16	Immediately
$606B_h$	0	Velocity demand value	T_PDO	INT32	-
$606C_h$	0	Velocity actual value	T_PDO	INT32	-
$606D_h$	0	Velocity window	R_PDO	UINT16	Immediately
$606E_h$	0	Velocity window time	R_PDO	UINT16	Immediately
$606F_h$	0	Velocity threshold	R_PDO	UINT16	Immediately
6071_h	0	Target Torque	R_PDO	INT16	Immediately
6074_h	0	Torque demand value	T_PDO	INT16	-
6075_h	0	Motor rated current	T_PDO	UINT32	-
6076_h	0	Motor rated torque	T_PDO	UINT32	-
6077_h	0	Torque actual value	T_PDO	INT16	-
6078_h	0	Current actual value	T_PDO	INT16	-
$607A_h$	0	Target position	R_PDO	INT32	Immediately
$607C_h$	0	Home offset	R_PDO	INT32	Next movement
$607D_h$	1	Min position limit	R_PDO	INT32	Immediately
$607D_h$	2	Max position limit	R_PDO	INT32	Immediately
$607F_h$	0	Max profile velocity	R_PDO	UINT32	Immediately
6080_h	0	Max motor speed	R_PDO	UINT32	Immediately
6081_h	0	Profile velocity	R_PDO	UINT32	Next movement
6083_h	0	Profile acceleration	R_PDO	UINT32	Next movement
6084_h	0	Profile deceleration	R_PDO	UINT32	Next movement
6085_h	0	Quick stop deceleration	R_PDO	UINT32	Next movement
6086_h	0	Motion profile type	R_PDO	INT16	Next movement
6087_h	0	Torque slope	R_PDO	UINT32	Immediately
6093_h	1	Numerator (Position factor)	R_PDO	UINT32	Immediately
6093_h	2	Speed constant (Position factor)	R_PDO	UINT32	Immediately
6098_h	0	Homing method	R_PDO	INT8	Next movement
6099_h	1	Speed during search for switch	R_PDO	UINT32	Next movement

0198441113938, V2.00, 10.2011

Index	Subindex	Object	PDO	Data type	Takes effect
6099_h	2	Speed during search for zero	R_PDO	UINT32	Next movement
$609A_h$	0	Homing accelertation	R_PDO	UINT32	Next movement
$60B0_h$	0	Position offset	R_PDO	INT32	Immediately
$60B1_h$	0	Velocity offset	R_PDO	INT32	Immediately
$60B2_h$	0	Torque offset	R_PDO	INT16	Immediately
$60C0_h$	0	Interpolation sub mode select	R_PDO	INT16	Immediately
$60C1_h$	1	Parameter 1 of ip function	R_PDO	UINT16	Immediately
$60C1_h$	2	Parameter 2 of ip function	R_PDO	UINT16	Immediately
$60C1_h$	3	Parameter 3 of ip function	R_PDO	INT16	Immediately
$60C2_h$	1	Interpolation time units	R_PDO	UINT8	Next movement
$60C2_h$	2	Interpolation time index	R_PDO	INT8	Next movement
$60C5_h$	0	Max acceleration	R_PDO	UINT32	Next movement
$60C6_h$	0	Max deceleration	R_PDO	UINT32	Next movement
$60F2_h$	0	Position option code	R_PDO	UINT16	Next movement
$60F4_h$	0	Following error actual value	T_PDO	INT32	-
$60FC_h$	0	Position demand value	T_PDO	INT32	-
$60FF_h$	0	Target velocity	R_PDO	INT32	Immediately
6502_h	0	Supported drive modes	T_PDO	-	-

0198441113938, V2.00, 10.2011

3.4 Synchronization

The synchronization 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 transmitted to all reachable devices by a network device and can be evaluated by the devices that support synchronous PDOs.

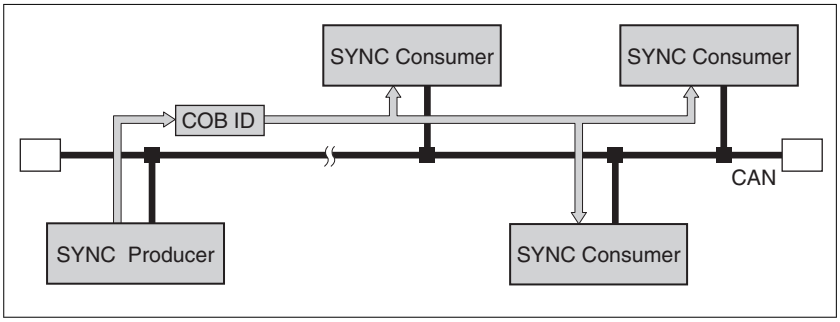


Figure 23: SYNC message

Time values for synchronization

Two time values define the behavior of synchronous data transmission:

- 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 span during which the synchronous PDO messages must be received and transmitted. The time window is set with the object `Synchronous window length (1007h)`.

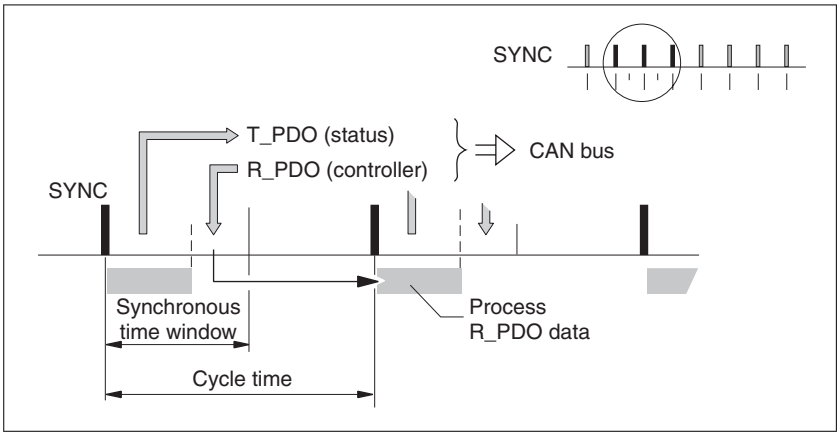


Figure 24: Synchronization times

Synchronous data transmission

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

0198441113938, V2.00, 10.2011

Cyclic ad acyclic data transmission

Synchronous exchange of messages can be cyclic or acyclic.

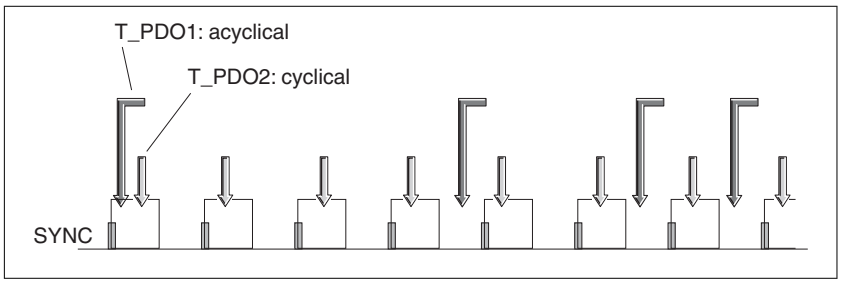


Figure 25: Cyclic and acyclic transmission

In the case of cyclic transmission, PDO messages are exchanged continuously in a specified cycle, for example with each SYNC message.

If a synchronous PDO message is transmitted acyclically, it can be transmitted or received at any time; however, it will not be valid until the next SYNC message.

Cyclic or acyclic behavior of a PDO is specified in the subindex `transmission type (02h)` of the corresponding PDO parameter, for example, in the object `1st receive PDO parameter (1400h: 02h)` for R_PDO1.

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 initialization of the network with the object `COB-ID SYNC Message (1005h)` .

"Start" PDO

With the default settings of the PDOs, R_PDO1 ... R_PDO4 and T_PDO1 ... T_PDO4 are received and transmitted asynchronously. T_PDO2 ... T_PDO3 are transmitted additionally after the event timer has elapsed. The synchronization allows an operating mode to be started simultaneously on multiple devices so that, for example, the feed of a portal drive with several motors can be synchronized.

0198441113938, V2.00, 10.2011

3.5 Emergency service

The Emergency Service signals internal device errors via the CAN bus. The error message is transmitted to the network devices with an EMCY object according to the Consumer-Producer relationship.

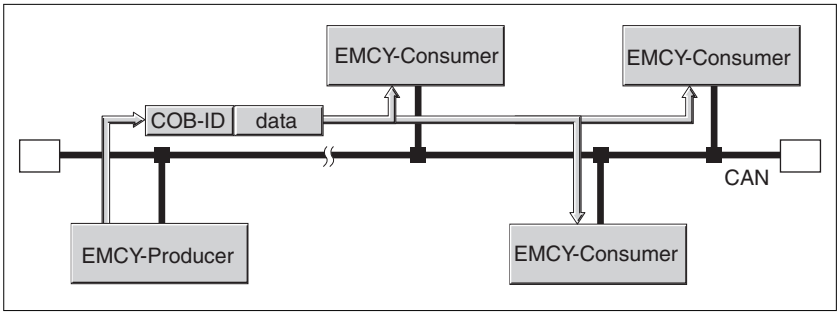


Figure 26: Error message via EMCY objects

Boot-up message

The communication profile DS301, version 3.0, defines an additional task for the EMCY object: sending a boot-up message. A boot-up message informs the network devices that the device that transmitted the message is ready for operation in the CAN network.

The boot-up message is transmitted 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 the operating state **9** Fault as per the CANopen state machine. At the same time, it transmits an EMCY message with error register and error code.

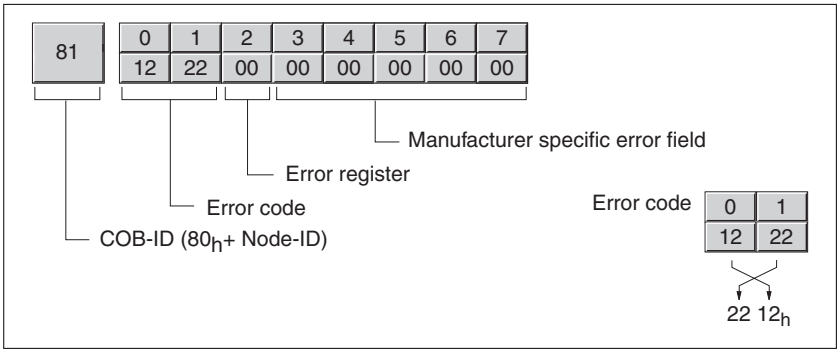


Figure 27: EMCY message

Bytes 0, 1 - Error code, value is also saved in the object `Error code (603Fh)`

Byte 2 - Error register, value is also saved in the object `Error register (1001h)`, see "7.2 Error register"

Bytes 3, 4 - Reserved

Byte 5 - PDO: Number of the PDO

Bytes 6, 7 - Vendor-specific error code

COB ID

The COB ID for each device on the network supporting an EMCY object is determined on the basis of the node address:

COB ID = Function code EMCY object (80_h) + node ID

0198441113938, V2.00, 10.2011

The function code of the COB ID can be changed with the object COB-ID emergency(1014_h).

Error register and error code

The error register contains bit-coded information on the error. Bit 0 remains set as long as an error is active. The remaining bits identify the error type. The exact cause of error can be determined on the basis of the error code. The error code is transmitted in Intel format as a 2 byte value; the bytes must be reversed for evaluation.

See chapter "7 Diagnostics and troubleshooting" for a list of the error messages and error responses by the device as well as remedies.

Error memory

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 5 error messages are stored in the objects P4-00 (2400_h) to P4-04 (2404_h) in the order in which the errors occurred.

0198441113938, V2.00, 10.2011

3.6 Network management services

Network management (NMT) is part of the CANopen communication profile; it is used to initialize the network and the network devices and to start, stop and monitor the network devices during operation on the network.

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

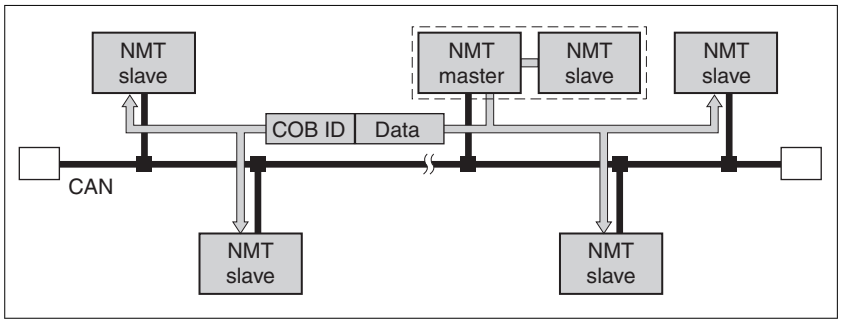


Figure 28: NMT services via the master-slave relationship

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

NMT services

NMT services can be divided into 2 groups:

- Services for device control, to initialize devices for CANopen communication and to control the behavior of devices during operation on the network
- Services:for connection monitoring

3.6.1 NMT services for device control

NMT state machine

The NMT state machine describes the initialization and states of an NMT slave during operation on the network.

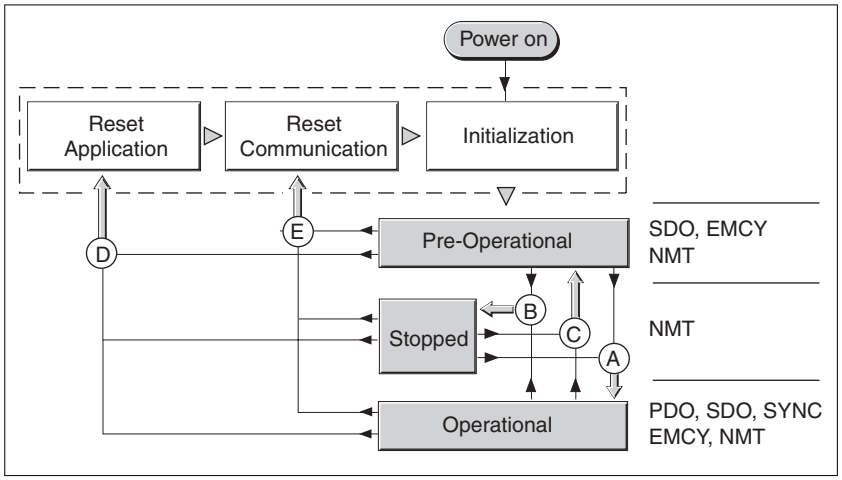


Figure 29: NMT state machine and available communication objects

To the right, the graphic shows the communication objects that can be used in the specific network state.

Initialization

An NMT slave automatically runs through an initialization phase after the supply voltage is switched on (power on) to prepare it for CAN bus

0198441113938, V2.00, 10.2011

operation. On completion of the initialization, the slave switches to the operating state "Pre Operational" and sends a boot-up message. From now on, an NMT master can control the operational behavior of an NMT slave on the network via 5 NMT services, represented in the above illustration by the letters A to E.

NMT service	Transition	Meaning
Start remote node (Start network node)	A	Transition to operating state "Operational" Start normal operation on the network
Stop remote node (Stop network node)	B	Transition to operating state "Stopped" Stops communication of the network device on the network. If connection monitoring is active, it remains on. If the power stage is enabled (operating state "Operation Enabled" or "Quick Stop"), an error of error class 2 is triggered. The motor is stopped and the power stage disabled.
Enter Pre-Operational (Transition to "Pre-Operational")	C	Transition to operating state "Pre-Operational" The communication objects except for PDOs can be used. The operating state "Pre-Operational" can be used for configuration via SDOs: - PDO mapping - Start of synchronization - Start of connection monitoring
Reset node (Reset node)	D	Transition to operating state "Reset application" Load stored data of the device profiles and automatically switch via operating state "Reset communication" to "Pre-Operational".
Reset communication (Reset communication data)	E	Transition to operating state "Reset communication" Load stored data of the communication profile and automatically transition to operating state "Pre-Operational". If the power stage is enabled (operating state "Operation Enabled" or "Quick Stop"), an error of error class 2 is triggered. The motor is stopped and the power stage disabled.

Persistent data memory

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

NMT message

The NMT services for device control are transmitted as unconfirmed messages with the COB ID = 0 . By default, they have the highest priority on the CAN bus.

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

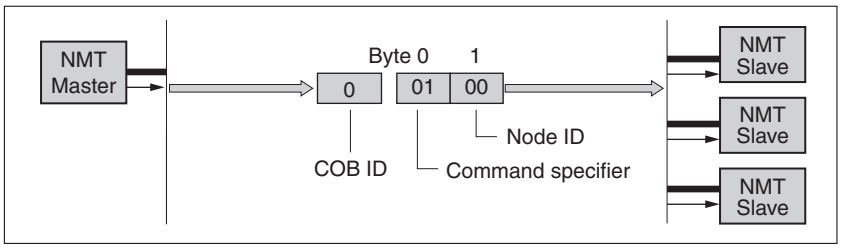


Figure 30: NMT message

0198441113938, V2.00, 10.2011

The first byte, the "Command specifier", indicates the NMT service used.

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 recipient of an NMT message with a node address between 1 and 127 ($7F_h$). A message with node address "0" is broadcast to all reachable NMT slaves.

3.6.2 NMT services for connection monitoring

Connection monitoring monitors the communication status of network devices.

3 NMT services for connection monitoring are available:

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

3.6.2.1 Node guarding / Life guarding

COB ID

The communication object `NMT error control (700h+node-Id)` is used for connection monitoring. The COB ID for each NMT slave is determined on the basis of the node address:

COB ID = function code `NMTerror control (700h) + node-Id`.

Structure of the NMT message

After a request from the NMT master, the NMT slave responds with one data byte.

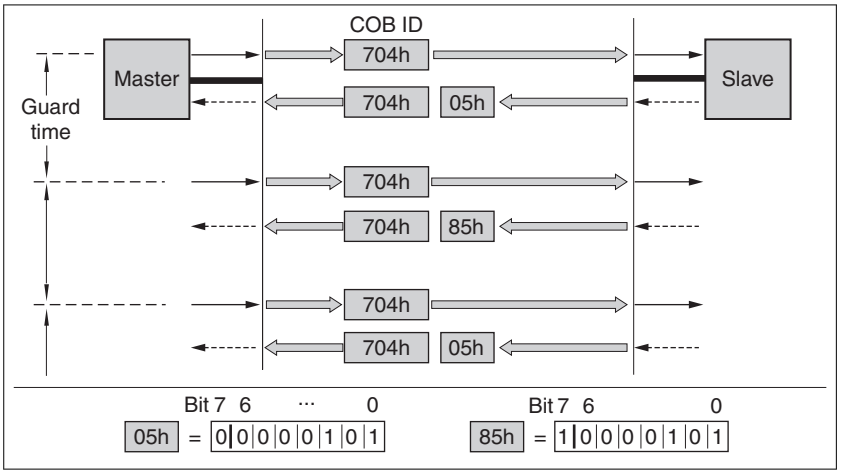


Figure 31: Acknowledgement of the NMT slave

Bits 0 to 6 identify the NMT state of the slave:

0198441113938, V2.00, 10.2011

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

After each "guard time" interval, bit 7 switches toggles between "0" and "1", so the NMT master can detect and ignore a second response within the "guard time" interval. The first request when connection monitoring is started begins with bit 7 = 0.

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

Connection monitoring remains active in the NMT state "Stopped".

Configuration

Node Guarding/Life Guarding is configured via:

- Guard time ($100C_h$)
- Life time factor ($100D_h$)

Connection error

The NMT master signals a connection error to the master program if:

- the slave does not respond within the "guard time" period
- the NMT state of the slave has changed without a request by the NMT master.

Figure 32 shows an error message after the end of the third cycle because of a missing response from an NMT slave.

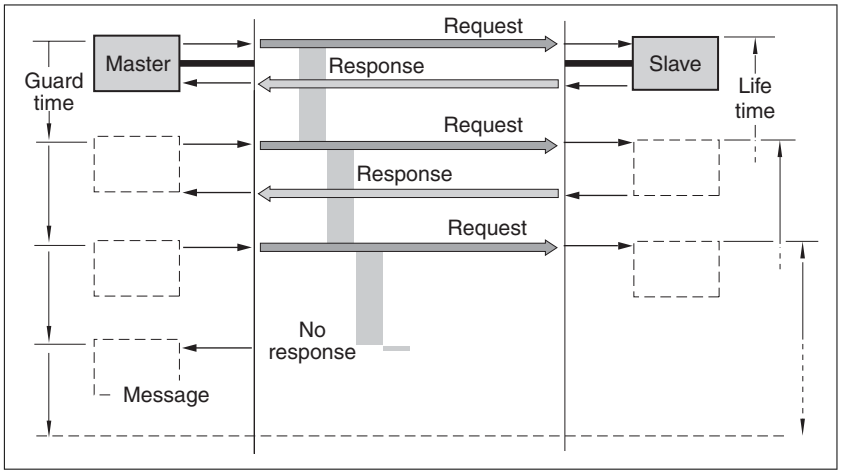


Figure 32: "Node Guarding" and "Life Guarding" with time intervals

0198441113938, V2.00, 10.2011

3.6.2.2 Heartbeat

The optional Heartbeat protocol replaces the node guarding/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 (1017h)`. One or several consumers can receive this message. `Producer heartbeat time (1016 h) = 0` deactivates heartbeat monitoring.

The relationship between producer and consumer can be configured with objects. If a consumer does not receive a signal within the period of time set with `Consumer heartbeat time (1016h)`, it generates an error message (heartbeat event). `Consumer heartbeat time (1016h) = 0` deactivates monitoring by a consumer.

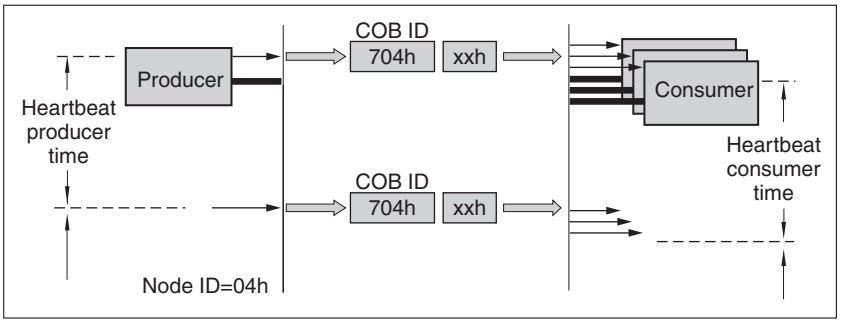


Figure 33: "Heartbeat" monitoring

Data byte for NMT state 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 increments of 1 ms steps; the values for the consumer must not be less than the values 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 already active during the NMT state transition to "Pre-Operational", "Heartbeat" monitoring starts with sending of the boot-up message. The boot-up message is a Heartbeat message with one data byte 00_h.

Devices can monitor each other via "Heartbeat" messages. They assume the function of consumer and producer at the same time.

0198441113938, V2.00, 10.2011

4 Installation

4

⚠ WARNING

SIGNAL AND DEVICE INTERFERENCE

Signal interference can cause unexpected responses of the device.

- Install the wiring in accordance with the EMC requirements.
- Verify compliance with the EMC requirements.

Failure to follow these instructions can result in death, serious injury or equipment damage.

For information on installation of the device and connecting the device to the fieldbus see the product manual.

0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

5 Commissioning

5

⚠ WARNING

LOSS OF CONTROL

The product is unable to detect an interruption of the network link if connection monitoring is not active.

- Verify that connection monitoring is on.
- The shorter the time for monitoring, the faster the detection of the interruption.

Failure to follow these instructions can result in death, serious injury or equipment damage.

⚠ WARNING

UNINTENDED OPERATION

- Do not write values to reserved parameters.
- Do not write values to parameters unless you fully understand the function.
- Run initial tests without coupled loads.
- Verify the use of the word sequence with fieldbus communication.
- Do not establish a fieldbus connection unless you have fully understood the communication principles.
- Only start the system if there are no persons or obstructions in the hazardous area.

Failure to follow these instructions can result in death, serious injury or equipment damage.

Using the library considerably facilitates controlling the device. The library is available for download from the Internet.
http://www.schneider-electric.com

5.1 Commissioning the device

For installation in the network, the device must first be properly installed (mechanically and electrically) and commissioned.

▸ Commission the device as per product manual.

5.2 Address and baud rate

Up to 64 devices can be addressed in a CAN bus network segment and up to 127 devices in the extended network. Each device is identified by a unique address. The default node address for a device is 0.

0198441113938, V2.00, 10.2011

The default baud rate set in the parameter P3-01 is 250 kBaud.

Each device must be assigned its own node address, i.e. any given node address may be assigned only once in the network.

Setting address and baud rate

After the initialization, the CAN interface must be configured. You must assign a unique network address (node address) to each device. The transmission rate (baud rate) must be the same for all devices in the network.

- Set the transmission rate in the parameter P3-01 to meet the requirements of your network.
- Enter the network address. The network address is stored in the parameter P3-05.

The settings are valid for CANopen and for CANmotion.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via field-bus
P3-01	CANopen baud rate **0:** 125 kBaud **1:** 250 kBaud **2:** 500 kBaud **4:** : 1 Mbaud Changed settings become active the next time the product is switched on.	- 0 1 4	UINT16 R/W per. -	CANopen 2301_h
P3-05	CANopen address hexadecimal (node number) Changed settings become active the next time the product is switched on.	- 00 - 7F	UINT16 R/W per. -	CANopen 2305_h

0198441113938, V2.00, 10.2011

6 Operation

6

⚠ WARNING

UNINTENDED OPERATION

- Do not write values to reserved parameters.
- Do not write values to parameters unless you fully understand the function.
- Run initial tests without coupled loads.
- Verify the use of the word sequence with fieldbus communication.
- Do not establish a fieldbus connection unless you have fully understood the communication principles.
- Only start the system if there are no persons or obstructions in the hazardous area.

Failure to follow these instructions can result in death, serious injury or equipment damage.

The chapter "Operation" describes the basic operating states, operating modes and functions of the device.

Using the library considerably facilitates controlling the device. The library is available for download from the Internet.
http://www.schneider-electric.com

0198441113938, V2.00, 10.2011

6.1 Indication of the operating state

After switching on and when an operating mode is started, the product goes through a number of operating states. The operating states are internally monitored and influenced by monitoring functions

The parameter `StatusWord` provides information on the operating state of the device and the processing status of the operating mode.

Graphical representation The state diagram is represented as a flow chart.

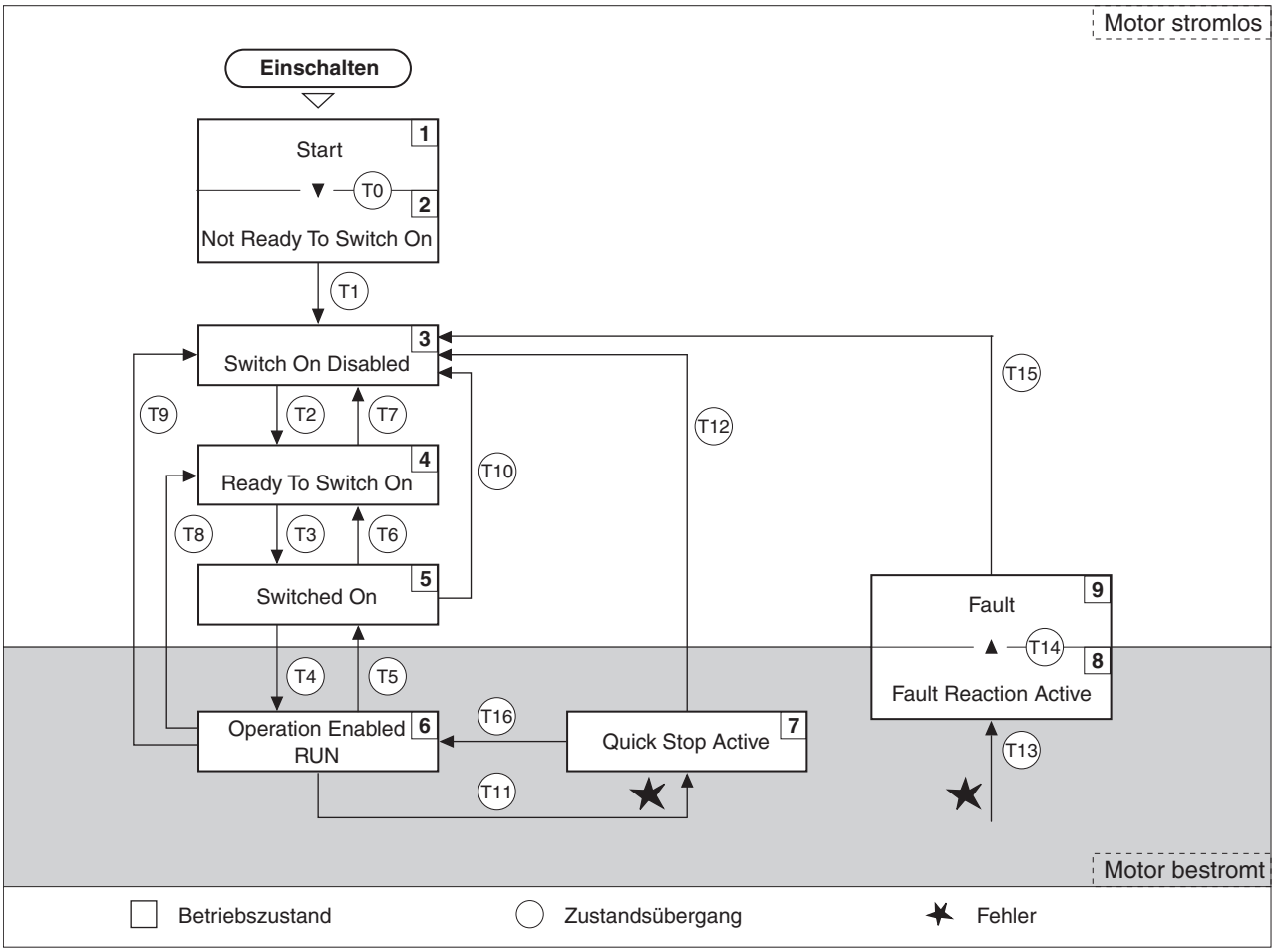


Figure 34: State diagram

0198441113938, V2.00, 10.2011

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via fieldbus
StausWord	DriveCom status word Bit assignments: Bits 0 ... 3: Status bits Bit 4: Voltage enabled Bits 5 ... 6: Status bits Bit 7: Warning Bit 8: Reserved Bit 9: Remote Bit 10: Target reached Bit 11: Assignment can be set via parameter DS402intLim Bit 12: Operating mode-specific Bit 13: Operating mode-specific Bit 14: LimP Bit 15: LimN	-	UINT16 R/- - -	CANopen 6041:0$_h$

Bits 0, 1, 2, 3, 5 and 6 Bits 0, 1, 2, 3, 5 and 6 of the StausWord parameter provide information on the operating state.

Operating state	Bit 6 Switch On Disabled	Bit 5 Quick Stop	Bit 3 Fault	Bit 2 Operation Enabled	Bit 1 Switch On	Bit 0 Ready To Switch On
2 Not Ready To Switch On	0	X	0	0	0	0
3 Switch On Disabled	1	X	0	0	0	0
4 Ready To Switch On	0	1	0	0	0	1
5 Switched On	0	1	0	0	1	1
6 Operation Enabled	0	1	0	1	1	1
7 Quick Stop Active	0	0	0	1	1	1
8 Fault Reaction Active	0	X	1	1	1	1
9 Fault	0	X	1	0	0	0

Bit 4 Bit 4=1 indicates whether the DC bus voltage is correct. If the voltage is missing or is too low, the device does not transition from operating state 3 to operating state 4.

Bit 7 Bit 7 is 1 if parameter _WarnActive contains a warning message. The movement is not interrupted. The bit remains set as long as a warning message is contained in parameter _WarnActive. The bit remains set for at least 100ms, even if a warning message is active for a shorter time. The bit is reset immediately in the case of a "Fault Reset".

Bit 8 Reserved.

Bit 9 If bit 9 is set, the device carries out commands via the fieldbus. If Bit 9 is reset, the device is controlled via a different interface. In such a case, it is still possible to read or write parameters via the fieldbus.

Bit 10 Bit 10 is used for monitoring the current operating mode. Details can be found in the chapters on the individual operating modes.

Bit 11 Reserved.

Bit 12 Bit 12 is used for monitoring the current operating mode. Details can be found in the chapters on the individual operating modes.

0198441113938, V2.00, 10.2011

Bit 13 Bit 13 only becomes "1" in the case of an error which needs to be remedied prior to further processing.

Bit 14 Bit 14 changes to "1" when the limit switch LimP is triggered.

Bit 15 Bit 15 changes to "1" when the limit switch LimN is triggered.

0198441113938, V2.00, 10.2011

6.2 Changing the operating state

It is possible to switch between operating states via the parameter `ControlWord`.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via fieldbus
`ControlWord`	control word Refer to chapter Operation, Operating States, for bit coding information. Bit 0: Switch on Bit 1: Enable Voltage Bit 2: Quick Stop Bit 3: Enable Operation Bit 4..6: Operating mode specific Bit 7: Fault Reset Bit 8: Halt Bit 9: Reserved Bits 10 ... 15: Reserved (must be 0) Changed settings become active immediately.	-	UINT16 R/W - -	CANopen 6040:0_h

Bits 0, 1, 2, 3 and 7 Bits 0, 1, 2, 3 and 7 of the parameter `ControlWord` allow you to switch between the operating states.

Fieldbus command	State transitions	State transition to	Bit 7, Fault Reset	Bit 3, Enable operation	Bit 2, Quick Stop	Bit 1, Enable Voltage	Bit 0, Switch On
Shutdown	T2, T6, T8	**4**: Ready To Switch On	X	X	1	1	0
Switch On	T3	**5** Switched On	X	X	1	1	1
Disable Voltage	T7, T9, T10, T12	**3** Switch On Disabled	X	X	X	0	X
Quick Stop	T7, T10 T11	**3** Switch On Disabled **7** Quick Stop Active	X	X	0	1	X
Disable Operation	T5	**5** Switched On	X	0	1	1	1
Enable Operation	T4, T16	**6** Operation Enabled	X	1	1	1	1
Fault Reset	T15	**3** Switch On Disabled	0->1	X	X	X	X

Bits 4 ... 6 Bits 4 to 6 are used for the operating mode-specific settings. Details can be found in the descriptions of the individual operating modes in this chapter.

Bit 8 A "Halt" can be triggered with bit 8=1.

Bits 9 ... 15 Reserved.

6.3 Starting and changing an operating mode

The parameter `Mode of operation` (6060_h) is used to set the desired operating mode.

0198441113938, V2.00, 10.2011

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via field-bus
`Mode of operation`	Operating mode **-1 / Jog**: Jog **0 / Reserved**: Reserved **1 / Profile Position**: Profile Position **3 / Profile Velocity**: Profile Velocity **4 / Profile Torque**: Profile Torque **6 / Homing**: Homing **7 / Interpolated Position**: Interpolated Position **8 / Cyclic Synchronous Position:** Cyclic Synchronous Position Changed settings become active immediately.	- -1 - 8	INT8 R/W - -	CANopen 6060:0$_h$

- Set the operating mode with the parameter `Mode of operation Display`.

The parameter `Mode of operation Display` can be used to read the current operating mode.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via field-bus
`Mode of operation Display`	Operating mode **-1 / Jog**: Jog **0 / Reserved**: Reserved **1 / Profile Position**: Profile Position **3 / Profile Velocity**: Profile Velocity **4 / Profile Torque**: Profile Torque **6 / Homing**: Homing **7 / Interpolated Position**: Interpolated Position **8 / Cyclic Synchronous Position:** Cyclic Synchronous Position Changed settings become active immediately.	- -1 - 8	INT8 R/W - -	CANopen 6061:0$_h$

0198441113938, V2.00, 10.2011

6.4 Operating mode Profile Position

Description

In the operating mode Profile Position, a movement to a desired target position is performed.

Definition of the unit "Pulse"

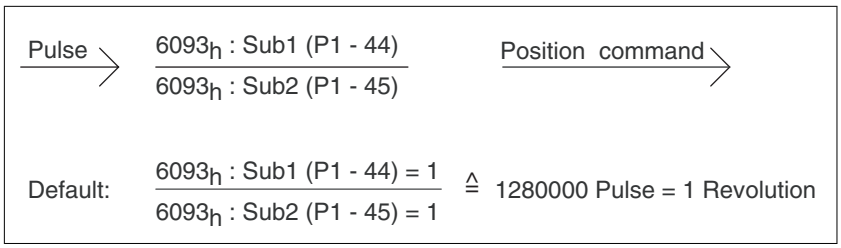


Figure 35: Definition of Pulse

Procedure

- Set [`Mode of operation`:(6060_h)] to operating mode Profile Position (1).
- Set [`Target Position`:($607A_h$)] to the target position (unit = pulse).
- Set [`Profile velocity`:(6081_h)] to Profile Velocity (unit = pulses per second).
- Set [`Profile acceleration`:(6083_h)] to the value for the acceleration ramp (unit = time in milliseconds from 0 to 3000 min^{-1}).
- Set [`Profile decceleration`:(6084_h)] to the value for the deceleration ramp (unit = time in milliseconds from 3000 to 0 min^{-1}).
- Set [`ControlWord`:(6040_h)] to start the movement.
- Query [`Position actual value`:(6064_h)] to get the actual position of the motor.
- Query [`StatusWord`:(6041_h)] to get the current status of following error, set-point acknowledge and target reached.

0198441113938, V2.00, 10.2011

Optional Additional information on the operating mode Profile Position

- Query [`Position demand value`:(6062_h)] to get the internal reference value (unit = pulse).
- Query [`Position actual value`:(6063_h)] to get the actual position value (unit = increments).

Following error:

- Set [`Following error window`:(6065_h)] to the permissible following error (unit = pulse).
- Query [`Following error actual value`:($60F4_h$)] to get the current following error (unit = pulse).

Standstill window:

- Set [`Position window`:(6067_h)] to the value for the standstill window. If the difference between the target position and the current motor position remains in the standstill window for the time `Position window time`:(6065_h), the target position is considered to have been reached (unit = pulse).
- Set [`Position window time`:(6068_h)] to the value for the standstill window. If the difference between the target position and the current motor position remains in the standstill window for the time `Position window time`:(6065_h), the target position is considered to have been reached (unit = pulse).

List of associated objects

Index	Subindex	Object	PDO	Data type	Takes effect
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
6062_h	0	Position demand value	T_PDO	INT32	-
6063_h	0	Position actual Value	T_PDO	INT32	-
6064_h	0	Position actual Value	T_PDO	INT32	-
6065_h	0	Following error window	R_PDO	UINT32	-
6067_h	0	Position window	R_PDO	UINT32	-
6068_h	0	Position window time	R_PDO	UINT16	Immediately
6081_h	0	Profile velocity	R_PDO	UINT32	Next movement
6083_h	0	Profile acceleration	R_PDO	UINT32	Next movement
6084_h	0	Profile deceleration	R_PDO	UINT32	Next movement
6093_h	1	Numerator (Position factor)	R_PDO	UINT32	Immediately
6093_h	2	Speed constant (Position factor)	R_PDO	UINT32	Immediately
$60F2_h$	0	Position option code	R_PDO	UINT16	Next movement
$60F4_h$	0	Following error actual value	T_PDO	INT32	-
$60FC_h$	0	Position demand value	T_PDO	INT32	-

0198441113938, V2.00, 10.2011

6.4.1 Example: Profile Position

Starting the operating mode

The operating mode must be set in the parameter `Mode of operation` (6060_h. Writing the parameter value activates the operating mode.

The movement is started via the control word.

Control word

The bits 4 ... 6 and the bit 8 in the parameter `ControlWord`(6040_h) start a movement.

Bit 5: Change setpoint immediately	Bit 4: New target value	Meaning
0	0->1	Starts a movement to a target position. Target values transmitted during a movement become immediately effective and are executed at the target. The movement is stopped at the current target position. [1)]
1	0->1	Starts a movement to a target position. Target values transmitted during a movement become immediately effective and are executed at the target. The movement is not stopped at the current target position. [1)]

1) Target values include target position, target velocity, acceleration and deceleration.

Parameter value	Meaning
Bit 6: Absolute / relative	0: Absolute movement 1: Relative movement
Bit 8: Halt	Stop movement with "Halt"

Terminating the operating mode

The operating mode is terminated when the motor is at a standstill and one of the following conditions is met:

- Target position reached
- Stop caused by "Halt" or "Quick Stop"
- Stop caused by an error

Status word

Information on the current movement is available via bits 10 and 12 ... 15 in the parameter `StatusWord`:(6041_h).

Parameter value	Meaning
Bit 10: Target reached	0: Target position not reached 1: Target position reached
Bit 12: Target value acknowledge	0: New position possible 1: New target position accepted
Bit 13: x_err	1: following error
Bit 14: LimP	1: Positive limit switch triggered
Bit 15: LimN	1: Negative limit switch triggered

0198441113938, V2.00, 10.2011

6.4.1.1 Example Node address 1

Work step COB ID / data	Object Value
▸ Activate R_PDO2 601 / 23 01 14 01 01 03 00 04 ◃ 581 / 60 01 14 01 00 00 00 00	1401:1$_h$ 0400 0301$_h$
▸ Activate T_PDO2 601 / 23 01 18 01 81 02 00 04 ◃ 581 / 60 01 18 01 00 00 00 00	1801:1$_h$ 0400 0281$_h$
▸ Set acceleration to 2000 601 / 23 83 60 00 D0 07 00 00 ◃ 581 / 60 83 60 00 00 00 00 00	6083$_h$ 0000 07D0$_h$
▸ Set deceleration to 4000 601 / 23 84 60 00 A0 0F 00 00 ◃ 581 / 60 84 60 00 00 00 00 00	6084$_h$ 0000 0FA0$_h$
▸ Set target velocity to 4000 601 / 23 81 60 00 A0 0F 00 00 ◃ 581 / 60 81 60 00 00 00 00 00	6081$_h$ 0000 0FA0$_h$
▸ NMT Start remote node 0 / 01 00 ◃ T_PDO2 with status word 281 / 31 66 00 00 00 00	
▸ Enable power stage with R_PDO2 301 / 00 00 00 00 00 00 301 / 06 00 00 00 00 00 301 / 0F 00 00 00 00 00 ◃ T_PDO2 (operating state: **6** Operation Enabled) 281 / 37 42 00 00 00 00	
▸ Starting the operating mode 601 / 2F 60 60 00 01 00 00 00 ◃ 581 / 60 60 60 00 00 00 00 00	6060$_h$ 01$_h$
▸ Check operating mode [1)] 601 / 40 61 60 00 00 00 00 00 ◃ Operating mode active 581 / 4F 61 60 00 01 61 01 00	6061$_h$ 01$_h$
▸ R_PDO2: Start relative movement with NewSetpoint=1 301 / 5F 00 30 75 00 00 ◃ T_PDO2 with status word and position actual value 281 / 37 12 00 00 00 00 ◃ Target position reached 281 / 37 56 30 75 00 00	
▸ R_PDO2: NewSetpoint=0 301 / 4F 00 30 75 00 00	

1) The operating mode must be checked until the device has activated the specified operating mode.

0198441113938, V2.00, 10.2011

6.5 Operating mode Interpolated Position

Description

In the operating mode Interpolated Position, movements are made to cyclically set reference positions.

- The master cyclically send a SYNC frame (0x80).
- With each PDO, the master sends the next reference position Xi, the difference X_i and the control word to the slave.
- While the next SYNC frame is received, the device interpolates from X_{i-1} to X_i.
- There is no input data buffer since this would cause a delay.

Extrapolation, jitter compensation

- If a SYNC object is received with a delay, the last acceleration is used to extrapolate the velocity and the position.
- If the SYNC object is not received for 2 cycles, the device stops and generates an error message.

PDO Rx/Tx mapping example

- PDOs from master to slave

  32 bit reference position (increments)

  16 bit standstill window (increments)

  $\Delta X_i = (X_{i+1}-X_{i-1})/2$ (applies to the velocity as well)

  16 bit control word

PDO from producer to consumer (each PDO contains the 8 bytes as described below (xxx ??? Grafik).

Procedure

- ▸ Set [`Mode of operation`:(6060_h)] to operating mode Interpolated Position (7).
- ▸ Set [`Interpolated sub mode select`:($60C0_h$)].
- ▸ Set [`Interpolated time period`:($60C2_h$)] to the cycle time of the SYNC signal.

  ($60C2_h$ Sub-1) Interpolation time units. The value range is 1ms to 20ms.

  ($60C2_h$ Sub-2) Interpolation time index. The default value is -3 which corresponds to a time unit of 10^{-3} seconds.
- ▸ PDO and SDO settings.

  Set 1400_h Sub-1 for PDO RxCobId.

  Set 1400_h Sub-2 for PDO receive type (normally 01_h).

  If these settings are used, the master must send a SYNC signal and PDO data each cycle.
- ▸ Drive PDO Rx:

  $60C1_h$ Sub-1 for Pos Cmd (low word).

  6040_h Sub-0 for control word.
- ▸ The content of the PDO transmit data can be adapted to the requirements of the master.
- ▸ The master sends NMT to start the operating mode.

NOTE: Due to the different cycle times of the SYNC signals, it may be necessary to adjust the setting of parameter P3-09. xxx yyy

0198441113938, V2.00, 10.2011

Starting the operating mode

An initialization sequence must be written to start the operating mode. After the initialization sequence, the operating mode can be started via the control word.

NOTE: In the operating mode Interpolated Position, the scaling factor of the user-defined unit usr_p must be set to 1 min^{-1}/131072. Among other things, this scaling factor is written by means of the initialization sequence.

Index	Subindex	Length in bytes	Value	Meaning
1400_h	1_h	4	80000200_h + node id	Deactivate R_PDO1
1800_h	1_h	4	80000180_h + node id	Deactivate T_PDO1
1401_h	1_h	4	00000300_h + node id	Activate R_PDO2
1801_h	1_h	4	00000280_h + node id	Activate T_PDO2
1402_h	1_h	4	80000400_h + node id	Deactivate R_PDO3
1802_h	1_h	4	80000380_h + node id	Deactivate R_PDO3
1403_h	1_h	4	80000500_h + node id	Deactivate R_PDO4
1803_h	1_h	4	80000480_h + node id	Deactivate R_PDO4
1400_h	2	1	1_h	Activate cyclic transmission of R_PDO1
1800_h	2_h	1	1_h	Activate cyclic transmission of T_PDO1
6040_h	0_h	2	0_h	Control word = 0
6040_h	0_h	2	80_h	Perform Fault Reset
1600_h	0_h	1	0_h	Change PDO mapping for R_PDO1
1600_h	1_h	4	60400010_h	Map control word
1600_h	2_h	4	$60C10120_h$	Map reference position for Interpolated Position
1600_h	0_h	1	2_h	Finalize mapping for R_PDO2
$1A00_h$	0_h	1	0_h	Change PDO mapping for T_PDO2
$1A00_h$	1_h	4	60410010_h	Map status word
$1A00_h$	2_h	4	60640020_h	Map Position actual Value
$1A00_h$	0_h	1	2_h	Finalize mapping for T_PDO2
$212C_h$	0_h	4	1_h	Position scaling: Denominator
$212D_h$	0_h	4	1_h	Position scaling: Numerator
6060_h	0_h	1	7_h	Select operating mode Interpolated Position
$60C2_h$	1_h	1	2	Cycle time 2 ms (example)

List of associated objects

0198441113938, V2.00, 10.2011

Index	Subindex	Object	PDO	Data type	Takes effect
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
6093_h	1	Numerator (Position factor)	R_PDO	UINT32	Immediately
6093_h	2	Speed constant (Position factor)	R_PDO	UINT32	Immediately
$60C0_h$	0	Interpolation sub mode select	R_PDO	INT16	Immediately
$60C1_h$	1	Parameter 1 of ip function	R_PDO	UINT16	Immediately
$60C1_h$	2	Parameter 2 of ip function	R_PDO	UINT16	Immediately
$60C1_h$	3	Parameter 3 of ip function	R_PDO	INT16	Immediately

0198441113938, V2.00, 10.2011

6.6 Operating mode Homing

Description

In the operating mode Homing, a movement is performed to a defined position. This position is defined as the reference point.

Procedure

- Set [`Mode of operation`:(6060_h)] to operating mode Homing (6).
- Set [`Home offset`:($607C_h$)].
- Set [`Home method`:(6098_h)], the value range is 1 to 35 and specifies the different homing methods.
- Set [`Home speeds`:(6099_h Sub-1)] to the value for velocity for the search for the limit switches (unit = min^{-1}).
- Set [`Home speeds`:(6099_h Sub-2)] to the value for velocity for the search for the index pulse (unit = min^{-1}).
- Set [`Home acceleration`:(6099_h Sub-2)] to the value for the acceleration ramp (unit = milliseconds form 0 to 3000 min^{-1}).
- Set [`Controlword`:(6040_h)] to start the operating mode.
- Start Homing
- Query [`Statusword`:(6041_h)] to get the device status.

List of associated objects

Index	Subindex	Object	PDO	Data type	Takes effect
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
$607C_h$	0	Home offset	R_PDO	INT32	Next movement
6093_h	1	Numerator (Position factor)	R_PDO	UINT32	Immediately
6093_h	2	Speed constant (Position factor)	R_PDO	UINT32	Immediately
6098_h	0	Homing method	R_PDO	INT8	Next movement
6099_h	1	Speed during search for switch	R_PDO	UINT32	Next movement
6099_h	2	Speed during search for zero	R_PDO	UINT32	Next movement
$609A_h$	0	Homing acceleration	R_PDO	UINT32	Next movement

6.6.1 Example: Homing

Starting the operating mode

The operating mode must be set in the parameter `Mode of operation`:(6060_h). Writing the parameter value activates the operating mode.

The movement is started via the control word.

0198441113938, V2.00, 10.2011

Control word

Bits 4 in the parameter `ControlWord`(6040_h) starts a movement, bit 8 terminates the movement.

Parameter value	Meaning
Bit 4: Homing operation start	Start Homing
Bit 5: Reserved	Not relevant for this operating mode
Bit 6: Reserved	Not relevant for this operating mode
Bit 8: Halt	Stop movement with "Halt"

Terminating the operating mode

The operating mode is terminated when the motor is at a standstill and one of the following conditions is met:

- Homing successful
- Stop caused by "Halt" or "Quick Stop"
- Stop caused by an error

Status word

Information on the current movement is available via bits 10 and 12 ... 15 in the parameter `StatusWord`:(6041_h).

Parameter value	Meaning
Bit 10: Target reached	0: Homing not completed 1: Homing completed
Bit 12: Homing attained	1: Homing successfully completed
Bit 13: x_err	1: Homing error
Bit 14: LimP	1: Positive limit switch triggered
Bit 15: LimN	1: Negative limit switch triggered

0198441113938, V2.00, 10.2011

6.6.1.1 Example Node address 1

Work step COB ID / data	Object Value
▸ Velocity for searching the limit switch to 100 601 / 23 99 60 01 64 00 00 00 ◃ 581 / 60 99 60 01 00 00 00 00	6099:1$_h$ 0000 0064$_h$
▸ Velocity for moving away from switch to 10 601 / 23 99 60 02 0A 00 00 00 ◃ 581 / 60 99 60 02 00 00 00 00	6099:2$_h$ 0000 000A$_h$
▸ NMT Start remote node 0 / 01 00 ◃ T_PDO1 with status word 181 / 31 62	
▸ Enable power stage with R_PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 ◃ T_PDO1 (operating state: **6** Operation Enabled) 181 / 37 42	
▸ Starting the operating mode 601 / 2F 60 60 00 06 00 00 00 ◃ 581 / 60 60 60 00 00 00 00 00	6060$_h$ 06$_h$
▸ Check operating mode [1)] 601 / 40 61 60 00 00 00 00 00 ◃ Operating mode active 581 / 4F 61 60 00 06 61 01 00	6061$_h$ 06$_h$
▸ Select method 17 601 / 2F 98 60 00 11 00 00 00 ◃ 581 / 60 98 60 00 00 00 00 00	6098$_h$ 11$_h$
▸ Start reference movement (Homing operation start) 201 / 1F 00 ◃ T_PDO1 reference movement active 181 / 37 02 ◃ T_PDO1 reference movement terminated 181 / 37 D6	

1) The operating mode must be checked until the device has activated the specified operating mode.

0198441113938, V2.00, 10.2011

6.7 Operating mode Profile Velocity

Description

In the operating mode Profile Velocity, a movement is made with a desired target velocity.

Procedure

- Set [`Mode of operation`:(6060_h)] to operating mode Profile Velocity (3).
- Set [`Controlword`:(6040_h)] to start the operating mode.
- Set [`Profile acceleration`:(6083_h)] to the value for the acceleration ramp (unit = time in milliseconds from 0 to 3000 min^{-1}).
- Set [`Profile decceleration`:(6084_h)] to the value for the deceleration ramp (unit = time in milliseconds from 3000 to 0 min^{-1}).
- Set [`Target velocity`:($60FF_h$)] to the target velocity (unit = 0.1 min^{-1}). If the power stage is enabled, the new target velocity will become active immediately and the movement will start. The value is reset to zero if the operating mode is changed, the power stage is disabled or a Quick Stop is triggered.
- Query [`Statusword`:(6041_h)] to get the device status.

Optional:

- Query [`Velocity demand value`:($606B_h$)] to get the reference velocity (unit = 0.1 min^{-1}).
- Query [`Velocity actual value`:($60C3_h$)] to get the actual velocity (unit = 0.1 min^{-1}).
- Set [`Velocity window`:($606D_h$)] to the value of the velocity window (unit = 0.1 min^{-1}).
- Set [`Velocity window time`:($606E_h$)] to the duration in the velocity window required to consider the velocity to have been reached unit = milliseconds).
- Query [`Velocity threshold`:($60F4_h$)] to set the standstill window (unit = 0.1 min^{-1}).

List of associated objects

Index	Subindex	Object	PDO	Data type	Takes effect
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
$606B_h$	0	Velocity demand value	T_PDO	INT32	-
$606C_h$	0	Velocity actual value	T_PDO	INT32	-
$606D_h$	0	Velocity window	R_PDO	UINT16	Immediately
$606E_h$	0	Velocity window time	R_PDO	UINT16	Immediately
$606F_h$	0	Velocity threshold	R_PDO	UINT16	Immediately
$60FF_h$	0	Target velocity	R_PDO	INT32	Immediately

0198441113938, V2.00, 10.2011

6.7.1 Example: Profile Velocity

Starting the operating mode

The operating mode must be set in the parameter `Mode of operation`: (6060_h). Writing the parameter value activates the operating mode.

The parameter `Target velocity`: ($60FF_h$) starts the movement.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via fieldbus
`Target velocity`	Target velocity for operating mode Profile Velocity Changed settings become active immediately.	$0.1min^{-1}$ - 0 -	INT32 R/W - -	CANopen 60FF:0_h

Control word

Bit 8 in parameter `ControlWord`(6040_h) is used to stop a movement with "Halt".

Parameter value	Meaning
Bit 4: Reserved	Not relevant for this operating mode
Bit 5: Reserved	Not relevant for this operating mode
Bit 6: Reserved	Not relevant for this operating mode
Bit 8: Halt	Stop movement with "Halt"
Bit 9: Change on setpoint	Not relevant for this operating mode

Terminating the operating mode

The operating mode is terminated when the motor is at a standstill and one of the following conditions is met:

- Stop caused by "Halt" or "Quick Stop"
- Stop caused by an error

Status word

Information on the current movement is available via bits 10 and 12 in the parameter `StatusWord`: (6041_h).

Parameter value	Meaning
Bit 10: Target reached	0: Target velocity not reached 1: Target velocity reached
Bit 12: Velocity	0: Velocity = >0 1: Velocity = 0
Bit 14: LimP	1: Positive limit switch triggered
Bit 15: LimN	1: Negative limit switch triggered

0198441113938, V2.00, 10.2011

6.7.1.1 Example Node address 1

Work step COB ID / data	Object Value
▸ Activate R_PDO3 601 / 23 02 14 01 01 04 00 04 ◃ 581 / 60 02 14 01 00 00 00 00	1402:1$_h$ 0400 0401$_h$
▸ Activate T_PDO3 601 / 23 02 18 01 81 03 00 04 ◃ 581 / 60 02 18 01 00 00 00 00	1802:1$_h$ 0400 0381$_h$
▸ Set acceleration to 2000 601 / 23 83 60 00 D0 07 00 00 ◃ 581 / 60 83 60 00 00 00 00 00	6083$_h$ 0000 07D0$_h$
▸ NMT Start remote node 0 / 01 00 ◃ T_PDO3 with status word 381 / 31 66 00 00 00 00	
▸ Enable power stage with R_PDO3 401 / 00 00 00 00 00 00 401 / 06 00 00 00 00 00 401 / 0F 00 00 00 00 00 ◃ T_PDO3 (operating state: **6** Operation Enabled) 381 / 37 46 00 00 00 00	
▸ Starting the operating mode 601 / 2F 60 60 00 03 00 00 00 ◃ 581 / 60 60 60 00 00 00 00 00	6060$_h$ 03$_h$
▸ Check operating mode [1)] 601 / 40 61 60 00 00 00 00 00 ◃ Operating mode active 581 / 4F 61 60 00 03 61 01 00	6061$_h$ 03$_h$
▸ R_PDO3: Specification of target velocity 1000 401 / 0F 00 E8 03 00 00 ◃ T_PDO2 with status word and velocity actual value 381 / 37 02 00 00 00 00 ◃ Target velocity reached 381 / 37 06 E8 03 00 00	
▸ Terminate operating mode with "Quick Stop" with R_PDO3 401 / 0B 00 00 00 00 00 ◃ T_PDO3 with status word 381 / 17 66 00 00 00 00	
▸ Clear "Quick Stop" with R_PDO3 401 / 0F 00 00 00 00 00 ◃ T_PDO3 with status word 381 / 37 46 00 00 00 00	

1) The operating mode must be checked until the device has activated the specified operating mode.

0198441113938, V2.00, 10.2011

6.8 Operating mode Profile Torque

Description

In the operating mode Profile Torque, a movement is made with a desired target torque.

Procedure

- Set [`Mode of operation`:(6060_h)] to operating mode Profile Torque (4).
- Set [`Controlword`:(6040_h)] to start the operating mode.

  When the operating mode is started, the target torque is set to zero.
- Set [`Torque slope`:(6087_h)] to the value of the slope of the motion profile for the torque (unit = milliseconds from 0 to 100% torque).
- Set [`Target torque`:(6071_h)] to the value for the target torque (unit = 0.1% of nominal torque. The value is reset to zero if the operating mode is changed, the power stage is disabled or a Quick Stop is triggered.

Optional:

- Query [`Torque demand value`:(6074_h)] to get the value of the torque limitation (unit = increments of 0.1 % of the nominal torque).
- Query [`Torque rated current`:(6075_h)] to get the nominal current depending on the motor and the drive (unit = multiples of mA).
- Query [`Torque actual value`:(6077_h)] to get the actual torque (unit = increments of 0.1 % of the nominal torque).
- Query [`Current actual value`:(6078_h)] to get the actual current (unit = increments of 0.1 % of the nominal current).

List of associated objects

Index	Subindex	Object	PDO	Data type	Takes effect
6040_h	0	Controlword	R_PDO	UINT16	Immediately
6041_h	0	Statusword	T_PDO	UINT16	-
6060_h	0	Modes of Operation	R_PDO	INT8	Immediately
6061_h	0	Modes of Operation Display	T_PDO	INT8	-
6071_h	0	Target Torque	R_PDO	INT16	Immediately
6074_h	0	Torque demand value	T_PDO	INT16	-
6075_h	0	Motor rated current	T_PDO	UINT32	-
6077_h	0	Torque actual value	T_PDO	INT16	-
6087_h	0	Torque slope	R_PDO	UINT32	Immediately

0198441113938, V2.00, 10.2011

6.8.1 Example: Profile Torque

Starting the operating mode

The operating mode must be set in the parameter `Mode of operation`: (6060_h). Writing the parameter value activates the operating mode.

The parameter `Target torque`: (6071_h) starts the movement.

Parameter name	Description	Unit Minimum value Factory setting Maximum value	Data type R/W Persistent Expert	Parameter address via fieldbus
`PTtq_target`	Target torque for operating mode Profile Torque 100.0 % correspond to the continuous stall. In increments of 0.1 %. Changed settings become active immediately.	0,1 % -3000 0 3000	INT16 R/W - -	CANopen 6071:0_h

Control word

Bit 8 in parameter `ControlWord(6040h)` is used to stop a movement with "Halt".

Parameter value	Meaning
Bit 4: Reserved	Not relevant for this operating mode
Bit 5: Reserved	Not relevant for this operating mode
Bit 6: Reserved	Not relevant for this operating mode
Bit 8: Halt	Stop movement with "Halt"
Bit 9: Change on setpoint	Not relevant for this operating mode

Terminating the operating mode

The operating mode is terminated when the motor is at a standstill and one of the following conditions is met:

- Stop caused by "Halt" or "Quick Stop"
- Stop caused by an error

Status word

Information on the movement is available via bit 10 in the parameter `StatusWord`: (6041_h).

Parameter value	Meaning
Bit 10: Target reached	0: Target torque not reached 1: Target torque reached

0198441113938, V2.00, 10.2011

6.8.1.1 Example Node address 1

Work step COB ID / data	Object Value
▸ NMT Start remote node 0 / 01 00 ◃ T_PDO1 with status word 181 / 31 62	
▸ Enable power stage with R_PDO1 201 / 00 00 201 / 06 00 201 / 0F 00 ◃ T_PDO1 (operating state: **6** Operation Enabled) 181 / 37 62	
▸ Starting the operating mode 601 / 2F 60 60 00 04 00 00 00 ◃ 581 / 60 60 60 00 00 00 00 00	6060_h 04_h
▸ Check operating mode [1] 601 / 40 61 60 00 00 00 00 00 ◃ Operating mode active 581 / 4F 61 60 00 04 61 01 00	6061_h 04_h
▸ Target torque set to 100 (10.0%) 601 / 2B 71 60 00 64 00 00 00 ◃ 581 / 60 71 60 00 00 00 00 00 ◃ Target torque reached 181 / 37 06	6071_h 64_h
▸ Terminate operating mode with "Quick Stop" with R_PDO1 201 / 0B 00 ◃ T_PDO1 with status word 181 / 17 66	
▸ Clear "Quick Stop" with R_PDO1 201 / 0F 00 ◃ T_PDO1 with status word 181 / 37 46	

1) The operating mode must be checked until the device has activated the specified operating mode.

0198441113938, V2.00, 10.2011

7 Diagnostics and troubleshooting

7

This chapter describes the various types of diagnostics and provides troubleshooting assistance.

7.1 Error diagnostics via integrated HMI

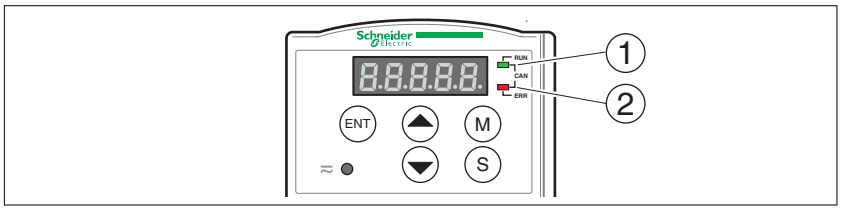


Figure 36: integrated HMI

The internal HMI has 2 status LEDs

(1) RUN (green)
(2) ERR (red)

0198441113938, V2.00, 10.2011

The illustration below shows the fieldbus communication states.

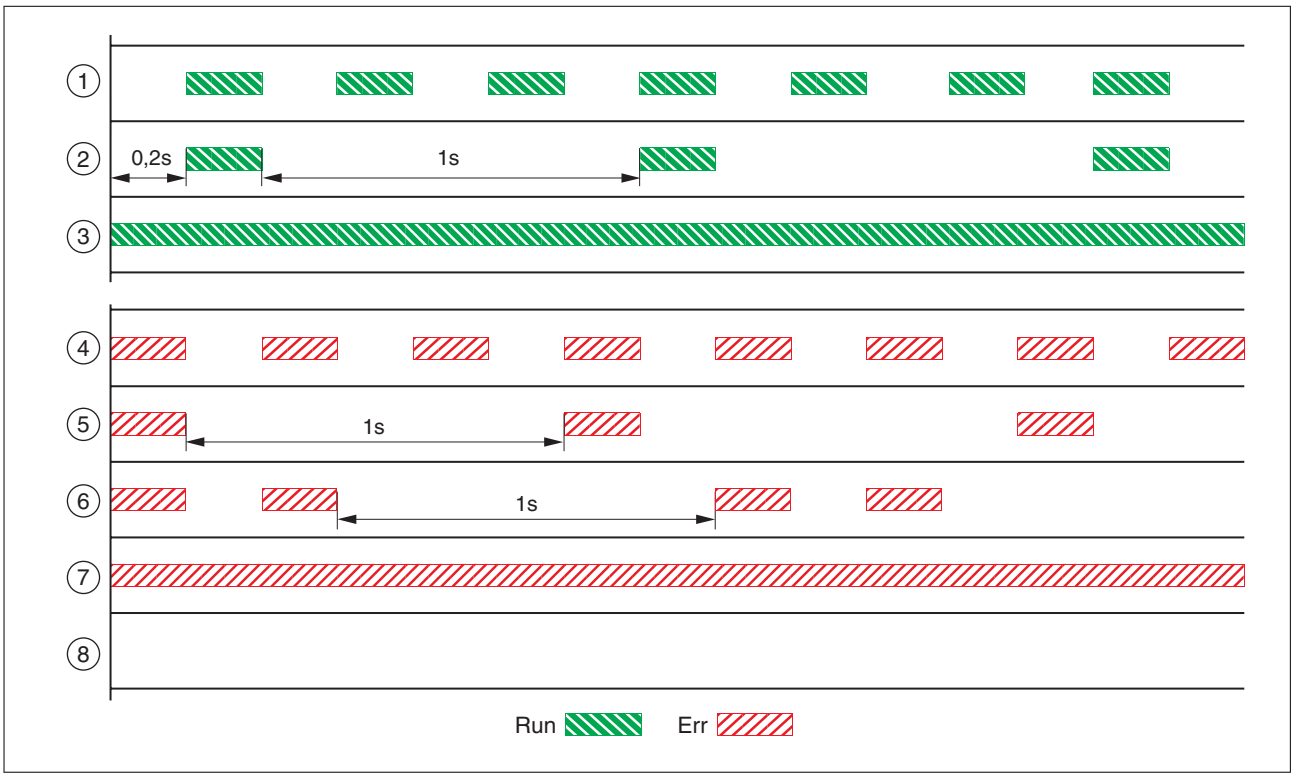


Figure 37: Signals of the CAN bus status LEDs (Run=GN; Err=RD)

(1) NMT state PRE-OPERATIONAL
(2) NMT state STOPPED
(3) NMT state OPERATIONAL
(4) Incorrect settings,
for example, invalid node address
(5) Warning limit reached,
for example after 16 incorrect transmission attempts
(6) Monitoring event (node guarding)
(7) CAN is BUS-OFF,
for example after 32 incorrect transmission attempts.
(8) Fieldbus communication without error message.

0198441113938, V2.00, 10.2011

7.2 Error register

The object `Error register(1001h)` indicates the error of a device in bit-coded form. The exact cause of error can 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	-	Reserved
2	-	Reserved
3	-	Reserved
4	Communication	Network communication error
5	Device profile-specific	Error during execution as per device profile
6	-	Reserved
7	Manufacturer-specific	Vendor-specific error message

0198441113938, V2.00, 10.2011

7.3 Communication Alarm List

Emergency Object

Byte	0	1	2	3	4	5	6	7
Content	Emergency Error Code		Error register	Panel Alam Code		N/A		

Display	Error Name	Error Description	Clear
AL185	CAN Bus Error	CAN bus Error Counter exceeds 128	NMT-ResetNode or Re-PowerOn
AL111	SDO Rx Overrun Error	SDO Rx Overrun (receive two SDO packet in 1ms)	NMT-ResetNode or 6040_h fault reset
AL112	PDO Rx Overrun Error	PDO Rx Overrun (receive two PDO (same COBID)packet in 1 1ms)	NMT-ResetNode or 6040_h fault reset
AL121	PDO Index Error	Index error when accessing PDO object	NMT-ResetNode or 6040_h fault reset
AL122	PDO Sub-Index Error	Sub-Index error when accessing PDO object	NMT-ResetNode or 6040_h fault reset
AL123	PDO Rx Write Data Size Error	Data type(size) error when accessing PDO object (ex. Write 32bit data into 16bit data size)	NMT-ResetNode or 6040_h fault reset
AL124	PDO Rx Write Data Range Error	Data range error when accessing PDO object	NMT-ResetNode or 6040_h fault reset
AL125	PDO Rx Write R-Only Error	Object is read-only when PDO writing	NMT-ResetNode or 6040_h fault reset
AL126	PDO Mapping Error	Object is not mapped to the PDO	NMT-ResetNode or 6040_h fault reset
AL127	PDO Rx Write SON Error	Object does not allow to write when Servo On	NMT-ResetNode or 6040_h fault reset
AL128	PDO Tx/Rx Read EEPROM Error	Object error when reading from EEPROM	NMT-ResetNode or 6040_h fault reset
AL129	PDO Rx Write EEPROM Error	Object error when writing to EEPROM	NMT-ResetNode or 6040_h fault reset
AL130	PDO Tx/Rx R/W EEPROM Range Error	Invalid Range when accessing EEPROM	NMT-ResetNode or 6040_h fault reset
AL131	PDO Tx/Rx R/W EEPROM CheckSum Error	Checksum error when accessing EEPROM	NMT-ResetNode or 6040_h fault reset
AL132	PDO Tx/Rx Write EEPROM Zone Error	Password error when writing encryption zone	NMT-ResetNode or 6040_h fault reset
AL3E1	SYNC Error	Interal adjuster not successful because of SYNC disorder	NMT-ResetNode or 6040_h fault reset
AL3E2	SYNC too early	Drive receive two SYNC in one period	NMT-ResetNode or 6040_h fault reset
AL3E3	SYNC not received	SYNC not received in two period	NMT-ResetNode or 6040_h fault reset
AL3E4	Internal Cmd Generation Error	Internal generation has not enough time to process	NMT-ResetNode or 6040_h fault reset
AL3E5	SYNC invalid	SYNC period 1006_h value invalid	NMT-ResetNode or 6040_h fault reset

0198441113938, V2.00, 10.2011

7.3.1 ErrorCode order by Alarm

Display	Description	32bit-ErrorCode (16bit-Error-Code + 16bit-Additional Info)
AL001	Over current	2310-0001$_h$
AL002	Over voltage	3110-0002$_h$
AL003	Low voltage	3120-0003$_h$
AL004	Motor type not match	7122-0004$_h$
AL005	Re-generation error	3210-0005$_h$
AL006	Over load	3230-0006$_h$
AL007	Over speed	8400-0007$_h$
AL008	Abnormal, AL pulse cmd	8600-0008$_h$
AL009	Pulse error exceed	8611-0009$_h$
AL010	MC WatchDog Timeout	0000-0010$_h$
AL011	Encoder error	7305-0011$_h$
AL012	Calibration error	6320-0012$_h$
AL013	DI: EMGS alarm	5441-0013$_h$
AL014	DI: N-limit	5443-0014$_h$
AL015	DI: P-limit	5442-0015$_h$
AL016	Over Heat of Motor	4210-0016$_h$
AL017	EEPROM error	5330-0017$_h$
AL018	OA/OB Freq. too fast	7306-0018$_h$
AL019	Com-port not OK	7510-0019$_h$
AL020	Com-port timeout	7520-0020$_h$
AL021	Reserved	Reserved
AL022	Main power lack phase	3130-0022$_h$
AL023	Pre-overload	3231-0023$_h$
AL024	Encoder magnet field error	7305-0024$_h$
AL025	Encoder counter error	7305-0025$_h$
AL026	Encoder data error	7305-0026$_h$
AL027	Encoder reset error	7305-0027$_h$
AL030	Motor protection error	7121-0030$_h$
AL031	UVW wiring error	3300-0031$_h$
AL040	Full close-loop pos error	8610-0040$_h$
AL099	Firmware upgrade, Notify perform ROM upgrading	5500-0099$_h$
AL111	SDO Rx Overrun Error	8110-0111$_h$
AL112	PDO Rx Overrun Error	8110-0112$_h$
AL121	PDO Index Error	8200-0121$_h$
AL122	PDO Sub-Index Error	8200-0122$_h$
AL123	PDO Rx Write Data Size Error	8200-0123$_h$

0198441113938, V2.00, 10.2011

Display	Description	32bit-ErrorCode (16bit-Error-Code + 16bit-Additional Info)
AL124	PDO Rx Write Data Range Error	8200-0124$_h$
AL125	PDO Rx Write R-Only Error	8200-0125$_h$
AL126	PDO Mapping Error	8200-0126$_h$
AL127	PDO Rx Write SON Error	8200-0127$_h$
AL128	PDO Tx/Rx Read EEPROM Error	8200-0128$_h$
AL129	PDO Rx Write EEPROM Error	8200-0129$_h$
AL130	PDO Tx/Rx R/W EEPROM Range Error	8200-0130$_h$
AL131	PDO Tx/Rx R/W EEPROM CheckSum Error	8200-0131$_h$
AL132	PDO Tx/Rx Write EEPROM Zone Error	8200-0132$_h$
AL180	Life guard error or heart beat error	
AL185	CAN Bus Error	8120-0185$_h$
AL201	CANopen load/save 1010/1011 error	6310-0201$_h$
AL283	Software P-limit	5444-0283$_h$
AL285	Software N-limit	5445-0285$_h$
AL3E1	SYNC Error	6200-0301$_h$
AL3E2	SYNC too early	6200-0302$_h$
AL3E3	SYNC not received	6200-0303$_h$
AL3E4	Internal Cmd Generation Error	6200-0304$_h$
AL3E5	SYNC invalid	6200-0305$_h$

0198441113938, V2.00, 10.2011

7.3.2 SDO Abort Codes

Abort Code	Description
05040001_h	Client/server command specifier not valid or unknown
06010002_h	Attempt to write a read only object
06020000_h	Object does not exist in the object dictionary
06040041_h	Object cannot be mapped to the PDO
06040042_h	The number and length of the objects to be mapped would exceed PDO length
06060000_h	Access failed due to an hardware error (store or restore error)
06070010_h	Data type does not match, length of service parameter does not match
06090011_h	Sub-index does not exist
06090030_h	Value range of parameter exceeded(only for write access)
08000000_h	General error
$080000a1_h$	Object error when reading from EEPROM
$080000a2_h$	Object error when writing to EEPROM
$080000a3_h$	Invalid Range when accessing EEPROM
$080000a4_h$	Checksum error when accessing EEPROM
$080000a5_h$	Password error when writing encryption zone
08000020_h	Data cannot be transferred or stored to the application (store or restore signature error)
08000021_h	Data cannot be transferred or stored to the application because of the local control(store or restore while wrong state)
08000022_h	Object is on the fly

0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

8 Object dictionary

8

Object type

Object Name	Comments
VAR	A single value such as an UNSIGNED8, Boolean, float, INTEGER16 etc.
ARRAY	A multiple data field object where each data field is a sample variable of the SAME basic data type e.g. array of UNSIGNED16 etc. Sub-index 0 is of UNSIGNED8 and therefore not part of the ARRAY data.
RECORD	A multiple data field object where the data fields may be any combination of simple variables. Sub-index 0 is of UNSIGNED8 and therefore not part of the RECORD data.

Data Type Please refer to CANopen Standard 301.

0198441113938, V2.00, 10.2011

Overview object dictionary entries

Index	Object Type	Name	DataType	Access	Mappable
CANopen DS301					
1000_h	VAR	Device type	UNSIGNED 32	RO	No
1001_h	VAR	Error register	UNSIGNED 8	RO	Yes
1003_h	ARRAY	Pre-defined error field	UNSIGNED 32	RW	No
1005_h	VAR	COB-ID SYNC	UNSIGNED 32	RW	No
1006_h	VAR	Communication cycle period	UNSIGNED 32	RW	No
$100C_h$	VAR	Guard time	UNSIGNED 16	RW	No
$100D_h$	VAR	Life time factor	UNSIGNED 8	RW	No
1010_h	ARRAY	Store parameters	UNSIGNED 32	RW	No
1014_h	VAR	COB-ID EMCY	UNSIGNED 32	RO	No
1016_h	ARRAY	Consumer heartbeat time	UNSIGNED 32	RW	No
1017_h	VAR	Producer heartbeat time	UNSIGNED 16	RW	No
1018_h	RECORD	Identity object	UNSIGNED 32	RO	No
1400_h~03_h	RECORD	Receive PDO parameter	UNSIGNED 16/32	RW	No
1600_h~03_h	RECORD	Receive PDO mapping	UNSIGNED 32	RW	No
1800_h~03_h	RECORD	Transmit PDO parameter	UNSIGNED 16/32	RW	No
$1A00_h$~03_h	RECORD	Transmit PDO mapping	UNSIGNED 32	RW	No

Index	Object Type	Name	DataType	Access	Mappable
CANopen DS402					
6040_h	VAR	Control word	UNSIGNED 16	RW	Yes
6041_h	VAR	Status word	UNSIGNED 16	RO	Yes
$605B_h$	VAR	Shutdown option code	INTEGER16	RW	No

0198441113938, V2.00, 10.2011

Index	Object Type	Name	DataType	Access	Mappable
$605E_h$	VAR	Fault reaction option code	INTEGER16	RW	No
6060_h	VAR	Modes of operation	INTEGER8	RW	Yes
6061_h	VAR	Modes of operation display	INTEGER8	RO	Yes
6062_h	VAR	Position demand value	INTEGER32	RO	Yes
6063_h	VAR	Position actual value*	INTEGER32	RO	Yes
6064_h	VAR	Position actual value	INTEGER32	RO	Yes
6065_h	VAR	Following error window	UNSIGNED32	RW	Yes
6067_h	VAR	Position windows	UNSIGNED32	RW	Yes
6068_h	VAR	Position window time	UNSIGNED16	RW	Yes
$606B_h$	VAR	Velocity demand value	INTEGER32	RO	Yes
$606C_h$	VAR	Velocity actual value	INTEGER32	RO	Yes
$606D_h$	VAR	Velocity window	UNSIGNED16	RW	Yes
$606E_h$	VAR	Velocity window time	UNSIGNED16	RW	Yes
$606F_h$	VAR	Velocity threshold	UNSIGNED16	RW	Yes
6071_h	VAR	Target torque	INTEGER16	RW	Yes
6074_h	VAR	Torque demand value	INTEGER16	RO	Yes
6075_h	VAR	Motor rated current	UNSIGNED32	RO	Yes
6077_h	VAR	Torque actual value	UNSIGNED16	RO	Yes
6078_h	VAR	Current actual value	INTEGER16	RO	Yes
$607A_h$	VAR	Target position	INTEGER32	RW	Yes

0198441113938, V2.00, 10.2011

Index	Object Type	Name	DataType	Access	Mappable
607C$_h$	VAR	Home Offset	INTEGER32	RW	Yes
6081$_h$	VAR	Profile velocity	UNSIGNED 32	RW	Yes
6083$_h$	VAR	Profile acceleration	UNSIGNED 32	RW	Yes
6084$_h$	VAR	Profile deceleration	UNSIGNED 32	RW	Yes
6085$_h$	VAR	Quick stop deceleration	UNSIGNED 32	RW	Yes
6087$_h$	VAR	Torque slope	UNSIGNED 32	RW	Yes
6093$_h$	ARRAY	Position factor	UNSIGNED 32	RW	Yes
6098$_h$	VAR	Homing method	INTEGER8	RW	Yes
6099$_h$	ARRAY	Homing speeds	UNSIGNED 32	RW	Yes
609A$_h$	VAR	Homing acceleration	UNSIGNED 32	RW	Yes
60C0$_h$	VAR	Interpolation sub mode select	INTEGER16	RW	Yes
60C1$_h$	ARRAY	Interpolation data record	UNSIGNED 16	RW	Yes
60C2$_h$	RECORD	Interpolation time period	SIGNED8	RW	Yes
60F4$_h$	VAR	Following error actual value	INTEGER32	RO	Yes
60FC$_h$	VAR	Position demand value	INTEGER32	RO	Yes
60FF$_h$	VAR	Target velocity	INTEGER32	RW	Yes
6502$_h$	VAR	Supported drive modes	UNSIGNED 32	RO	Yes
Manufacturer spedific					
2xxx	VAR	Keypad Mapping	INTEGER16/32	RW	Yes

0198441113938, V2.00, 10.2011

Object 1000_h: Device Type Object 1001_h: Error Register

INDEX	1000_h
Name	Device type
Object Code	VAR
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	04020192css

Device	Additional information	Device profile number	
Mode bits	Type		
Bit arrange	31~24	23~16	15~0
Servo drive	04_h	02_h	0192_h

INDEX	1001_h
Name	Error register
Object Code	VAR
Data Type	UNSIGNED8
Access	RO
PDO Mapping	Yes
Value Range	UNSIGNED8
Default Value	0

0198441113938, V2.00, 10.2011

Object 1003_h: Pre-defined Error Field

INDEX	1003_h
Name	Pre-defined error field
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Number of errors
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	0~5
Default Value	0

Sub-Index	1~5
Description	Standard error field
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

Object 1005_h: COB-ID SYNC message

INDEX	1005_h
Name	COB-ID SYNC message
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	80h

Object 1006_h: Communication Cycle Period

INDEX	1006_h
Name	Communication cycle period
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

0198441113938, V2.00, 10.2011

Object $100C_h$: Guard Time

INDEX	$100C_h$
Name	Guard Time
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	0

Object $100D_h$: Life Time Factor

INDEX	$100D_h$
Name	Life Time Factor
Object Code	VAR
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	0

Object 1010_h: Store parameters

INDEX	1010_h
Name	Store parameters
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Largest sub-index supported
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	1
Default Value	1

Sub-Index	1
Description	Save all parameters
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	1

0198441113938, V2.00, 10.2011

Object 1014_h: COB-ID Emergency Object

INDEX	1014_h
Name	COB-ID Emergency message
Object Code	VAR
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	80_h + Node-ID

Object 1016_h: Consumer Heartbeat Time Object 1017_h: Producer Heartbeat Time

INDEX	1016_h
Name	Consumer Heartbeat Time
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Number entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	1
Default Value	1

Sub-Index	1
Description	Consumer Heartbeat Time
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

INDEX	1017_h
Name	Producer Heartbeat Time
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	0

0198441113938, V2.00, 10.2011

Object 1018_h: Identity Object

INDEX	1018_h
Name	Identity Object
Object Code	RECORD
Data Type	Identity
Access	RO
PDO Mapping	No

Sub-Index	0
Description	Number of entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	3
Default Value	3

Sub-Index	1
Description	Vendor ID
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	1DDh

Sub-Index	2
Description	Product code
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	6000h

Sub-Index	3
Description	Revision number
Data Type	UNSIGNED32
Access	RO
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	02000001_h

0198441113938, V2.00, 10.2011

Object 1400_h ~ 1403_h: Receive PDO Communication Parameter

INDEX	1400_h ~ 1403_h
Name	Receive PDO parameter
Object Code	RECORD
Data Type	PDO CommPar
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Largest sub-index supported
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1
Description	COB-ID used by PDO
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	Default Node-ID: 0 Index 1400_h: 200_h + Node-ID Index 1401_h: 300_h + Node-ID Index 1402_h: 400_h + Node-ID Index 1403_h: 500_h + Node-ID

Sub-Index	2
Description	Reception type
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	0

0198441113938, V2.00, 10.2011

Object 1600_h ~ 1603_h: Receive PDO Mapping Parameter

INDEX	1600_h ~ 1603_h
Name	Receive PDO mapping
Object Code	RECORD
Data Type	PDO Mapping
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Number of mapped application objects in PDO
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	0: deactivated 1~8: activated
Default Value	0

Sub-Index	1~8
Description	PDO mapping for the nth application object to be mapped
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

0198441113938, V2.00, 10.2011

Object 1800_h ~ 1803_h: Transmit PDO Communication Parameter

INDEX	1800_h ~ 1803_h
Name	Transmit PDO parameter
Object Code	RECORD
Data Type	PDO CommPar
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Largest sub-index supported
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	3
Default Value	3

Sub-Index	1
Description	COB-ID used by PDO
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	Default Node-ID: 0 Index 1800_h: 180_h + Node-ID Index 1801_h: 280_h + Node-ID Index 1802_h: 380_h + Node-ID Index 1803_h: 480_h + Node-ID

Sub-Index	2
Description	Transmission type
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	0

Sub-Index	5
Description	Event timer
Data Type	UNSIGNED16
Access	RW
PDO Mapping	No
Value Range	0: not used
UNSIGNED16	
Default Value	0

0198441113938, V2.00, 10.2011

Object $1A00_h$ ~ $1A03_h$: Transmit PDO Mapping Parameter

INDEX	$1A00_h$ ~ $1A03_h$
Name	Transmit PDO mapping
Object Code	RECORD
Data Type	PDO Mapping
Access	RW
PDO Mapping	No

Sub-Index	0
Description	Number of mapped application objects in PDO
Data Type	UNSIGNED8
Access	RW
PDO Mapping	No
Value Range	0: deactivated 1~8: activated
Default Value	0

Sub-Index	1~8
Description	PDO mapping for the nth application object to be mapped
Data Type	UNSIGNED32
Access	RW
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

Object 6040_h: Controlword

INDEX	6040_h
Name	Controlword
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Default Value	0

0198441113938, V2.00, 10.2011

Bit Definition

15~9	8	7	6~4	3	2	1	0
N/A	Halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on

NOTE: If Host wants to make Drive servo-on, the bit0, bit1 bit2 and bit3 of Controlword must be activated. It means Host would send OD-6040_h for $0F_h$ to make Drive servo-on

Bit	Operation mode				
	pp	hm	ip	pv	pt
4	New set-point (positive trigger)	Start Homing	Enable ip mode	N/A	N/A
5	Change set immediately	N/A	N/A	N/A	N/A
6	abs/rel	N/A	N/A	N/A	N/A

Object 6041_h: Statusword

INDEX	6041_h
Name	Statusword
Object Code	VAR
Data Type	UNSIGNED16
Access	RO
PDO Mapping	Yes
Value Range	UNSIGNED16
Default Value	0

7	6	5	4	3	2	1	0
warning	switch on disabled	Quick stop	Voltage enabled	Fault	Operation enabled	Switched on	Ready to switch on

14,15	12,13	11	10	9	8
manufacturerspecific	operation mode specific	internal limit active	Target reached	Remote	manufacture r specific

Bit	Operation mode				
	pp	hm	ip	pv	pt
12	Set-point acknowledge	Homing attained	IP mode active	Zero Speed	N/A
13	Following error	Homing error	N/A	N/A	N/A
14	N/A	N/A	Sync OK	N/A	N/A
15	N/A	N/A	N/A	N/A	N/A

0198441113938, V2.00, 10.2011

Object $605B_h$: Shutdown option code

INDEX	$605B_h$
Name	Shutdown option code
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	0
Comment	0:Disable drive function -1:Dynamic break enable

Object $605E_h$: Fault reaction option code

INDEX	$605E_h$
Name	Fault reaction option code
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	2
Comment	0:Disable drive, motor is free to rotate 1:slow down on slow down ramp 2:slow down on quick stop ramp

0198441113938, V2.00, 10.2011

Object 6060_h: Modes of operation Object 6061_h: Modes of operation display

INDEX	6060_h
Name	Modes of operation
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	INTEGER8
Default Value	0
Comment	-1: Jog mode 0: Reserved 1: Profile position mode 3: Profile velocity mode 4: Profile torque mode 6: Homing mode 7: Interpolated Position mode

INDEX	6061_h
Name	Modes of operation display
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	INTEGER8
Default Value	0

Object 6062_h: Position demand value

INDEX	6062_h
Name	Position demand value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	Pos cmd calculated by Interpolation theory Unit: pulse

0198441113938, V2.00, 10.2011

Object 6063_h: Position demand value

INDEX	6063_h
Name	Position actual value*
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	Unit: increments

Object 6064_h: Position actual value

INDEX	6064_h
Name	Position actual value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	Unit: pulse

Object 6065_h: Following error window

INDEX	6065_h
Name	Following error window
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	0
Comment	Unit: pulse

0198441113938, V2.00, 10.2011

Object 6067_h: Position window Object 6068_h: Position window time

INDEX	6067_h
Name	Position window
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Comment	Unit: pulse

INDEX	6068_h
Name	Position window time
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Comment	Unit: millisecond

Object $606B_h$: Velocity demand value

INDEX	$606B_h$
Name	Velocity demand value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Comment	Unit: 0.1rpm

Object $606C_h$: Velocity actual value

INDEX	$606C_h$
Name	Velocity actual value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Comment	Unit: 0.1rpm

0198441113938, V2.00, 10.2011

Object $606D_h$: Velocity window

INDEX	$606D_h$
Name	Velocity window
Object Code	VAR
Data Type	INTEGER16
Access	RO
PDO Mapping	Yes
Value Range	INTEGER16
Comment	Unit: 0.1rpm

Object $606E_h$: Velocity window time

INDEX	$606E_h$
Name	Velocity window time
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Comment	Unit: millisecond

Object $606F_h$: Velocity threshold Object 6071_h: Target torque

INDEX	$606F_h$
Name	Velocity threshold
Object Code	VAR
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Comment	Unit: 0.1rpm

INDEX	6071_h
Name	Target torque
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Comment	Unit: per thousand of rated torque

0198441113938, V2.00, 10.2011

Object 6074_h: Torque demand value

INDEX	6074_h
Name	Torque demand value
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Comment	Unit: per thousand of rated torque

Object 6075_h: Motor rated current

INDEX	6075_h
Name	Motor rated current
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Comment	Unit: milliamp

Object 6077_h: Torque actual value

INDEX	6077_h
Name	Torque actual value
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Comment	Unit: per thousand of rate torque

Object 6078_h: Current actual value

INDEX	6078_h
Name	Current actual value
Object Code	VAR
Data Type	INTEGER16
Access	RO
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	0
Comment	Unit: per thousand of rated current

0198441113938, V2.00, 10.2011

Object $607A_h$: Target position Object $607C_h$: Home Offset

INDEX	$607A_h$
Name	Target position
Object Code	VAR
Data Type	INTEGER32
Access	RW
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	For Profile position mode 6060_h=1 Unit: pulse

INDEX	$607C_h$
Name	Home offset
Object Code	VAR
Data Type	INTEGER32
Access	RW
PDO Mapping	Yes
Value Range	INTEGER32
Default Value	0
Comment	Unit : pulse

Object 6081_h: Profile velocity

INDEX	6081_h
Name	Profile Velocity
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	10000
Comment	For Profile position mode 6060_h=1 Unit: pulse per second

Object 6083_h: Profile acceleration

INDEX	6083_h
Name	Profile acceleration
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	1~UNSIGNED32
Default Value	200
Comment	For Profile position mode 6060_h=1 Unit: millisecond (time from 0rpm to 3000rpm)

0198441113938, V2.00, 10.2011

Object 6084_h: Profile deceleration

INDEX	6084_h
Name	Profile deceleration
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	1~UNSIGNED32
Default Value	200
Comment	For Profile position mode 6060_h=1 Unit: millisecond (time from 0rpm to 3000rpm)

Object 6085_h: Quick stop deceleration Object 6087_h: Torque slope

INDEX	6085_h
Name	Quick stop acceleration
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	0
Comment	Unit: millisecond (time from 0rpm to 3000rpm)

INDEX	6087_h
Name	Torque slope
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	0
Comment	Unit: millisecond (time from 0 to 100% rated torque)

0198441113938, V2.00, 10.2011

Object 6093_h: Position factor

INDEX	6093_h
Name	Position factor
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Comment	Position factor = Numerator / Feed_constant

Sub-Index	0
Description	Number of entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1
Description	Numerator
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Default Value	1
Comment	Same as P1-44

Sub-Index	2
Description	Feed constant
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Default Value	1
Comment	Same as P1-45

Object 6098_h: Homing method

INDEX	6098_h
Name	Homing method
Object Code	VAR
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	0~35
Default Value	0

0198441113938, V2.00, 10.2011

Object 6099_h: Homing speeds Object $609A_h$: Homing acceleration

INDEX	6099_h
Name	Homing speeds
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes

Sub-Index	0
Description	Number of entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	Yes
Value Range	2
Default Value	2

Sub-Index	1
Description	Speed during search for switch
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	1~2000
Default Value	100
Comment	Uint:rpm

Sub-Index	2
Description	Speed during search for zero
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	1~500
Default Value	20
Comment	Uint:rpm

INDEX	$609A_h$
Name	Homing acceleration
Object Code	VAR
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	100
Comment	Unit: millisecond (time of acc from 0rpm to 3000rpm)

0198441113938, V2.00, 10.2011

Object $60C0_h$: Interpolation sub mode select

INDEX	$60C0_h$
Name	Interpolation sub mode select
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	0
Comment	0: manufacturer specific (Linear interpolation -not need pos difference[OD-60C1sub3]) -1: manufacturer specific (Schneider definition -need pos difference[OD-60C1sub3])

0198441113938, V2.00, 10.2011

Object $60C1_h$: Interpolation data record

INDEX	$60C1_h$
Name	Interpolation data record
Object Code	ARRAY
Data Type	UNSIGNED32
Access	RW
PDO Mapping	Yes
Comment	Set this record by PDO every T msec before SYNC message Where T is specified by 1006_h

Sub-Index	0
Description	Number of entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	3
Default Value	3

Sub-Index	1
Description	Pos_Cmd (Low Word)
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Default Value	0
Comment	Unit: low word of 32-bit pulse

Sub-Index	2
Description	Pos_Cmd (High Word)
Data Type	UNSIGNED16
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED16
Default Value	0
Comment	Unit: high word of 32-bit pulse

Sub-Index	3
Description	Velocity – Pos_Cmd difference
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	0
Comment	UXi = (Xi+1 - Xi-1)/2 (it is also the same as velocity) Unit: pulse

0198441113938, V2.00, 10.2011

Object $60C2_h$: Interpolation time period

INDEX	$60C2_h$
Name	Interpolation time period
Object Code	RECORD
Data Type	UNSIGNED8
Access	RW
PDO Mapping	Yes
Comment	The unit of the interpolation time unit is given in $10^{\text{interpolation time index seconds}}$

Sub-Index	0
Description	Number of entries
Data Type	UNSIGNED8
Access	RO
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1
Description	Interpolation time units
Data Type	UNSIGNED8
Access	RW
PDO Mapping	Yes
Value Range	UNSIGNED8
Default Value	1

Sub-Index	2
Description	Interpolation time index
Data Type	INTEGER8
Access	RW
PDO Mapping	Yes
Value Range	-128~63
Default Value	-3

Object $60F4_h$: Following error actual value

INDEX	$60F4_h$
Name	Following error actual value
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Comment	Unit: pulse

0198441113938, V2.00, 10.2011

Object $60FC_h$: Position demand value

INDEX	$60FC_h$
Name	Position demand value*
Object Code	VAR
Data Type	INTEGER32
Access	RO
PDO Mapping	Yes
Value Range	INTEGER32
Comment	Unit: increment

Object $60FF_h$: Target velocity

INDEX	$60FF_h$
Name	Target velocity
Object Code	VAR
Data Type	INTEGER32
Access	RW
PDO Mapping	Yes
Value Range	INTEGER32
Comment	Unit: 0.1rpm

0198441113938, V2.00, 10.2011

Object 6502_h: Supported drive modes

INDEX	6502_h
Name	Supported drive modes
Object Code	VAR
Data Type	UNSIGNED32
Access	Ro
PDO Mapping	Yes
Value Range	UNSIGNED32
Default Value	EF_h

31	16	15	10	9	8	7	6	5	4	3	2	1	0
Manufacturer specific		Reserved		CST	CSV	CSP	ip	hm	Reserved	tq	pv	vl	pp
MSB													LSB

Object $2xxx_h$: Keypad mapping

INDEX	$2xxx_h$
Name	Keypad mapping register
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	N/A

Object 2xxx is defined Keypad mapping.

If user wants to use CANopen protocol for simulate Keypad press, he could read and write Keypad parameter via SDO protocol.

Pa-**bc**<==>2a**BC** $_h$
BC is hexadecimal format of **bc**

Example: Object 2305_h: Node-ID [P3-05]

INDEX	2300_h
Name	Node-ID
Object Code	VAR
Data Type	INTEGER16
Access	RW
PDO Mapping	Yes
Value Range	INTEGER16
Default Value	0_h

0198441113938, V2.00, 10.2011

Example 2:
Object 212C$_h$: Electronic Gear [P1-44]

INDEX	212C$_h$
Name	Electronic Gear
Object Code	VAR
Data Type	INTEGER32
Access	RW
PDO Mapping	Yes
Value Range	INTEGER32

0198441113938, V2.00, 10.2011

9 Glossary

9

9.1 Units and conversion tables

The value in the specified unit (left column) is calculated for the desired unit (top row) with the formula (in the field).

Example: conversion of 5 meters [m] to yards [yd]
5 m / 0.9144 = 5.468 yd

9.1.1 Length

	in	ft	yd	m	cm	mm
in	-	/ 12	/ 36	* 0.0254	* 2.54	* 25.4
ft	* 12	-	/ 3	* 0.30479	* 30.479	* 304.79
yd	* 36	* 3	-	* 0.9144	* 91.44	* 914.4
m	/ 0.0254	/ 0.30479	/ 0.9144	-	* 100	* 1000
cm	/ 2.54	/ 30.479	/ 91.44	/ 100	-	* 10
mm	/ 25.4	/ 304.79	/ 914.4	/ 1000	/ 10	-

9.1.2 Mass

	lb	oz	slug	kg	g
lb	-	* 16	* 0.03108095	* 0.4535924	* 453.5924
oz	/ 16	-	* $1.942559*10^{-3}$	* 0.02834952	* 28.34952
slug	/ 0.03108095	/ $1.942559*10^{-3}$	-	* 14.5939	* 14593.9
kg	/ 0.45359237	/ 0.02834952	/ 14.5939	-	* 1000
g	/ 453.59237	/ 28.34952	/ 14593.9	/ 1000	-

9.1.3 Force

	lb	oz	p	dyne	N
lb	-	* 16	* 453.55358	* 444822.2	* 4.448222
oz	/ 16	-	* 28.349524	* 27801	* 0.27801
p	/ 453.55358	/ 28.349524	-	* 980.7	* $9.807*10^{-3}$
dyne	/ 444822.2	/ 27801	/ 980.7	-	/ $100*10^{3}$
N	/ 4.448222	/ 0.27801	/ $9.807*10^{-3}$	* $100*10^{3}$	-

9.1.4 Power

	HP	W
HP	-	* 746
W	/ 746	-

0198441113938, V2.00, 10.2011

9.1.5 Rotation

	min^{-1} (RPM)	**rad/s**	**deg./s**
min^{-1} (RPM)	-	* π / 30	* 6
rad/s	* 30 / π	-	* 57.295
deg./s	/ 6	/ 57.295	-

9.1.6 Torque

	lb·in	**lb·ft**	**oz·in**	**Nm**	**kp·m**	**kp·cm**	**dyne·cm**
lb·in	-	/ 12	* 16	* 0.112985	* 0.011521	* 1.1521	* $1.129*10^6$
lb·ft	* 12	-	* 192	* 1.355822	* 0.138255	* 13.8255	* $13.558*10^6$
oz·in	/ 16	/ 192	-	* $7.0616*10^{-3}$	* $720.07*10^{-6}$	* $72.007*10^{-3}$	* 70615.5
Nm	/ 0.112985	/ 1.355822	/ $7.0616*10^{-3}$	-	* 0.101972	* 10.1972	* $10*10^6$
kp·m	/ 0.011521	/ 0.138255	/ $720.07*10^{-6}$	/ 0.101972	-	* 100	* $98.066*10^6$
kp·cm	/ 1.1521	/ 13.8255	/ $72.007*10^{-3}$	/ 10.1972	/ 100	-	* $0.9806*10^6$
dyne·cm	/ $1.129*10^6$	/ $13.558*10^6$	/ 70615.5	/ $10*10^6$	/ $98.066*10^6$	/ $0.9806*10^6$	-

9.1.7 Moment of inertia

	lb·in^2	**lb·ft^2**	**kg·m^2**	**kg·cm^2**	**kp·cm·s^2**	**oz·in^2**
lb·in^2	-	/ 144	/ 3417.16	/ 0.341716	/ 335.109	* 16
lb·ft^2	* 144	-	* 0.04214	* 421.4	* 0.429711	* 2304
kg·m^2	* 3417.16	/ 0.04214	-	* $10*10^3$	* 10.1972	* 54674
kg·cm^2	* 0.341716	/ 421.4	/ $10*10^3$	-	/ 980.665	* 5.46
kp·cm·s^2	* 335.109	/ 0.429711	/ 10.1972	* 980.665	-	* 5361.74
oz·in^2	/ 16	/ 2304	/ 54674	/ 5.46	/ 5361.74	-

9.1.8 Temperature

	°F	**°C**	**K**
°F	-	(°F - 32) * 5/9	(°F - 32) * 5/9 + 273.15
°C	°C * 9/5 + 32	-	°C + 273.15
K	(K - 273.15) * 9/5 + 32	K - 273.15	-

9.1.9 Conductor cross section

AWG	1	2	3	4	5	6	7	8	9	10	11	12	13
mm^2	42.4	33.6	26.7	21.2	16.8	13.3	10.5	8.4	6.6	5.3	4.2	3.3	2.6

AWG	14	15	16	17	18	19	20	21	22	23	24	25	26
mm^2	2.1	1.7	1.3	1.0	0.82	0.65	0.52	0.41	0.33	0.26	0.20	0.16	0.13

0198441113938, V2.00, 10.2011

9.2 Terms and Abbreviations

See chapter "2.5 Standards and terminology" for information on the pertinent standards on which many terms are based. Some terms and abbreviations may have specific meanings with regard to the standards.

AC	Alternating current
CAN	(**C**ontroller **A**rea **N**etwork), standardized open fieldbus as per ISO 11898, allows drives and other devices from different manufacturers to communicate.
CANopen	Device- and manufacturer-independent description language for communication via the CAN bus
CiA	**C**AN **i**n **A**utomation, CAN interest group, standardization group for CAN and CANopen.
COB	**C**ommunication **OB**ject, transport unit in a CAN network.
COB ID	**C**ommunication **OB**ject **ID**entifier; uniquely identifies each communication object in a CAN network
DC	Direct current
DOM	**D**ate **o**f **m**anufacturing: The nameplate of the product shows the date of manufacture in the format DD.MM.YY or in the format DD.MM.YYYY. Example: 31.12.09 corresponds to December 31, 2009 31.12.2009 corresponds to December 31, 2009
DriveCom	Specification of the DSP402 state machine was created in accordance with the DriveCom specification.
DS301	Standardizes the CANopen communication profile
DSP402	Standardizes the CANopen device profile for drives
E	Encoder
I/O	Inputs/outputs
EDS	(**E**lectronic **D**ata **S**heet); contains the specific properties of a product.
Input device	A device that can be connected via the RS232 interface; either the HMI or a PC with commissioning software.
Electronic gear	Calculation of a new output velocity for the motor movement based on the input velocity and the values of an adjustable gear ratio; calculated by the drive system.
EMCY object	Emergency Object
EMC	Electromagnetic compatibility
Encoder	Sensor that converts a measured distance or angle into an electrical signal. This signal is evaluated by the drive to determine the actual position of a shaft (rotor) or a driving unit.
Limit switch	Switches that signal overtravel of the permissible range of travel.
Power stage	The power stage controls the motor. The power stage generates current for controlling the motor on the basis of the positioning signals from the controller.

0198441113938, V2.00, 10.2011

Error	Discrepancy between a detected (computed, measured or signaled) value or condition and the specified or theoretically correct value or condition.
Fatal error	In the case of fatal error, the product is no longer able to control the motor so that the power stage must be immediately disabled.
Fault	Fault is a state that can be caused by an error. Further information can be found in the pertinent standards such as IEC 61800-7, ODVA Common Industrial Protocol (CIP).
Fault reset	A function used to restore the drive to an operational state after a detected error is cleared by removing the cause of the error so that the error is no longer active.
Error class	Classification of errors into groups. The different error classes allow for specific responses to errors, for example by severity.
Heartbeat	Used for unconfirmed connection acknowledgement messages from network devices.
HMI	Human Machine Interface
Life guarding	For monitoring the connection of an NMT master
Mapping	Assignment of object dictionary entries to PDOs
Node ID	Node address assigned to a device on the network.
NMT	Network Management (NMT), part of the CANopen communication profile; tasks include initialization of the network and devices, starting, stopping and monitoring of devices
Node guarding	Monitoring of the connection to the slave at an interface for cyclic data traffic.
Object dictionary	List of the parameters, values and functions available in the device. Each entry is uniquely referenced via index (16 bit) and subindex (8 bit).
Parameter	Device data and values that can be read and set (to a certain extent) by the user.
PDO	Process Data Object
Persistent	Indicates whether the value of the parameter remains in the memory after the device is switched off.
Quick Stop	The Quick Stop function can be used for fast deceleration of a movement in the case of an error or via a command.
R_PDO	Receive PDO
SDO	Service Data Object
SYNC object	Synchronization object
T_PDO	Transmit PDO
Warning	If the term is used outside the context of safety instructions, a warning alerts to a potential problem that was detected by a monitoring function. A warning does not cause a transition of the operating state.
Factory setting	Factory settings when the product is shipped

0198441113938, V2.00, 10.2011

10 Index

10

0198441113938, V2.00, 10.2011

D

E

0198441113938, V2.00, 10.2011

0198441113938, V2.00, 10.2011

N

O

P

0198441113938, V2.00, 10.2011

Q

R

S

0198441113938, V2.00, 10.2011

T

U

V

W

0198441113938, V2.00, 10.2011