

L-force *Drives*



Communication Manual

Servo Drives 930



931E

CANopen

This documentation applies to 931E servo inverters.

Document history

Material No.	Version			Description
13190599	2.0	02/2007	TD19	First edition



Tip!

Current documentation and software updates concerning Lenze products can be found on the Internet in the "Services & Downloads" area under

<http://www.Lenze.com>

Important note:

Software is provided to the user "as is". All risks regarding the quality of the software and any results obtained from its use remain with the user. The user should take appropriate security precautions against possible maloperation.

We do not accept any responsibility for direct or indirect damage caused, e.g. loss of profit, loss of orders or adverse commercial effects of any kind.

All trade names listed in this documentation are trademarks of their respective owners.

© 2007 Lenze GmbH & Co KG Kleinantriebe, Hans-Lenze-Straße 1, D-32699 Extertal

No part of this documentation may be reproduced or made accessible to third parties without written consent by Lenze GmbH & Co KG Kleinantriebe.

All information given in this documentation has been selected carefully and complies with the hardware and software described.

Nevertheless, discrepancies cannot be ruled out. We do not take any responsibility or liability for any damage that may occur.

Necessary corrections will be included in subsequent editions.

1	Preface	7
1.1	Introduction	7
1.2	About this Communication Manual	8
1.3	Legal regulations	9
2	Safety instructions	10
2.1	Persons responsible for safety	10
2.2	General safety instructions	11
2.3	Definition of notes used	12
3	Technical data	13
3.1	Communication data	13
4	Electrical installation	14
4.1	Wiring according to EMC	14
4.2	Electrical connections of CANopen	15
4.3	Connection of CAN bus slave	16
4.4	Connection of CAN bus master	17
5	CANopen communication	18
5.1	About CANopen	18
5.1.1	Structure of the CAN data telegram	18
5.1.2	Identifier	19
5.1.3	Node address (node ID)	19
5.1.4	User data	20
5.2	Parameter data transfer (SDO transfer)	21
5.2.1	Telegram structure	21
5.2.2	Reading parameters (example)	25
5.2.3	Writing parameters (example)	26
5.3	Process data transfer (PDO transfer)	27
5.3.1	Telegram structure	27
5.3.2	Available process data objects	27
5.3.3	Objects for PDO parameterisation	28
5.3.4	Description of the objects	37
5.3.5	Example of a process data telegram	39
5.3.6	Activation of the PDOs	40
5.4	Sync telegram	41
5.4.1	Telegram structure	41
5.4.2	Synchronisation of the process data	41
5.4.3	Description of the objects	42

i Contents

5.5	Network management (NMT)	43
5.5.1	Communication phases of the CAN network (NMT)	43
5.5.2	Telegram structure	44
5.6	Emergency telegram	46
5.6.1	Telegram structure	46
5.6.2	Description of the objects	48
5.7	Heartbeat telegram	49
5.7.1	Telegram structure	49
5.7.2	Description of the objects	51
5.8	Boot-up telegram	52
5.8.1	Telegram structure	52
6	Commissioning	53
6.1	Activation of CANopen	53
6.2	Speed control	54
6.2.1	Parameterising of a process data object (TPDO and RPDO)	54
6.2.2	Parameterising of the motor and the current controller	57
6.2.3	Parameterising of the speed control	58
6.2.4	Running through the state machine	59
6.3	Position control	61
6.3.1	Parameterising of the homing run	61
6.3.2	Running through the state machine	63
7	Parameter setting	67
7.1	Loading and saving of parameter sets	67
7.1.1	Overview	67
7.1.2	Description of the objects	69
7.2	Conversion factors (factor group)	70
7.2.1	Overview	70
7.2.2	Description of the objects	72
7.3	Power stage parameters	74
7.3.1	Overview	74
7.3.2	Description of the objects	74
7.4	Current controller and motor adaptation	76
7.4.1	Overview	76
7.4.2	Description of the objects	77
7.5	Speed controller	79
7.5.1	Overview	79
7.5.2	Description of the objects	79
7.6	Position controller (position control function)	80
7.6.1	Overview	80
7.6.2	Description of the objects	82

7.7	Analog inputs	85
7.7.1	Overview	85
7.8	Digital inputs and outputs	85
7.8.1	Overview	85
7.8.2	Description of the objects	85
7.9	Limit switches	86
7.9.1	Overview	86
7.9.2	Description of the objects	86
7.10	Device information	87
7.10.1	Description of the objects	87
8	Device control	88
8.1	State diagram	88
8.1.1	Overview	88
8.1.2	State diagram of the drive controller	89
8.1.3	States of the drive controller	91
8.1.4	State transitions of the drive controller	92
8.1.5	Control word	93
8.1.6	Controller state	96
8.1.7	Status word	97
9	Operating modes	99
9.1	Setting of the operating mode	99
9.1.1	Overview	99
9.1.2	Description of the objects	99
9.2	Speed control	101
9.2.1	Overview	101
9.2.2	Description of the objects	103
9.3	Homing	104
9.3.1	Overview	104
9.3.2	Description of the objects	105
9.3.3	Control of the homing run	106
9.4	Positioning	107
9.4.1	Overview	107
9.4.2	Description of the objects	108
9.4.3	Functional description	109
9.5	Synchronous position selection	111
9.5.1	Overview	111
9.5.2	Description of the objects	112
9.5.3	Functional description	114

i Contents

9.6	Torque control	117
9.6.1	Overview	117
9.6.2	Description of the objects	118
10	Appendix	119
10.1	Index table	119
11	Index	143

1 Preface

1.1 Introduction

The competitive situation in the mechanical and system engineering sector requires new means to optimise the production costs. This is why modular machine and system engineering is becoming increasingly more important, since individual solutions can now be set up easily and cost-effectively from a single modular system.

Lenze fieldbus systems in industrial applications

For an optimal communication between the single modules of a system, fieldbus systems are increasingly used for process automation. Lenze offers the following communication modules for the standard fieldbus systems:

- ▶ PROFIBUS-DP
- ▶ CANopen

Decision support

The decision for a fieldbus system depends on many different factors. The following overviews will help you to find the solution for your application.

PROFIBUS-DP

For bigger machines with bus lengths of more than 100 metres, INTERBUS or PROFIBUS-DP (PROFIBUS-**D**ecentralised **P**eriphery) are frequently used. The PROFIBUS-DP is always used together with a master control (PLC) – here the PROFIBUS master transmits e.g. the setpoints to the single PROFIBUS stations (e. g. Lenze controllers).

When using the data transfer rate of 1.5 Mbps typical for the PROFIBUS-DP, the sensors and actuators receive the process data. Due to the data transmission mode and the telegram overhead, a bus cycle time results at 1.5 Mbps, which is sufficient to control e. g. conveyors. If, for technical reasons, the process data must be transmitted faster to the sensors and actuators, the PROFIBUS can also be operated with a data transmission rate of maximally 12 Mbps.

CANopen

CANopen is a communication protocol specified to the CiA (CAN in Automation) user group. Lenze can provide communication modules for controls with CANopen masters. These modules are compatible with the specification DS 301 V4.01.

1**Preface**

About this Communication Manual

1.2**About this Communication Manual****Target group**

This manual is directed at all persons who carry out the dimensioning, installation, commissioning and settings of the 931 series drive controllers.

Together with the catalogue, it provides the project planning basis for the manufacturer of plants and machinery.

Contents

The CAN manual supplements the software manual and mounting instructions which are included in the scope of supply:

- ▶ The features and functions are described in detail.
- ▶ It provides detailed information on the possible applications.
- ▶ Parameter setting is explained with the help of examples.
- ▶ In case of doubt, the supplied mounting instructions are always valid.

How to find information

- ▶ The table of contents and the index help you to find all information about a certain topic.
- ▶ Descriptions and data on other Lenze products can be found in the corresponding catalogues, operating instructions and manuals.
- ▶ You can request Lenze documents from your responsible Lenze sales partner or download it as a PDF file from the Internet.

1.3 Legal regulations

Labelling	Nameplate	CE identification	Manufacturer
	Lenze drive controllers are definitely identified by the contents of the nameplate.	In compliance with the EC Low-Voltage Directive	Lenze GmbH & Co KG Kleinantriebe Postfach 10 13 52 D-31763 Hameln
Application as directed	<p>931E servo inverters</p> <ul style="list-style-type: none"> ● must only be operated under the operating conditions prescribed in these Instructions. ● are components <ul style="list-style-type: none"> – for open and closed loop control of variable speed drives with synchronous motors. – for installation in a machine – for assembly with other components to form a machine. ● are electric units for the installation into control cabinets or similar closed electrical operating areas. ● comply with the requirements of the Low-Voltage Directive. ● are not machines for the purpose of the Machinery Directive. ● are not to be used as domestic appliances, but only for industrial purposes. <p>Drive systems with 931E servo inverters</p> <ul style="list-style-type: none"> ● comply with the EMC Directive if they are installed according to the guidelines of CE-typical drive systems. ● can be used <ul style="list-style-type: none"> – for operation on public and non-public mains – for operation in industrial premises. ● The user is responsible for the compliance of his application with the EC Directives. <p>Any other use shall be deemed as inappropriate!</p>		
Liability	<ul style="list-style-type: none"> ● The information, data, and notes in these instructions met the state of the art at the time of printing. Claims on modifications referring to controllers which have already been supplied cannot be derived from the information, illustrations, and descriptions. ● The specifications, processes, and circuitry described in these Instructions are for guidance only and must be adapted to your own specific application. Lenze does not take responsibility for the suitability of the process and circuit proposals. ● Lenze does not accept any liability for damage and operating interference caused by: <ul style="list-style-type: none"> – disregarding the Operating Instructions – unauthorised modifications to the drive controllers – operating errors – improper working on and with the drive controllers 		
Warranty	<ul style="list-style-type: none"> ● Terms of warranty: see Sales and Delivery Conditions of Lenze GmbH & Co KG Kleinantriebe. ● Warranty claims must be made to Lenze immediately after detecting the deficiency or fault. ● The warranty is void in all cases where liability claims cannot be made. 		
Disposal	Material	Recycle	Dispose
	Metal	●	-
	Plastic	●	-
	Assembled PCBs	-	●

2 Safety instructions

Persons responsible for safety

2 Safety instructions

2.1 Persons responsible for safety

Operator

An operator is any natural or legal person who uses the drive system or on behalf of whom the drive system is used.

The operator or his safety officer is obliged

- ▶ to ensure the compliance with all relevant regulations, instructions and legislation.
- ▶ to ensure that only qualified personnel work on and with the drive system.
- ▶ to ensure that the personnel have the Operating Instructions available for all work.
- ▶ to ensure that all unqualified personnel are prohibited from working on and with the drive system.

Qualified personnel

Qualified personnel are persons who - due to their education, experience, instructions, and knowledge about relevant standards and regulations, rules for the prevention of accidents, and operating conditions - are authorised by the person responsible for the safety of the plant to perform the required actions and who are able to recognise potential hazards.

(Definition for skilled personnel to VDE 105 or IEC 364)

2.2 General safety instructions

- ▶ These safety instructions are not claimed to be complete. In case of questions and problems, please contact your Lenze representative.
- ▶ At the time of delivery, the drive controller meets the state of the art and basically ensures safe operation.
- ▶ The information given in this manual refers to the specified hardware and software versions of the modules.
- ▶ The drive controller is a source of danger if
 - unqualified personnel work with and on the drive controller.
 - the drive controller is used inappropriately.
- ▶ The procedural notes and circuit details given in this manual are suggestions and their transferability to the respective application has to be checked.
- ▶ Ensure by appropriate measures that there is no risk of injury or death to persons or risk of damage to property in the event of a drive controller failure.
- ▶ Operate the drive system only when it is in a proper state.
- ▶ Retrofittings, modifications or redesigns of the drive controller are basically prohibited. Lenze must be contacted in all cases.

2 Safety instructions

Definition of notes used

2.3 Definition of notes used

The following signal words and symbols are used in this documentation to indicate dangers and important information:

Safety instructions

Structure of safety instructions:



Danger!

(characterises the type and severity of danger)

Note

(describes the danger and gives information about how to prevent dangerous situations)

Pictograph and signal word	Meaning
Danger!	Danger of personal injury through dangerous electrical voltage. Reference to an imminent danger that may result in death or serious personal injury if the corresponding measures are not taken.
Danger!	Danger of personal injury through a general source of danger. Reference to an imminent danger that may result in death or serious personal injury if the corresponding measures are not taken.
Stop!	Danger of property damage. Reference to a possible danger that may result in property damage if the corresponding measures are not taken.

Application notes

Pictograph and signal word	Meaning
Note!	Important note to ensure trouble-free operation
Tip!	Useful tip for simple handling
	Reference to another documentation

3 Technical data

3.1 Communication data

Communication		
Communication profile	DS 301, DSP 402	
Communication medium		RS232
Network topology		Without repeater: line / with repeaters: line or tree
CAN node		Slave
Baud rate (in kbps)		10, 20, 50, 100, 125, 250, 500
Max. cable length per bus segment		1200 m (dependent on baud rate and cable type used)
Bus connection		RJ45

4 Electrical installation

Wiring according to EMC

4 Electrical installation

4.1 Wiring according to EMC

General notes	<ul style="list-style-type: none"> ● The electromagnetic compatibility of the drive depends on the type of installation and the care taken. Especially observe: <ul style="list-style-type: none"> – Assembly – Shielding – Earthing ● In the case of differing installations, the evaluation of the conformity to the EMC Directive requires the system to be checked for compliance with the EMC limit values. This applies, for instance, to: <ul style="list-style-type: none"> – Use of unshielded cables ● The user is responsible for compliance with the EMC Directive. <ul style="list-style-type: none"> – If the following measures are observed, you can assume that no EMC problems will occur during operation and that the EMC Directive / EMC law is met. – If devices are operated close to the system which do not meet the CE requirements regarding the noise immunity according to EN 61000-4-2, these devices may be electromagnetically impaired by the drive.
Assembly	<ul style="list-style-type: none"> ● Electrical contacting of the mounting plate: <ul style="list-style-type: none"> – Mounting plates with conductive surface (galvanised or stainless steel) enable a permanent contact. – Painted plates are not suitable for an EMC-compliant installation. ● If you use several mounting plates: <ul style="list-style-type: none"> – Contact the mounting plates to each other over a large area (e.g. with copper strips). ● Route signal cables separately from mains cables. ● Route the cables as close as possible to the reference potential. Freely suspended cables act like aerials.
Shielding	<ul style="list-style-type: none"> ● If possible, only use braided cables. ● The overlap rate of the shield should be higher than 80%. ● Always use metal or metallised connectors for the serial data cable coupling. Connect the shield of the data cable to the connector shell. ● Use metal cable clamps to attach the shield braid. ● Connect the shield to the shield bus in the control cabinet. ● Connect the shields of analog control cables at one end.
Earthing	<ul style="list-style-type: none"> ● Electrical contacting of the mounting plate: <ul style="list-style-type: none"> – Mounting plates with conductive surface (galvanised or stainless steel) enable a permanent contact. – Painted plates are not suitable for an EMC-compliant installation. ● If you use several mounting plates: <ul style="list-style-type: none"> – Contact the mounting plates to each other over a large area (e.g. with copper strips). ● Route signal cables separately from mains cables. ● Route the cables as close as possible to the reference potential. Freely suspended cables act like aerials.

4.2 Electrical connections of CANopen

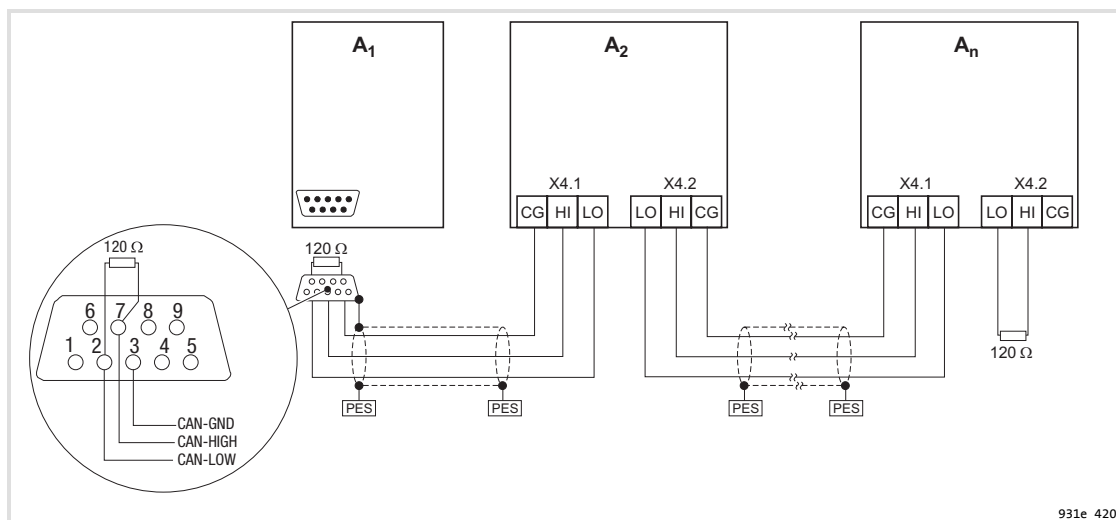


Fig. 1 Basic wiring of CANopen with Sub-D connector to the master

A₁ Node 1 - master (e.g. PLC)

A₂ Node 2 - slave (e.g. drive controller 931E)

A_n Node n - slave, n = max. 128

Specification of the transmission cable

Please observe our recommendations for signal cables.

Bus cable specification	
Cable resistance	135 - 165 Ω/km, (f = 3 - 20 MHz)
Capacitance per unit length	≤ 30 nF/km
Loop resistance	< 110 Ω/km
Wire diameter	> 0.64 mm
Wire cross-section	> 0.34 mm ²
Wires	double twisted, insulated and shielded

- ▶ Connection of the bus terminating resistors:
 - One resistor of 120 Ω each at the first and last bus node
- ▶ Communication protocol
 - CANopen (CAL-based communication profile DS 301/DSP 402)
- ▶ Bus extension:
 - 25 m for max. data transfer rate of 1 Mbps
 - Up to 1 km for reduced data transfer speed
- ▶ Signal level according to ISO 11898
- ▶ Up to 128 bus nodes possible
- ▶ Access to all Lenze parameters

4 Electrical installation

Connection of CAN bus slave

4.3 Connection of CAN bus slave

Features

- ▶ Parameter selection
- ▶ Data exchange between drive controllers
- ▶ Connection of operator and input devices
- ▶ Connection of higher-level controls
- ▶ Baud rates of 125, 250, 500 kBaud

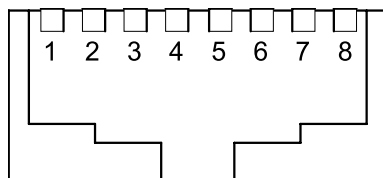


Stop!

An external 120 Ω terminating resistor is required to terminate the bus system.

Connection plan for RJ45 socket

X4.1 / X4.2



931E-001.iso

Fig. 2 Connection of CAN bus (X4.1, X4.2)

Pin no.	Meaning	Comment
1	CAN-HIGH	CAN-HIGH (high is dominant)
2	CAN-LOW	CAN-LOW (low is dominant)
3	CAN-GND	CAN ground
4	—	Reserved
5	—	Reserved
6	CAN-SHLD	CAN shield (hardware version 1.1 and higher)
7	CAN-GND	CAN ground
8	—	Reserved



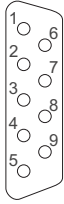
Tip!

An RJ45 bus terminating connector is available for the 931E drive controllers. Please contact Lenze.

4.4 Connection of CAN bus master

The below table shows the assignment of a 9-pin Sub-D socket such as provided by most CAN masters for the connection of field devices.

Connection of the CAN bus to a 9-pin Sub-D socket

View	Pin	Signal	Explanation
	1	—	Reserved
	2	CAN-LOW	CAN-LOW (low is dominant)
	3	CAN-GND	CAN ground
	4	—	Reserved
	5	(CAN-SHLD)	Optional CAN shield
	6	(GND)	Optional ground
	7	CAN-HIGH	CAN-HIGH (high is dominant)
	8	—	Reserved
	9	(CAN-V+)	Optional external CAN voltage supply

Tab. 1 CAN Sub-D socket

5 CANopen communication

About CANopen

Structure of the CAN data telegram

5 CANopen communication

5.1 About CANopen

The CANopen protocol is a standardised layer 7 protocol for the CAN bus. This layer is based on the CAN application layer (CAL), which has been developed as a universal protocol.

In practice, however, it became clear that applications with CAL were too complex for the user. CANopen is a uniform, easy-to-use structure which has been developed to provide a connection for CAN devices from different manufacturers.

5.1.1 Structure of the CAN data telegram

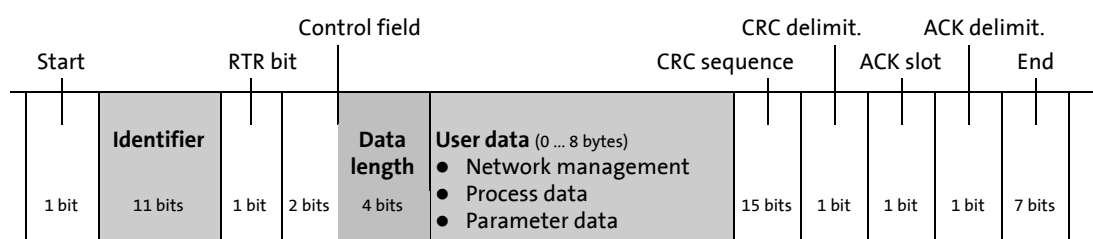


Fig. 3 Basic structure of the CAN telegram



Note!

To the user, only the identifier, the data length and the user data are relevant. All other data of the CAN telegram is automatically processed by the system.

5.1.2 Identifier

The principle of the CAN communication is based on a message-oriented data exchange between one sender and many receivers. All nodes can send and receive quasi-simultaneously.

The *identifier* in the CAN telegram - also called *COB ID (communication object identifier)* - is used to control which node is to receive a sent message. In addition to the addressing, the identifier contains information on the priority of the message and on the type of the user data.

With the exception of the network management and the sync telegram, the identifier contains the node address of the drive:

Identifier (COB ID) = basic identifier + adjustable node address (node ID)

The identifier assignment is specified in the CANopen protocol.

The ex works default setting of the basic identifier is:

Object	Direction		Basic identifier
	From the drive	To the drive	Hex
NMT			0
Sync			80
Emergency	X		80
PDO1 (process data channel 1)	TPDO1		180
	RPDO1	X	200
PDO2 (process data channel 2)	TPDO2		280
	RPDO2	X	300
SDO1 (parameter data channel 1)		X	580
			600
Heartbeat/boot-up	X		700

5.1.3 Node address (node ID)

Each node of the CAN network must be assigned with a node address (also called *node ID*) within the valid address range for unambiguous identification.

- ▶ A node address may not be assigned more than once within a network.

5 CANopen communication

About CANopen

User data

5.1.4 User data

The master and the drive controller communicate with each other by exchanging data telegrams via the CAN bus.

The user data range of the CAN telegram contains network management data, parameter data or process data:

- ▶ Network management data (NMT data)
 - Network service: E.g. all CAN nodes can be addressed at the same time.
- ▶ Process data (PDO, process data objects)
 - Process data is transferred via the process data channel.
 - Process data can be used to control the drive controller.
 - The master can directly access the process data. The data is, for instance, directly assigned to the I/O area of the master. It is necessary that the control and the drive controller can exchange data within a very short time interval. For this purpose, small amounts of data can be transferred cyclically.
 - Process data is not stored in the drive controller.
 - Process data is transferred between the master and the drive controllers to ensure a continuous exchange of current input and output data.
 - Examples for process data are, for instance, setpoints and actual values.
- ▶ Parameter data (SDO, service data objects)
 - Parameters are set, for instance, for the initial system set-up during commissioning or when the material is changed on a production machine.
 - Parameter data is transferred by means of so-called SDOs via the parameter data channel. The transfer is acknowledged by the receiver, i.e. the sender gets a feedback about the transfer being successful or not.
 - The parameter data channel enables the access to all CANopen indexes.
 - Parameter changes are automatically stored in the drive controller.
 - In general, the transfer of parameters is not time-critical.
 - Examples for parameter data are, for instance, operating parameters, diagnostic information and motor data.

5.2 Parameter data transfer (SDO transfer)

5.2.1 Telegram structure

The telegram for parameter data has the following structure:

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
Error code									

► The following subchapters explain in detail the different parts of the telegram.

Identifier

11 bits	4 bits	User data (up to 8 bytes)							
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4

With the exception of the network management and the sync telegram, the identifier contains the node address of the drive:

Identifier (COB ID) = basic identifier + adjustable node address (node ID)

The identifier assignment is specified in the CANopen protocol.

The ex works default setting of the basic identifier is:

Object	Direction		Basic identifier
	From the drive	To the drive	Hex
SDO (parameter data channel)	X		580
		X	600

CANopen communication

Parameter data transfer (SDO transfer)

Telegram structure

Command code

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
						Error code			

The command code contains the services for writing and reading parameters and the information on the length of the user data.

Structure of the command code:

	Bit 7 MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 LSB	
Write command code	CS			0	Length		e	s	Comment CS: command specifier User data length is coded in bits 2 and 3: • 00 = 4 bytes • 01 = 3 bytes • 10 = 2 bytes • 11 = 1 byte
Write command / write request	0	0	1	0	x	x	1	1	
Response to write command / write response	0	1	1	0	x	x	0	0	
Read command code	CS			0	Length		e	s	
Read command / read request	0	1	0	0	x	x	0	0	
Response to read command / read response	0	1	0	0	x	x	1	1	
Error command code	CS			0	Length		e	s	
Error response	1	0	0	0	0	0	0	0	

The command code specifies whether a value is to be read or written. The command code also determines the data length (1 byte, 2 bytes, 4 bytes).

	4-byte data (5th ... 8th byte)	2-byte data (5th and 6th byte)	1-byte data (5th byte)
Write command code	hex	hex	hex
Write command / write request (Send parameters to the drive)	23	2B	2F
Response to write command / write response (Response of the drive controller to the write request (acknowledgement))	60	60	60
Read command code			
Read command / read request (Request to read a parameter from the drive controller)	40	40	40
Response to read command / read response (Response to the read request with the actual value)	43	4B	4F
Error command code			
Error response (The drive controller signals a communication error)	80	80	80

Index low byte / index high byte

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4

The object to be addressed is contained in bytes 2 and 3 of the telegram.

- ▶ The value for the index is split up into low byte and high byte and entered in the left-justified Intel format.

Subindex

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4

- ▶ If an object (e.g. controller parameter) consists of several sub-objects, the sub-objects are addressed via subindexes. The number of the corresponding subindex is entered in byte 4 of the telegram. (See following tables for sub-objects).
- ▶ If an object has no sub-objects, the value "0" is entered in byte 4 of the telegram. (See following sub-object tables).

Data (data 1 ... data 4)

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4

For the data of the parameter up to 4 bytes (data 1 ... data 4) are available.

The data is represented in the left-justified Intel format with data 1 as the LSB and data 4 as the MSB.

CANopen communication

Parameter data transfer (SDO transfer)

Telegram structure

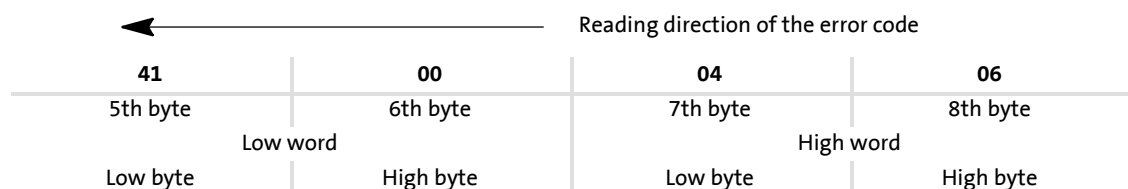
Error code (F0 ... F3)

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	F0	F1	F2	F3
						Error code			

- ▶ **Byte 1:**
Code **80_h** in the **command code** byte indicates that an error has occurred.
- ▶ **Bytes 2, 3 and 4:**
These bytes contain the **index** (bytes 2 and 3) and the **subindex** (byte 4) at which an error occurred.
- ▶ **Bytes 5 to 8:**
The data bytes 5 to 8 contain the **error code**. The error code is represented opposite to the direction of reading.

Example:

The representation of the error code **06 04 00 41_h** in bytes 5 to 8



The below table lists the meanings of the error codes:

Error code				Explanation
F3	F2	F1	F0	
06	01	00	00	Access to object is not supported
06	01	00	01	Attempt to read a write-only object
06	01	00	02	Attempt to write to a read-only object
06	02	00	00	Object does not exist in the object directory
06	04	00	41	Object cannot be mapped to the PDO
06	04	00	42	The number and length of objects to be mapped would exceed PDO length.
06	07	00	10	Data type does not match, length of service parameter does not match
06	07	00	12	Data type does not match, length of service parameter is too large
06	07	00	13	Data type does not match, length of service parameter is too small
06	09	00	11	Subindex does not exist
06	09	00	30	Value range of parameter exceeded
06	09	00	31	Parameter values too large
06	09	00	32	Parameter values too small
08	00	00	20	Data cannot be transferred/saved to the application.
08	00	00	21	Data cannot be transferred/saved to the application due to local control.
08	00	00	22	Data cannot be transferred/saved to the application due to current device state.

5.2.2 Reading parameters (example)

Problem

The numerator setting (object 6093_01) of the drive controller with node address 1 is to be read via the parameter channel.

Telegram to the drive controller

	Value	Info
Identifier	= Basic identifier + node address = 600 + 1 = 601 _h	<ul style="list-style-type: none"> Basic identifier for parameter channel = 600_h Node address = 1
Data length	= 08	
Command code	= 40 _h	<ul style="list-style-type: none"> “Read request” command (request to read a parameter)
Index	= 6093 _h	<ul style="list-style-type: none"> Index of the position_factor
Subindex	= 1	<ul style="list-style-type: none"> Subindex = 1 (numerator)
Data 1	= 00 _h	<ul style="list-style-type: none"> Read request only
Data 2	= 00 _h	
Data 3	= 00 _h	
Data 4	= 00 _h	
Data 1 ... 4	= 00 00 00 00 _h	

11 bits	4 bits	User data							
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
601 _h	08 _h	40 _h	93 _h	60 _h	01 _h	00 _h	00 _h	00 _h	00 _h

Telegram from the drive controller

	Value	Info
Identifier	= Basic identifier + node address = 580 + 1 = 581 _h	<ul style="list-style-type: none"> Basic identifier for parameter channel = 580_h Node address = 1
Data length	= 08	
Command code	= 43 _h	<ul style="list-style-type: none"> “Read response” command (response to the read request with the actual value)
Index	= 6093 _h	<ul style="list-style-type: none"> Index of the position_factor
Subindex	= 1	<ul style="list-style-type: none"> Subindex = 1 (numerator)
Data 1	= C0 _h	<ul style="list-style-type: none"> Assumption: The set numerator value is 00 03 4B C0_h (216000_d).
Data 2	= 4B _h	
Data 3	= 03 _h	
Data 4	= 00 _h	
Data 1 ... 4	= C0 4B 03 00 _h	

11 bits	4 bits	User data							
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
581 _h	08 _h	43 _h	93 _h	60 _h	01 _h	C0 _h	4B _h	03 _h	00 _h

5

CANopen communication

Parameter data transfer (SDO transfer)

Writing parameters (example)

5.2.3

Writing parameters (example)**Problem**

The numerator (object 6093_01) of the drive controller with node address 1 is to be set to 216000 via the SDO (parameter data channel).

Telegram to the drive controller

	Value	Info
Identifier	= Basic identifier + node address = 600 + 1 = 601 _h	<ul style="list-style-type: none"> Basic identifier for parameter channel = 600_h Node address = 1
Data length	= 08	
Command code	= 23 _h	<ul style="list-style-type: none"> “Write request” command (send parameter to the drive)
Index	= 6093 _h	<ul style="list-style-type: none"> Index of the position_factor
Subindex	= 1	<ul style="list-style-type: none"> Subindex = 1 (numerator)
Data 1	= C0 _h	<ul style="list-style-type: none"> Assumption: The numerator value to be set is to be 00 03 4B C0_h (216000_d).
Data 2	= 4B _h	
Data 3	= 03 _h	
Data 4	= 00 _h	
Data 1 ... 4	= C0 4B 03 00 _h	

11 bits	4 bits	User data							
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
601 _h	08 _h	23 _h	93 _h	60 _h	01 _h	C0 _h	4B _h	03 _h	00 _h

Telegram from the drive controller (acknowledgement for faultless execution)

	Value	Info
Identifier	= Basic identifier + node address = 580 + 1 = 581 _h	<ul style="list-style-type: none"> Basic identifier for parameter channel = 580_h Node address = 1
Data length	= 08	
Command code	= 60 _h	<ul style="list-style-type: none"> “Write response” command (acknowledgement from the drive controller)
Index	= 6093 _h	<ul style="list-style-type: none"> Index of the position_factor
Subindex	= 1	<ul style="list-style-type: none"> Subindex = 1 (numerator)
Data 1 ... 4	= 00 00 00 00 _h	<ul style="list-style-type: none"> Acknowledgement only

11 bits	4 bits	User data							
Identifier	Data length	Command code	Index low byte	Index high byte	Subindex	Data 1	Data 2	Data 3	Data 4
581 _h	08 _h	60 _h	93 _h	60 _h	01 _h	00 _h	00 _h	00 _h	00 _h

5.3 Process data transfer (PDO transfer)

Process data objects (PDOs) can be used, for instance, for the fast event-controlled transfer of data. The PDO transfers one or several parameters specified in advance. Unlike with an SDO, the transfer of a PDO is not acknowledged. After the PDO activation, all receivers must therefore always be able to process any arriving PDOs. This usually means a considerable software load on the master. However, this disadvantage is compensated by the advantage that the master does not need to cyclically poll the parameters transferred by a PDO, which results in a significant reduction of the CAN bus load.

Example:

The master wants to know when the drive controller has completed the positioning from A to B.

When SDOs are used for this purpose, the master continuously (e.g. every millisecond) has to poll the **status word** object, i.e. the load on the bus is high.

When a PDO is used, right from the start of the application the drive controller is parameterised in such a way that it transmits a PDO containing the **status word** object as soon as the **status word** object changes.

Instead of polling continuously, the master automatically receives a corresponding message as soon as the event has occurred.

The following types of process data telegram are distinguished

- ▶ Process data telegrams **to** the drive controller: Receive PDO (RPDOx)
- ▶ Process data telegrams **from** the drive controller: Transmit PDO (TPDOx)

5.3.1 Telegram structure

The telegram for process data has the following structure:

11 bits	4 bits	User data (up to 8 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

5.3.2 Available process data objects

The drive controller is provided with two transmit and two receive PDOs.

Almost all objects of the object directory can be entered in (mapped to) the PDOs, i.e. the PDO contains for instance the actual speed value or actual position value as data. The drive controller must know in advance which data is to be transferred because the PDO only contains user data and no information about the type of the parameter.

In this way almost all kinds of data telegrams can be defined. The settings required are described in the following chapters.

5

CANopen communication

Process data transfer (PDO transfer)

Objects for PDO parameterisation

5.3.3 Objects for PDO parameterisation

Two transmit PDOs (TPDO) and two receive PDOs (RPDO) are available in the drive controller. The different objects of the PDOs are identical.

1. Transmit PDO

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1800 _h	Transmit PDO1 Communication Parameter								
0	number_of_entries		00 _h {1 _h }	04 _h	REC	UINT8	RO	—	
									Maximum number of supported subindexes.
			03 _h						3 subindexes are supported.
1	COB-ID_used_by_PDO	00000181 _h	00000181 _h {1 _h }	000001FF _h	—	UINT32	RW	—	
									Identifier of transmit PDO1, (180 _h + node address). For processing, bits 30 and 31 must be set (parameterisation of mapping).
			Bit no. Value						
			0 - 10 x						11-bit identifier
			11 - 28 0						The extended identifier (bit 29) is not supported.
			29 0						Each bit of this range must be "0".
			30 0						RTR of this PDO is permitted (Lenze).
			1						RTR of this PDO is not permitted (unadjustable).
			31 0						PDO is active
			1						PDO is inactive
2	transmission_type	255	0 {1}	240, 254, 255	—	UINT8	RW	—	
									Setting of the transmission mode
			0						Function is switched off
			n = 1 ... 240						By entering a value n, this PDO is accepted with every n-th sync.
			n = 254, 255						Event-controlled transmission mode
3	inhibit_time	0	0 {0.1 ms}	65535	—	UINT16	RW	—	
									Setting of the minimum delay time between two PDOs. The time can only be changed if the PDO is not active (subindex 1, bit 31 = 1)

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
1A00 _h	Transmit PDO1 Mapping Parameter							
	0 number_of_mapped_objects		00 _h {1 _h }	04 _h	REC	UINT32	RW	—
								Maximum number of supported subindexes.
				01 _h				1 subindex is supported.
	1 first_mapped_object	60410010 _h		{1 _h }		—	UINT32	RW
								Entry of the COB ID of the first mapped object.
2 second_mapped_object					—	UINT32	RW	—
								Not supported.
...								
4 fourth_mapped_object					—	UINT32	RW	—
								Not supported.

5

CANopen communication

Process data transfer (PDO transfer)

Objects for PDO parameterisation

2. Transmit PDO

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1801 _h	Transmit PDO2 Communication Parameter								
0	number_of_entries		00 _h	{1 _h }	04 _h	REC	UINT8	RO	—
									Maximum number of supported subindexes
			03 _h						3 subindexes are supported.
1	COB-ID_used_by_PDO	00000281 _h	00000281 _h	{1 _h }	000002FF _h	—	UINT32	RW	—
									Identifier of transmit PDO2, (280 _h + node address). For processing, bits 30 and 31 must be set (parameterisation of mapping).
			Bit no.	Value					
			0 - 10	x					11-bit identifier
			11 - 28	0					The extended identifier (bit 29) is not supported.
			29	0					Each bit of this range must be "0".
			30	0					RTR of this PDO is permitted (Lenze)
				1					RTR of this PDO is not permitted (unadjustable)
			31	0					PDO is active
				1					PDO is inactive
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—
									Setting of the transmission mode
			0						Function is switched off
			n = 1 ... 240						By entering a value n, this PDO is accepted with every n-th sync.
			n = 254, 255						Event-controlled transmission mode
3	inhibit_time	0	0	{0.1 ms}	65535	—	UINT16	RW	—
									Setting of the minimum delay time between two PDOs. The time can only be changed if the PDO is not active (subindex 1, bit 31 = 1)

Index	Name	Possible settings		Characteristics			
		Lenze	Selection	Description			
1A01 _h	Transmit PDO2 Mapping Parameter						
	0 number_of_mapped_objects		00 _h {1 _h } 04 _h	REC	UINT32	RW	—
							Maximum number of supported subindexes.
			02 _h				2 subindexes are supported.
	1 first_mapped_object	60410010 _h	{1 _h }	—	UINT32	RW	—
							Entry of the COB ID of the first mapped object.
	2 second_mapped_object	60610008 _h	{1 _h }	—	UINT32	RW	—
						Entry of the COB ID of the second mapped object.	
3 third_mapped_object				—	UINT32	RW	—
							Not supported.
4 fourth_mapped_object				—	UINT32	RW	—
							Not supported.

5

CANopen communication

Process data transfer (PDO transfer)

Objects for PDO parameterisation

1. Receive PDO

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1400 _h	Receive PDO1 Communication Parameter								
0	number_of_entries		00 _h	{1 _h }	04 _h	REC	UINT8	RO	—
									Maximum number of supported subindexes
			02 _h						2 subindexes are supported.
1	COB-ID_used_by_PDO	00000201 _h	00000201 _h	{1 _h }	000002FF _h	—	UINT32	RW	—
									Identifier of receive PDO1 (200 _h + node address) For processing, bits 30 and 31 must be set (parameterisation of mapping).
			Bit no.	Value					
			0 - 10	x					11-bit identifier
			11 - 28	0					The extended identifier (bit 29) is not supported.
			29	0					Each bit of this range must be "0".
			30	0					RTR of this PDO is permitted (Lenze) RTR = remote transmission request
				1					RTR of this PDO is not permitted (unadjustable)
			31	0					PDO is active
				1					PDO is inactive
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—
									Setting of the transmission mode
			0						Function is switched off
			n = 1 ... 240						By entering a value n, this PDO is accepted with every n-th sync.
			n = 254, 255						Event-controlled transmission mode, PDO is accepted immediately

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
1600 _h	Receive PDO1 Mapping Parameter							
	0 number_of_mapped_objects		00 _h {1 _h }	04 _h	REC	UINT32	RW	—
								Maximum number of supported subindexes.
				01 _h				1 subindex is supported.
	1 first_mapped_object	60400010 _h		{1 _h }		—	UINT32	RW
								Entry of the COB ID of the first mapped object.
2 second_mapped_object					—	UINT32	RW	—
								Not supported.
...								
4 fourth_mapped_object					—	UINT32	RW	—
								Not supported.

5

CANopen communication

Process data transfer (PDO transfer)

Objects for PDO parameterisation

2. Receive PDO

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1401 _h	Receive PDO2 Communication Parameter								
0	number_of_entries		00 _h	{1 _h }	04 _h	REC	UINT8	RO	—
									Maximum number of supported subindexes
			02 _h						2 subindexes are supported.
1	COB-ID_used_by_PDO	00000301 _h	00000301 _h	{1 _h }	000003FF _h	—	UINT32	RW	—
									Identifier of receive PDO2 (300 _h + node address) For processing, bits 30 and 31 must be set (parameterisation of mapping).
			Bit no.	Value					
			0 - 10	x					11-bit identifier
			11 - 28	0					The extended identifier (bit 29) is not supported.
			29	0					Each bit of this range must be "0".
			30	0					RTR of this PDO is permitted (Lenze)
				1					RTR = remote transmission request
				1					RTR of this PDO is not permitted (unadjustable)
			31	0					PDO is active
				1					PDO is inactive
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—
									Setting of the transmission mode
			0						Function is switched off
			n = 1 ... 240						By entering a value n, this PDO is accepted with every n-th sync.
			n = 254, 255						Event-controlled transmission mode, PDO is accepted immediately

Index	Name	Possible settings		Characteristics			
		Lenze	Selection	Description			
1601 _h	Receive PDO2 Mapping Parameter						
	0 number_of_mapped_objects		00 _h {1 _h } 04 _h	REC	UINT32	RW	—
							Maximum number of supported subindexes.
			02 _h				2 subindexes are supported.
	1 first_mapped_object	60400010 _h	{1 _h }	—	UINT32	RW	—
							Entry of the COB ID of the first mapped object.
	2 second_mapped_object	60600008 _h	{1 _h }	—	UINT32	RW	—
							Entry of the COB ID of the second mapped object.
	3 third_mapped_object			—	UINT32	RW	—
							Not supported.
4 fourth_mapped_object			—	UINT32	RW	—	
						Not supported.	

5

CANopen communication

Process data transfer (PDO transfer)

Objects for PDO parameterisation

1. Transmit masking

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
2014 _h	Transmit PDO1 Mask								
	0	number_of_entries				ARR	UINT8	RO	—
									Maximum number of supported subindexes
	1	tpdo1_transmit_mask_low	FFFFFFF _h	00000000 _h {1 _h }	FFFFFFF _h	—	UINT32	RW	—
								Mask for masking out individual bits of the PDOs.	
	2	tpdo1_transmit_mask_high	FFFFFFF _h	00000000 _h {1 _h }	FFFFFFF _h	—	UINT32	RW	—
									Mask for masking out individual bits of the PDOs.

2. Transmit masking

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
2015 _h	Transmit PDO2 Mask								
	0	number_of_entries				ARR	UINT8	RO	—
									Maximum number of supported subindexes
	1	tpdo2_transmit_mask_low	FFFFFFF _h	00000000 _h {1 _h }	FFFFFFF _h	—	UINT32	RW	—
								Mask for masking out individual bits of the PDOs.	
	2	tpdo2_transmit_mask_high	FFFFFFF _h	00000000 _h {1 _h }	FFFFFFF _h	—	UINT32	RW	—
									Mask for masking out individual bits of the PDOs.

5.3.4 Description of the objects

Identifier of the PDO (COB_ID_used_by_PDO)

The identifier on which the respective PDO is to be sent or received must be entered in the **COB_ID-used_by_PDO** object. If bit 31 is set, the respective PDO is deactivated. This is the default setting for all PDOs. In addition, bit 30 (no RTR allowed) must be set for every access.

The COB ID can only be changed if the PDO is deactivated, i.e. if bit 31 is set. For changing the COB ID, you therefore have to keep to the following sequence:

- ▶ Read out the COB ID
- ▶ Write the read-out COB ID + C0000000_h
- ▶ Write the new COB ID + C0000000_h
- ▶ Write the new COB ID, the PDO is active again.

Transmission mode (transmission_type and inhibit_time)

For each PDO, the event leading to a message being sent (transmit PDO) or evaluated (receive PDO) can be defined:

Value	Meaning	Permitted for
00 _h - F0 _h	Sync telegram The numerical value specifies how many sync telegrams are ignored between two transmissions before the PDO is - sent (TPDO) or - evaluated (RPDO).	TPDO RPDO
FE _h	Cyclic The TPDO is cyclically updated and sent by the drive controller. The time interval is specified with the inhibit_time object. RPDOS, however, are evaluated immediately after the receipt.	TPDO (RPDO)
FF _h	Change The TPDO is sent if at least one bit of the PDO data has changed. The inhibit_time can be used to additionally specify the minimum time interval (in 100 μs steps) between the transmission of two PDOs.	TPDO

The use of all other values is not permitted.

Number of objects to be transferred (number_of_mapped_objects)

This object specifies how many objects are to be mapped into the corresponding PDO. The following restrictions have to be taken into account:

- ▶ It is not possible to map more than 4 objects per PDO
- ▶ A PDO can have a maximum of 64 bits (8 bytes).

CANopen communication

Process data transfer (PDO transfer)

Description of the objects

Objects to be transferred (first_mapped_object ... fourth_mapped_object)

For every object to be contained in the PDO, the drive controller must know the corresponding index, subindex and length. The specified length must be identical to the length specified in the object dictionary. It is not possible to map parts of an object.

The mapping information has the following format:

Index	Subindex	Length
16 bits	8 bits	8 bits

- ▶ Index: Main index of the object to be mapped (hex)
- ▶ Subindex: Subindex of the object to be mapped (hex)
- ▶ Length: Length of the object (hex)

The following mandatory procedure serves to simplify the mapping:

1. The number of the mapped objects is set to 0.
2. The first_mapped_object ... fourth_mapped_object parameters can be written (the total length of all objects is not relevant at this time).
3. The number of the mapped objects is set to a value between 1 ... 4. The total length of these objects must not exceed 64 bits.

Masking (transmit_mask_high and transmit_mask_low)

If "change" is selected for the **transmission_type**, the TPDO is always sent if at least 1 bit of the TPDO has changed.

However, very often it is necessary to send the TPDO only if a certain bit has changed. For this purpose, the TPDO can be provided with a mask. Only those bits of the TPDO are evaluated which are set to "1" in the mask.

In the default setting all bits of the masks are set.

5.3.5 Example of a process data telegram

The following objects are to be transferred together in a PDO:

- ▶ Status word, index 6041_00_h (controller control),
- ▶ Modes_of_operation_display, index 6061_00_h (operating mode)
- ▶ Digital_inputs, index 60FD_00_h (digital inputs)

The first transmit PDO (TPDO 1) is to be used, and it is always to be sent if one of the digital inputs changes, but not more often than every 10 ms. The identifier used for this PDO is to be 187_h.

1. Delete the number of objects.

Description	Name	Value
To enable the change of the object mapping, the number of objects has to be set to zero.	number_of_mapped_objects	0

2. Parameterise the objects which are to be mapped.

Description	Name	Value
The objects listed above have to be composed to form a 32-bit value:		
Index = 6041 _h , subindex = 00 _h , length = 10 _h (UINT16)	first_mapped_object	60410010 _h
Index = 6061 _h , subindex = 00 _h , length = 08 _h (INT8)	second_mapped_object	60610008 _h
Index = 60FD _h , subindex = 00 _h , length = 20 _h (UINT32)	third_mapped_object	60FD0020 _h

3. Parameterise the number of objects.

Description	Name	Value
The PDO has to contain 3 objects	number_of_mapped_objects	3 _h

4. Parameterise the transmission mode.

Description	Name	Value
The PDO has to be sent if one or several of the above digital inputs change.	transmission_type	FF _h
In order to ensure that only changes of the digital inputs cause the transmission of the PDO, it is masked in such a way that only the above 16 bits of the object 60FD _h "come through".	transmit_mask_high	00FFFF00 _h
	transmit_mask_low	00000000 _h
The PDO is to be sent not more often than every 10 ms (100 × 100 μs).	inhibit_time	64 _h

5. Parameterise the identifier.

Description	Name	Value
The PDO is to be sent with the identifier 187 _h . If the PDO is active, it first has to be deactivated.		
Read out the identifier:	cob_id_used_by_pdo	00000181 _h
Set bit 31 (deactivate PDO):	cob_id_used_by_pdo	C0000181 _h
Write new identifier:	cob_id_used_by_pdo	C0000187 _h
Activate PDO by deleting bit 31:	cob_id_used_by_pdo	40000187 _h



Note!

The parameterisation of the PDO can only be changed if the network state (NMT) is not **operational**.

5

CANopen communication

Process data transfer (PDO transfer)

Activation of the PDOs

5.3.6

Activation of the PDOs

The following criteria must be met to enable the drive controller to send or receive PDOs:

- ▶ The **number_of_mapped_objects** object must be non-zero.
- ▶ Bit 31 of the **cob_id_used_for_pdos** object must be deleted.
- ▶ The communication state of the controller must be **operational** (see chapter 5.5, network management).

The following criterion must be met to enable the parameterisation of PDOs:

- ▶ The communication state of the drive controller must not be **operational**.

5.4 Sync telegram

The sync telegram is an additional and special telegram which enables the drive controller to cyclically read / accept process data.

5.4.1 Telegram structure

11 bits	4 bits									
Identifier	Data length									

The identifier on which the drive controller receives the sync telegram is permanently set to 080_h. The data length is 0.

5.4.2 Synchronisation of the process data

The sync telegram is the trigger point for data acceptance in the drive controller and it starts the sending process of the drive controller. Cyclic process data processing requires an appropriate generation of the sync telegram.

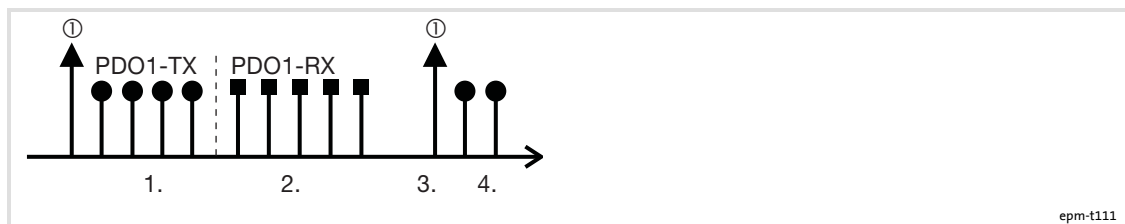


Fig. 4 Synchronisation of cyclic process data by means of a sync telegram (without consideration of asynchronous data)

① Sync telegram

Transmission sequence

1. After the sync telegram has been received, the cyclic process data are sent from the drive controllers to the master. The data is read by the master as process input data.
2. When the sending process is completed, the process output data (of the master) is received by the drive controllers.
3. The data is accepted by the drive controllers with the next sync telegram.
4. All other telegrams (e.g. for parameters or event-controlled process data) are accepted asynchronously by the drive controllers after the transmission has been completed.

5 CANopen communication

Sync telegram

Description of the objects

5.4.3 Description of the objects

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1005 _h	0 COB-ID_sync_message	00000080 _h	00000080 _h	{1 _h }	00000080 _h	VAR	UINT32	RW	—
									Synchronisation object identifier 80 _h .
									Bit no. Value
									0 - 10 x
									11 - 28 0
									29 0
									30 0
									Device does not generate sync telegrams.
									1
									Device generates sync telegrams.
									31 x
									Any

5.5 Network management (NMT)

Via the network management, the master can carry out state changes for the entire CAN network. For this purpose, the identifier with the highest priority (000_n) is reserved.

5.5.1 Communication phases of the CAN network (NMT)

Regarding communication, the drive distinguishes between the following states:

State	Explanation
"Initialisation"	Initialisation starts when the controller is switched on. In this phase, the controller does not take part in the bus data transfer. It is also possible in every NMT state to restart the entire initialisation or parts of it by transferring special telegrams (see "State transitions"). In this case, all parameters already set are overwritten with their standard values. After initialisation has been completed, the controller is automatically set to the state "pre-operational".
"Pre-operational" (before being ready for operation)	The controller can receive parameter data. Process data is ignored.
"Operational" (ready for operation)	The controller can receive parameter data and process data.
"Stopped"	Only network management telegrams can be received.

5 CANopen communication

Network management (NMT)

Telegram structure

5.5.2 Telegram structure

11 bits	4 bits	User data (2 bytes)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	CS	NI						

Via the NMT, commands can be sent to one or all drive controllers. Each command consists of two bytes. The first byte contains the command code (command specifier, CS) and the second byte contains the node address (node ID, NI) of the addressed drive controller. Via the node address zero, all nodes of the network can be addressed simultaneously. It is thus, for instance, possible to reset all drive controllers simultaneously. The drive controllers do not acknowledge the NMT commands. The successful execution can only be inferred indirectly e.g. from the switch-on message after a reset.

The NMT states of the CANopen nodes are defined in a state diagram. Via the CS byte in the NMT message state changes can be initiated. These changes are mainly orientated towards the target state.

In the NI parameter, the node address of the drive controller has to be specified. If all nodes of the network are to be addressed (broadcast), the parameter must be set to zero.



Note!

Communication via process data is only possible with a state change to "operational"!

Example:

For changing the state of all nodes on the bus from "pre-operational" to "operational" via the CAN master, the following identifier and user data must be set in the telegram:

- ▶ Identifier: 00 (broadcast telegram)
- ▶ User data: 0100 (hex)

State transitions

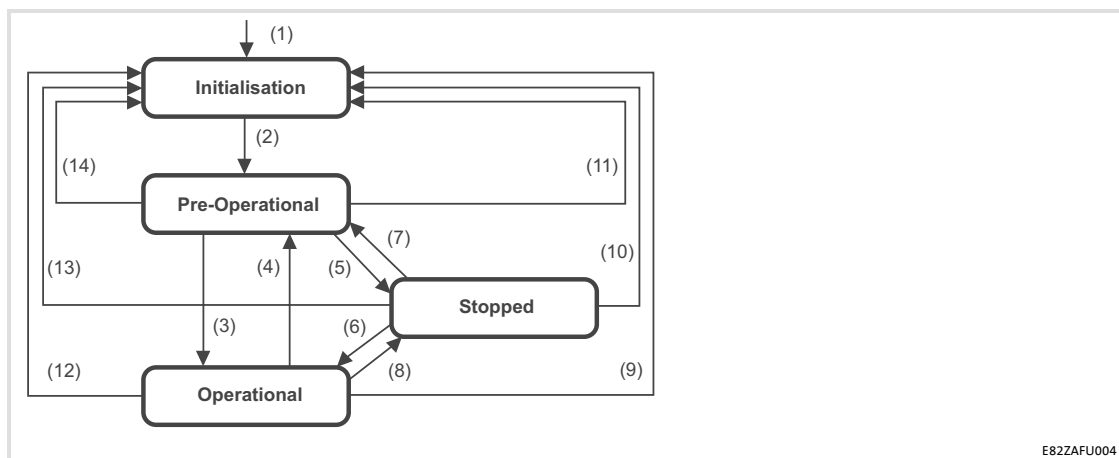


Fig. 5 Network management state transitions

State transition	Command (hex)	Network state after change	Effect on process and parameter data after state change
(1)	-	Initialisation	At power-on the initialisation is started automatically. During the initialisation, the drive controller does not take part in the data transfer. After the initialisation is completed, a boot-up message with an own identifier is sent from the drive controller to the master and the drive controller automatically changes to the pre-operational state.
(2)	-	Pre-operational	In this phase, the master decides how the drive controller/s is/are to participate in the communication.
From this moment on, the master changes the states for the entire network. A target address, which is part of the command, specifies the receiver/s.			
(3), (6)	01 xx	Operational	Network management telegrams, sync, emergency, process data (PDO) and parameter data (SDO) are active (corresponds to "start remote node") Optional: Event-controlled and time-controlled process data (PDO) are sent once in the case of a change.
(4), (7)	80 xx	Pre-operational	Network management telegrams, sync, emergency and parameter data (SDO) are active (corresponds to "enter pre-operational state")
(5), (8)	02 xx	Stopped	Only network management telegrams can be received.
(9)	82 xx	Initialisation	Initialisation of all parameters in the communication module with the stored values (corresponds to "reset node")
(10)			Initialisation of communication-relevant parameters (CIA DS 301) in the communication module with the stored values (corresponds to "reset communication")
(11)	81 xx	Initialisation	Initialisation of communication-relevant parameters (CIA DS 301) in the communication module with the stored values (corresponds to "reset communication")
(12)			
(13)			
(14)			

xx = 00_{hex}

xx = node ID

With this assignment, all devices connected are addressed by the telegram. The state can be changed for all devices at the same time.
If a node address is specified, only the state of the addressed device will be changed.

5 CANopen communication

Emergency telegram
Telegram structure

5.6 Emergency telegram

The drive controller monitors the functioning of its main components (including voltage supply, power stage, angle encoder evaluation, technology slots). In addition, the motor (temperature, angle encoder) and the limit switches are checked continuously. Incorrect parameter settings can also cause error messages (division by zero, etc.).

If an error occurs, the error number is indicated on the display of the drive controller. If several errors occur at the same time, the message with the highest priority (the lowest number) is displayed.

5.6.1 Telegram structure

11 bits	4 bits	Error code		1001 _h					
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	E0	E1	R0	00	00	00	00	00

The drive controller sends an emergency telegram if an error occurs. The identifier of this message is composed of the identifier **80_h** and the **node address** of the drive controller concerned.

The emergency telegram consists of eight bytes. The first and second byte contain the **error_code**. In the third byte there is an additional error code (object 1001_h). The fourth to eighth byte are always set to zero.

The following error codes may appear:

Error cause	Display	2nd byte	1st byte	3rd byte	4th ... 8th byte
		E1	E0	R0	
Motor overtemperature	03	43	10		00 ... 00
Insufficient temperature/overtemperature of power electronics	04	42	10		00 ... 00
SINCOS supply error	05	73	92		00 ... 00
SINCOS-RS485 communication error	06	73	91		00 ... 00
SINCOS track signal error	07	73	90		00 ... 00
Resolver track signal error / carrier failure	08	73	80		00 ... 00
5V-electronic supply error	09	51	13	05	00 ... 00
12V-electronic supply error	10	51	14		00 ... 00
24V-supply error (out of range)	11	51	12		00 ... 00
Offset current measurement error	13	52	10		00 ... 00
DC bus / power stage overcurrent	14	23	20		00 ... 00
DC bus undervoltage	15	32	20		00 ... 00
DC bus overvoltage	16	32	10		00 ... 00
I ² t-error of motor (I ² t at 100%)	19	23	12		00 ... 00
I ² t-error of drive controller (I ² t at 100%)	20	23	11		00 ... 00
I ² t at 80%	26	23	80		00 ... 00
Motor temperature 5°C below maximum	27	43	80		00 ... 00
Power stage temperature 5°C below maximum	28	42	80		00 ... 00
Following error monitoring	29	86	11		00 ... 00
Limit switch error	31	86	12	01	00 ... 00
Time-out during quick stop	35	61	99		00 ... 00
Homing error	36	8A	80		00 ... 00
Error: Motor and angle encoder identification	40	61	97		00 ... 00
Course program: Unknown command	43	61	93		00 ... 00
Course program: Invalid jump target	44	61	92		00 ... 00
No MMC available	49	76	80		00 ... 00
Error during MMC initialisation	50	76	81		00 ... 00
Error in MMC parameter set	51	76	82		00 ... 00
RS232 communication error	56	75	10		00 ... 00
Position data record error	57	61	91		00 ... 00
Operating mode error	58	63	80		00 ... 00
Error in preliminary positioning calculation	60	61	90		00 ... 00
Stack overflow	62	61	80		00 ... 00
Checksum error	63	55	81		00 ... 00
Initialisation error	64	61	87		00 ... 00

5 CANopen communication

Emergency telegram

Description of the objects

5.6.2 Description of the objects

Index	Name	Possible settings		Characteristics							
		Lenze	Selection	Description							
1001 _h	0 error_register					VAR	UINT8	RO	MAP		
						Reading-out of the error register.					
				Bit no.	Meaning						
				0	generic error			An unspecified error has occurred (flag is set for every error message).			
				1	current						
				2	voltage						
				3	temperature						
				4	communication error			Communication error (CAN overrun).			
				5	device profile specific						
				6	reserved						
				7	manufacturer specific			Manufacturer-specific error.			
1003 _h	0 pre_defined_error_field	0	0	0	{1}	255	ARR	UINT8	RW	—	
							Reading-out of the number of stored error messages. When an error has occurred, the error has to be acknowledged to activate the power stage (bit 7, control word index 6040 _h).				
				0			Memory is cleared				
				1	standard_error_field_0			—	UINT32	RO	—
						Reading-out of last error message					
	...										
	4 standard_error_field_3						—	UINT32	RO	—	
						Reading-out of error message					
1014 _h	0 COB-ID_emergency_message	00000081 _h	00000000 _h	{1 _h }	00000081 _h	VAR	UINT32	RW	—		
						Emergency object identifier, 080 _h + node address					
				Bit no.	Value						
				0 - 10	x			11-bit identifier			
				11 - 28	0			The extended identifier (bit 29) is not supported.			
				29	0			Each bit of this range must be "0".			
				30	0			Reserved			
31	0			Emergency object is valid							
				1		Emergency object is invalid					

5.7 Heartbeat telegram

The heartbeat telegram is implemented to monitor the communication between the drive controller and the master. For this purpose, the controller cyclically sends messages to the master. The master can check the cyclic transmission of these messages and initiate corresponding measures if they are missing. The heartbeat telegram is sent with the identifier **700_h (1792_d) + node address**. It only contains 1 byte of user data and the NMT state of the drive controller. The data length is 1.

In addition to the monitoring by the master, the bus system can be monitored by the drive controller. For this purpose, the drive controller monitors the acknowledgement of the heartbeat telegram. The absence of acknowledgements indicates that there is no other active drive controller on the bus system or that the bus system is damaged by a cable break.

The following response, which can be a warning, a quick stop or the immediate disconnection of the power stage, can be defined in the error management.

5.7.1 Telegram structure

11 bits	4 bits	User data (1 byte)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	N							



Note!

If the acknowledgement of the heartbeat telegram is missing in the heartbeat procedure, this can – depending on the error management settings – cause an error. This error can only be acknowledged via DIN9 or the parameter setting program (SDC) or a restart of the drive controller (for more detailed information please refer to the software manual).

CANopen communication

Heartbeat telegram

Telegram structure

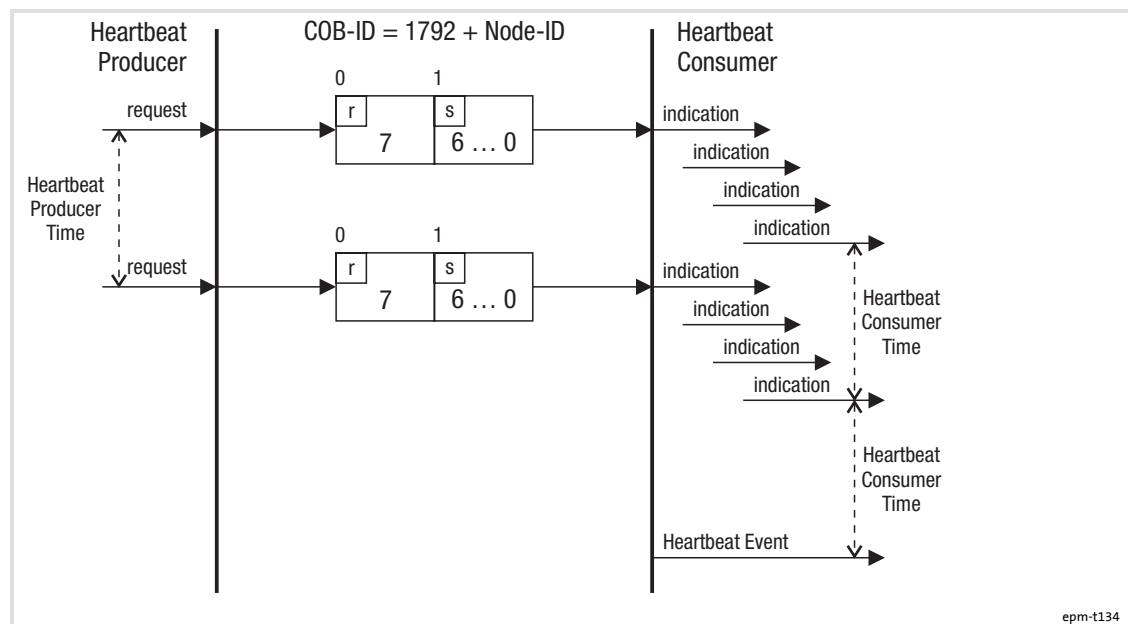


Fig. 6 Heartbeat telegram

- r Reserved
- s State of the Heartbeat Producer

Heartbeat producer

The drive controller transmits a state telegram on the fieldbus and can thus be monitored by other bus devices.

The settings are made under index 1017_h.

- ▶ The producer heartbeat is automatically started if a time > 0 is entered under index 1017_h and the drive controller changes to the operational state.
- ▶ When the cycle time has expired, the drive controller transmits the state telegram on the fieldbus.
- ▶ A reset changes the state to operational.

Device state (bits 1 ... 6) of the heartbeat producer:

Command (hex)	State
00	Boot-up
05	Operational
04	Stopped
7F	Pre-operational

5.7.2 Description of the objects

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
1017 _h	0 producer_heartbeat_time	0	0	{1 ms}	65536	VAR	UINT16	RW	—
			0						Function is deactivated

5 CANopen communication

Heartbeat telegram

Boot-up telegram

5.8 Boot-up telegram

After the supply voltage has been switched on or after a reset, the drive controller sends the boot-up telegram indicating that the initialisation phase is completed. The controller then is in the NMT state **pre-operational**.

5.8.1 Telegram structure

11 bits	4 bits	User data (1 byte)							
		1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Identifier	Data length	0							

The structure of the boot-up telegram is almost identical to the structure of the heartbeat telegram.

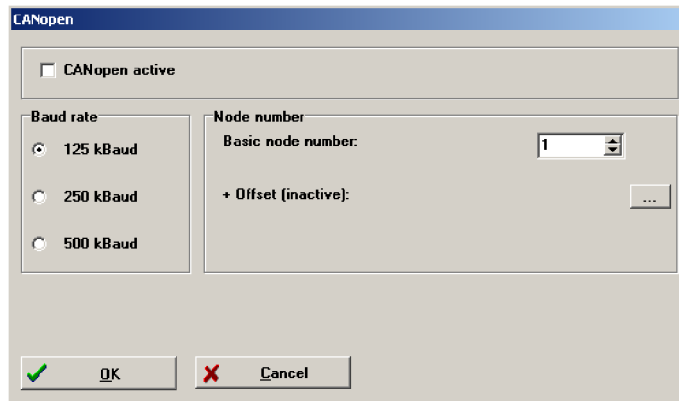
The boot-up telegram is also sent with the identifier **700_h + node address**. The data length is 1.

The only difference is that a zero is sent instead of the NMT state. For boot-up telegrams, too, the sending device expects – depending on the error management setting – the receipt of the telegram to be acknowledged by the other bus devices.

6 Commissioning

6.1 Activation of CANopen

The CAN interface is activated once with the CANopen protocol via the serial interface of the drive controller. The CAN protocol is activated via the CANopen window of the small drive control (SDC).



931e_380

Three different parameters have to be set:

- ▶ Basic node number (node address)

In order to unambiguously identify the nodes in the network, each node must be assigned a node address which may only appear once in the network. Via this node address the device is addressed.

As an additional option, the node address of the drive controller can be made dependent on the external wiring. If this function is activated, the input combination of the digital inputs DIN4 and DIN5 (or the DIN0 ... DIN5, depending on the selected evaluation of AIN / DIN in the 'digital inputs' menu) is added to the basic node address once after a reset.

- ▶ Baud rate

This parameter determines the baud rate (in kBaud / kbps) used on the CAN bus. Adapt the baud rate to the length of the transmission cable.

- ▶ CANopen active

Activating communication in this field activates the CAN communication. The CAN communication parameters cannot be changed anymore until communication has been deactivated again.



Note!

Please observe that the parameterisation of the CANopen functionality is only preserved after a reset if the parameter set of the drive controller has been saved in the EEPROM of the drive controller.

6 Commissioning

Speed control
Parameterising of a process data object (TPDO and RPDO)

6.2 Speed control

The purpose of this example is to show how a speed control can be commissioned via the CAN bus.

1. Use/activation of the transmit PDO1 (transmission of actual speed and status word) and of the receive PDO1 (setpoint speed)
2. Control of the network management
3. Parameterisation of the motor, current and speed controller
4. Definition of the operating mode (speed control)
5. Selection of a speed setpoint
6. Commissioning of the speed controller via the state machine

6.2.1 Parameterising of a process data object (TPDO and RPDO)

This example shows the adaptation and activation of a transmit PDO (TPDO) and a receive PDO (RPDO). The TPDO transfers the actual speed and the status word. Via the RPDO a higher-level control specifies the speed setpoint.

The following table lists and explains the different SDO accesses for parameterising the TPDO. When the network state is 'operational', the PDO is set to the identifier 181_h. The actual speed and the status word are transferred with a cycle time of 10 ms.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
			Data length		Low byte	High byte					
1	Network management (NMT) For the parameterisation of the PDO, the network management is set to the 'pre-operational' mode (80 _h).	00	2	80	00	00	00	00	00	00	00
2	Deactivation of the TPDO The PDO is deactivated by setting bit 31.	601	8	23	00	18	01	81	01	00	C0
3	Deletion of the number of objects To enable the changing of the object mapping, the number of objects (number_of_mapped_objects) must be set to zero.	601	5	2F	00	1A	00	00	00	00	00
4	Parameterisation of the first object to be mapped The index of the object to be mapped 1A00_01 _h (first_mapped_object) and the length of the corresponding variable type must be specified here. The first object to be mapped is the actual speed (index 606C_00 _h) with a length of 32 bits (20 _h).	601	8	23	00	1A	01	20	00	6C	60
5	Parameterisation of the second object to be mapped The second object to be mapped (second_mapped_object) is the status word (index 6041_00 _h) with a length of 16 bits (10 _h).	601	8	23	00	1A	02	10	00	41	60
6	Definition of the number of objects In this example, 2 mapped objects (actual speed and status word) are to be transferred (number_of_mapped_objects).	601	5	2F	00	1A	00	02	00	00	00
7	Parameterisation of the transmission mode The PDO is to be transferred at cyclic intervals (FF _h is entered for the transmission_type).	601	5	2F	00	18	02	FF	00	00	00
8	Definition of the transmission cycle time The transmission cycle time (inhibit_time) is to be set to 10 ms (100 × 100 μs).	601	6	2B	00	18	03	64	00	00	00
9	Activation of the TPDO The TPDO is activated by cancelling bit 31.	601	8	23	00	18	01	81	01	00	40
10	Network management (NMT) For the parameterisation of the PDO, the network management is set to the 'operational' mode (01 _h).	00	2	01	00	00	00	00	00	00	00

Tab. 2 Example of parameterising a transmit PDO

6

Commissioning

Speed control

Parameterising of a process data object (TPDO and RPDO)

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Low byte	High byte					
1	Network management (NMT) For the parameterisation of the PDO, the network management is set to the 'pre-operational' mode (80 _h).	00	2	80	00	00	00	00	00	00	00
2	Deactivation of the RPDO The RPDO is deactivated by setting bit 31.	601	8	23	00	14	01	01	02	00	C0
3	Deletion of the number of objects To enable the changing of the object mapping, the number of objects (number_of_mapped_objects) must be set to zero.	601	5	2F	00	16	00	00	00	00	00
4	Parameterisation of the first object to be mapped The index of the object to be mapped (first_mapped_object) and the length of the corresponding variable type must be specified here. The first object to be mapped is the setpoint speed (index 60FF_00 _h) with a length of 32 bits (20 _h).	601	8	23	00	16	01	20	00	FF	60
5	Definition of the number of objects In this example, 1 mapped object (setpoint speed) is to be transferred (number_of_mapped_objects).	601	5	2F	00	16	00	01	00	00	00
6	Parameterisation of the transmission mode The PDO is to be transferred at cyclic intervals (FF _h is entered for the transmission_type).	601	5	2F	00	14	02	FF	00	00	00
7	Activation of the RPDO The PDO is activated by cancelling bit 31.	601	8	23	00	14	01	01	02	00	40
8	Network management (NMT) For the parameterisation of the PDO, the network management is set to the 'operational' mode (01 _h).	00	2	01	00	00	00	00	00	00	00

Tab. 3 Example of parameterising a receive PDO

6.2.2 Parameterising of the motor and the current controller

In addition to the motor parameters (rated current, number of pole pairs), the safety-relevant parameters (max. current, i^2t tripping criterion) also have to be specified in advance. The current controller parameters can be adapted as well.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Data length	Low byte					
1	Entry of the rated motor current The rated motor current (motor_rated_current) is entered in mA (example: 5000 mA corresponds to 1388 _h)	601	8	23	75	60	00	88	13	00	00
2	Definition of the maximum current To restrict the current (max_current) / the maximum torque, the current is limited to twice the rated motor current (corresponds to 2000 / 07D0 _h).	601	6	2B	73	60	00	D0	07	00	00
3	Definition of the number of poles For a 2-pole-pair motor, the number of poles (pole_number) is 4.	601	5	2F	4D	60	00	04	00	00	00
4	Setting of the i^2t tripping time The tripping time (iit_ratio_time) is set to 5000 ms (corresponds to 1388 _h).	601	6	2B	10	64	03	88	13	00	00
5	Setting of the current controller (K_p) A gain (torque_control_gain) of $K_p = 2$ (corresponds to 0200 _h) is selected.	601	6	2B	2F	60	01	00	02	00	00
6	Setting of the current controller (T_n) A reset time (torque_control_time) of $T_n = 10$ ms (corresponds to 2710 _h) is selected.	601	6	2B	F6	60	02	10	27	00	00

Tab. 4 Example of parameterising the current controller and the motor type

6 Commissioning

Speed control

Parameterising of the speed control

6.2.3 Parameterising of the speed control

Before a control can be put into operation, it is often necessary to adapt the controller parameters in order to ensure a dynamic and sufficiently damped operational performance. The dimensioning of the controller parameters depends on the existing system / the respective process and must be executed in advance.

The following short example shows the selection and subsequent parameterisation of the speed control. In addition to the control parameters (K_p , T_n), the limit values required for safe operation (maximum speed, maximum acceleration and maximum braking deceleration) are specified.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Low byte	High byte					
1	Definition of the operating mode For the operating mode (modes_of_operation), the speed control (03) is selected.	601	5	2F	60	60	00	03	00	00	00
2	Setting of the speed controller (K_p) A gain (velocity_control_gain) of $K_p = 2$ (corresponds to 200 _h) is selected.	601	6	2B	F9	60	01	00	02	00	00
3	Setting of the speed controller (T_n) A reset time (velocity_control_time) of $T_n = 10$ ms (corresponds to 2710 _h) is selected.	601	6	2B	F9	60	02	10	27	00	00
4	Setting of the actual speed value filter The time constant of the speed filter (velocity_control_filter_time) is set to 0.5 ms (corresponds to 1F4 _h).	601	6	2B	F9	60	04	F4	01	00	00
5	Definition of the maximum speed The maximum speed (end_velocity) is set to 3000 rpm (corresponds to 0BB8 _h).	601	8	23	82	60	00	B8	0B	00	00
6	Definition of the maximum acceleration The maximum acceleration (profile_acceleration) is 20000 rev × 4096 inc/rev × 1/mi n/s (corresponds to 4E20000 _h).	601	8	23	83	60	00	00	00	E2	04
7	Definition of the maximum braking deceleration The maximum deceleration (profile_deceleration) is 20000 rev × 4096 inc/rev × 1/mi n/s (corresponds to 4E20000 _h).	601	8	23	84	60	00	00	00	E2	04

Tab. 5 Parameterisation of a speed controller

6.2.4 Running through the state machine

After all parameters required for the cascade control (current and speed control) have been defined, the drive can be commissioned via the state machine. First a speed setpoint is defined and sent once via an SDO access and once via the RPDO. Then the run through the state machine follows.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Low byte	High byte					
1	Selection of the speed setpoint via SDO access The speed setpoint (target_ velocity) is set to 1000 rpm.	601	8	23	FF	60	00	E8	03	00	00
2	Selection of the speed setpoint via RPDO The speed setpoint is set to 1000 rpm via the RPDO. For speed selection either method 1 or method 2 can be used.	201	4	E8	03	00	00	00	00	00	00
3	State query (read)	601	4	40	41	60	00	00	00	00	00
4	Control word: acknowledge error If an error has occurred, it can be reset with the 'fault reset' command after the error cause has been eliminated. If there is no error, continue with step 6.	601	6	2B	40	60	00	08	00	00	00
5	State query (read)	601	4	40	41	60	00	00	00	00	00
6	Control word: shut down With the 'shut down' command, the state is adapted to ready to switch on .	601	6	2B	40	60	00	06	00	00	00
7	State query (read)	601	4	40	41	60	00	00	00	00	00
8	Control word: switch on With the 'switch on' command, the state is adapted to switched on .	601	6	2B	40	60	00	07	00	00	00
9	State query (read)	601	4	40	41	60	00	00	00	00	00
10	Control word: enable operation With the 'enable operation' command, the state is adapted to operation enable . Voltage is now applied to the motor and the setpoint is being approached.	601	6	2B	40	60	00	0F	00	00	00
11	State query (read)	601	4	40	41	60	00	00	00	00	00
12	Operation of the control While the control is in operation, further changes (e.g. of the setpoint) can be made.	—	—	—	—	—	—	—	—	—	—
13	Control word: disable voltage This command switches off the drive and sets it to the switch on disabled state.	601	6	2B	40	60	00	00	00	00	00

Tab. 6 Commissioning of speed control via the state machine

Commissioning

Speed control

Running through the state machine

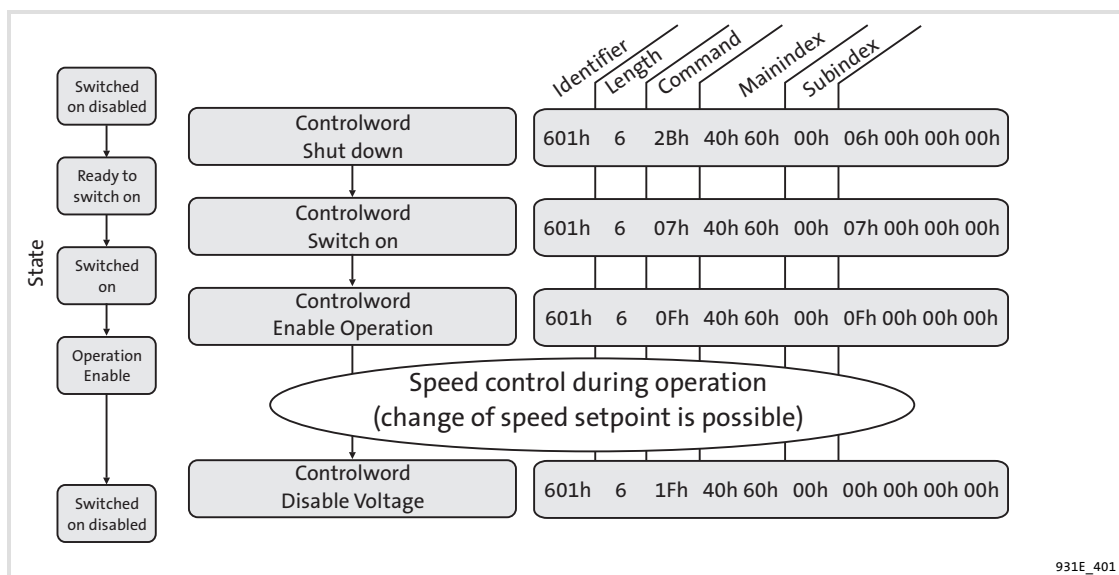


Fig. 7 Representation of a state machine during speed control commissioning

6.3 Position control

The aim of this example is to show the principle of parameterising and executing homing runs. A drive controller with the node address 1 is assumed to be the communicating node. In addition, the commissioning of a position control is explained.

The lower-level current and speed control must be set according to chapters 6.2.2 and 6.2.3. The following description assumes these drive controllers being set accordingly.

6.3.1 Parameterising of the homing run

Prior to the execution of the homing run, the homing method, the homing speeds and accelerations must be defined. Then the homing run can be performed.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Data length						
1	Selection of the homing operating mode For the operating mode (modes_of_operation), homing (06) is selected.	601	5	2F	60	60	00	06	00	00	00
2	Definition of the homing method For the homing method, travel to the limit switch in negative direction under consideration of the zero pulse is set (value 1).	601	5	2F	98	60	00	01	00	00	00
3	Setting of the fast homing speed The travelling speed used for searching for the limit switch (speed_during_search_for_switch) is set to 100 rpm.	601	6	2B	99	60	01	64	00	00	00
4	Setting of the slow homing speed The travelling speed used for searching for the zero pulse (speed_during_search_for_zero) is set to 50 rpm.	601	6	2B	99	60	02	32	00	00	00

Tab. 7 Parameterisation of homing

The homing state is indicated in the status word. Bit 12 shows whether the started homing process is completed (homing_attained) or whether it is still running.

In difference to the other operating modes, an additional step is required for running through the state machine when the state change **operation enabled** has been made. Homing can then be started with bit 4 of the control word.

6

Commissioning

Position control

Parameterising of the homing run

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Data length						
1	State query (read) Each change of the state must be executed depending on the initial state. After a state change you have to wait for the state change being indicated in the status word.	601	4	40	41	60	00	00	00	00	00
2	Control word: shut down With the 'shut down' command, the state is adapted to Ready_To_Switch_On .	601	6	2B	40	60	00	06	00	00	00
3	State query (read) (Explanation see step 1)	601	4	40	41	60	00	00	00	00	00
4	Control word: switch on With the switch on command, the state is adapted to Switched_On .	601	6	2B	40	60	00	07	00	00	00
5	State query (read) (Explanation see step 1)	601	4	40	41	60	00	00	00	00	00
6	Control word: enable operation With the enable operation command, the state is adapted to Operation_Enable . Voltage is now applied to the motor. However, the homing run does not start yet.	601	6	2B	40	60	00	0F	00	00	00
7	State query (read) (Explanation see step 1)	601	4	40	41	60	00	00	00	00	00
8	Control word: Enable operation and homing start The enable operation and homing start commands start the homing run.	601	6	2B	40	60	00	1F	00	00	00
9	State query (read) At this point the homing run is executed. Homing is completed when bit 12 (homing attained) is set in the status word.	601	4	40	41	60	00	00	00	00	00
10	Control word: disable voltage This command switches off the drive and sets it to the Switch_On_Disabled state.	601	6	2B	40	60	00	00	00	00	00

Tab. 8 Execution of the homing run by means of the state machine

6.3.2 Running through the state machine

When the homing run has been performed, the position control can be executed. This requires that the target position is defined. In addition the position controller, the required control accuracy, and the ramps and the speed for the profile generator must be parameterised.

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Low byte	High byte					
			Data length								
1	Definition of the operating mode For the operating mode (modes_of_operation), the position control (01) is selected.	601	5	2F	60	60	00	01	00	00	00
2	Setting of the position controller (K_p) A gain (position_control_gain) of $K_p = 2$ (corresponds to 0200 _h) is selected.	601	6	2B	FB	60	01	00	02	00	00
3	Setting of the maximum positioning speed The maximum speed (position_control_v_max) is set to 1000 rpm.	601	8	23	FB	60	04	E8	03	00	00
4	Definition of the profile speed The profile velocity is used to define the travelling speed of the positioning process (v = 100 rpm).	601	8	23	81	60	00	64	00	00	00
5	Definition of the final speed The end velocity is used to define the travelling speed at the end of the positioning process. Must be set to 0 m/s!	601	8	23	82	60	00	00	00	00	00
6	Setting of the profile acceleration The profile_acceleration object is used to define the acceleration.	601	8	23	83	60	00	10	27	00	00
7	Setting of the profile deceleration The profile_deceleration object is used to define the deceleration.	601	8	23	84	60	00	10	27	00	00
8	Setting of the position window The position window (position_error_tolerance_window) is used to define a range in which the drive controller does not intervene. One revolution corresponds to an entry of 65536. Here 1/100 rev (655) is entered.	601	8	23	FB	60	05	8F	02	00	00

6

Commissioning

Position control

Running through the state machine

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Low byte	High byte					
9	Definition of the position window The target position (target_position) is assumed to be reached if the actual position of the position controller (position_actual_value) is within a window (position_window) around the target position. The value selected is 1/100 rev.	601	8	23	67	60	00	8F	02	00	00
10	Definition of the position time If the actual position is within the position window for the time specified here (position_window_time), the target reached bit is set in the status word. The time is set to 100 ms.	601	6	2B	68	60	00	64	00	00	00

Tab. 9 Parameterisation of the position control

A change in the position is performed – as for all other operating modes as well – via a change of the state machine. This is described below:

No.	Description	Identifier	Control field	Command code	Index		Subindex	Data 1	Data 2	Data 3	Data 4
					Data length	Low byte					
1	Selection of the position setpoint via SDO access The position setpoint (target_position) is set to 1000 rev (1 rev = 4096 increments).	601	8	23	7A	60	00	00	00	E8	03
2	State query (read)	601	4	40	41	60	00	00	00	00	00
3	Control word: acknowledge error If an error has occurred, it can be reset with the fault reset command after the error cause has been eliminated. If there is no error, continue with step 4.	601	6	2B	40	60	00	08	00	00	00
4	State query (read)	601	4	40	41	60	00	00	00	00	00
5	Control word: shut down With the 'shut down' command, the state is adapted to Ready_To_Switch_On .	601	6	2B	40	60	00	06	00	00	00
6	State query (read)	601	4	40	41	60	00	00	00	00	00
7	Control word: switch on With the switch on command, the state is adapted to Switched_On .	601	6	2B	40	60	00	07	00	00	00
8	State query (read)	601	4	40	41	60	00	00	00	00	00
9	Control word: enable operation With the enable operation command, the state is adapted to Operation_Enable . Voltage is now applied to the motor. The target is not yet being approached.	601	6	2B	40	60	00	0F	00	00	00
10	State query (read)	601	4	40	41	60	00	00	00	00	00
11	Control word: enable operation and new setpoint With the enable operation and new setpoint commands, the state is adapted to Operation_Enable . Voltage is now applied to the motor and the setpoint is being approached.	601	6	2B	40	60	00	1F	00	00	00
12	State query (read)	601	4	40	41	60	00	00	00	00	00
13	Operation of the control During operation further changes (e.g. of the setpoint) can be made.	—	—	—	—	—	—	—	—	—	—
14	Control word: disable voltage This command switches off the drive and sets it to the Switch_On_Disabled state.	601	6	2B	40	60	00	00	00	00	00

Tab. 10 Commissioning of position control via the state machine

Commissioning

Position control

Running through the state machine

In Fig. 8, the state changes and the corresponding states are represented graphically. The process of running through the state machine is independent of the selected operating mode (torque, speed or position control).

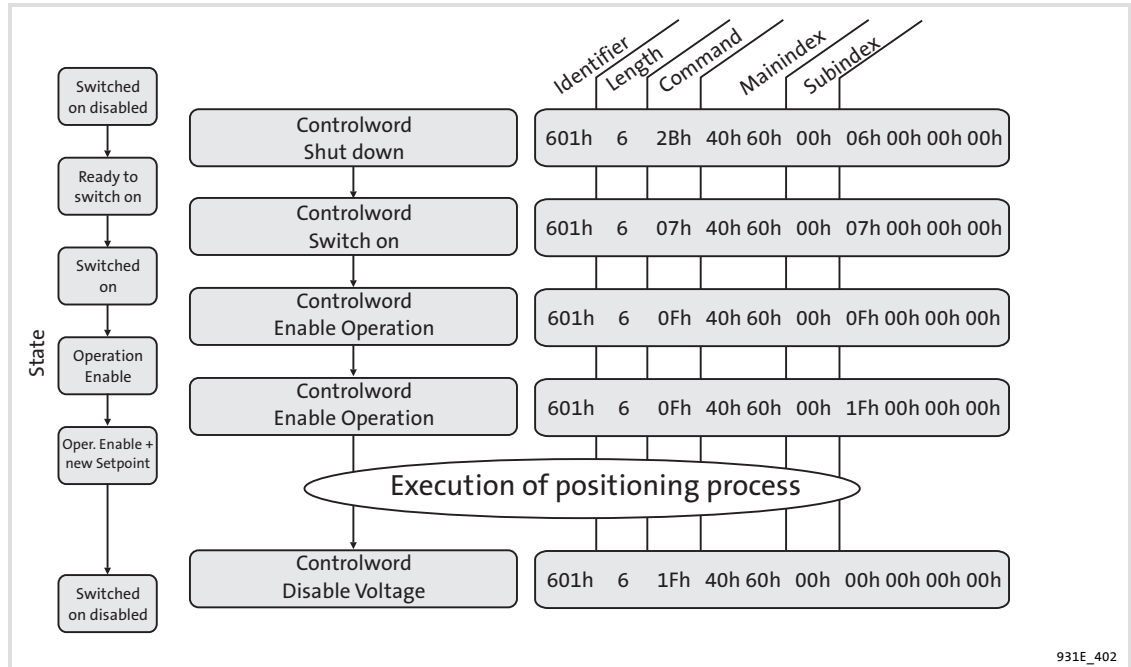


Fig. 8 Representation of a state machine during position control commissioning

931E_402

7 Parameter setting

Before the drive controller can perform the required task (torque or speed control or positioning), several controller parameters have to be adapted to the motor used and to the specific application. For this purpose you should keep to the sequence given in the following chapters.

These chapters first explain the parameterisation and then the device control and the use of the different operating modes.

7.1 Loading and saving of parameter sets

7.1.1 Overview

The drive controller is provided with three parameter sets:

► **Current parameter set**

This parameter set is stored in the volatile memory (RAM) of the drive controller. It can be read or written to as desired with the Small Drive Control parameterisation program or via the CAN bus. When switching on the drive controller, the **application parameter set** is copied into the **current parameter set**.

► **Default parameter set**

This parameter set is the unchangeable parameter set of the drive controller which is preset by default. By writing into the CANopen object 1011_01h (**restore_all_default_parameters**), the **default parameter set** can be copied into the **current parameter set**. This copying process is only possible if the power stage is switched off.

► **Application parameter set**

The **current parameter set** can be saved in the non-volatile flash memory. The saving process is initiated by a write access to the CANopen object 1010_01h (**save_all_parameters**). When the drive controller is switched on, the **application parameter set** is automatically copied into the **current parameter set**.

The following diagram illustrates the connections between the different parameter sets.

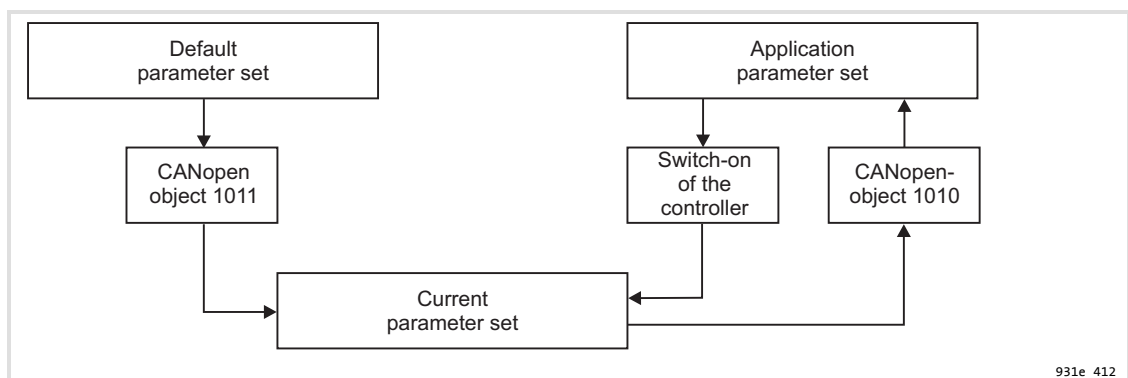


Fig. 9 Connections between the parameter sets

Parameter setting

Loading and saving of parameter sets

Overview

There are two possible variants for the parameter set management:

1. The parameter set is created with the Small Drive Control (SDC) parameterisation program and transferred to the different drive controllers. When this method is used, only the objects which can solely be accessed via CANopen have to be set via the CAN bus.

The disadvantage of this method is that the parameterisation software is required for the commissioning of a new drive controller or in the event of a repair being necessary (replacement of the drive controller). Therefore this method only makes sense for single controllers.

2. This variant is based on the fact that most application-specific parameter sets differ from the **default parameter set** in only a few parameters. Due to this, it is possible to recreate the **current parameter set** via the CAN bus after every switching on of the system.

For this purpose, the higher-level control first loads the **default parameter set** (calling of the CANopen object 1011_01h **restore_all_default_parameters**). Then only the differing objects are transferred. The whole process takes less than 1 second per drive controller. The advantage of this method is that it also functions for unparameterised drive controllers, so that the commissioning of new systems or the replacement of individual controllers does not cause any problems and the parameterisation software is not needed.



Note!

We recommend to use the second variant. Please observe that not all parameters can be set via CAN. If the setting of these parameters is required, the first variant has to be used.



Stop!

Uncontrolled rotation of the motor

An incorrect parameter set can cause uncontrolled rotation of the motor.

Possible consequences:

- ▶ This may result in property damage.

Protective measures:

- ▶ Before the initial switch-on of the power stage, ensure that the drive controller really contains the correct parameter set.

7.1.2 Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
1010 _h	0	store_parameters			ARR	UINT8	RO	—		
					Not used.					
	1	save_all_parameters	00000001 _h	00000000 _h	{1 _h }	65766173 _h	—	UINT32	RW	—
				00000000 _h			Acceptance of default parameter set in application parameter set.			
			00000000 _h			Default parameter set is not accepted.				
			65766173 _h	Save		Default parameter set is accepted.				
1011 _h	0	restore_parameters			ARR	UINT8	RO	—		
					Query on the number of possible objects.					
	1	restore_all_default_parameters	00000001 _h	00000000 _h	{1 _h }	64616F6C _h	—	UINT32	RW	—
							Loading of the default parameter set, only possible if the power stage is deactivated. The CAN communication parameters (node no., baud rate and operating mode) remain unchanged.			
			64616F6C _h	Load		Loading of default parameter set.				
			00000001 _h			Read access: Reset to default values.				

7

Parameter setting

Conversion factors (factor group)

Overview

7.2

Conversion factors (factor group)

7.2.1

Overview

Drive controllers are used in numerous applications, for instance as direct drives, with downstream gearboxes, for linear drives, etc.

To enable simple parameterisation for all these different applications, the drive controller can, by means of the factory group, be parameterised in such a way that the user can enter or read out all quantities (e.g. the speed) directly on the drive in the units desired (e.g. position values in millimetres and speeds in millimetres per second in the case of a linear axis).

The drive controller then converts the entries to internal units by means of the factor group. For every physical quantity (position, speed and acceleration) there is a conversion factor for adapting the user units to the own application.

The units set by the factor group are generally designated as **positions_units**, **speed_units** or **acceleration_units**.

The below diagram illustrates the function of the factor group.

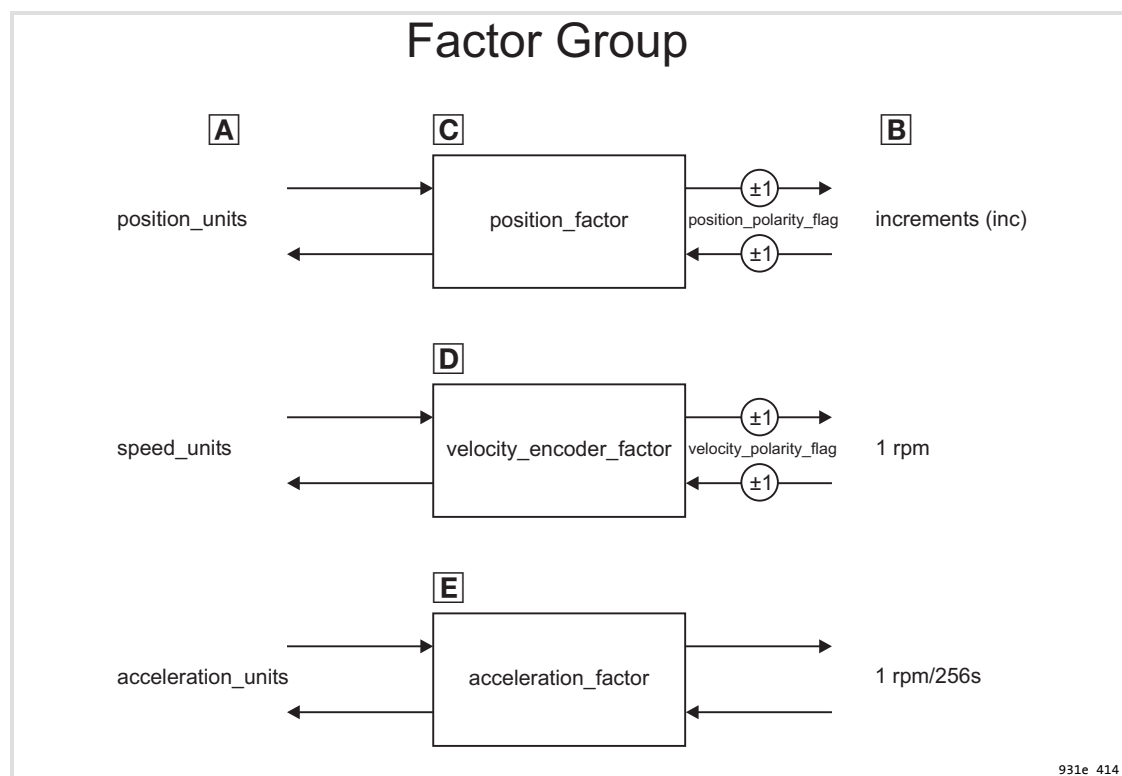


Fig. 10

Factor group

- A** User units
- B** Controller-internal units
- C** Position
- D** Speed
- E** Acceleration

In the controller, all parameters are stored in the form of internal units. When they are written or read out, they are converted by means of the factor group.



Note!

The factor group should be set prior to the first parameterisation and should not be changed during a parameterisation process.

The factory group is set to the following units as default:

Quantity	Designation	Unit	Explanation
Length	position_units	increments	65535 increments per revolution
Speed	speed_units	rpm	Revolutions per minute
Acceleration	acceleration_units	rpm/s	Speed increase per second

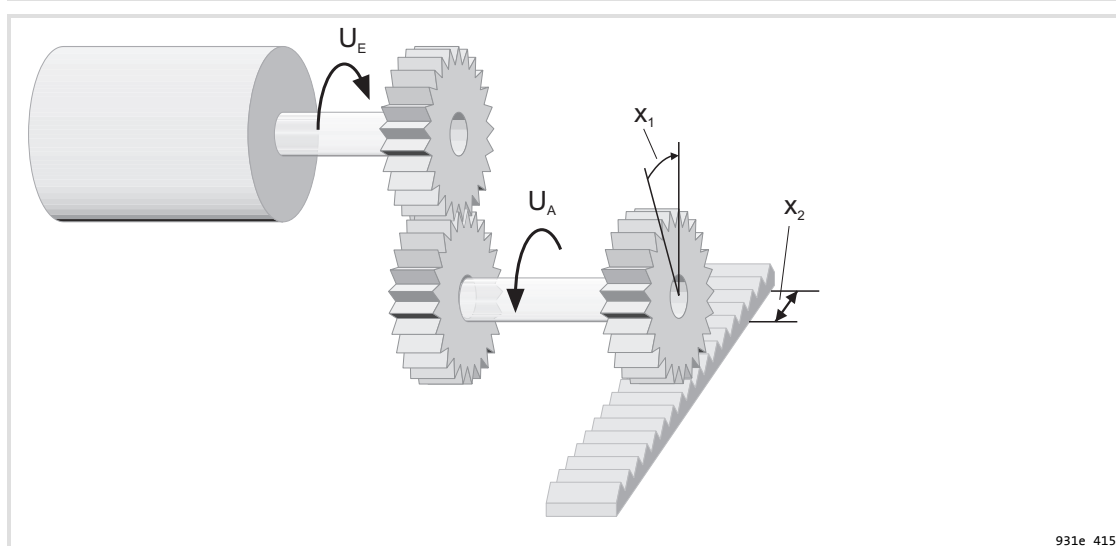


Fig. 11 Overview: Factor Group

- U_E Input-end speed
- U_A Output-end speed
- x_1 Position_units in degrees
- x_2 Position_units in mm

7

Parameter setting

Conversion factors (factor group)

Description of the objects

7.2.2

Description of the objects**Object 6093_n: position_factor**

The **position_factor** object serves to convert all length units of the application from **position_units** to the internal unit of increments (65535 increments correspond to 1 revolution).

The numerator and the denominator have to be written separately into the drive controller. It can, therefore, be necessary to bring the fraction to integers by appropriate expansions.

The **position_factor** is calculated using the following formula:

$$position_factor = \frac{numerator}{divisor}$$

Example: The unit required at the output end is 1 rev (position_unit)

1. 1 inc = 1/65535 rev
2. Result:
Numerator = 1
Divisor = 65535

Object 6094_h: velocity_encoder_factor

The **velocity_encoder_factor** object serves to convert all speed values of the application from **speed_units** to the internal unit of **revolutions per minute**.

The object consists of two parts: A factor for conversion of internal length units to **position_units** and a factor for conversion of internal time units to user-defined time units.

The **velocity_encoder_factor** is calculated using the following formula:

$$\text{velocity_encoder_factor} = \frac{\text{numerator}}{\text{divisor}}$$

Object 6097_h: acceleration_factor

The **acceleration_factor** object serves to convert all acceleration values of the application from **acceleration_units** to the internal unit of **revolutions per minute per 256 seconds**.

The object consists of two parts: A factor for conversion of internal length units to **position_units** and a factor for conversion of internal time units to user-defined time units.

The **acceleration_factor** is calculated using the following formula:

$$\text{acceleration_factor} = \frac{\text{numerator}}{\text{divisor}}$$

Object 607E_h: polarity

Index	Name	Possible settings				Characteristics					
		Lenze	Selection			Description					
607E _h	0 polarity	00 _h	00 _h	{04 _h }	40 _h , 80 _h , C0 _h	VAR	UINT8	RW	MAP	Setting of the signs of position and speed values. By changing the sign, the direction of rotation can be inverted. Often it makes sense to set both flags to the same value.	
			Bit 6	40 _h	0	multiply by 1					velocity_polarity flag
					1	multiply by -1					
			Bit 7	80 _h	0	multiply by 1					position_polarity flag
					1	multiply by -1					

7 Parameter setting

Power stage parameters

Overview

7.3 Power stage parameters

7.3.1 Overview

The power stage of the drive controller comprises several safety functions, some of which can be parameterised:

- ▶ Controller enable logic (software and hardware enable)
- ▶ Overcurrent monitoring
- ▶ Overvoltage / undervoltage monitoring of the DC bus
- ▶ Power stage monitoring

7.3.2 Description of the objects

Object 6510_10_n: enable_logic



Danger!

Dangerous voltage

Even if the digital input DIN9 (controller enable) is not set, a dangerous voltage may be applied to the motor.

Possible consequences:

- ▶ This means a danger to life when working on the motor.

Protective measures:

- ▶ Disconnect the motor from the mains before starting to work on it.

With a rising edge and a HIGH level at the digital input DIN9 **controller enable** is detected. To enable the activation of the controller's power stage, the drive controller must be in the "ready for operation" state.

If this is the case, the drive controller changes to the "switch on power stage" state and the microprocessor activates the power transistors.

This in turn means that a previously switched-on power stage cannot be switched off if the microprocessor is defective. The only possibility to switch off the power stage in this case is to switch off the DC bus and the logic supply.

The drive controller's response to the cancellation of the signal depends on the operating mode:

▶ **Positioning operation and speed-controlled operation**

When the signal is cancelled, the motor is braked along a defined deceleration ramp. The power stage is switched off when the motor speed falls below 10 rpm and the eventually existing holding brake has been applied.

▶ **Torque-controlled operation**

When the signal is cancelled, the power stage is immediately switched off. Simultaneously an eventually existing holding brake is applied. This means that the motor coasts without braking or is only stopped by the eventually existing holding brake.

Index	Name	Possible settings			Characteristics			
		Lenze	Selection		Description			
6510 _h 10	enable_logic	2	0 {1}	2	—	UINT16	RW	MAP
			0		Setting of the power stage enable.			
			1		Digital inputs of power stage enable and controller enable must be activated.			
			2		Digital inputs of power stage enable and controller enable and interface RS232 must be activated.			

7

Parameter setting

Current controller and motor adaptation

Overview

7.4

Current controller and motor adaptation**Stop!****Motor and drive controller overload**

The current controller parameters and the current limitation can be set incorrectly and cause overloading of the motor and the drive controller.

Possible consequences:

- ▶ The motor and the drive controller can be damaged within a very short period of time.

Protective measures:

- ▶ Before switching on the drive controller, ensure that the current controller parameters and the current limitation are correctly set.

7.4.1

Overview**Stop!****Uncontrolled rotation of the motor**

If the phase sequence is reversed in the motor or angle encoder cable, positive feedback effects may occur resulting in the motor speed not being controllable.

Possible consequences:

- ▶ This may result in property damage.

Protective measures:

- ▶ Before switching on the motor, check that the motor cable and the angle encoder cable are connected in the correct phase relation.

The parameter set of the drive controller has to be adapted to the connected motor and to the cable set used. This concerns the following parameters:

- ▶ Rated current: Depends on the motor
- ▶ Overload capacity: Depends on the motor
- ▶ Number of poles: Depends on the motor
- ▶ Current controller: Depends on the motor
- ▶ Direction of rotation: Depends on the motor and on the phase sequence in the motor cable and the angle encoder cable
- ▶ Offset angle: Depends on the motor and on the phase sequence in the motor cable and the angle encoder cable

This data has to be specified by means of the Small Drive Control program at the first use of a motor type. For several motors, there are ready-made parameter sets available from Lenze. Please observe that the direction of rotation and the offset angle also depend on the cable set used.

7.4.2 Description of the objects

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
6075 _h	0 motorRatedCurrent	500	0 {1 mA}	VAR	UINT32	RW	MAP	Input value for I_r (specified on the motor nameplate). The value must be smaller than the rated controller current. If this index is changed, the index 6073 _h maxCurrent also must be parameterised again.
6073 _h	0 maxCurrent	2023	0 {motorRatedCurrent/1000} 65535	VAR	INT16	RW	MAP	Input value for I_{max} . The value for index 6075 _h motorRatedCurrent must be entered before this value can be input.
604D _h	0 poleNumber	2	2 {2} 254	VAR	UINT8	RW	MAP	Number of motor poles.
6410 _h	0 motorData			REC	UINT8	RO	—	Reading-out of the motor data.
	3 iitTimeMotor	2000	0 {1 ms} 10000	—	UINT16	RW	—	Setting of the time interval for which the motor can be fed with I_{max} (index 6073 _h). When this time has expired, the motor is automatically limited to the value set under 6075 _h .
	4 iitRatioMotor		{1 %}	—	UINT16	RO	MAP	Reading-out of the actual utilisation ratio of the I^2t limitation.
	10 phaseOrder	1	0 {1} 1	—	UINT16	RW	MAP	Setting of the phase sequence. See 'angle encoder' dialog box in the SDC parameterisation program.
			0					Phase sequence: left
			1					Phase sequence: right
	11 resolverOffsetAngle	11360	-32767 {1 inc} 32767	—	INT16	RW	MAP	Setting of the angle encoder orientation with respect to the permanent magnetic field of the rotor. See 'angle encoder offset angle' dialog box in the SDC parameterisation program. Conversion: $\alpha = \text{offset angle} \times 32767/180^\circ$ $= 11360$ (with offset angle = 62.4°)

7

Parameter setting

Current controller and motor adaptation

Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
2415 _h	0 current_limitation				REC	INT8	RO	—		
					Limitation von I_{max} (independently of the operating mode). Torque-limited speed operation is possible.					
1	limit_current_input_channel	00 _h	00 _h	{1 _h }	04 _h	—	INT8	RW	—	
						Setpoint source for the limiting torque.				
			0	No limitation						
			1	AIN0						
			2	AIN2						
			3	RS232						
2	limit_current	0	0	{1 mA}	100000	—	INT32	RW	—	
						<ul style="list-style-type: none"> Setpoint source RS232, CAN: limitation of the torque-proportional current Setpoint sources AIN0, AIN1: Selection of the scaling factor for the analog inputs. The current in mA corresponds to an applied voltage of 10 V. 				
60F6 _h	0 torque_control_parameters				REC	UINT8	RO	—		
					Reading-out of PI-controlled current controller data. The gain and the time constant apply to both the field-generating and the torque-generating current controller.					
		1	torque_control_gain	256	0	{1}	32 × 256	—	UINT16	RW
						Setting of the proportional gain of the current controller. From the SDC program: $K_p = 1.0$ Setting here: $1.0 \times 256 = 256$ (100 _h)				
2	torque_control_time	2000	100	{1 μs}	65500	—	UINT16	RW	—	
						Setting of the current controller time constant. From the SDC program: $T_n = 2$ ms Setting here: 2 ms = 2000 μs				

7.5 Speed controller

7.5.1 Overview

The parameter set of the drive controller has to be adapted to the application. Especially the gain strongly depends on masses which may be coupled to the motor. At the commissioning of the system, the data has to be optimised by means of the Small Drive Control program.



Stop!

Uncontrolled vibrations

Incorrect settings of the speed controller parameters can cause heavy vibrations.

Possible consequences:

- ▶ Parts of the system may be damaged.

Protective measures:

- ▶ Before switching on the drive controller, ensure that the speed controller parameters are correctly set.

7.5.2 Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
60F9 _h	0 velocity_control_parameter_set				REC	UINT8	RO	—	Reading-out of the speed controller data.	
	1 velocity_control_gain	179	26	{1}	64 × 256	—	UINT16	RW	MAP	Setting of the speed controller gain. From the SDC program: $K_p = 0.7$ Setting here: $0.7 \times 256 = 179$
	2 velocity_control_time	8000	200	{1 μs}	32000	—	UINT16	RW	MAP	Setting of the speed controller time constant. From the SDC program: $T_n = 8 \text{ ms}$ Setting here: $8 \text{ ms} = 8000 \mu\text{s}$
	4 velocity_control_filter_time	1600	200	{1 μs}	32000	—	UINT16	RW	MAP	Setting of the filter time constant for the actual speed value. The filter serves to reduce the measurement noise. From the SDC program: $T = 1.6 \text{ ms}$ Setting here: $1.6 \text{ ms} = 1600 \mu\text{s}$

7

Parameter setting

Position controller (position control function)

Overview

7.6

Position controller (position control function)

7.6.1

Overview

This chapter describes all parameters required for the position controller. The position setpoint (**position_demand_value**) from the trajectory generator is applied to the input of the position controller. In addition, the position controller is fed with the actual position (**position_actual_value**) from the angle encoder (resolver, incremental encoder, etc.). The behaviour of the position controller can be influenced by parameters. To keep the position control loop stable, the output quantity can be limited (**control_effort**). The output quantity is fed to the speed controller as the speed setpoint. All input and output quantities of the position controller are converted from the application-specific units to the corresponding internal units of the drive controller in the **factor group**. The internal quantities are marked with an asterisk.

The following subfunctions are defined in this chapter:

1. Following error (Following_Error)

The following error is the deviation of the actual position (**position_actual_value**) from the position setpoint (**position_demand_value**). If this following error exceeds the value specified in the following error window (**following_error_window**) for a certain time, bit 13 **following_error** of the **status word** object is set. The permissible time interval can be defined via the **following_error_time_out** object.

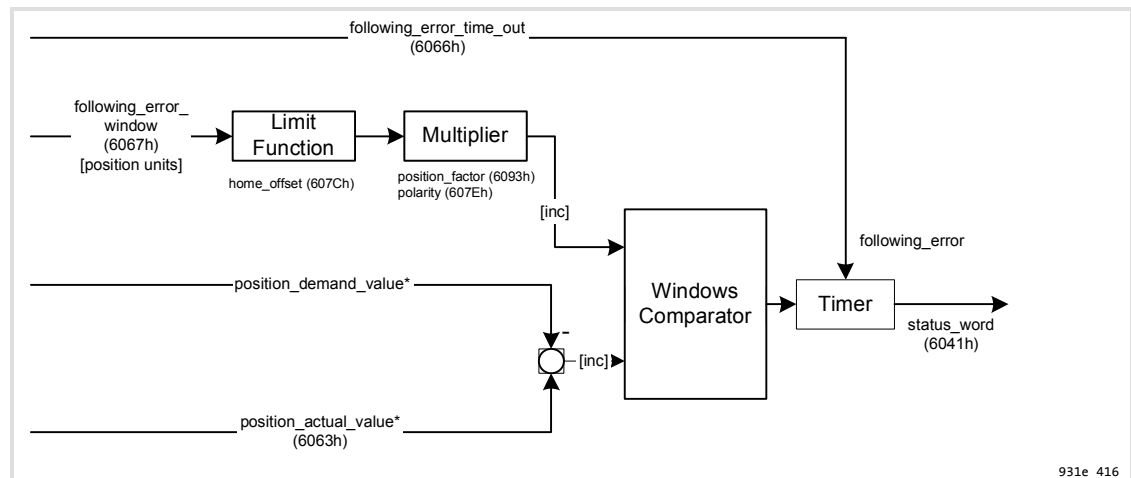


Fig. 12 Following error - functional survey

Fig. 13 shows the definition of the window function for the "following error" message. The range between $x_i - x_0$ and $x_i + x_0$ is defined symmetrically around the set position (**position_demand_value**) x_i . The positions x_{t2} and x_{t3} are, for instance, located outside this window (**following_error_window**). If the drive leaves this window and does not enter it again within the time specified in the **following_error_time_out** object, then bit 13 **following_error** of the **status word** is set.

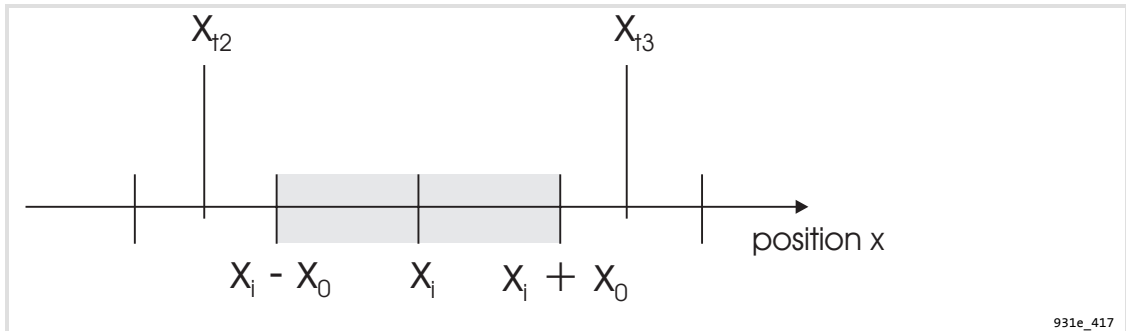


Fig. 13 Following error

2. Position reached

This function allows you to define a position window around the target position (**target_position**). If the actual position of the drive is within this range for a certain time (the **position_window_time**), then bit 10 **target_reached** of the **status word** is set.

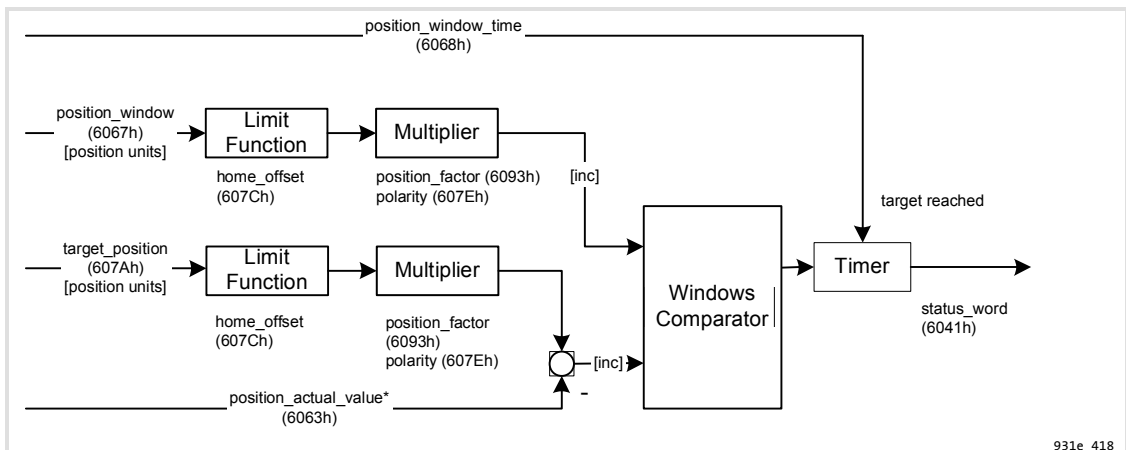


Fig. 14 Position reached - functional survey

7

Parameter setting

Position controller (position control function)

Description of the objects

Fig. 15 shows the definition of the window function for the "position reached" message. The positioning range between $x_i - x_0$ and $x_i + x_0$ is defined symmetrically around the target position (**target_position**) x_i . The positions x_{t0} and x_{t1} are, for instance, located within this position window (**position_window**). When the drive enters this window, a timer is started in the controller. When the timer has reached the time specified in the **position_window_time** object and the drive has not left the valid range between $x_i - x_0$ and $x_i + x_0$ during this time, then bit 10 **target_reached** of the **status word** is set. If the drive leaves the valid range, both bit 10 and the timer are set to zero.

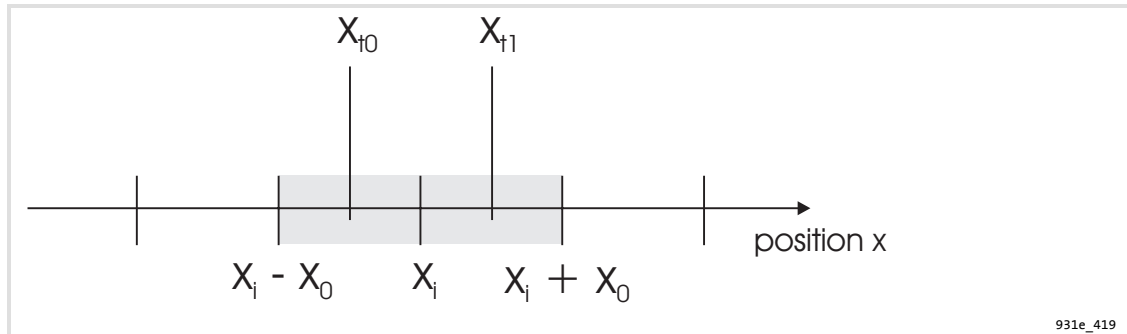


Fig. 15 Position reached

7.6.2

Description of the objects

The parameter set of the drive controller has to be adapted to the application. At the commissioning of the system, the data of the position controller has to be optimised by means of the Small Drive Control program.

**Stop!****Uncontrolled vibrations**

Incorrect settings of the position controller parameters can cause heavy vibrations.

Possible consequences:

- ▶ Parts of the system may be damaged.

Protective measures:

- ▶ Before switching on the drive controller, ensure that the position controller parameters are correctly set.

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
60FB _h	0 position_control_parameter_set				REC	UINT8	RO	—	
					Reading-out of the position controller data. The position controller operates with internal feedforwarding so that deviation control is minimised and the controller settling time is reduced.				
1	position_control_gain	52	0	{1}	64 × 256	—	UINT16	RW	—
						Setting of the position controller gain. From the SDC program: $K_p = 0.2$ Setting here: $0.2 \times 256 = 52$			
4	position_control_v_max	500	0	{1 rpm}	32767	—	UINT32	RW	MAP
						Limitation of the position controller correction speed. This is required since even small position deviations can cause considerable correction speeds.			
5	position_error_tolerance_window	13	0	{1 inc}	65535	—	UINT32	RW	MAP
						Definition of the size of a position deviation up to which the position controller does not intervene (dead zone). This can be used for stabilisation purposes, for instance, if there is backlash present in the system.			
6062 _h	0 position_demand_value		-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RO	MAP
		Reading-out of the position setpoint. This value is supplied to the position controller by the trajectory generator.							
6063 _h	0 position_actual_value		-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RO	MAP
		Reading-out of the actual position. The unit can be set via the factor group.							
6064 _h	0 position_actual_value		-2 ³¹	{position units}	2 ³¹ -1	VAR	INT32	RO	MAP
		Reading-out of the actual position.							

7

Parameter setting

Position controller (position control function)

Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
6065 _h	0 following_error_window	9102	00000000 _h	{1 inc}	7FFFFFFF	VAR	UINT32	RW	MAP	<p>Symmetrical range around the position setpoint. If the actual position value is outside the range, a following error occurs and bit 13 of the status word is set. Causes for following errors:</p> <ul style="list-style-type: none"> • The drive is blocked • The positioning speed is too high • The acceleration values are too high • The value entered for the following_error_window index is too low • The position controller is not parameterised correctly <p>The unit can be set via the factor group.</p>
6066 _h	0 following_error_time_out	100	0	{1 ms}	27314	VAR	UINT16	RW	MAP	<p>If the following error lasts longer than 100 ms, bit 13 of the status word is set.</p>
60FA _h	0 control_effort			{speed units}		VAR	INT32	RO	MAP	<p>Reading-out of the position controller correction speed. The correction speed is the difference between the set position and the actual position with consideration of the gain. This value is internally supplied to the speed controller as a setpoint.</p>
6067 _h	0 position_window	1820	-2 ³¹	{1 inc}	2 ³¹ -1	VAR	UINT32	RW	MAP	<p>Symmetrical range around the target position. The target position is reached if the actual position is within this range for a certain time. The unit can be set via the factor group.</p>
6068 _h	0 position_window_time	0	0	{1 ms}	65535	VAR	UINT16	RW	MAP	<p>If the actual position is within the position window for as long as defined here, bit 10 of the status word is set.</p>

7.7 Analog inputs

7.7.1 Overview

The drive controller is provided with two analog inputs, which can be used, for instance, to define setpoints. These analog inputs can only be parameterised via the Small Drive Control program.

7.8 Digital inputs and outputs

7.8.1 Overview

All digital inputs of the drive controller can be read via the CAN bus and the two digital outputs can be set as desired.

7.8.2 Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
60FD _h	0 digital_inputs	0	00000000 _h {1} FFFFFFFF _h		VAR	UINT32	RO	MAP		
			Reading-out of the digital inputs.							
			Bit no.	Value	Digital input					
			0	00000001 _h	Neg. limit switch					
			1	00000002 _h	Pos. limit switch					
			3	00000008 _h	Interlock (controller or power stage enable is missing)					
			16 - 25	03FF0000 _h	DIN0 ... DIN9					
60FE _h	0 digital_outputs				ARR	UINT8	RO	—		
			Reading-out of the digital outputs.							
1	digital_outputs_data	0	00000000 _h {1 _h } FFFFFFFF _h		—	UINT32	RW	MAP		
			Bit no.	Value	Digital output					
			0	00000001 _h	Brake					
			16	00010000 _h	Ready for operation					
			17, 18	00060000 _h	DOUT1, DOUT2					
					Setting of the 2 outputs. The activation can be delayed by up to 1 ms. By reading back index 60FE_00 _h you can check when the outputs are really set.					

7 Parameter setting

Limit switches

Overview

7.9 Limit switches

7.9.1 Overview

Limit switches can be used to define the home position of the drive controller. More detailed information on the possible homing methods can be found in the chapter 'Homing operation mode'.

7.9.2 Description of the objects

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
6510 _h	11 limit_switch_polarity	1	0	{1}	1	—	INT16	RW	MAP
			Setting of the limit switch polarity. The value always refers to both limit switches.						
			0				NC contact		
			1						NO contact
15	limit_switch_deceleration	200000	0	{1 rpm/s}	200000	—	INT32	RW	MAP
			Setting of the deceleration used for braking when the limit switch is reached in normal operation (limit switch emergency stop ramp).						

7.10 Device information

7.10.1 Description of the objects

Index	Name	Possible settings		Characteristics			
		Lenze	Selection	Description			
1000 _h	0 device_type			VAR	UINT32	RO	—
				Identification of the bus device in a multi-axis system.			
			00020192 _h	931E servo inverter			
1018 _h	0 identity_object			REC	UINT8	RO	—
				Not used.			
	1 vendor_id			—	UINT32	RO	—
				Manufacturer code			
			0000003B _h				
	2 product_code			—	UINT32	RO	—
				Product code			
			00001111 _h	931E			
	3 revision_number			—	UINT32	RO	—
				Firmware version			
	4 serial_number			—	UINT32	RO	—
				Hardware serial number			
6510 _h	1 serial_number			—	UINT32	RO	MAP
				Reading-out of the serial number.			
	2 drive_code			—	UINT32	RO	MAP
				Reading-out of the identification.			
			0x00001111				
A1	drive_type			—	UINT32	RO	MAP
				Reading-out of the device type, see also index 1018_02 _h product_code.			
A9	firmware_main_version			—	UINT32	RO	MAP
				Reading-out the main version number of the firmware (product version).			
AA	firmware_custom_version			—	UINT32	RW	MAP
				Reading-out of the version number of the customer-specific firmware variant.			

8 Device control

State diagram

Overview

8 Device control

8.1 State diagram

8.1.1 Overview

The following chapter describes how the drive controller is controlled under CANopen, i.e. how, for instance, the power stage is switched on or how an error is acknowledged.



Stop!

Uncontrolled rotation of the motor

An incorrectly parameterised drive controller can cause uncontrolled rotation of the motor.

Possible consequences:

- ▶ This may result in property damage.

Protective measures:

- ▶ Before the initial switch-on of the power stage, ensure that the drive controller is really parameterised with the correct parameters.

Under CANopen, the entire control of the drive controller is implemented using two objects: The master can control the drive controller via the **control word**, and the state of the drive controller can be read back via the **status word** object. To explain the control of the drive controller, the following terms are used:

▶ State

Depending on the power stage being switched on or an error having occurred, the drive controller is in different states. The states defined under CANopen are described in this chapter.

Example: Switch_On_Disabled

▶ State transition

CANopen not only defines the states, but also how to get from one state to another (e.g. for acknowledging an error). State transitions are initiated by the master by setting bits in the **control word** or internally by the drive controller if, for instance, the controller detects an error.

▶ Command

To initiate state transitions, certain bit combinations must be set in the **control word**. Such a combination is called a command.

Example: Enable operation

▶ State diagram (state machine)

All states and state transitions together form the state diagram, i.e. the overview of all states and the possible transitions.

8.1.2 State diagram of the drive controller

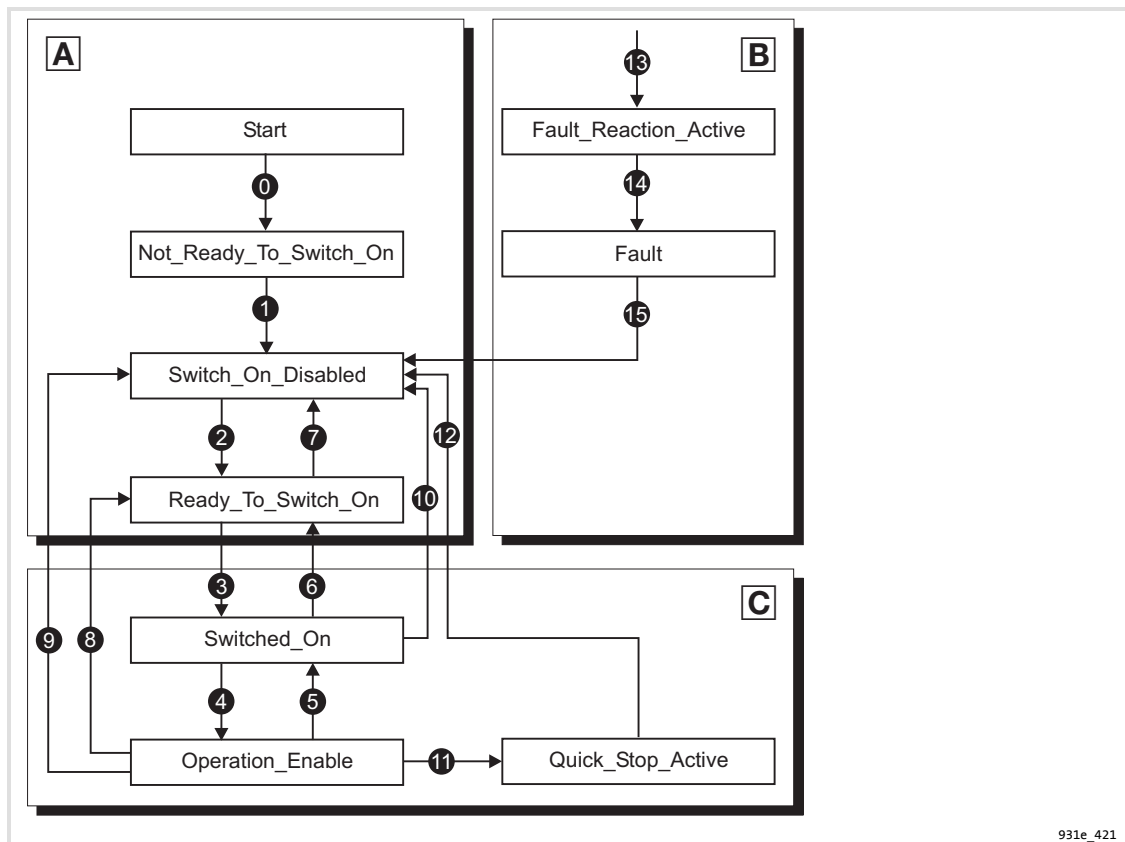


Fig. 16 State diagram of the drive controller

- Ⓐ Power disabled (power stage is inhibited)
- Ⓑ Fault (error)
- Ⓒ Power enabled (power stage is switched on)

**Danger!****Dangerous voltage**

Power stage inhibited means that the power transistors cannot be controlled anymore. However, a dangerous voltage may still be applied to the motor.

Possible consequences:

- ▶ This means a danger to life when working on the motor.

Protective measures:

- ▶ Disconnect the motor from the mains before starting to work on it.

After being switched on, the drive controller is initialised and finally reaches the **Switch_On_Disabled** state. In this state, the CAN communication is fully operational and the drive controller can be parameterised (e.g. the speed control operating mode can be set). The power stage is switched off and the shaft can thus be rotated freely.

Via the state transitions 2, 3, 4 (basically corresponding to the CAN controller enable) the **Operation_Enable** state is reached. In this state the power stage is switched on and the motor is controlled according to the set operating mode. It is therefore essential to ensure in advance that the drive is parameterised correctly and that a corresponding setpoint is set to zero.

Device control

State diagram

State diagram of the drive controller

If an error occurs, finally the **fault** state will be reached (no matter from which state you have started). Depending on the severity of the error, certain actions (e.g. an emergency braking) can be executed before the fault state is reached (**Fault_Reaction_Active**).

To execute the state transitions mentioned above, certain bit combinations must be set in the **control word**. The bits 0 ... 3 of the **control word** are evaluated together to initiate a state transition. In the following sections, first only the most important state transitions 2, 3, 4, 9 and 15 are explained. A table listing all states and state transitions can be found at the end of this chapter.

The following table contains in the first column the desired state transition and in the second column the required conditions (usually a command given by the master). The column **control word** shows how this command is generated, i.e. which bits are to be set in the **control word**. The **fault reset** command is generated by a positive edge change of bit 7.

Transition	Command	Control word (bits)					Action
		7	3	2	1	0	
2	Shut down and controller enable	x	x	1	1	0	None
3	Switch on	x	x	1	1	1	Switching on of the power stage
4	Enable operation	x	1	1	1	1	Control according to the set operating mode
9	Disable voltage	x	x	x	0	x	Power stage is inhibited. Motor can be rotated freely.
15	Fault reset and error eliminated	0->1	x	x	x	x	Error acknowledgement

Tab. 11 Important state transition of the drive controller
x not relevant

Example: Switching on of the power stage (drive controller must be parameterised)

The drive controller is in the **Switch_On_Disabled** state and is to be set into the **Operation_Enable** state. No other bits are to be set in the **control word**.

Transition	Old state	Control word (bits)				New state
		3	2	1	0	
2	Switch_On_Disabled	x	1	1	0	Ready_To_Switch_On ¹⁾
3	Ready_To_Switch_On	x	1	1	1	Switched_On ¹⁾
4	Switched_On	1	1	1	1	Operation_Enable ¹⁾

¹⁾ The master has to wait until the new state can be read back from the status word.

The transitions 3 and 4 can be summarised by directly setting the **control word** to 1111. For the state transition 2, the set bit 3 is not relevant.

8.1.3 States of the drive controller

State	Meaning
Not_Ready_To_Switch_On	The drive controller executes a self-test. The CAN communication is not yet working.
Switch_On_Disabled	The drive controller has completed the self-test. CAN communication is working.
Ready_To_Switch_On	The drive controller waits until the digital input DIN9 "controller enable" is at 24 V. (Controller enable logic "digital input and CAN").
Switched_On	The power stage can be switched on.
Operation_Enable ¹⁾	Voltage is applied to the motor and the motor is controlled according to the operating mode.
Quick_Stop_Active ¹⁾	The quick stop function is executed (see: quick_stop_option_code). Voltage is applied to the motor and the motor is controlled according to the quick stop function .
Fault_Reaction_Active ¹⁾	An error has occurred. Critical errors cause the immediate change to the fault state. For all other errors, the action selected in the fault_reaction_option_code is executed. Voltage is applied to the motor and the motor is controlled according to the fault reaction function .
Fault	An error has occurred. The motor is de-energised.

Tab. 12 States of the drive controller

¹⁾ The power stage is switched on

8

Device control

State diagram

State transitions of the drive controller

8.1.4

State transitions of the drive controller

Transition	Command	Control word (bits)					Action
		7	3	2	1	0	
0	Switched on or reset executed	Internal transition					Execution of self-test.
1	Self-test successful	Internal transition					Activation of CAN communication.
2	Shut down and controller enable	x	x	1	1	0	None
3	Switch on	x	x	1	1	1	None
4	Enable operation	x	1	1	1	1	On transition to the Operation_Enable state, the power stage is switched on.
5	Disable operation	x	0	1	1	1	Power stage is inhibited. Motor can be rotated freely.
6	Shut down	x	x	1	1	0	Power stage is inhibited. Motor can be rotated freely
7	Quick stop	x	x	0	1	x	None
8	Shut down	x	x	1	1	0	Power stage is inhibited. Motor can be rotated freely
9	Disable voltage	x	x	x	0	x	Power stage is inhibited. Motor can be rotated freely.
10	Disable voltage	x	x	x	0	x	Power stage is inhibited. Motor can be rotated freely.
11	Quick stop	x	x	0	1	x	Braking is initiated.
12	Disable voltage or braking completed	x	x	x	0	x	Power stage is inhibited. Motor can be rotated freely.
13	Error occurred	Internal transition					For non-critical errors response according to fault_reaction_option_code. For critical errors transition 14 follows.
14	Error handling is completed	Internal transition					Power stage is inhibited. Motor can be rotated freely.
15	Fault reset and error eliminated	0->1	x	x	x	x	Error acknowledgement (at rising edge).

Tab. 13 State transitions of the drive controller
x not relevant

8.1.5 Control word

The **control word** can be used to change the current state of the drive controller or to initiate a certain action directly (e.g. start of the homing run). The function of bits 4, 5, 6 and 8 depends on the current operating mode (**modes_of_operation**) of the drive controller.

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
6040 _h	0 controlword	0000 _h	0000 _h	{1 _h }	FFFF _h	VAR	UINT16	RW	MAP
						Change of the drive controller state. An action is initiated (e.g. homing run).			
			Bit no.	Value					
			0	0001 _h		Control of the state transitions. (These bits are evaluated together).			
			1	0002 _h					
			2	0004 _h					
			3	0008 _h					
			4	0010 _h		The function of these bits depends on the operating mode.			
			5	0020 _h					
			6	0040 _h					
			7	0080 _h		<ul style="list-style-type: none"> ● reset_fault At the transition from zero to one, the drive controller tries to acknowledge the pending errors. This is only possible if the cause of the error has been eliminated.			
			8	0100 _h		The bit function depends on the operating mode.			
			9	0200 _h		Set to zero.			
			10	0400 _h					
			11	0800 _h					
			12	1000 _h					
			13	2000 _h					
			14	4000 _h					
			15	8000 _h					

Device control

State diagram

Control word

The bits 0 ... 3 can be used to execute state transitions. The commands required for this purpose are listed in the below overview. The **fault reset** command is generated by a LOW/HIGH edge of bit 7.

Command	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0
Shut down	x	x	1	1	0
Switch on	x	x	1	1	1
Disable voltage	x	x	x	0	x
Quick stop	x	x	0	1	x
Disable operation	x	0	1	1	1
Enable operation	x	1	1	1	1
Fault reset	0->1	x	x	x	x

Tab. 14 Command overview

x not relevant



Note!

Since some of the state changes take some time, all state changes initiated via the **control word** must be read back via the **status word**.

Only if the new state can be read in the **status word**, a new command can be given via the **control word**.

Below the remaining bits of the **control word** are explained. Some bits have different meanings dependent on the operating mode (**modes_of_operation**):

Operation mode	Bit 4	Bit 5	Bit 6	Bit 8
Profile position mode	<ul style="list-style-type: none"> new_set_point <p>A rising edge indicates to the drive controller that a new travel task will be transferred.</p>	<ul style="list-style-type: none"> change_set_immediately <p>If this bit is not set, a new travel task will not be processed before an already running task has been completed. If the bit is set, the running positioning task will be aborted immediately and replaced by the new travel task.</p>	<ul style="list-style-type: none"> absolute/relative <p>If bit 6 is set, this means "absolute positioning". If bit 6 is not set, this means "relative positioning". If the bit is set, the drive controller refers the target position (target_position) of the current travel task to the set position (position_demand_value) of the position controller.</p>	<ul style="list-style-type: none"> stop <p>If the bit is set, the running positioning task is aborted. For braking, the profile_deceleration (index 6084_h) is used. After the process has been completed, bit 10 (target_reached) is set in the status word (index 6041_h). Deleting of the bit has no effect.</p>
Profile velocity mode	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> stop <p>If the bit is set, the speed is reduced to zero. For braking, the profile_deceleration (index 6084_h) is used. Deleting of the bit results in the drive controller being accelerated again.</p>
Profile torque mode	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> stop <p>If the bit is set, the torque is set to zero. Deleting of the bit results in the drive controller being accelerated again.</p>
Homing mode	<ul style="list-style-type: none"> start_homing_operation <p>A rising edge starts the parameterised homing run. A falling edge aborts the homing run being performed.</p>	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> stop <p>If the bit is set, the homing run being performed is aborted. Deleting of the bit has no effect.</p>
Interpolated position mode	<ul style="list-style-type: none"> enable_ip_mode <p>This bit has to be set if the interpolation data records are to be evaluated. It is acknowledged by bit 12 (ip_mode_active) in the status word.</p>	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> Reserved

8 Device control

State diagram
Controller state

8.1.6 Controller state

Similar to the combination of several **control word** bits initiating different state changes, the combination of different **status word** bits can be used to read out the current state of the drive controller.

The below table lists the possible states of the state diagram and the corresponding bit combinations indicating these states in the **status word**.

State	Bit 6	Bit 5	Bit 3	Bit 2	Bit 1	Bit 0	Mask	Value
	0040 _h	0020 _h	0008 _h	0004 _h	0002 _h	0001 _h		
Not_Ready_To_Switch_On	0	x	0	0	0	0	004F _h	0000 _h
Switch_On_Disabled	1	x	0	0	0	0	004F _h	0040 _h
Ready_to_Switch_On	0	1	0	0	0	1	006F _h	0021 _h
Switched_On	0	1	0	0	1	1	006F _h	0023 _h
Operation_Enable	0	1	0	1	1	1	006F _h	0027 _h
Fault	0	x	1	1	1	1	004F _h	000F _h
Fault_Reaction_Active	0	x	1	1	1	1	004F _h	000F _h
Quick_Stop_Active	0	0	0	1	1	1	006F _h	0007 _h

Tab. 15 Controller state
x not relevant

Example:

The above example shows which bits are to be set in the **control word** to enable the drive controller. Now the new state is to be read out from the **status word**:

Transition from **Switch_On_Disabled** to **Operation_Enable**:

- Write state transition 2 into the **control word**.
- Wait until the **Ready_To_Switch_On** state is indicated in the **status word**.
Transition 2: Control word = 0006_h, wait until (status word + 006F_h) = 0021_h is indicated ¹⁾
- State transitions 3 and 4 can be summarised and written to the **control word** together.
- Wait until the **Operation_Enable** state is indicated in the **status word**.
Transitions 3 and 4: Control word = 000F_h, wait until (status word + 006F_h) = 0027_h is indicated ¹⁾

The example is based on the assumption that no other bits are set in the **control word** (since only bits 0 ... 3 are important for the transitions).

1) For the identification of the states, the bits **not** set have to be evaluated as well (see table). Therefore the **status word** must be masked appropriately.

8.1.7 Status word

Index	Name	Possible settings			Characteristics			
		Lenze	Selection		Description			
6041 _h	0 statusword		0000 _h {1 _h }	FFFF _h	VAR	UINT16	RO	MAP
					Display of the controller state and of various events.			
			Bit no.	Value				
			0	0001 _h	State of the drive controller, see Tab. 15.			
			1	0002 _h	(These bits must be evaluated together).			
			2	0004 _h				
			3	0008 _h				
			4	0010 _h	<ul style="list-style-type: none"> ● voltage_disable This bit is set when the power stage transistors are switched on. Caution! In the event of a defect, the motor can still be energised.			
			5	0020 _h	State of the drive controller, see Tab. 15. <ul style="list-style-type: none"> ● quick_stop If the bit is deleted, the drive executes a quick stop according to index quick_stop_option_code.			
			6	0040 _h	State of the drive controller, see Tab. 15.			
			7	0080 _h	<ul style="list-style-type: none"> ● warning This bit is undefined. It must not be evaluated.			
			8	0100 _h	<ul style="list-style-type: none"> ● unused This bit is not used and must not be evaluated.			
			9	0200 _h	<ul style="list-style-type: none"> ● remote The bit is set if the controller enable logic is set correspondingly via index 6510_10 _h enable_logic.			
			10	0400 _h	The bit function depends on the operating mode.			
			11	0800 _h	<ul style="list-style-type: none"> ● internal_limit_active This bit indicates that the I ² t limitation is active.			
			12	1000 _h	The function of these bits depends on the operating mode.			
			13	2000 _h				
			14	4000 _h	<ul style="list-style-type: none"> ● unused This bit is not used and must not be evaluated.			
			15	8000 _h	<ul style="list-style-type: none"> ● reserved This bit must not be evaluated.			

**Note!**

The bits of the **status word** are not buffered. They indicate the current controller state.

In addition to the controller state, the **status word** indicates various events, i.e. each bit is assigned with a certain event (e.g. following error).

Some bits have different meanings dependent on the operating mode (**modes_of_operation**):

Operation mode	Bit 10	Bit 12	Bit 13
Profile position mode	<ul style="list-style-type: none"> target_reached <p>The bit is set if the current target position is reached and the current position (index position_actual_value) is within the parameterised position window (index 6067_h position_window). It is also set if the drive comes to standstill with the stop bit being set. The bit is deleted if a new target is selected.</p>	<ul style="list-style-type: none"> set_point_acknowledge <p>This bit is set if the drive controller has detected that bit 4 of the control word (new_set_point) is set. It is deleted again after bit 4 of the control word (new_set_point) has been set to zero.</p>	<ul style="list-style-type: none"> following_error <p>This bit is set if the current actual position (index position_actual_value) deviates that much from the set position (index position_demand_value) that the deviation is out of the parameterised tolerance window (indexes following_error_window, following_error_time_out).</p>
Profile velocity mode	<ul style="list-style-type: none"> target_reached <p>The bit is set if the actual speed (index 606C_h velocity_actual_value) of the drive is within the tolerance window. This tolerance window can be set via SDC.</p>	<ul style="list-style-type: none"> speed_0 <p>This bit is set if the current actual speed (index 606C_h velocity_actual_value) of the drive is within the corresponding tolerance window.</p>	<ul style="list-style-type: none"> Reserved
Homing mode	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> homing_attained <p>This bit is set if the homing run has been completed successfully.</p>	<ul style="list-style-type: none"> homing_error
Interpolated position mode	<ul style="list-style-type: none"> Reserved 	<ul style="list-style-type: none"> ip_mode_active <p>This bit indicates that interpolation is active and the interpolation data records are being evaluated. The bit is set if this has been requested by bit 4 of the control word (enable_ip_mode).</p>	<ul style="list-style-type: none"> Reserved

9 Operating modes

9.1 Setting of the operating mode

9.1.1 Overview

Below the operating modes specified in detail under CANopen are listed:

- ▶ Speed control (profile velocity mode)
- ▶ Homing (homing mode)
- ▶ Positioning (profile position mode)
- ▶ Synchronous position selection (interpolated position mode)
- ▶ Torque control (profile torque mode)

9.1.2 Description of the objects



Note!

The object 6060_h **modes_of_operation** can only be used to change the operating mode and the object 6061_h **modes_of_operation_display** can only be used to read back the operating mode.

Since changing the operating mode can take some time, you have to wait until the newly selected mode appears in the modes_of_operation_display object.

During this period it may happen that invalid operating modes are displayed for a short time.

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
6060 _h	0 modes_of_operation		1 {1}	7	VAR	INT8	RW	MAP
					Selection of the operating mode			
			1		Position control with positioning operation			
			3		Speed control with setpoint ramp			
			4		Torque control with setpoint ramp			
			6		Homing			
			7		Synchronous position selection			

9

Operating modes

Setting of the operating mode

Description of the objects

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
6061 _h	0 modes_of_operation_display			VAR	INT8	RO	MAP	
					Display of the operating mode. If operation via CANopen is not possible, an internal operating mode is displayed.			
			-1	Unknown operating mode / change of operating mode				
			1	Position control with positioning operation				
			2	Speed control without setpoint ramp				
			3	Speed control with setpoint ramp				
			4	Torque control with setpoint ramp				
			6	Homing				
			7	Synchronous position selection				
			-11	Internal positioning operation				
			-12	Internal speed control without setpoint ramp				
			-13	Internal speed control with setpoint ramp				
			-14	Internal torque control				
-15	Internal position control							

9.2 Speed control

9.2.1 Overview

The speed-controlled operation (profile velocity mode) comprises the following subfunctions:

- ▶ Setpoint generation by ramp generator
- ▶ Speed detection via angle encoder by differentiation
- ▶ Speed control with appropriate input and output signals
- ▶ Limitation of torque setpoint (**torque_demand_value**)
- ▶ Monitoring of actual speed (**velocity_actual_value**) with window function/threshold

The meaning of the parameters **profile_acceleration**, **profile_deceleration** and **quick_stop** is described in chapter "Positioning".

Operating modes

Speed control

Overview

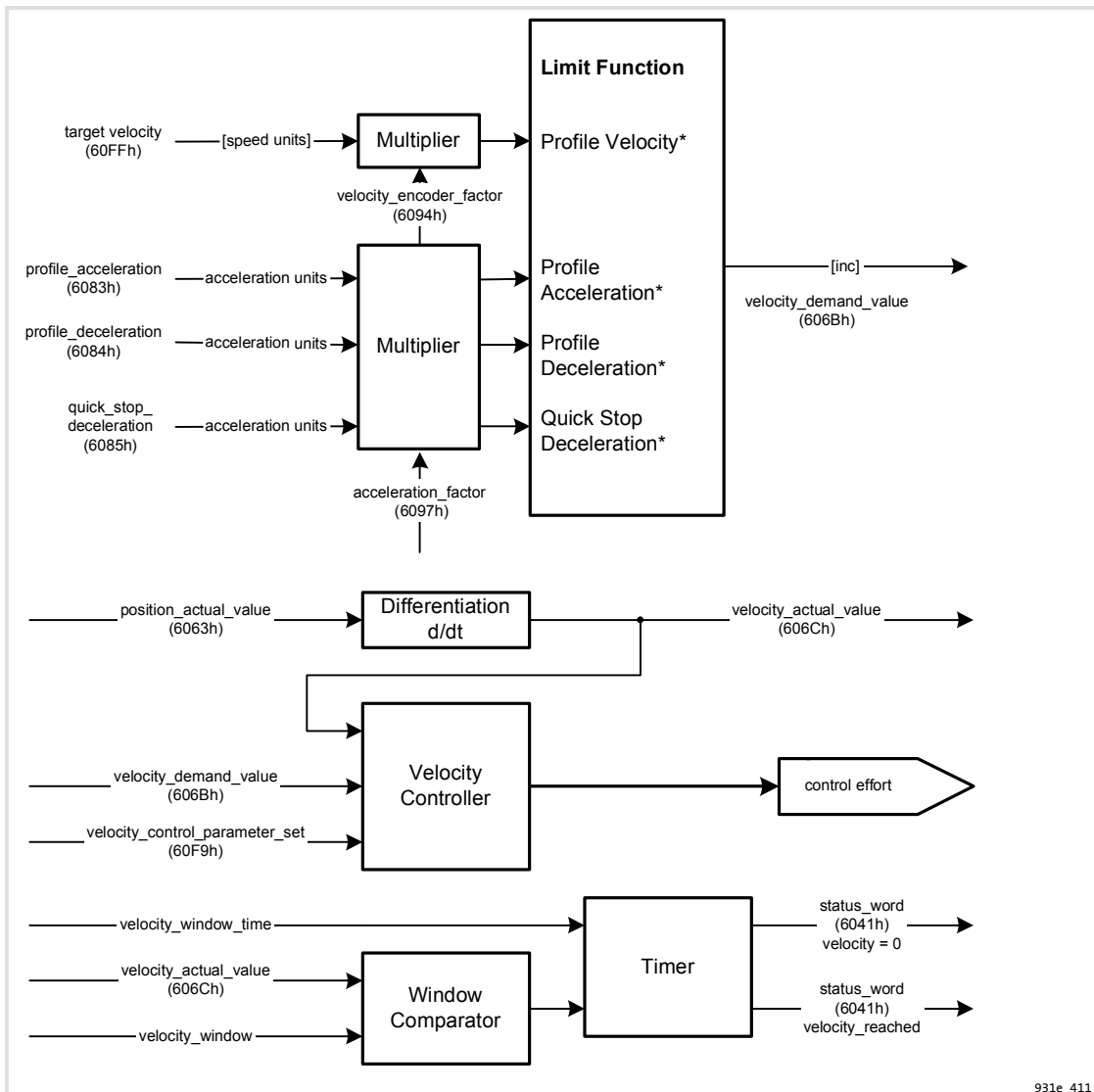


Fig. 17 Structure of speed-controlled operation (profile velocity mode)

931e_411

9.2.2 Description of the objects

Index	Name	Possible settings				Characteristics				
		Lenze	Selection			Description				
6069 _h	0 velocity_sensor_actual_value		{1 inc/s}			VAR	INT32	RO	MAP	Reading-out of the speed value directly on the encoder system. However, for determining the actual speed the object 606C _h should be used.
606B _h	0 velocity_demand_value		{1 rpm}			VAR	INT32	RO	MAP	Current speed setpoint. This value is affected by the setpoint of the ramp generator / the trajectory generator. If the position controller is activated, the correction speed of this controller is added.
606C _h	0 velocity_actual_value		{1 rpm}			VAR	INT32	RO	MAP	Reading-out of the actual speed.
6080 _h	0 max_motor_speed	32768	0	{1 rpm}	32768	VAR	UINT16	RW	MAP	Setting of the maximum speed. The speed setpoint is limited to this value.
60FF _h	0 target_velocity	0	-2 ³¹	{1 rpm}	2 ³¹ -1	VAR	INT32	RW	MAP	Setting of the setpoint selected for the ramp generator. The unit can be set via the factor group.

9 Operating modes

Homing
Overview

9.3 Homing

9.3.1 Overview

This chapter describes how the drive controller finds the start position (also called reference position, home position or zero position). There are different methods to determine this position. Sometimes the limit switches at the end of the positioning range are used. In order to increase the reproducibility as much as possible, some methods also integrate the zero pulse of the angle encoder used (resolver, incremental encoder etc.).

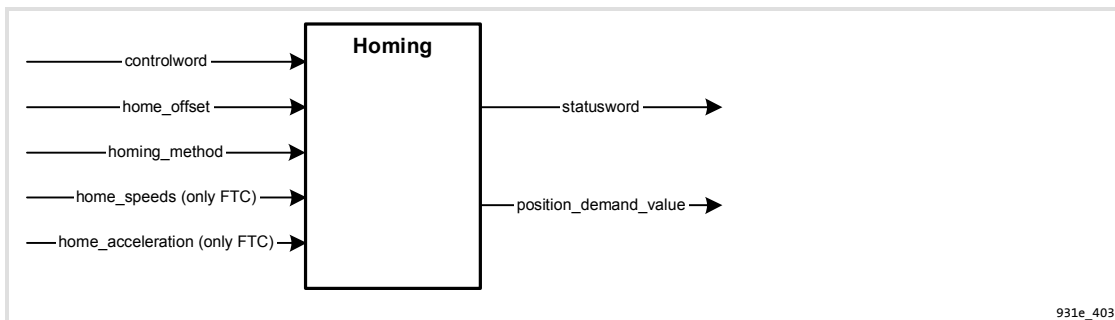


Fig. 18 Homing

The user can specify the speed, acceleration and type of homing. The **home_offset** object can be used to set the zero position of the drive to any position desired.

The value of the **home_offset** object is added to the value of the homing point (e.g. limit switch position).

There are two homing speeds. The higher search speed is used to find the limit switch / homing switch. In order to determine the exact position of the respective switching edge, the system changes over to the creep speed. The speeds and accelerations are set via the small drive control.



Note!

With CANopen, travelling to the zero position is usually not part of the homing run. If the drive controller knows all parameters required (e.g. the position of the zero pulse), no physical movement is executed.

9.3.2 Description of the objects

Index	Name	Possible settings			Characteristics						
		Lenze	Selection		Description						
607C _h	0 home_offset	0	-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RW	MAP	Shift of zero position with respect to home position.	
6098 _h	0 homing_method	17	-18	{1}	34	VAR	INT8	RW	—	<p>Selection of the homing variant.</p> <p>There are 3 homing signals:</p> <ul style="list-style-type: none"> • Negative and positive limit switches • Zero pulse (periodic) of the angle encoder • Negative and positive limit stops <p>If a method is selected for homing, the following settings are made:</p> <ul style="list-style-type: none"> • The reference source (neg./pos. limit switch, neg. / pos. limit stop) • The direction and the sequence of the homing run • The way in which the zero pulse of the used angle encoder is evaluated 	
			Value	Direction	Target	Ref. point for zero					
			-18	Positive	Limit stop	Limit stop					
			-17	Negative	Limit stop	Limit stop					
			-2	Positive	Limit stop	Zero pulse					
			-1	Negative	Limit stop	Zero pulse					
			1	Negative	Limit switch	Zero pulse					
			2	Positive	Limit switch	Zero pulse					
			17	Negative	Limit switch	Limit switch					
			18	Positive	Limit switch	Limit switch					
			33	Negative	Zero pulse	Zero pulse					
			34	Positive	Zero pulse	Zero pulse					
			35		No motion	Current actual position					
6099 _h	0 homing_speeds						ARR	UINT32	RO	—	Reading-out of the homing speed
	1 speed_during_search_for_switch	100	0	{1 rpm}	2 ³² -1	—	UINT32	RW	MAP	Speed for approaching the limit switch or the limit stop. Usually the speed is then changed and the final position is approached with a slower speed.	
	2 speed_during_search_for_zero	10	0	{1 rpm}	2 ³² -1	—	UINT32	RW	MAP	Selection of the slower speed	
609A _h	0 homing_acceleration	250	0	{1 rpm/s}	2 ³² -1	VAR	UINT32	RW	MAP	Selection of the acceleration and deceleration for the homing run. The value applies to all homing methods and to both homing speeds.	

9

Operating modes

Homing

Control of the homing run

9.3.3

Control of the homing run

The homing run is controlled by the **control word** and monitored by the **status word**. Homing is started by setting bit 4 in the **control word**. The successful completion is indicated by bit 12 being set in the **status word** object. Bit 13 being set in the **status word** object indicates that an error has occurred during the homing run. The error cause can be determined via the **error_register** and **pre_defined_error_field** objects.

Control word		Status word		
Bit 4	Meaning	Bit 13	Bit 12	Meaning
0	Homing is not active	0	0	Homing is not yet completed
0 -> 1	Start homing	0	1	Homing has been completed successfully
1	Homing is active	1	0	Homing has not been successful
1 -> 0	Interrupt homing	1	1	Forbidden state

9.4 Positioning

9.4.1 Overview

The target position (**target_position**) is transferred to the trajectory generator which then generates a position setpoint (**position_demand_value**) for the position controller. These two function blocks can be set independently of each other.

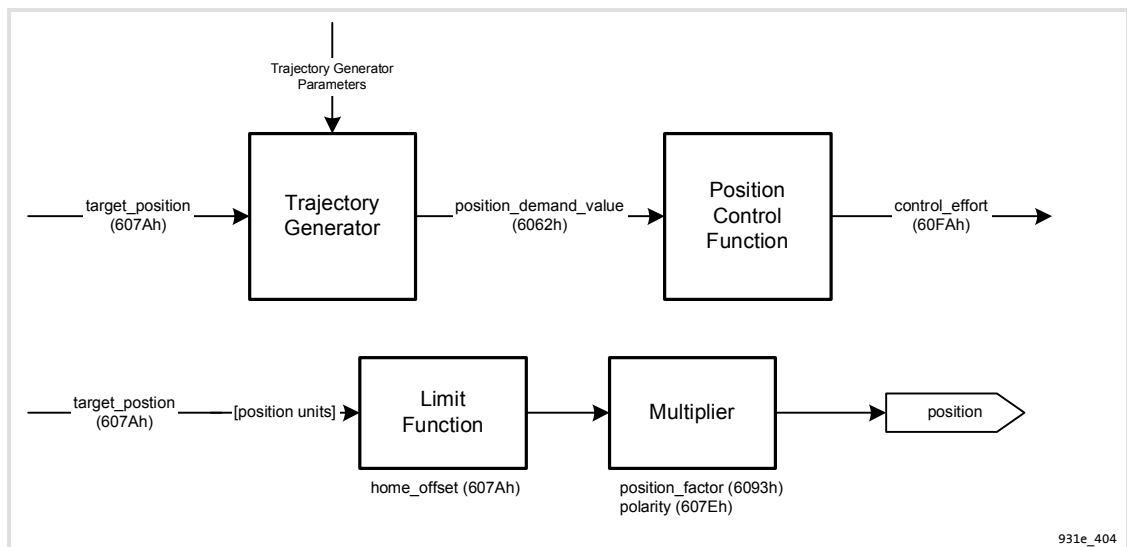


Fig. 19 Trajectory generator and position controller

All trajectory generator input variables are converted to the internal units of the drive controller by means of the factor group.

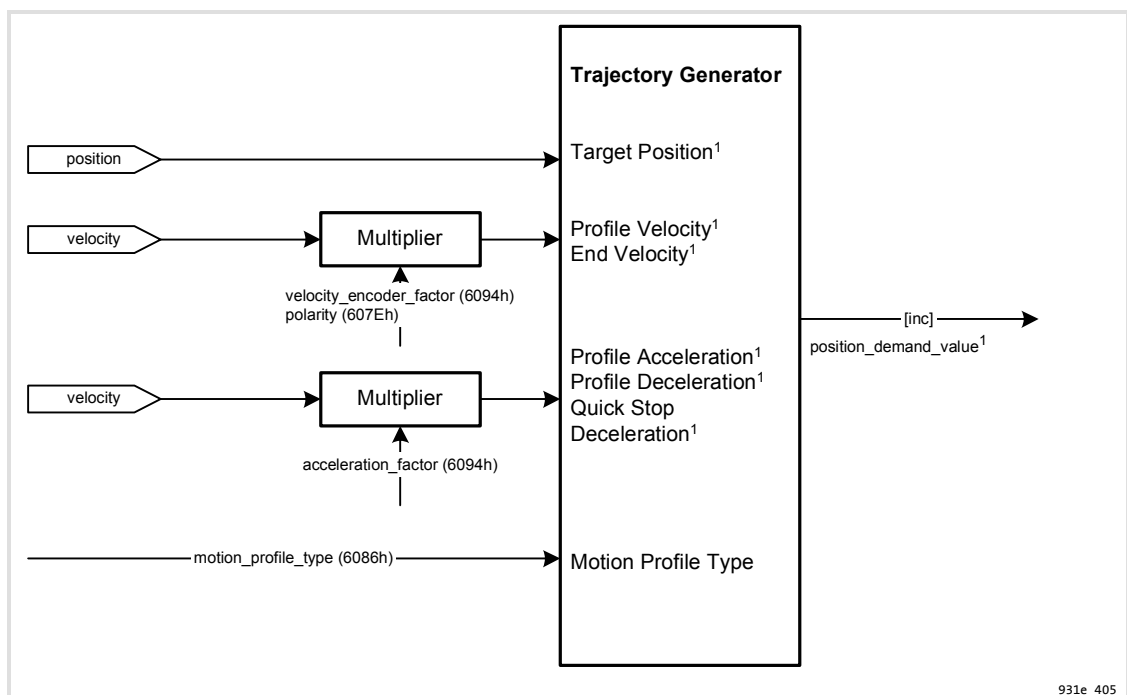


Fig. 20 Trajectory generator

1) Internal variables, normally not required by the user.

9

Operating modes

Positioning

Description of the objects

9.4.2 Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
607A _h	0 target_position	0	-2 ³¹	{1 inc}	2 ³¹⁻¹	VAR	INT32	RW	MAP	Entry of the target position (absolute or relative entry, see bit 6 of the control word). The current settings for speed, acceleration, braking deceleration and type of the travel profile must always be taken into account. The unit can be set via the factor group.
6081 _h	0 profile_velocity	0	0	{1 rpm}	2 ³¹⁻¹	VAR	UINT32	RW	MAP	Positioning: Speed at the end of the acceleration ramp. The unit can be set via the factor group.
6082 _h	0 end_velocity	0	0	{1 rpm}	2 ³¹⁻¹	VAR	UINT32	RW	MAP	Final speed The unit can be set via the factor group.
			0							Drive controller stops when the target position is reached.
			≠ 0							Gap-free positioning is not supported.
6083 _h	0 profile_acceleration	10000	0	{1 rpm/s}		VAR	UINT32	RW	MAP	Setting of the acceleration. The unit can be set via the factor group.
6084 _h	0 profile_deceleration	10000	0	{1 rpm/s}		VAR	UINT32	RW	MAP	Setting of the braking deceleration. The unit can be set via the factor group.
6085 _h	0 quick_stop_deceleration	14100	0	{1 rpm/s}		VAR	UINT32	RW	MAP	Setting of the braking deceleration in the event of a quick stop. The unit can be set via the factor group.
6086 _h	0 motion_profile_type	0	0	{1}	1	VAR	INT16	RW	MAP	Setting of the type of the positioning profile.
			0							Linear ramp
			1							Jerk-free acceleration

9.4.3 Functional description

There are two ways to transfer a target position to the drive controller:

► Simple travel task

When the drive controller has reached a target position, it signals this to the master with the **target_reached** bit (bit 10 in the **status word** object). In this operating mode, the drive controller stops when it has reached the target.

► Sequence of travel tasks

After the drive controller has reached a target, it immediately starts to approach the next target.

Interruption of a running task by a new travel task

The running positioning task is interrupted immediately and the new travel task is started.

These three methods are controlled by the **new_set_point** and **change_set_immediately** bits in the **control word** object and by the **set_point_acknowledge** bit in the **status word** object. The relationship between these bits is an interrogation/response relationship. This makes it possible to prepare a new travel task while the old one is still running.

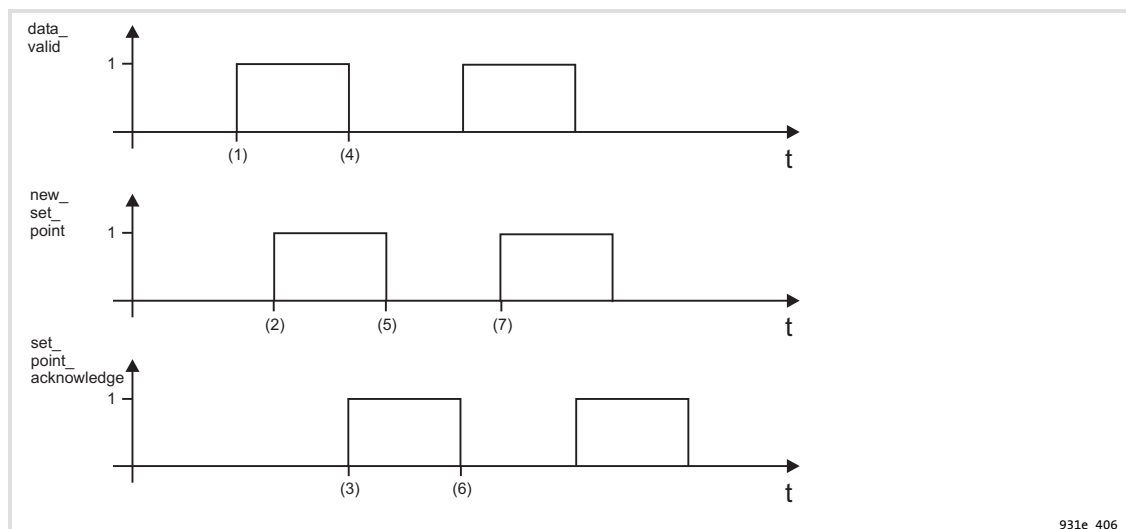


Fig. 21 Transfer of a travel task between master and drive controller

First the positioning data (target position, travelling speed and acceleration) are transferred to the drive controller. When the positioning data record is written completely (1), the master can start the positioning by setting the **new_set_point** bit in the **control word** to "1" (2).

After the drive controller has recognised the new data and transferred this data into its buffer, the controller signals this to the master by setting the **set_point_acknowledge** bit in the **status word** (3).

Then the master can start to write a new positioning data record to the drive controller (4) and delete the **new_set_point** bit again (5). The drive controller signals that it is ready to accept a new travel task (6) by setting the **set_point_acknowledge** bit to "0". Prior to this, the master may not start a new positioning (7).

Operating modes

Positioning

Functional description

In Fig. 22 a new positioning is only started if the last positioning has been completed completely. For this purpose, the master evaluates the **target_reached** bit in the **status word** object.

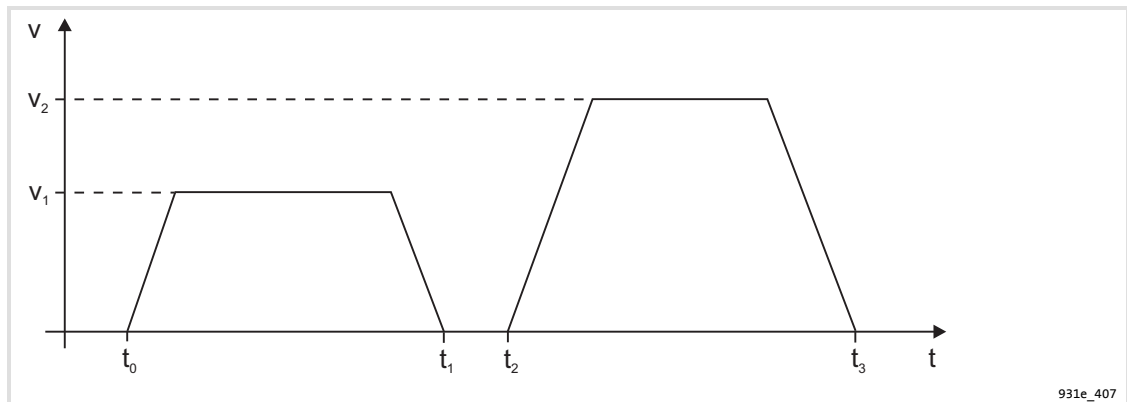


Fig. 22 Simple travel task

If the **new_set_point** bit as well as the **change_set_immediately** bit of the **control word** are set to "1", the master instructs the drive controller to start the new travel task immediately.

A travel task already being processed is interrupted in this case as shown in Fig. 23.

For this purpose, the master transfers the next target to the drive controller after the controller has signalled by deleting the **set_point_acknowledge** bit that it has read the buffer and started the corresponding positioning. Here it is important that the new travel task is transferred to the drive controller in time.

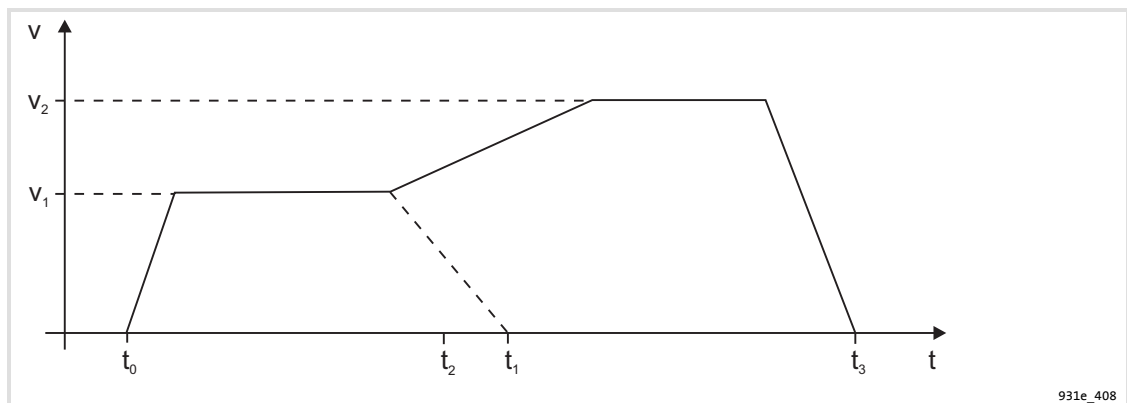


Fig. 23 Sequence of travel tasks

9.5 Synchronous position selection

9.5.1 Overview

The interpolated position mode (IP) enables the selection of position setpoints for multi-axis drive controller applications. For this purpose, a master provides synchronisation telegrams (sync) and position setpoints on a fixed time base (synchronisation interval).

Since the interval usually is larger than a position controller cycle, the drive controller independently interpolates the data values between two given position values as shown in the below graph.

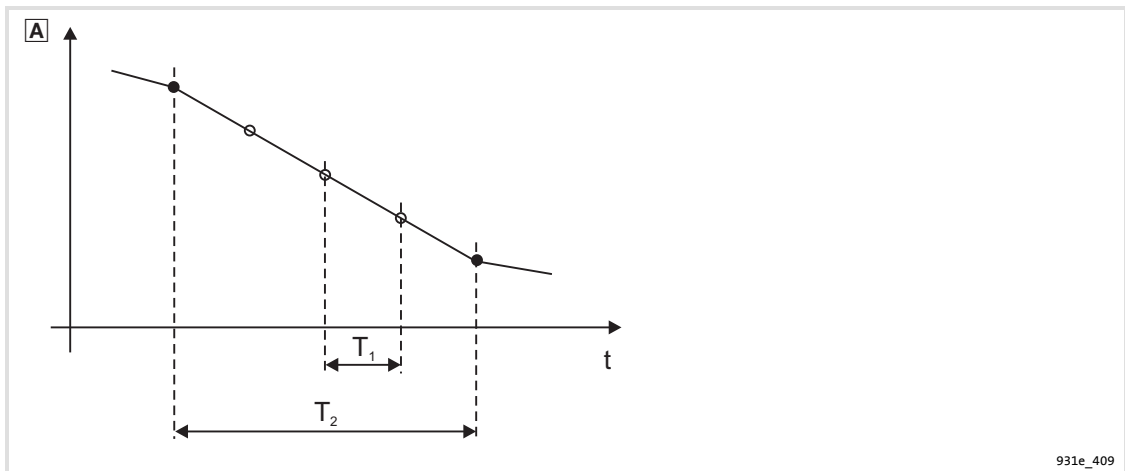


Fig. 24 Travel task: Linear interpolation between two data values

- ▣ A Position setpoints
- t_1 Position control interval
- t_2 Synchronisation interval
- Position setpoints specified by the control (ip_data_position)
- Interpolated position values

9

Operating modes

Synchronous position selection

Description of the objects

9.5.2 Description of the objects

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
60C0 _h	0 interpolated_submode_select	-2	-2	{}	-2	VAR	INT16	RW	MAP
			-2			Selection of the interpolation type.			
									Linear interpolation without buffer
60C1 _h	0 interpolated_data_record					REC	UINT8	RO	—
						Reading-out of the data record. The record consists of the position value (ip_data_position) and the optional control word (ip_data_controlword).			
1	ip_data_position	0	-2 ³¹	{1 inc}	2 ³² -1	—	INT32	RW	MAP
						Entry of the absolute position value.			
60C2 _h	0 interpolated_time_period					REC	UINT8	RO	—
						Reading-out of the synchronisation interval.			
1	ip_time_units	10	8	{1 ms}	40	—	UINT8	RW	MAP
			80	{0.1 ms}	400	Parameterisation of the synchronisation interval.			
2	ip_time_index	-3	-4	{1}	-3	—	INT8	RW	MAP
						Selection of the interval unit (ms or 1/10 ms).			
			-3			ip_time_units in 1 ms			
			-4			ip_time_units in 0.1 ms			

Index	Name	Possible settings		Characteristics							
		Lenze	Selection	Description							
60C4 _h	0	interpolated_data_configuration				REC	UINT8	RO	—	Reading-out of the buffer.	
	1	max_buffer_size				—	UINT32	RO	MAP	Reading-out of the size of the position buffer for interpolated position mode.	
	2	actual_buffer_size	0	0	{1}	2 ³² -1	—	UINT32	RW	MAP	Size of the actual position buffer for interpolated position mode.
	3	buffer_organization	0	0	{1}	255	—	UINT8	RW	MAP	Type of the buffer.
				0							FIFO
	4	buffer_position	0	0	{1}	65535	—	UINT16	RW	MAP	Access to the buffer.
	5	size_of_data_record	2	1	{1}	255	—	UINT8	WO	MAP	Size of one buffer element.
6	buffer_clear	0	0	{1}	1	—	UINT8	WO	MAP	Access to the buffer. Although there is no buffer available for the "linear interpolation without buffer" interpolation type, access must also be enabled in this case.	
			0								Clear buffer / access to 60C1 _h not enabled
			1								Access to 60C1 _h enabled

Operating modes

Synchronous position selection

Functional description

9.5.3

Functional description

Before the drive controller can be switched to the **interpolated position mode** operating mode, several settings have to be made.

These settings include the interpolation interval (**interpolation_time_period**), i.e. the time interval between two sync telegrams, and the interpolation type (**interpolation_submode_select**). Additionally, access to the position buffer must be enabled via the **buffer_clear** object.

Example:

Settings	Value	CAN object (COB) index	Entry
Interpolation type	-2	60C0 _n , interpolation_submode_select	-2
Time unit	0.1 ms	60C2_02 _h , interpolation_time_index	-04
Time interval	8 ms	60C2_01 _h , interpolation_time_units	80
Buffer enable	1	60C4_06 _h , buffer_clear	1
Sync generation		Sync (time base 8 ms)	

Activation of the interpolated position mode (IP) and synchronisation

The IP is activated via the **modes_of_operation** object. The **modes_of_operation_display** object can be used to read out whether the drive controller is in the **interpolated position mode** operating mode or not. From this instant on, the internal selectors are set in such a way that an internal position control is carried out. The setpoints received via CAN are interpolated first and then extrapolated (if required) to obtain the time interval set.

If the operating mode is reached, the transfer of position data to the drive can start. It makes sense that for this purpose the master first reads out the current actual position from the drive controller and cyclically writes this position to the controller as the new **interpolation_data_record** setpoint. **Control word** and the **status word** handshake bits activate the acceptance of the data by the drive controller. By setting the **enable_ip_mode** bit in the **control word**, the master indicates that the evaluation of the position data is to be started. Evaluation of the data records will not started before the drive controller has acknowledges this information via the **ip_mode_selected** status bit in the **status word**.

The below graphs show the assignments and the sequence in detail:

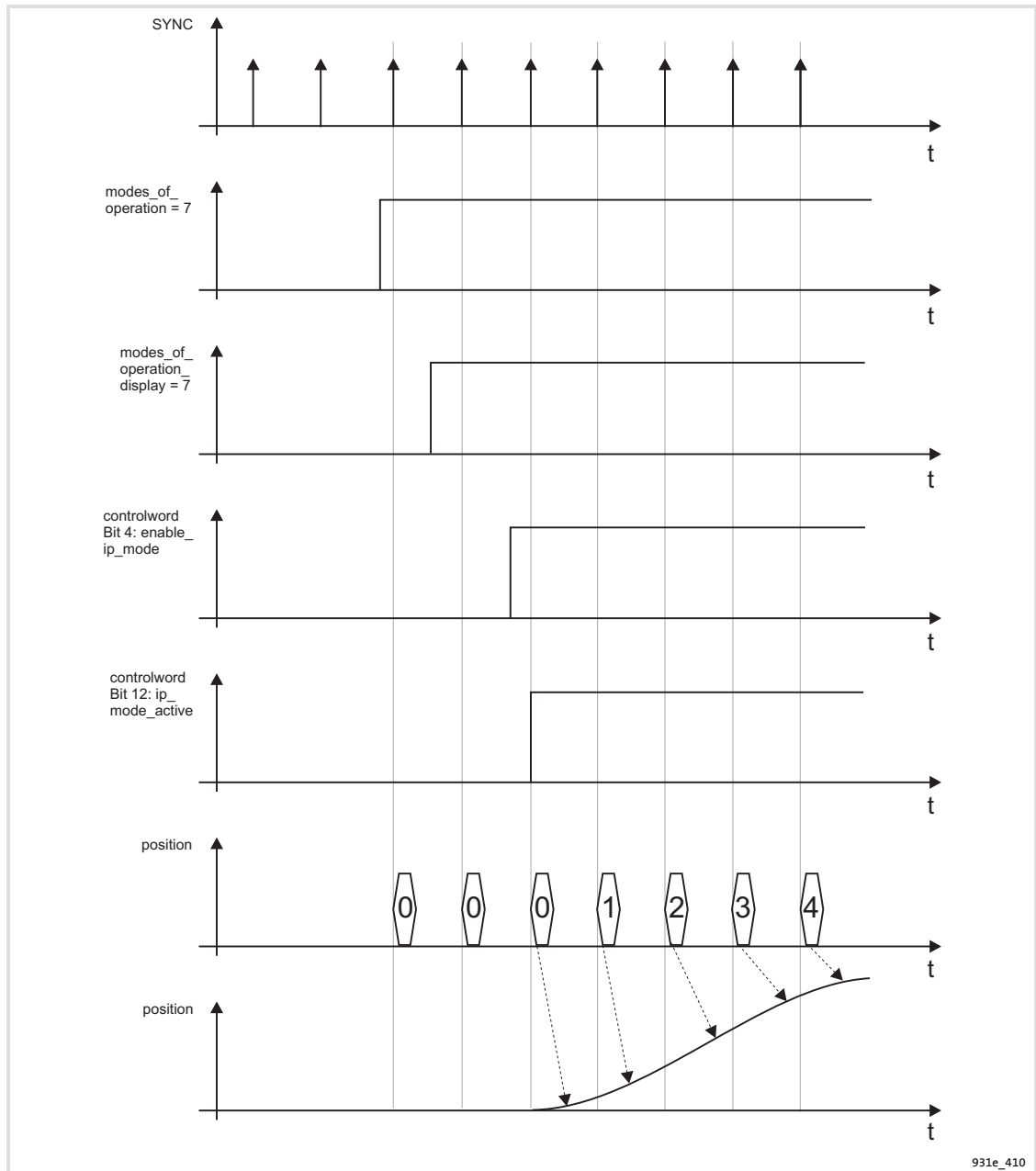


Fig. 25 IP switch-on and data release

Operating modes

Synchronous position selection

Functional description

No.	Event	CAN object
1	Generation of sync messages	
2	Request of IP operating mode	6060 _h , modes_of_operation = 07
3	Waiting for the operating mode being reached	6061 _h , modes_of_operation_display = 07
4	Reading-out of the current actual position	6064 _h , position_actual_value
5	Writing-back as the current set position	60C1_01 _h , ip_data_position
6	Start of the interpolation	6040 _h , control word, enable_ip_mode
7	Acknowledgement through the drive controller	6041 _h , status word, ip_mode_active
8	Change of the current set position according to trajectory	60C1_01 _h , ip_data_position

When the synchronous travel has been completed, further evaluation of position values can be suppressed by deleting the **enable_ip_mode** bit. It is then possible to switch to another operating mode if required.

Interpolation interruption in the event of errors

If a running interpolation (**ip_mode_active** is set) is interrupted by an error occurring in the drive controller, the drive first responds in the way specified for the respective error (e.g. cancelling of the controller enable and change to the **switch_on_disables** state).

In this case the interpolation can only be continued by switching on the IP parameters once more because the drive controller must again reach the **operation_enable** state which deletes the **ip_mode_active** bit.

9.6 Torque control

9.6.1 Overview

This chapter describes the torque-controlled operation. In this operating mode, an external **target-torque** setpoint can be specified for the drive controller. Thus, it is possible to use the drive controller also for those path controls shifting the position controller as well as the speed controller to an external computer.

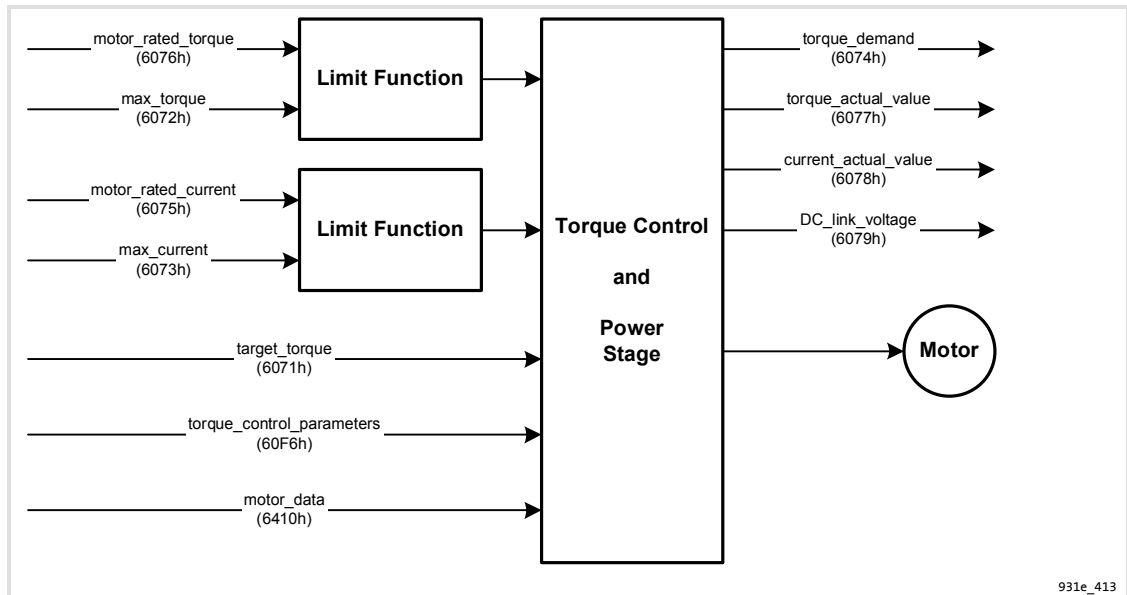


Fig. 26 Structure of torque-controlled operation

Ramp generators are not supported. If bit 8 (**stop**) of the **control word** is set, the current setpoint is set to zero. Correspondingly, the **target-torque** setpoint is limited to **max_torque** if bit 8 is deleted again.

All definitions in this document refer to rotary motors. If linear motors are used, all "torque" objects must refer to a "force". However, the objects do not appear twice and their names should not be changed.

The positioning (profile position mode) and speed control (profile velocity mode) operating modes require the torque controller for their functioning. For this reason the torque controller must always be parameterised.

9

Operating modes

Torque control

Description of the objects

9.6.2 Description of the objects

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
6071 _h	0 target_torque	0	-32768	{motorRatedTorque/1000}	32768	VAR	INT16	RW	MAP	Input value for the torque controller (torque control operating mode).
6072 _h	0 max_torque	2023	1000	{motorRatedTorque/1000}	65535	VAR	UINT16	RW	MAP	Input value for M_{max} . The value for index 6075 _h motorRatedCurrent must be entered before this value can be input.
6074 _h	0 torqueDemandValue			{motorRatedTorque/1000}		VAR	INT16	RO	MAP	Reading-out of the setpoint torque.
6076 _h	0 motorRatedTorque	1994	115	{0.001 Nm}	65535	VAR	UINT32	RW	MAP	Input value for M_r (specified on the motor nameplate).
6077 _h	0 torqueActualValue			{motorRatedTorque/1000}		VAR	INT16	RO	MAP	Actual torque value
6078 _h	0 currentActualValue			{motorRatedCurrent/1000}		VAR	INT16	RO	MAP	Reading-out of the actual current value.
6079 _h	0 DCLinkCircuitVoltage		0	{1 mV}	$2^{32}-1$	VAR	UINT32	RO	MAP	Reading-out of the DC-bus voltage.
60F6 _h	0 torqueControlParameters					REC	UINT8	RO	—	Reading-out of PI-controlled current controller data. The gain and the time constant apply to both the field-generating and the torque-generating current controller.
	1 torqueControlGain	256	0	{1}	32×256	—	UINT16	RW	—	Setting of the proportional gain of the current controller. From the SDC program: $K_p = 1.0$ Setting here: $1.0 \times 256 = 256$ (100 _h)
	2 torqueControlTime	2000	100	{1 μs}	65500	—	UINT16	RW	—	Setting of the current controller time constant. From the SDC program: $T_n = 2$ ms Setting here: 2 ms = 2000 μs

10 Appendix

10.1 Index table

- ▶ The indexes are numerically sorted in ascending order to form a "reference book".
- ▶ How to read the index table:

Column	Abbreviation			Meaning
Index	XXXX _h			Index XXXX _h
		01 _h		Subindex 1 of index XXXX _h
		02 _h		Subindex 2 of index XXXX _h
Name				Name of the index
Object type	REC			Record
	ARR			Array, composed type (field)
	VAR			Variable
Data type	UINT8			Unsigned integer, 1 byte without sign
	UINT16			Unsigned integer, 2 bytes without sign
	UINT32			Unsigned integer, 4 bytes without sign
	INT8			Integer, 1 byte with sign
	INT16			Integer, 2 bytes with sign
	INT32			Integer, 4 bytes with sign
Access	RO			Read only
	RW			Read and write
	WO			Write only
Mapping	–			No PDO mapping
	MAP			PDO mapping
Lenze				Lenze setting, value at delivery
Selection	1	{%}	99	Min. value {unit} max. value
Description				Short, important explanations

Index	Name	Possible settings			Characteristics							
		Lenze	Selection		Description							
1000 _h	0 device_type				VAR	UINT32	RO	—	Identification of the bus device in a multi-axis system.			
			00020192 _h							931E servo inverter		
1001 _h	0 error_register				VAR	UINT8	RO	MAP	Reading-out of the error register.			
			Bit no.	Meaning								
			0	generic error						An unspecified error has occurred (flag is set for every error message).		
			1	current								
			2	voltage								
			3	temperature								
			4	communication error						Communication error (CAN overrun).		
			5	device profile specific								
			6	reserved								
7	manufacturer specific						Manufacturer-specific error.					
1003 _h	0 pre_defined_error_field	0	0	{1}	255	ARR	UINT8	RW	—	Reading-out of the number of stored error messages. When an error has occurred, the error has to be acknowledged to activate the power stage (bit 7, control word index 6040 _h).		
			0						Memory is cleared			
			1	standard_error_field_0				—	UINT32	RO	—	Reading-out of last error message
			...									
4	standard_error_field_3				—	UINT32	RO	—	Reading-out of error message			
1005 _h	0 COB-ID_sync_message	00000080 _h	00000080 _h	{1 _h }	00000080 _h	VAR	UINT32	RW	—	Synchronisation object identifier 80 _h .		
			Bit no.	Value								
			0 - 10	x						11-bit identifier.		
			11 - 28	0						The extended identifier (bit 29) is not supported.		
			29	0						Each bit of this range must be "0".		
			30	0						Device does not generate sync telegrams.		
				1						Device generates sync telegrams.		
			31	x						Any		

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
1010 _h	0	store_parameters				ARR	UINT8	RO	—	
						Not used.				
	1	save_all_parameters	00000001 _h	00000000 _h	{1 _h }	65766173 _h	—	UINT32	RW	—
				00000000 _h			Acceptance of default parameter set in application parameter set.			
			00000000 _h			Default parameter set is not accepted.				
			65766173 _h	Save		Default parameter set is accepted.				
1011 _h	0	restore_parameters				ARR	UINT8	RO	—	
						Query on the number of possible objects.				
	1	restore_all_default_parameters	00000001 _h	00000000 _h	{1 _h }	64616F6C _h	—	UINT32	RW	—
							Loading of the default parameter set, only possible if the power stage is deactivated. The CAN communication parameters (node no., baud rate and operating mode) remain unchanged.			
			64616F6C _h	Load		Loading of default parameter set.				
			00000001 _h			Read access: Reset to default values.				
1014 _h	0	COB-ID_emergency_message	00000081 _h	00000000 _h	{1 _h }	00000081 _h	VAR	UINT32	RW	—
							Emergency object identifier, 080 _h + node address			
				Bit no.	Value					
				0 - 10	x		11-bit identifier			
				11 - 28	0		The extended identifier (bit 29) is not supported.			
				29	0		Each bit of this range must be "0".			
				30	0		Reserved			
				31	0		Emergency object is valid			
			1			Emergency object is invalid				
1017 _h	0	producer_heartbeat_time	0	0	{1 ms}	65536	VAR	UINT16	RW	—
							Time interval between two heartbeat telegrams. If the drive controller starts with a non-zero time value, the boot-up telegram is considered to be the first heartbeat.			
			0			Function is deactivated				

Index	Name	Possible settings		Characteristics															
		Lenze	Selection	Description															
1018 _h	0	identity_object				REC	UINT8	RO	—										
	Not used.																		
	1	vendor_id				—	UINT32	RO	—										
	Manufacturer code																		
				0000003B _h															
2	product_code					—	UINT32	RO	—										
										Product code									
											00001111 _h		931E						
3	revision_number					—	UINT32	RO	—										
Firmware version																			
4	serial_number					—	UINT32	RO	—										
Hardware serial number																			
1400 _h	Receive PDO1 Communication Parameter																		
0	number_of_entries		00 _h	{1 _h }	04 _h	REC	UINT8	RO	—										
			Maximum number of supported subindexes																
2 subindexes are supported.																			
1	COB-ID_used_by_PDO	00000201 _h	00000201 _h	{1 _h }	000002FF _h	—	UINT32	RW	—										
			Identifier of receive PDO1 (200 _h + node address) For processing, bits 30 and 31 must be set (parameterisation of mapping).																
			Bit no.	Value															
			0 - 10	x	11-bit identifier														
			11 - 28	0	The extended identifier (bit 29) is not supported.														
			29	0	Each bit of this range must be "0".														
			30	0	RTR of this PDO is permitted (Lenze) RTR = remote transmission request														
				1	RTR of this PDO is not permitted (unadjustable)														
31	0	PDO is active																	
	1	PDO is inactive																	
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—										
			Setting of the transmission mode																
			0	Function is switched off															
			n = 1 ... 240	By entering a value n, this PDO is accepted with every n-th sync.															
	n = 254, 255	Event-controlled transmission mode, PDO is accepted immediately																	

Index	Name	Possible settings		Characteristics										
		Lenze	Selection	Description										
1401 _h	Receive PDO2 Communication Parameter			00 _h	{1 _h }	04 _h	REC	UINT8	RO	—				
												Maximum number of supported subindexes		
	0	number_of_entries			02 _h					2 subindexes are supported.				
	1	COB-ID_used_by_PDO	00000301 _h	00000301 _h	{1 _h }	000003FF _h	—	UINT32	RW	—	Identifier of receive PDO2 (300 _h + node address) For processing, bits 30 and 31 must be set (parameterisation of mapping).			
							Bit no. Value							
							0 - 10	x						11-bit identifier
							11 - 28	0						The extended identifier (bit 29) is not supported.
							29	0						Each bit of this range must be "0".
							30	0						RTR of this PDO is permitted (Lenze) RTR = remote transmission request
									1					RTR of this PDO is not permitted (unadjustable)
							31	0						PDO is active
	1					PDO is inactive								
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—	Setting of the transmission mode				
						0					Function is switched off			
						n = 1 ... 240					By entering a value n, this PDO is accepted with every n-th sync.			
						n = 254, 255					Event-controlled transmission mode, PDO is accepted immediately			
1600 _h	Receive PDO1 Mapping Parameter			00 _h	{1 _h }	04 _h	REC	UINT32	RW	—				
											Maximum number of supported subindexes.			
	0	number_of_mapped_objects			01 _h				1 subindex is supported.					
	1	first_mapped_object	60400010 _h		{1 _h }	—	UINT32	RW	—	Entry of the COB ID of the first mapped object.				
	2	second_mapped_object								Not supported.				
	...													
4	fourth_mapped_object			—	UINT32	RW	—			Not supported.				

Index	Name	Possible settings		Characteristics			
		Lenze	Selection	Description			
1601 _h	Receive PDO2 Mapping Parameter						
0	number_of_mapped_objects		00 _h {1 _h } 04 _h	REC	UINT32	RW	—
				Maximum number of supported subindexes.			
			02 _h	2 subindexes are supported.			
1	first_mapped_object	60400010 _h	{1 _h }	—	UINT32	RW	—
				Entry of the COB ID of the first mapped object.			
2	second_mapped_object	60600008 _h	{1 _h }	—	UINT32	RW	—
				Entry of the COB ID of the second mapped object.			
3	third_mapped_object			—	UINT32	RW	—
				Not supported.			
4	fourth_mapped_object			—	UINT32	RW	—
				Not supported.			

Index	Name	Possible settings		Characteristics								
		Lenze	Selection	Description								
1800 _h	Transmit PDO1 Communication Parameter			00 _h	{1 _h }	04 _h	REC	UINT8	RO	—		
							Maximum number of supported subindexes.					
	0	number_of_entries		03 _h				3 subindexes are supported.				
	1	COB-ID_used_by_PDO	00000181 _h	00000181 _h	{1 _h }	000001FF _h	—	UINT32	RW	—		
							Identifier of transmit PDO1, (180 _h + node address). For processing, bits 30 and 31 must be set (parameterisation of mapping).					
							Bit no.	Value				
							0 - 10	x	11-bit identifier			
							11 - 28	0	The extended identifier (bit 29) is not supported.			
							29	0	Each bit of this range must be "0".			
							30	0	RTR of this PDO is permitted (Lenze).			
1								RTR of this PDO is not permitted (unadjustable).				
31	0	PDO is active										
	1	PDO is inactive										
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—			
						Setting of the transmission mode						
						0	Function is switched off					
						n = 1 ... 240	By entering a value n, this PDO is accepted with every n-th sync.					
			n = 254, 255	Event-controlled transmission mode								
3	inhibit_time	0	0	{0.1 ms}	65535	—	UINT16	RW	—			
						Setting of the minimum delay time between two PDOs. The time can only be changed if the PDO is not active (subindex 1, bit 31 = 1)						

Index	Name	Possible settings		Characteristics								
		Lenze	Selection	Description								
1801 _h	Transmit PDO2 Communication Parameter			00 _h	{1 _h }	04 _h	REC	UINT8	RO	—		
							Maximum number of supported subindexes					
	0	number_of_entries			03 _h			3 subindexes are supported.				
	1	COB-ID_used_by_PDO	00000281 _h	00000281 _h	{1 _h }	000002FF _h	—	UINT32	RW	—		
							Identifier of transmit PDO2, (280 _h + node address). For processing, bits 30 and 31 must be set (parameterisation of mapping).					
							Bit no.	Value				
							0 - 10	x	11-bit identifier			
							11 - 28	0	The extended identifier (bit 29) is not supported.			
							29	0	Each bit of this range must be "0".			
							30	0	RTR of this PDO is permitted (Lenze)			
1								RTR of this PDO is not permitted (unadjustable)				
31	0	PDO is active										
	1	PDO is inactive										
2	transmission_type	255	0	{1}	240, 254, 255	—	UINT8	RW	—			
						Setting of the transmission mode						
						0	Function is switched off					
						n = 1 ... 240	By entering a value n, this PDO is accepted with every n-th sync.					
					n = 254, 255	Event-controlled transmission mode						
3	inhibit_time	0	0	{0.1 ms}	65535	—	UINT16	RW	—			
						Setting of the minimum delay time between two PDOs. The time can only be changed if the PDO is not active (subindex 1, bit 31 = 1)						

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
1A00 _h	Transmit PDO1 Mapping Parameter									
	0 number_of_mapped_objects		00 _h	{1 _h }	04 _h	REC	UINT32	RW	—	
						Maximum number of supported subindexes.				
				01 _h			1 subindex is supported.			
	1 first_mapped_object	60410010 _h		{1 _h }		—	UINT32	RW	—	
						Entry of the COB ID of the first mapped object.				
2 second_mapped_object					—	UINT32	RW	—		
					Not supported.					
...										
4 fourth_mapped_object					—	UINT32	RW	—		
					Not supported.					
1A01 _h	Transmit PDO2 Mapping Parameter									
	0 number_of_mapped_objects		00 _h	{1 _h }	04 _h	REC	UINT32	RW	—	
						Maximum number of supported subindexes.				
				02 _h			2 subindexes are supported.			
	1 first_mapped_object	60410010 _h		{1 _h }		—	UINT32	RW	—	
						Entry of the COB ID of the first mapped object.				
2 second_mapped_object	60610008 _h		{1 _h }		—	UINT32	RW	—		
					Entry of the COB ID of the second mapped object.					
3 third_mapped_object					—	UINT32	RW	—		
					Not supported.					
4 fourth_mapped_object					—	UINT32	RW	—		
					Not supported.					
2014 _h	Transmit PDO1 Mask									
	0 number_of_entries					ARR	UINT8	RO	—	
						Maximum number of supported subindexes.				
1 tpdo1_transmit_mask_low	FFFFFFF _h	00000000 _h	{1 _h }	FFFFFFF _h	—	UINT32	RW	—		
					Mask for masking out individual bits of the PDOs.					
2 tpdo1_transmit_mask_high	FFFFFFF _h	00000000 _h	{1 _h }	FFFFFFF _h	—	UINT32	RW	—		
					Mask for masking out individual bits of the PDOs.					

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
2015 _h	Transmit PDO2 Mask				ARR	UINT8	RO	—		
					Maximum number of supported subindexes					
	1	tpdo2_transmit_mask_low	FFFFFFFF _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—
	Mask for masking out individual bits of the PDOs.									
2	tpdo2_transmit_mask_high	FFFFFFFF _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—	
Mask for masking out individual bits of the PDOs.										
2090 _h	0	velocity_ramps			REC	UINT8	RO	—		
					Not used.					
	2	velocity_acceleration_pos	01E84800 _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—
	Setting of the positive ramp acceleration.									
	3	velocity_deceleration_pos	01E84800 _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—
	Setting of the positive ramp deceleration.									
4	velocity_acceleration_neg	01E84800 _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—	
Setting of the negative ramp acceleration.										
5	velocity_deceleration_neg	01E84800 _h	00000000 _h	{1 _h }	FFFFFFFF _h	—	UINT32	RW	—	
Setting of the negative ramp deceleration.										
2415 _h	0	current_limitation			REC	INT8	RO	—		
					Limitation von I_{max} (independently of the operating mode). Torque-limited speed operation is possible.					
	1	limit_current_input_channel	00 _h	00 _h	{1 _h }	04 _h	—	INT8	RW	—
	Setpoint source for the limiting torque.									
	0	No limitation								
	1	AIN0								
2	AIN2									
3	RS232									
4	CAN									
2	limit_current	0	0	{1 mA}	100000	—	INT32	RW	—	
<ul style="list-style-type: none"> Setpoint source RS232, CAN: limitation of the torque-proportional current Setpoint sources AIN0, AIN1: Selection of the scaling factor for the analog inputs. The current in mA corresponds to an applied voltage of 10 V. 										
2C0A _h	0	lower_8bit_of_60FD			VAR	INT8	RO	MAP		
					Low byte (digital inputs and outputs).					

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
6040 _h	0 controlword	0000 _h	0000 _h	{1 _h }	FFFF _h	VAR	UINT16	RW	MAP
						Change of the drive controller state. An action is initiated (e.g. homing run).			
			Bit no.	Value					
			0	0001 _h		Control of the state transitions. (These bits are evaluated together).			
			1	0002 _h					
			2	0004 _h					
			3	0008 _h					
			4	0010 _h		The function of these bits depends on the operating mode.			
			5	0020 _h					
			6	0040 _h					
			7	0080 _h		<ul style="list-style-type: none"> ● reset_fault At the transition from zero to one, the drive controller tries to acknowledge the pending errors. This is only possible if the cause of the error has been eliminated.			
			8	0100 _h		The bit function depends on the operating mode.			
			9	0200 _h		Set to zero.			
			10	0400 _h					
			11	0800 _h					
			12	1000 _h					
			13	2000 _h					
			14	4000 _h					
			15	8000 _h					

Index	Name	Possible settings			Characteristics									
		Lenze	Selection		Description									
6041 _h	0	statusword		0000 _h	{1 _h }	FFF _h	VAR	UINT16	RO	MAP				
											Display of the controller state and of various events.			
				Bit no.		Value								
				0		0001 _h		State of the drive controller, see Tab. 15.						
				1		0002 _h		(These bits must be evaluated together).						
				2		0004 _h								
				3		0008 _h								
				4		0010 _h		<ul style="list-style-type: none"> voltage_disable This bit is set when the power stage transistors are switched on. Caution! In the event of a defect, the motor can still be energised.						
				5		0020 _h		State of the drive controller, see Tab. 15. <ul style="list-style-type: none"> quick_stop If the bit is deleted, the drive executes a quick stop according to index quick_stop_option_code.						
				6		0040 _h		State of the drive controller, see Tab. 15.						
				7		0080 _h		<ul style="list-style-type: none"> warning This bit is undefined. It must not be evaluated.						
				8		0100 _h		<ul style="list-style-type: none"> unused This bit is not used and must not be evaluated.						
				9		0200 _h		<ul style="list-style-type: none"> remote The bit is set if the controller enable logic is set correspondingly via index 6510_10 _h enable_logic.						
				10		0400 _h		The bit function depends on the operating mode.						
				11		0800 _h		<ul style="list-style-type: none"> internal_limit_active This bit indicates that the I ² t limitation is active.						
				12		1000 _h		The function of these bits depends on the operating mode.						
13		2000 _h												
14		4000 _h		<ul style="list-style-type: none"> unused This bit is not used and must not be evaluated.										
15		8000 _h		<ul style="list-style-type: none"> reserved This bit must not be evaluated.										
604D _h	0	pole_number	2	2	{2}	254	VAR	UINT8	RW	MAP				
							Number of motor poles.							

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
6060 _h	0 modes_of_operation		1	{1}	7	VAR	INT8	RW	MAP
						Selection of the operating mode			
			1			Position control with positioning operation			
			3			Speed control with setpoint ramp			
			4			Torque control with setpoint ramp			
			6			Homing			
			7			Synchronous position selection			
6061 _h	0 modes_of_operation_display					VAR	INT8	RO	MAP
						Display of the operating mode. If operation via CANopen is not possible, an internal operating mode is displayed.			
			-1			Unknown operating mode / change of operating mode			
			1			Position control with positioning operation			
			2			Speed control without setpoint ramp			
			3			Speed control with setpoint ramp			
			4			Torque control with setpoint ramp			
			6			Homing			
			7			Synchronous position selection			
			-11			Internal positioning operation			
			-12			Internal speed control without setpoint ramp			
			-13			Internal speed control with setpoint ramp			
			-14			Internal torque control			
			-15			Internal position control			
			6062 _h	0 position_demand_value		-2 ³¹	{1 inc}	2 ³¹ -1	VAR
						Reading-out of the position setpoint. This value is supplied to the position controller by the trajectory generator.			
6063 _h	0 position_actual_value		-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RO	MAP
						Reading-out of the actual position. The unit can be set via the factor group.			
6064 _h	0 position_actual_value		-2 ³¹	{position units}	2 ³¹ -1	VAR	INT32	RO	MAP
						Reading-out of the actual position.			

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
6065 _h	0 following_error_window	9102	00000000 _h	{1 inc}	7FFFFFFF	VAR	UINT32	RW	MAP	<p>Symmetrical range around the position setpoint. If the actual position value is outside the range, a following error occurs and bit 13 of the status word is set.</p> <p>Causes for following errors:</p> <ul style="list-style-type: none"> • The drive is blocked • The positioning speed is too high • The acceleration values are too high • The value entered for the following_error_window index is too low • The position controller is not parameterised correctly <p>The unit can be set via the factor group.</p>
6066 _h	0 following_error_time_out	100	0	{1 ms}	27314	VAR	UINT16	RW	MAP	<p>If the following error lasts longer than 100 ms, bit 13 of the status word is set.</p>
6067 _h	0 position_window	1820	-2 ³¹	{1 inc}	2 ³¹ -1	VAR	UINT32	RW	MAP	<p>Symmetrical range around the target position. The target position is reached if the actual position is within this range for a certain time.</p> <p>The unit can be set via the factor group.</p>
6068 _h	0 position_window_time	0	0	{1 ms}	65535	VAR	UINT16	RW	MAP	<p>If the actual position is within the position window for as long as defined here, bit 10 of the status word is set.</p>
6069 _h	0 velocity_sensor_actual_value			{1 inc/s}		VAR	INT32	RO	MAP	<p>Reading-out of the speed value directly on the encoder system. However, for determining the actual speed the object 606C_h should be used.</p>
606B _h	0 velocity_demand_value			{1 rpm}		VAR	INT32	RO	MAP	<p>Current speed setpoint. This value is affected by the setpoint of the ramp generator / the trajectory generator. If the position controller is activated, the correction speed of this controller is added.</p>

Index	Name	Possible settings			Characteristics					
		Lenze	Selection		Description					
606C _h	0 velocity_actual_value		{1 rpm}		VAR	INT32	RO	MAP	Reading-out of the actual speed.	
6071 _h	0 target_torque	0	-32768	{motorRatedTorque/1000}	32768	VAR	INT16	RW	MAP	Input value for the torque controller (torque control operating mode).
6072 _h	0 max_torque	2023	1000	{motorRatedTorque/1000}	65535	VAR	UINT16	RW	MAP	Input value for M_{max} . The value for index 6075 _h motorRatedCurrent must be entered before this value can be input.
6073 _h	0 max_current	2023	0	{motorRatedCurrent/1000}	65535	VAR	INT16	RW	MAP	Input value for I_{max} . The value for index 6075 _h motorRatedCurrent must be entered before this value can be input.
6074 _h	0 torque_demand_value		{motorRatedTorque/1000}			VAR	INT16	RO	MAP	Reading-out of the setpoint torque.
6075 _h	0 motorRatedCurrent	500	0	{1 mA}		VAR	UINT32	RW	MAP	Input value for I_r (specified on the motor nameplate). The value must be smaller than the rated controller current. If this index is changed, the index 6073 _h maxCurrent also must be parameterised again.
6076 _h	0 motorRatedTorque	1994	115	{0.001 Nm}	65535	VAR	UINT32	RW	MAP	Input value for M_r (specified on the motor nameplate).
6077 _h	0 torque_actual_value		{motorRatedTorque/1000}			VAR	INT16	RO	MAP	Actual torque value
6078 _h	0 current_actual_value		{motorRatedCurrent/1000}			VAR	INT16	RO	MAP	Reading-out of the actual current value.
6079 _h	0 DC_link_circuit_voltage		0	{1 mV}	$2^{32}-1$	VAR	UINT32	RO	MAP	Reading-out of the DC-bus voltage.

Index	Name	Possible settings				Characteristics				
		Lenze	Selection			Description				
607A _h	0 target_position	0	-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RW	MAP	<p>Entry of the target position (absolute or relative entry, see bit 6 of the control word).</p> <p>The current settings for speed, acceleration, braking deceleration and type of the travel profile must always be taken into account.</p> <p>The unit can be set via the factor group.</p>
607C _h	0 home_offset	0	-2 ³¹	{1 inc}	2 ³¹ -1	VAR	INT32	RW	MAP	<p>Shift of zero position with respect to home position.</p>
607E _h	0 polarity	00 _h	00 _h	{04 _h }	40 _h , 80 _h , C0 _h	VAR	UINT8	RW	MAP	<p>Setting of the signs of position and speed values. By changing the sign, the direction of rotation can be inverted.</p> <p>Often it makes sense to set both flags to the same value.</p>
			Bit 6	40 _h	0	multiply by 1	velocity_polarity flag			
					1	multiply by -1				
			Bit 7	80 _h	0	multiply by 1	position_polarity flag			
					1	multiply by -1				
6080 _h	0 max_motor_speed	32768	0	{1 rpm}	32768	VAR	UINT16	RW	MAP	<p>Setting of the maximum speed.</p> <p>The speed setpoint is limited to this value.</p>
6081 _h	0 profile_velocity	0	0	{1 rpm}	2 ³¹ -1	VAR	UINT32	RW	MAP	<p>Positioning: Speed at the end of the acceleration ramp.</p> <p>The unit can be set via the factor group.</p>
6082 _h	0 end_velocity	0	0	{1 rpm}	2 ³¹ -1	VAR	UINT32	RW	MAP	<p>Final speed</p> <p>The unit can be set via the factor group.</p>
			0			Drive controller stops when the target position is reached.				
			≠ 0			Gap-free positioning is not supported.				
6083 _h	0 profile_acceleration	10000	0	{1 rpm/s}		VAR	UINT32	RW	MAP	<p>Setting of the acceleration.</p> <p>The unit can be set via the factor group.</p>

Index	Name	Possible settings			Characteristics								
		Lenze	Selection		Description								
6084 _h	0 profile_deceleration	10000	0	{1 rpm/s}	VAR	UINT32	RW	MAP	Setting of the braking deceleration. The unit can be set via the factor group.				
6085 _h	0 quick_stop_deceleration	14100	0	{1 rpm/s}	VAR	UINT32	RW	MAP	Setting of the braking deceleration in the event of a quick stop. The unit can be set via the factor group.				
6086 _h	0 motion_profile_type	0	0	{1}	1	VAR	INT16	RW	MAP	Setting of the type of the positioning profile.			
			0						Linear ramp				
			1						Jerk-free acceleration				
6093 _h	0 position_factor					ARR	UINT32	RO	—	Conversion of length units (positions_units) to the internal unit (inc).			
			1	numerator	1	0	{1}	2 ³² -1	—		UINT32	RW	MAP
			2	divisor	1	0	{1}	2 ³² -1	—		UINT32	RW	MAP
6094 _h	0 velocity_encoder_factor					ARR	UINT32	RO	—	Conversion of speed values (speed_units) to the internal unit (rpm).			
			1	numerator	1	0	{1}	2 ³² -1	—		UINT32	RW	MAP
			2	divisor	1	0	{1}	2 ³² -1	—		UINT32	RW	MAP
										Ratio between revolutions at the output (rev _{OUT}) and movement in position units (degree or mm).			

Index	Name	Possible settings				Characteristics					
		Lenze	Selection			Description					
6097 _h	0 acceleration_factor					ARR	UINT32	RO	—	Conversion of acceleration values (acceleration_units) to the internal unit (rpm/256s).	
	1 numerator	1	0	{1}	2 ³² -1	—	UINT32	RW	MAP	Proportional to the gearbox ratio between input-end (rev _{IN}) and output-end (rev _{OUT}) speed.	
	2 divisor	1	0	{1}	2 ³² -1	—	UINT32	RW	MAP	Ratio between revolutions at the output (rev _{OUT}) and movement in position units (degree or mm).	
6098 _h	0 homing_method	17	-18	{1}	34	VAR	INT8	RW	—		
			Value	Direction	Target	Ref. point for zero					
			-18	Positive	Limit stop	Limit stop					Selection of the homing variant.
			-17	Negative	Limit stop	Limit stop					There are 3 homing signals:
			-2	Positive	Limit stop	Zero pulse					• Negative and positive limit switches
			-1	Negative	Limit stop	Zero pulse					• Zero pulse (periodic) of the angle encoder
			1	Negative	Limit switch	Zero pulse					• Negative and positive limit stops
			2	Positive	Limit switch	Zero pulse					If a method is selected for homing, the following settings are made:
			17	Negative	Limit switch	Limit switch					• The reference source (neg./pos. limit switch, neg. / pos. limit stop)
			18	Positive	Limit switch	Limit switch					• The direction and the sequence of the homing run
			33	Negative	Zero pulse	Zero pulse					• The way in which the zero pulse of the used angle encoder is evaluated
			34	Positive	Zero pulse	Zero pulse					
35		No motion	Current actual position								
6099 _h	0 homing_speeds					ARR	UINT32	RO	—	Reading-out of the homing speed	
	1 speed_during_search_for_switch	100	0	{1 rpm}	2 ³² -1	—	UINT32	RW	MAP	Speed for approaching the limit switch or the limit stop. Usually the speed is then changed and the final position is approached with a slower speed.	
	2 speed_during_search_for_zero	10	0	{1 rpm}	2 ³² -1	—	UINT32	RW	MAP	Selection of the slower speed	

Index	Name	Possible settings				Characteristics					
		Lenze	Selection		Description						
609A _h	0 homing_ acceleration	250	0	{1 rpm/s}	2 ³² -1	VAR	UINT32	RW	MAP		
							Selection of the acceleration and deceleration for the homing run. The value applies to all homing methods and to both homing speeds.				
60C0 _h	0 interpolated_ submode_ select	-2	-2	{}	-2	VAR	INT16	RW	MAP		
							Selection of the interpolation type.				
60C1 _h	0 interpolated_ data_ record					REC	UINT8	RO	—		
							Reading-out of the data record. The record consists of the position value (ip_data_position) and the optional control word (ip_data_controlword).				
	1 ip_data_ position	0	-2 ³¹	{1 inc}	2 ³² -1	—	INT32	RW	MAP		
							Entry of the absolute position value.				
60C2 _h	0 interpolated_ time_ period					REC	UINT8	RO	—		
							Reading-out of the synchronisation interval.				
	1 ip_time_ units	10	8	{1 ms}	40	—	UINT8	RW	MAP		
			80	{0.1 ms}	400	Parameterisation of the synchronisation interval.					
	2 ip_time_ index	-3	-4	{1}	-3	—	INT8	RW	MAP		
							Selection of the interval unit (ms or 1/10 ms).				
							ip_time_units in 1 ms				
								ip_time_units in 0.1 ms			

Index	Name	Possible settings			Characteristics						
		Lenze	Selection		Description						
60C4 _h	0	interpolated_data_configuration				REC	UINT8	RO	—		
	Reading-out of the buffer.										
	1	max_buffer_size				—	UINT32	RO	MAP		
	Reading-out of the size of the position buffer for interpolated position mode.										
	2	actual_buffer_size	0	0	{1}	2 ³² -1	—	UINT32	RW	MAP	
	Size of the actual position buffer for interpolated position mode.										
	3	buffer_organization	0	0	{1}	255	—	UINT8	RW	MAP	
Type of the buffer.											
FIFO											
4	buffer_position	0	0	{1}	65535	—	UINT16	RW	MAP		
Access to the buffer.											
5	size_of_data_record	2	1	{1}	255	—	UINT8	WO	MAP		
Size of one buffer element.											
6	buffer_clear	0	0	{1}	1	—	UINT8	WO	MAP		
			Access to the buffer. Although there is no buffer available for the "linear interpolation without buffer" interpolation type, access must also be enabled in this case.								
			0	Clear buffer / access to 60C1 _h not enabled							
1	Access to 60C1 _h enabled										
60F6 _h	0	torque_control_parameters				REC	UINT8	RO	—		
	Reading-out of PI-controlled current controller data. The gain and the time constant apply to both the field-generating and the torque-generating current controller.										
	1	torque_control_gain	256	0	{1}	32 × 256	—	UINT16	RW	—	
Setting of the proportional gain of the current controller. From the SDC program: K _p = 1.0 Setting here: 1.0 × 256 = 256 (100 _h)											
2	torque_control_time	2000	100	{1 μs}	65500	—	UINT16	RW	—		
Setting of the current controller time constant. From the SDC program: T _n = 2 ms Setting here: 2 ms = 2000 μs											

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
60F9 _h	0	velocity_control_parameter_set				REC	UINT8	RO	—
						Reading-out of the speed controller data.			
	1	velocity_control_gain	179	26 {1} 64 × 256	—	UINT16	RW	MAP	
						Setting of the speed controller gain. From the SDC program: $K_p = 0.7$ Setting here: $0.7 \times 256 = 179$			
2	velocity_control_time	8000	200 {1 μs} 32000	—	UINT16	RW	MAP		
					Setting of the speed controller time constant. From the SDC program: $T_n = 8 \text{ ms}$ Setting here: $8 \text{ ms} = 8000 \mu\text{s}$				
4	velocity_control_filter_time	1600	200 {1 μs} 32000	—	UINT16	RW	MAP		
					Setting of the filter time constant for the actual speed value. The filter serves to reduce the measurement noise. From the SDC program: $T = 1.6 \text{ ms}$ Setting here: $1.6 \text{ ms} = 1600 \mu\text{s}$				
60FA _h	0	control_effort		{speed units}	VAR	INT32	RO	MAP	
					Reading-out of the position controller correction speed. The correction speed is the difference between the set position and the actual position with consideration of the gain. This value is internally supplied to the speed controller as a setpoint.				

Index	Name	Possible settings			Characteristics				
		Lenze	Selection		Description				
60FB _h	0	position_control_parameter_set				REC	UINT8	RO	—
						Reading-out of the position controller data. The position controller operates with internal feedforwarding so that deviation control is minimised and the controller settling time is reduced.			
	1	position_control_gain	52	0 {1} 64 × 256	—	UINT16	RW	—	
						Setting of the position controller gain. From the SDC program: K _p = 0.2 Setting here: 0.2 × 256 = 52			
	4	position_control_v_max	500	0 {1 rpm} 32767	—	UINT32	RW	MAP	
					Limitation of the position controller correction speed. This is required since even small position deviations can cause considerable correction speeds.				
60FD _h	0	digital_inputs	0	00000000 _h {1} FFFFFFFF _h	VAR	UINT32	RO	MAP	
					Reading-out of the digital inputs.				
				Bit no. Value Digital input					
				0 00000001 _h Neg. limit switch					
				1 00000002 _h Pos. limit switch					
				3 00000008 _h Interlock (controller or power stage enable is missing)					
				16 - 25 03FF0000 _h DIN0 ... DIN9					
60FE _h	0	digital_outputs			ARR	UINT8	RO	—	
					Reading-out of the digital outputs.				
	1	digital_outputs_data	0	00000000 _h {1 _h } FFFFFFFF _h	—	UINT32	RW	MAP	
				Bit no. Value Digital output					
				0 00000001 _h Brake		Setting of the 2 outputs. The activation can be delayed by up to 1 ms. By reading back index 60FE_00 _h you can check when the outputs are really set.			
				16 00010000 _h Ready for operation					
				17, 18 00060000 _h DOUT1, DOUT2					

Index	Name	Possible settings				Characteristics				
		Lenze	Selection			Description				
60FF _h	0 target_velocity	0	-2 ³¹	{1 rpm}	2 ³¹ -1	VAR	INT32	RW	MAP	Setting of the setpoint selected for the ramp generator. The unit can be set via the factor group.
6410 _h	0 motor_data					REC	UINT8	RO	—	Reading-out of the motor data.
	3 iit_time_motor	2000	0	{1 ms}	10000	—	UINT16	RW	—	Setting of the time interval for which the motor can be fed with I _{max} (index 6073 _h). When this time has expired, the motor is automatically limited to the value set under 6075 _h .
	4 iit_ratio_motor			{1 ‰}		—	UINT16	RO	MAP	Reading-out of the actual utilisation ratio of the I ² t limitation.
10	phase_order	1	0	{1}	1	—	UINT16	RW	MAP	Setting of the phase sequence. See 'angle encoder' dialog box in the SDC parameterisation program.
			0							Phase sequence: left
			1							Phase sequence: right
11	resolver_offset_angle	11360	-32767	{1 inc}	32767	—	INT16	RW	MAP	Setting of the angle encoder orientation with respect to the permanent magnetic field of the rotor. See 'angle encoder offset angle' dialog box in the SDC parameterisation program. Conversion: $\alpha = \text{offset angle} \times 32767/180^\circ$ $= 11360$ (with offset angle = 62.4°)

Index	Name	Possible settings		Characteristics				
		Lenze	Selection	Description				
6510 _h	0 drive_data			REC	UINT8	RO	—	
				Not used.				
	1 serial_number			—	UINT32	RO	MAP	
				Reading-out of the serial number.				
	2 drive_code			—	UINT32	RO	MAP	
				Reading-out of the identification.				
10	enable_logic	2	0 {1}	2	—	UINT16	RW	MAP
					Setting of the power stage enable			
			0		Digital inputs of power stage enable + controller enable			
			1		Digital inputs of power stage enable + controller enable + RS232			
			2		Digital input of power stage enable + controller enable + CAN			
11	limit_switch_polarity	1	0 {1}	1	—	INT16	RW	MAP
					Setting of the limit switch polarity. The value always refers to both limit switches.			
			0		NC contact			
			1		NO contact			
15	limit_switch_deceleration	200000	0 {1 rpm/s}	200000	—	INT32	RW	MAP
					Setting of the deceleration used for braking when the limit switch is reached in normal operation (limit switch emergency stop ramp).			
A1	drive_type			—	UINT32	RO	MAP	
				Reading-out of the device type, see also index 1018_02 _h product_code.				
A9	firmware_main_version			—	UINT32	RO	MAP	
				Reading-out the main version number of the firmware (product version).				
AA	firmware_custom_version			—	UINT32	RW	MAP	
				Reading-out of the version number of the customer-specific firmware variant.				

11 Index

A

Activation of CANopen, 53
 Actual position value (position units), 82
 Actual value, position in position_units (position_actual_value), 82
 Analog inputs, 85
 Angle encoder offset, 76
 Application as directed, 9
 Approach new position, 109

B

Boot-up telegram, 52

C

Cable specification, 15
 CAN network, communication phases, 43
 CANopen
 - Communication, 18
 - Electrical connections, 15
 COB ID, 19
 Command code, 22
 Commissioning, 53
 Communication, CANopen, 18
 Communication data, 13
 Communication object identifier, 19
 Communication phases, 43
 Communication profile, 13
 Connection plan for RJ45, 16
 Control of the drive controller, 88
 Control of the homing run, 106
 Controller
 - Application as directed, 9
 - Labelling, 9
 Conversion factors, 70
 Correction speed, 82
 Current controller, 76
 Current limitation, 76

D

Definition of notes used, 12
 Device control, 88
 Device status of the heartbeat producer, 50
 Digital inputs, 85
 Digital outputs, 85
 Drive controller enable logic, 74

E

E82ZAFPC00x, baud rate, 13
 Electrical connections, CANopen, 15
 Electrical installation, 14
 EMC
 - Assembly, 14
 - Earthing, 14
 - Shielding, 14
 Emergency, 46
 Emergency telegram, 46
 Enable logic, 74
 Error, controller error, 46
 Error messages, 24
 Error register, 46

F

Factor group, 70
 Fault, 91
 Fault reaction active, 91
 Following error, 80
 - Error window, 82
 Following error time-out time, 82
 Following error window, 82

H

Heartbeat producer, 50
 Heartbeat telegram, 49
 Homing, 104
 Homing switch, 86

11 Index

I

Identifier, 19
 Inputs, analog, 85
 Installation, CE-typical drive system, assembly, 14
 Installation, electrical, 14

L

Labelling, controller, 9
 Legal regulations, 9
 Liability, 9
 Limit switch, 86
 Loading and saving of parameter sets, 67

M

Manufacturer, 9
 Monitoring of communication, 52
 Motor adaptation, 76
 Motor data, 76
 Motor parameters, rated current, 76

N

Network management (NMT), 43
 Network topology, 13
 Node address, 19
 Node ID, 19
 Not ready to switch on, 91
 Notes, definition, 12
 Number of pole pairs, 76
 Number of poles, 76

O

Offset of the angle encoder, 76
 Operating mode, setting of, 99
 Operating modes, 99
 Operation enable, 91
 Operator, 10

P

Parameter data transfer (SDO transfer), 21
 Parameter setting, 67
 PDO, 27
 PDOs, available, 27

Peak motor current, 76
 Personnel, qualified, 10
 Phase sequence, 76
 Position control, 61
 Position controller, 80
 - Output of, 82
 Position controller gain, 82
 Position controller output, 82
 Position controller parameters, 82
 Position controller time constant, 82
 Position reached message, 81
 Position setpoint (position units), 82
 Positioning, 107
 - Handshake, 109
 Power stage parameters, 74
 Process data objects, available, 27
 Process data transfer (PDO transfer), 27
 PROFIBUS-DP function module, communication medium, 13
 Profile velocity mode, 101

Q

Quick stop active, 91

R

Rated motor current, 76
 Reading parameters, 25
 Ready to switch on, 91

S

Safety instructions, 10
 - Definition, 12
 - General, 11
 - Structure, 12
 Scaling factors, 70
 SDO, 20
 Setpoint, position (position units), 82
 Setting of the operating mode, 99
 Shielding, EMC, 14
 Specification of the transmission cable, 15
 Speed control, 54, 101
 Speed controller, 79
 Start positioning, 109

State

- Fault, 91
- Fault reaction active, 91
- Not ready to switch on, 91
- Operation enable, 91
- Quick stop active, 91
- Ready to switch on, 91
- Switch on disabled, 91
- Switched on, 91

State diagram, 88 , 89

State diagram of the drive controller, 89

State machine , 89

Switch on disabled, 91

Switched on, 91

Sync telegram, 41

Synchronous position selection, 111

T

Target position window, 82

Target torque, 117

Target window, Position window, 82

Target window time, 82

Technical data, 13

Torque control, 117

Torque limitation, source, 76

Trajectory generator, 107

Transmission cable, specification, 15

Transmission parameters for PDOs, 28

U

User data, 21 , 22 , 23 , 24 , 27

V

Preface, 7

W

Warranty, 9

Waste disposal, 9

Writing parameters, 26






Notes





Lenze GmbH & Co KG Kleinantriebe
Hans-Lenze-Straße 1
D-32699 Extertal
Germany

KHB 13.0002-EN
2.0
© 02/2007
TD19

 +49 (0) 51 54 82-0
 Service 00 80 00 24 4 68 77 (24 h helpline)
 Service +49 (0) 51 54 82-1112
E-Mail Lenze@Lenze.de
Internet www.Lenze.com