CAN Bus

Fieldbus Interface SERVOSTAR 400/600

CANopen®

Edition 11/2018
Translation of the original manual

For safe and proper use,
follow these instructions.
Keep them for future reference.

Previous editions

Edition	Comments
01 / 1999	First edition, valid from software version 1.57
08 / 1999	Layout, minor corrections
03 / 2001	Expanded Object Dictionary, valid from firmware version 4.20
08 / 2001	Minor corrections, index expanded, object dictionary expanded, valid from firmware version 4.80
09 / 2002	new design, TPDO 34 added, minor corrections
09 / 2006	new design, expanded Object Dictionary, valid from firmware version 6.57
08 / 2007	Minor correction, symbols, product brand, standards
07 / 2009	Can logo, product brand
12 / 2009	Minor corrections, symbols according to ANSI Z535
12 / 2010	New company name
07 / 2014	Design cover page, warning notes updated
04 / 2016	Warning signs updated, european directives updated, safe voltage changed to 50V
07 / 2016	Use as directed updated, several graphics improved
11 / 2018	Layout of the warning notes updated, user expertise updated, new readers note on cover page

Technical changes to improve the performance of the equipment may be made without prior notice !

Page

1 General
1.1 About this manual . . . 7
1.2 Target group . . . 7
1.3 Hints for the online edition (PDF format) . . . 7
1.4 Use as directed of the CANopen interface . . . 8
1.5 Symbols used . . . 8
1.6 Abbreviations used . . . 9
1.7 Basic features implemented by CANopen . . . 9
1.8 Response to BUSOFF communication faults . . . 10
1.9 System requirements . . . 10
1.10 Transmission rate and procedure . . . 10

2 Installation / Setup
2.1 Important notes . . . 11
2.2 Setting the station address and transmission rate . . . 12
2.3 CANopen interface . . . 13
2.4 CAN-bus cable . . . 13
2.5 Guide to Setup . . . 14
2.6 Important configuration parameters for CAN bus operation . . . 15

3 CANopen communication profile
3.1 General description of CAN . . . 17
3.2 Construction of the COB Identifier . . . 18
3.3 Definition of the used data types . . . 18
3.3.1 Basic data types . . . 19
3.3.1.1 Unsigned Integer . . . 19
3.3.1.2 Signed Integer . . . 19
3.3.2 Mixed data types . . . 19
3.3.3 Extended data types . . . 20
3.3.3.1 Octet String . . . 20
3.3.3.2 Visible String . . . 20
3.4 Communication Objects . . . 20
3.4.1 Network Management Objects (NMT) . . . 21
3.4.2 Synchronization Object (SYNC) . . . 21
3.4.3 Time-Stamp Object (TIME) . . . 21
3.4.4 Emergency Object (EMCY) . . . 21
3.4.4.1 Application of the Emergency Object . . . 22
3.4.4.2 Composition of the Emergency Object . . . 22
3.4.5 Service Data Objects (SDO) . . . 23
3.4.5.1 Composition of the Service Data Object . . . 23
3.4.5.2 Initiate SDO Download Protocol . . . 24
3.4.5.3 Download SDO Segment Protocol . . . 24
3.4.5.4 Initiate SDO Upload Protocol . . . 24
3.4.5.5 Upload SDO Segment Protocol . . . 24
3.4.5.6 Abort SDO Protocol . . . 25
3.4.6 Process Data Object (PDO) . . . 25
3.4.6.1 Transmission modes . . . 26
3.4.6.2 Trigger modes . . . 27
3.4.7 Nodeguard . . . 27

Page

4 CANopen Drive Profile

4.1 Emergency Messages . . . 29
4.2 General definitions . . . 31
4.2.1 General Objects . . . 31
4.2.1.1 Object 1000h: Device Type (DS301) . . . 31
4.2.1.2 Object 1001h: Error register (DS301) . . . 31
4.2.1.3 Object 1002h: Manufacturer Status Register (DS301) . . . 32
4.2.1.4 Object 1003h: Pre-Defined Error Field (DS301) . . . 33
4.2.1.5 Object 1004h: Number of supported PDOs (DS301) . . . 34
4.2.1.6 Object 1005h: COB-ID of the SYNC Message (DS301) . . . 35
4.2.1.7 Object 1006h: Communication Cycle Period (DS301) . . . 35
4.2.1.8 Object 1007h: Synchronous window length (DS301) . . . 36
4.2.1.9 Object 1008h: Manufacturer Device Name (DS301) . . . 36
4.2.1.10 Object 100Ah: Manufacturer Software Version (DS301) . . . 36
4.2.1.11 Object 100Bh: Node-ID (DS301) . . . 37
4.2.1.12 Object 100Ch: Guard Time (DS301) . . . 37
4.2.1.13 Object 100Dh: Lifetime Factor (DS301) . . . 37
4.2.1.14 Object 100Eh: COB-ID Nodeguarding (DS301) . . . 38
4.2.1.15 Object 100Fh: Number of the supported Objects (DS301) . . . 38
4.2.1.16 Object 1010h: Store Parameters (DS301) . . . 39
4.2.1.17 Object 1012h: COB-ID for Time-Stamp Message (DS301) . . . 40
4.2.1.18 Object 1013h: High Resolution Time-Stamp (in preparation) (DS301) . . . 40
4.2.1.19 Object 1014h: COB-ID for Emergency Message (DS301) . . . 40
4.2.1.20 Object 1018h: Identity Object (DS301) . . . 41
4.3 PDO Mapping . . . 42
4.3.1 Receive-PDOs (RPDO) . . . 43
4.3.1.1 Description of predefined Receive-PDOs . . . 43
4.3.1.1.1 PDO controlword (1) (DS402) . . . 43
4.3.1.1.2 PDO Receive ASCII Channel (21) . . . 44
4.3.1.1.3 PDO Current/Speed Setpoint (22) . . . 44
4.3.1.1.4 PDO Setpoint 2 (32) . . . 44
4.3.1.1.5 PDO Trajectory (33) . . . 45
4.3.1.1.6 PDO Motion Block (34) . . . 45
4.3.1.1.7 PDO Start Motion Block (35) . . . 46
4.3.1.1.8 PDO Free Definition (37 to 40) . . . 46
4.3.1.1.9 PDO master position for CAN slaves (41) . . . 46
4.3.1.2 Object Decription . . . 47
4.3.1.2.1 Object 1400-1403h: 1st-4th Receive-PDO communication parameter (DS301) . . . 47
4.3.1.2.2 Object 1600-1603h: 1st-4th Receive-PDO mapping parameter (DS301) . . . 47
4.3.1.2.3 Object 2600-2603h: 1st-4th Receive PDO select . . . 47
4.3.1.2.4 Object 2721h: Configuration of Receive-PDO 33 . . . 47
4.3.2 Transmit-PDOs (TPDO) . . . 48
4.3.2.1 Description of the predefined Transmit-PDOs . . . 48
4.3.2.1.1 PDO Status Word (1) (DS402) . . . 48
4.3.2.1.2 PDO Transmit ASCII Channel (21) . . . 49
4.3.2.1.3 PDO Actual Position (22) . . . 49
4.3.2.1.4 PDO Enhanced Status (23) . . . 49
4.3.2.1.5 PDO Actual Position 2 (32) . . . 49
4.3.2.1.6 PDO Incremental Actual Position (33) . . . 50
4.3.2.1.7 PDO Position Treshold (34) . . . 50
4.3.2.1.8 PDO Free Definition (37 to 40) . . . 50
4.3.2.1.9 PDO Internal Master Position Setpoint (41) . . . 50
4.3.2.2 Object Description . . . 51
4.3.2.2.1 Object 1800-1803h: 1st-4th Transmit-PDO communcation param. (DS301) . . . 51
4.3.2.2.2 Object 1A00-1A03h: 1st-4th Transmit-PDO mapping parameter (DS301) . . . 51
4.3.2.2.3 Object 2A00-2A03h: 1st-4th Transmit-PDO select (DS301) . . . 51
4.3.2.2.4 Object 2014-2017h: 1st-4th Mask 1 to 4 for Transmit-PDO . . . 52

Page

4.4 Device Control (dc) . . . 52
4.4.1 Status Machine (DS402) . . . 53
4.4.1.1 States of the Status Machine . . . 53
4.4.1.2 Transitions of the status machine . . . 54
4.4.2 Object description . . . 55
4.4.2.1 Object 6040h: Controlword (DS402) . . . 55
4.4.2.2 Object 6041h: Statusword (DS402) . . . 56
4.4.2.3 Object 6060h: modes_of_operation (DS402) . . . 58
4.4.2.4 Object 6061h: modes_of_operation_display (DS402) . . . 59
4.5 Factor Groups (fg) (DS402) . . . 59
4.5.1 General Information . . . 59
4.5.1.1 Factors . . . 59
4.5.1.2 Relationship between physical and internal units . . . 59
4.5.2 Object Description . . . 60
4.5.2.1 Object 608Bh: velocity_notation_index (DS402) . . . 60
4.5.2.2 Object 608Ch: velocity_dimension_index (DS402) . . . 60
4.5.2.3 Object 6093h: position_factor (DS402) . . . 61
4.5.2.4 Object 6094h: velocity_encoder_factor (DS402) . . . 62
4.5.2.5 Object 6097h: acceleration_factor (DS402) . . . 64
4.6 Manufacturer-specific Current and Speed-Mode . . . 65
4.6.1 Object 2060h: Digital current or speed setpoint . . . 65
4.6.2 Objekt 2061h: Current limitation . . . 65
4.7 Setup data for manufacturer-specific Jogging/Homing mode . . . 66
4.7.1 Object 2024h: Setup operation for position mode (SERVOSTAR) . . . 66
4.8 Positioning data for position mode (SERVOSTAR) . . . 68
4.8.1 Object 2020h: Position controller . . . 68
4.8.2 Object 2022h: Position data for position mode . . . 71
4.9 Object 2050h: Auxiliary variable for digital inputs . . . 77
4.10 Latch function . . . 78
4.10.1 Object 2026h: Latch enable . . . 78
4.10.2 Object 2082h: 32/24-bit Latch positive . . . 78
4.10.3 Object 2083h: 32/24-bit Latch negative . . . 79
4.10.4 Object 2084h: 16-bit Latch positive . . . 79
4.10.5 Object 2085h: 16-bit Latch negative . . . 80
4.10.6 Object 2087h: Latch Positions Digital Input 1 . . . 81
4.11 Manufacturer specific values . . . 82
4.11.1 Object 2070h: Actual values . . . 82
4.11.2 Objekt 6077h: Torque actual value . . . 89
4.11.3 Objekt 60C2h: Interpolation time period . . . 89
4.12 Freely available, mappable PLC variables, Objects 2030h / 2090h . . . 90
4.12.1 Object 2030h: DP-Ram-Variables 9-16 (write only) . . . 90
4.12.2 Object 2090h: DP-Ram-Variables 1-8 (read only) . . . 90
4.13 Dummy Variables, Objects 2031h / 2071h . . . 91
4.13.1 Object 2031h: Dummy variables for mapping (WO) . . . 91
4.13.2 Object 2071h: Dummy variables for mapping (RO) . . . 92
4.14 Profile Velocity Mode (pv) (DS402) . . . 93
4.14.1 General Information . . . 93
4.14.2 Objects that are defines in this section . . . 93
4.14.3 Objects that are defines in other sections . . . 93
4.14.4 Object Description . . . 93
4.14.4.1 Object 606Ch: velocity_actual_value* (DS402) . . . 93
4.14.4.2 Object 60FFh: target_velocity (DS402) . . . 94
4.15 Position Control Function (pc) (DS402) . . . 94
4.15.1 General Information . . . 94
4.15.2 Objects that are defined in this section . . . 94
4.15.3 Objects that are defined in other sections . . . 94
4.15.4 Object Decription . . . 95
4.15.4.1 Object 6063h: position_actual_value* (DS402) . . . 95
4.15.4.2 Object 6064h: position_actual_value (DS402) . . . 95

Page

4.16 Homing Mode (hm) (DS402) . . . 96
4.16.1 General Information . . . 96
4.16.2 Objects that are defined in this section . . . 96
4.16.3 Objects that are defined in other sections . . . 96
4.16.4 Object Description . . . 96
4.16.4.1 Object 607Ch: home_offset (DS402) . . . 96
4.16.4.2 Object 6098h: homing_method (DS402) . . . 97
4.16.4.2.1 Description of the homing methods . . . 98
4.16.4.3 Object 6099h: homing_speeds (DS402) . . . 98
4.16.4.4 Object 609Ah: homing_acceleration (DS402) . . . 98
4.16.5 Homing Mode Sequence . . . 99
4.17 Profile Position Mode (pp) . . . 99
4.17.1 General Information . . . 99
4.17.2 Objects that are defined in this section . . . 99
4.17.3 Objects that are defined in other sections . . . 100
4.17.4 Object Description . . . 100
4.17.4.1 Object 607Ah: target_position (DS402) . . . 100
4.17.4.2 Object 607Bh: position_range_limit (DS402) . . . 101
4.17.4.3 Object 6081h: profile_velocity (DS402) . . . 101
4.17.4.4 Object 6083h: profile_acceleration (DS402) . . . 102
4.17.4.5 Object 6084h: profile_adeceleration (DS402) . . . 102
4.17.4.6 Object 6086h: motion_profile_type (DS402) . . . 103
4.17.5 Functional Description . . . 103

5 The Object Channel

5.1 Object Description . . . 107
5.1.1 Object > 3500h Manufacturer Specific Object Channel . . . 107

6 Appendix

6.1 Setup examples . . . 117
6.1.1 Basic testing of the connection control<->SERVOSTAR . . . 117
6.1.2 Example of operating the Status Machine . . . 118
6.1.3 Example of PDO usage . . . 119
6.1.4 Example of Homing . . . 121
6.1.5 Example of Motion Block Processing . . . 122
6.1.6 Example of Profile Positioning Mode usage . . . 123
6.1.7 ASCII Communication . . . 125
6.1.8 Test for SYNC-telegrams . . . 126
6.1.9 SYNC-Object . . . 126
6.1.10 Emergency-Object . . . 126
6.2 Special Applications . . . 127
6.2.1 External Trajectory . . . 127
6.2.1.1 Position controller in the servo amplifier . . . 127
6.2.1.2 Position controller in the control system . . . 130
6.3 Object Dictionary . . . 131
6.4 New configuration of SERVOSTAR 400/600 . . . 137
6.5 Index . . . 138

1 General

1.1 About this manual

This manual describes the setup, range of functions and software protocol of the SERVOSTAR 400/600 servo amplifiers with the CANopen communication profile. It forms part of the complete documentation for the SERVOSTAR 400/600 family of servo amplifiers.

The installation and setup of the servo amplifier, as well as all standard functions, are described in the corresponding instructions manual.

Other parts of the complete documentation for the digital servo amplifier series:

Title	Publisher
Instructions manual SERVOSTAR 400/600	Kollmorgen
Online-Help with object reference (setup software)	Kollmorgen

Additional documentation:

Title	Publisher
CAN Application (CAL) for Industrial Applications	CiA e.V.
Draft Standards 301 (from Version 4.0), 401	CiA e.V.
CAN Specification Version 2.0	CiA e.V.
ISO 11898 ... Controller Area Network (CAN) for high-speed communication	

1.2 Target group

This manual addresses personnel with the following qualifications:

Transport :	only by personnel with knowledge of handling electrostatically sensitive components.
Unpacking:	only by electrically qualified personnel.
Installation :	only by electrically qualified personnel.
Setup :	only by qualified personnel with extensive knowledge of electrical engineering and drive technology
Programming:	Software developers, CAN bus project-planners

The qualified personnel must know and observe the following standards:
IEC 60364, IEC 60664, and regional accident prevention regulations

Qualified Personnel only!

During operation there are deadly hazards, with the possibility of death, severe injury or material damage.

- The user must ensure that the safety instructions in this manual are followed.
- The user must ensure that all personnel responsible for working with the servo amplifier have read and understood the instructions manual.

Training courses are available on request. Specific examples for individual tasks can be found in the applications section (⇒ 6.1 and 6.2) of this manual.

1.3 Hints for the online edition (PDF format)

Bookmarks:
Table of contents and index are active bookmarks.
Table of contents and index in the text:
The lines are active cross references. Click on the desired line and the appropriate page is accessed.
Page/chapter numbers in the text:
Page/chapter numbers with cross references are active. Click at the page/chapter number to reach the indicated target.

1.4 Use as directed of the CANopen interface

Please observe the chapter “Permitted use” in the instructions manual for the servo amplifier.
The interface is a component part of the SERVOSTAR 400/600 series of digital servo amplifiers.
The CANopen interface serves only for the connection of the servo amplifier to a master via the CAN-bus.
The servo amplifiers are components that are built into electrical apparatus or machinery, and can only be setup and operated as integral components of such apparatus or machinery.

NOTE

We only guarantee the conformity of the servo amplifier with the directives listed in the EU Declaration of Conformity, if the components that we specify are used, and the installation regulations are followed.

1.5 Symbols used

Symbol	Indication
DANGER	Indicates a hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
NOTICE	This is not a safety symbol. Indicates situations which, if not avoided, could result in property damage.
NOTE	This is not a safety symbol. This symbol indicates important notes.
	Warning of a danger (general). The type of danger is specified by the warning text next to it.
	Warning of danger from electricity and its effects.
	Warning of danger from automatic start.

1.6 Abbreviations used

The abbreviations used in this manual are explained in the table below.

Abbrev.	Meaning
BTB/RTO	Ready to operate (standby)
COB	Communication Object
COB-ID	Communication Object Identifier
EEPROM	Electrically erasable/programmable memory
EMC	Electromagnetic compatibility
ISO	International Standardization Organization
km	1000 m
LED	Light-emitting diode
MB	Megabyte
NSTOP	Limit switch for CCW (left) rotation
PC	Personal Computer
PDO	Process Data Object
PSTOP	Limit switch for CW (right) rotation
RAM	Volatile memory
ROD	Incremental position encoder
RXPDO	Receive PDO
SDO	Service Data Object
TXPDO	Transmit PDO

1.7 Basic features implemented by CANopen

When working with the position controller that is integrated in SERVOSTAR 400/600 digital servo amplifiers, the following functions are available:

Setup and general functions:

— homing, set reference point
— jogging, with a variable speed
— provision of a digital setpoint for speed and torque control

Positioning functions:

— execution of a motion task from the motion block memory of the servo amplifier
— execution of a direct motion task
— absolute trajectory

Data transfer functions:

— transmit a motion task to the motion block memory of the servo amplifier
  A motion task consists of the following elements:
  » position setpoint (absolute task) or path setpoint (relative task)
  » speed setpoint
  » acceleration time, braking time, rate-of-change/jolt limiting (in preparation)
  » type of motion task (absolute/relative)
  » number of a following task (with or without pause)
— read a motion task from the motion block memory of the servo amplifier
— read actual values
— read the error register
— read the status register
— read/write control parameters

1.8 Response to BUSOFF communication faults

The communication fault BUSOFF is directly monitored and signaled by Level 2 (CAN controller). This message may have various causes.

A few examples:
— telegrams are transmitted, although there is no other CAN node connected
— CAN nodes have different transmission rates
— the bus cable is faulty
— faulty cable termination causes reflections on the cable.

A BUSOFF is only signaled by the SERVOSTAR 400/600 if another CAN node is connected and at least one Object was successfully transmitted to start off with. The BUSOFF condition is signaled by the error message F23. If the output stage is enabled for the execution of a motion task at the moment when this fault occurs, then the drive is braked to a stop, using the emergency stop ramp, and the output stage is disabled.

1.9 System requirements

— servo amplifier SERVOSTAR 400/600
— master station with a CAN-bus interface (e.g. PC with CAN interface)

1.10 Transmission rate and procedure

— bus connection and bus medium: CAN-standard ISO 11898 (CAN high-speed)
— transmission rate: max. 1Mbit/s
possible settings for the servo amplifier:
10, 20, 50, 100, 125, 250, 333, 500 (default), 666, 800, 1000 kbit/s

2 Installation / Setup

2.1 Important notes

DANGER High Voltages up to 900V!

Risk of electric shock. Residual charges in the capacitors can still have dangerous levels several minutes after switching off the supply voltage. Power and control connections can still be live, even though the motor is not rotating.

- Install and wire up the equipment only while it is not electrically connected.
  Make sure that the control cabinet is safely isolated (lock-out, warning signs etc.).The individual supply voltages will not be switched on until setup is carried out.
- Measure the voltage in the intermediate (DC-link) circuit and wait until it has fallen below 50V.

WARNING Automatic Start!

Risk of death or serious injury for humans working in the machine. Drives with servo amplifiers in fieldbus systems are remote-controlled machines. They can start to move at any time without previous warning.

- Implement appropriate protective measures to ensure that any unintended start-up of the machines cannot result in dangerous situations for personnel or machinery.
- The user is responsible for ensuring that, in the event of a failure of the servo amplifier, the drive is set to a state that is functional safe, for instance with the aid of a safe mechanical brake.
- Software limit-switches are not a substitute for the hardware limit-switches in the machine.

NOTICE

Assemble the servo amplifier as described in the instructions manuals for SERVOSTAR 400/600. Observe all safety instructions in the installation instructions that belong to the servo amplifier. Follow all the notes on mounting position, ambient conditions, wiring, and fusing.

2.2 Setting the station address and transmission rate

During setup it makes sense to use the keypad on the servo amplifier's front panel to preset the station addresse and the Baud rate for communication (see setup instructions in the instructions manual).

Setting the **station address**

The station address can be set in three different ways:

- by using the pushbuttons on the front panel
  (see instructions manual for SERVOSTAR 400/600)
- In the setup software DRIVE.EXE, on the screen page “Basic settings”
- Using the ASCII command sequence: ADDR nn ⇒ SAVE ⇒ COLDSTART (nn = address)

NOTE

After changing the station address and/or baud rate you must turn the 24V auxiliary supply for the servo amplifier off and on again for reset.

The address range can be expanded from 1 ... 63 to 1 ... 127 by using the ASCII-Object MDRV.

Setting the **transmission rate**

The CAN transmission rate can be set in three different ways:

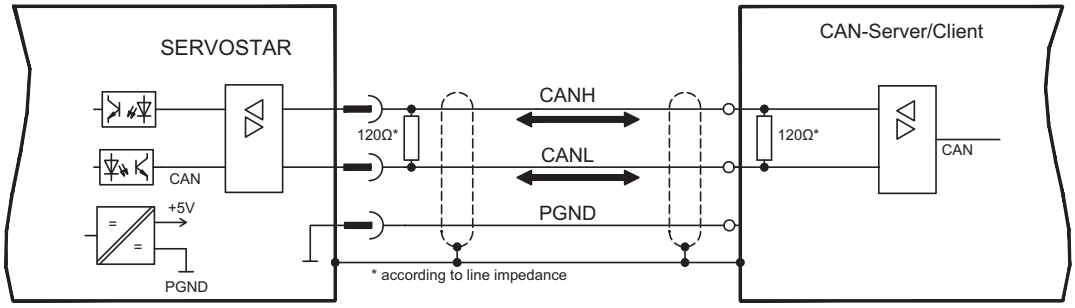
- by using the pushbuttons on the front panel
  (see instructions manual for SERVOSTAR 400/600)
- In the setup software DRIVE.EXE, on the screen page “Basic settings”
- Using the ASCII command sequence: CBAUD bb ⇒ SAVE ⇒ COLDSTART
  (bb = transmission rate in kbit/s)

Possible transmission rates are: 20, 50, 100, 125, 250, 333, 500 (default), 666, 800, 1000 kbit/s.

2.3 CANopen interface

The interface for connection to the CAN-bus (default : 500 kBaud). The interface is at the same electrical potential as the RS232 interface. The analog setpoint inputs can still be used.

We can supply special clamp-sleeve connectors, that can easily be made up for bus operation. In addition, an adapter can be used for the option slot (Option -2CAN-), which provides options for pass-through wiring and the attachment of 120Ω termination resistors. The pin assignments correspond to ISO 11898 (CAN).



NOTE For the purpose of equipotential bonding, AGND must be connected to the controller! For additional information, see the servo amplifier instructions manual.

2.4 CAN-bus cable

In accordance with ISO 11898 you should use a bus cable with a characteristic impedance of 120Ω. The usable cable length for reliable communication is reduced as the transmission rate is increased. The following values that we have measured can be used as a guide. They should not, however, be interpreted as limiting values:

Cable data:	characteristic impedance	100 ... 120Ω
	cable capacitance	max. 60 nF/km
	lead resistance (loop)	159.8 Ω/km

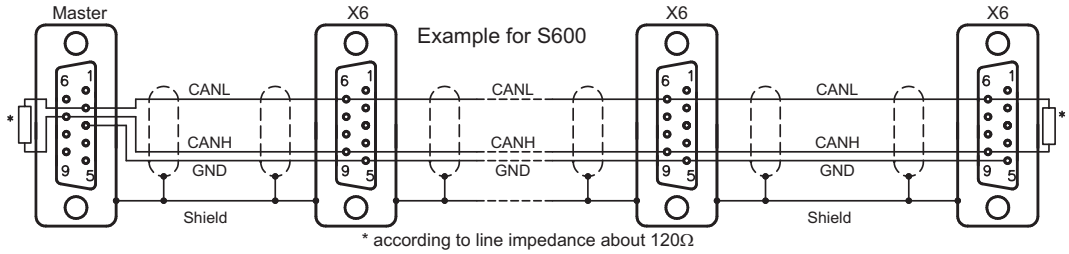
Cable length, dependent on the transmission rate

Transmission rate: kbit/s	Max. cable length: m
1000	20
500	70
250	115

Longer transmission distances may be achieved with a lower cable capacitance (max. 30 nF/km) and lower lead resistance (loop, 115 Ω/km).
(characteristic impedance 150 ± 5Ω ⇒ termination resistance 150 ± 5Ω).

For EMC reasons, the SubD connector housing must fulfill the following requirements:

— metal or metallic housing
— provision for connecting the cable shielding in the housing, with a large contact area.



We supply special clamp-sleeve connectors (Order No. DE-90650), that can easily be made up for bus operation. In addition, an adapter can be used for the S600 option slot (Option -2CAN-, Order No. DE-101174), which provides options for pass-through wiring and the attachment of 120Ω termination resistors. The pin assignments correspond to the CANopen standard DS 301.

2.5 Guide to Setup

NOTICE

Only professional personnel with extensive knowledge of control and drive technology are allowed to setup the servo amplifier.

Check assembly / installation	Check that all the safety instructions in the installation instructions for the servo amplifier and this manual have been observed and implemented. Check the setting for the station address and baud rate.
Connect PC, start setup software	Use the setup software DRIVE.EXE to set the parameters for the servo amplifier.
Setup basic functions	Start up the basic functions of the servo amplifier and optimize the current and speed controllers. This section of the setup is described in the online help of the setup software.
Save parameters	When the parameters have been optimized, save them in the servo amplifier.
Start up communication	The altered parameters will only become effective after a software-reset (warm boot). To do this, change to the screen page “Status” and operate the reset button. It is required, that the software protocol described in Chapter 4 is implemented in the master. Adjust the transmission rate of the SERVOSTAR 400/600 to match the master.
Test communication	Recommendation : request the Emergency Object.

WARNING: **Automatic Start!**
Risk of death or serious injury for humans working in the machine. The drive performing unplanned movements during commissioning cannot be ruled out. Make sure that, even if the drive starts to move unintentionally, no danger can result for personnel or machinery. The measures you must take in this regard for your task are based on the risk assessment of the application.

Setup position controller	Setup the position controller, as described in the setup software online help.

2.6 Important configuration parameters for CAN bus operation

The following parameters are important for CAN operation:

1. CBAUD (Object 3515_h Sub-index 01_h): transmission rate for the CAN bus

2. ADDR (Object 3505_h Sub-index 01_h): The *ADDR* command defines the fieldbus address of the amplifier. After making a change to the address, all the parameters must be saved in the EEPROM, and the amplifier must be switched off and on again.

3. AENA (Object 3506_h Sub-index 01_h): This can be used to define the state of the software enable when the amplifier is switched on. The software enable provides an external control with the facility of enabling or disabling the output stage through software control. On amplifiers that function with an analog setpoint (OPMODE=1,3), the software enable is set automatically when the amplifier is switched on, so that these amplifiers are immediately ready to operate (provided that the hardware enable is present). For all other amplifiers, the software enable is set to the value of *AENA* at switch-on. The variable *AENA* also has a function for the reset of the amplifier after a fault (via digital input 1 or through the ASCII command *CLRFAULT*). For errors that can be reset through software, after the error/fault has been cleared, the software enable is set to the state of *AENA*. In this way, the response of the amplifier for a software reset is analogous to the switch-on behavior.

4. DRVCNFG (Object 3672_h Sub-index 01_h): The configuration variable *DRVCNFG* can be used to activate various additional CANopen functions in the amplifier.

Bit0	=1	CANopen switch-on telegram, 0 bytes long
	=0	CANopen switch-on telegram, 8 bytes long
Bit1	=1	enable and disable affect the CANopen status machine. The CANopen status machine follows the internal status of the servo amplifier. If this status changes (e.g. hardware disable) the CANopen status machine is automatically updated (with a corresponding Emergency Message).
	=0	the CANopen status machine is not affected.
Bit2	=1	Object length is checked, an Emergency Object is generated if the Object length is incorrect
	=0	the Object length is not checked.
Bit3	=1	all communication- and mapping parameters can be saved with Object 1010_h, sub-index 02_h ($\Rightarrow$ 4.2.1.16) This saved mapping is selected at switch-on of the amplifier.
	=0	At switch-on of SERVOSTAR 400/600, the default mapping configuration is set.

5. MDRV (Object 3639_h, sub-index 01_h): with this variable, the multidrive mode for the setup-software can be engaged (MDRV = 1). In this case only the first three RPDOs and TPDOs are available. If all four RPDOs and TPDOS are required, MDRV must be set to 0.

This pape has been deliberately left blank.

3 CANopen communication profile

This chapter describes the basic services and Communication Objects of the CANopen communication profile DS 301, which are used in the SERVOSTAR 400/600.

NOTE It is assumed that the basic operating functions of the communication profile are known, and available as reference documentation.

3.1 General description of CAN

The transmission method that is used here is defined in ISO 11898 (Controller Area Network CAN for high-speed communication).
The Layer-1/2 protocol (Physical Layer/Data Link Layer) that is implemented in all CAN modules provides, amongst other things, the requirements for data.

Data transport or data request is made by means of a data telegram (Data Frame) with up to 8 bytes of user data, or by a data request telegram (Remote Frame).

Communication Objects are labeled by an 11-bit Identifier (ID) that also determines the priority of Objects.

A Layer-7 protocol (Application Layer) was developed, to decouple the application from the communication. The service elements that are provided by the Application Layer make it possible to implement an application that is spread across the network. These service elements are described in the CAN Application Layer (CAL) for Industrial Applications.
The communication profile CANopen and the drive profile are mounted on the CAL.

The basic structure of a Communication Object is shown in the following diagram:

S O M	COB-ID	R T R	CTRL	Data Segment	CRC	A C K	EOM

SOM	Start of message
COB-ID	COB Identifier (11-bit)
RTR	Remote Transmission Request
CTRL	Control Field (e.g. Data Length Code)
Data Segment	0 ... 8 byte (Data-COB)
	0 byte (Remote-COB)
CRC	Cyclic Redundancy Check
ACK	Acknowledge slot
EOM	End of message

3.2 Construction of the COB Identifier

The following diagram shows the layout of the COB Identifier (COB-ID). The Function Code defines the interpretation and priority of the particular Object.

10	9	8	7	6	5	4	3	2	1	0
Code				Module-ID						

Bit 0 .. 6 Module ID (station number, range 1 ... 63; is set up in the operator software or the servo amplifier, ⇒ 2.2)

Bit 7... 10 Function Code (number of the Communication Object that is defined in the server)

NOTE

If an invalid station number (=0 or >63) is set, then the module will be set internally to 1. The ASCII Object MDRV can be used to expand the address range from 63 through to127.

The following tables show the default values for the COB Identifier after switching on the servo amplifier. The objects, which are provided with an index (Communication Parameters at Index), can have a new ID assigned after the initialization phase. The indices in brackets are optional.

Predefined broadcast Objects (send to all nodes):

Object	Function code (binary)	Resulting COB-IDs		Communication parameters at index
		Dec.	Hex.	
NMT	0000	0	0_h	—
SYNC	0001	128	80_h	(1005_h)
TIME	0010	256	100_h	—

Predefined Peer-to-Peer Objects (node sends to node):

Object	Function code (binary)	Resulting COB-IDs		Communication parameters at index	Piority
		Dec.	Hex.		
EMERGENCY	0001	129...255	81_h...FF_h	—	high
TPDO 1	0011	385...511	181_h...$1FF_h$	1800_h	
RPDO 1	0100	513...639	201_h...$27F_h$	1400_h	
TPDO 2	0101	641...767	281_h...$2FF_h$	1801_h	
RPDO 2	0110	769...895	301_h...$37F_h$	1401_h	
TPDO 3	0110	897...1023	381_h...$3FF_h$	1802_h	
RPDO 3	1000	1025...1151	401_h...$47F_h$	1402_h	
TPDO 4	1001	1153...1279	481_h...$4FF_h$	1803_h	
RPDO 4	1010	1281...1407	501_h...$57F_h$	1403_h	
SDO (tx*)	1011	1409...1535	581_h...$5FF_h$		
SDO (rx*)	1100	1537...1663	601_h...$67F_h$		
Nodeguard	1110	1793...1919	701_h...$77F_h$	($100E_h$)	low

* tx = direction of transmission: SERVOSTAR ⇒ Master
rx = direction of transmission: Master ⇒ SERVOSTAR

3.3 Definition of the used data types

This chapter defines the data types that are used. Each data type can be described by bit-sequences. These bit-sequences are grouped into “Octets” (bytes). The so-called **“Little – Endian” format** (a.k.a. Intel format) is used for numerical data types (see also: DS301 Application Layer “General Description of Data Types and Encoding Rules”).

3.3.1 Basic data types

3.3.1.1 Unsigned Integer

Data in the basic data type UNSIGNEDn define exclusively positive integers.
The value range is from 0 ... 2^n-1. The bit sequence b = b_0 ... b_{n-1} defines the value

$$UNSIGNEDn(b) = b_{n-1}\,2^{n-1} + ... + b_1\,2^1 + b_0\,2^0$$

Example: the value 266 = $10A_h$ is transmitted in the data type UNSIGNED16, in the form of two *octets* (1^{st} octet = $0A_h$, 2^{nd} octet = 01_h).

Transmission syntax for the data type UNSIGNEDn

Octet number	1.	2.	3.	4.	5.	6.	7.	8.
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
UNSIGNED48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
UNSIGNED56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

3.3.1.2 Signed Integer

Data in the basic data type INTEGERn define both positive and negative integers.
The value range is from -2^{n-1}-1 ... 2^{n-1}-1. The bit sequence b = $b_0..b_{n-1}$ defines the value

$$INTEGERn(b) = b_{n-2}\,2^{n-2} + ... + b_1\,2^1 + b_0\,2^0 \text{ with } b_{n-1} = 0$$

Negative numbers are represented as 2's complement, which means:

$$INTEGERn(b) = -\,INTEGERn(b) - 1 \text{ with } b_{n-1} = 1$$

Example: the value -266 = $FEF6_h$ is transmitted in the data type INTEGER16, in the form of two octets (1^{st} octet = $F6_h$, 2^{nd} octet = FE_h).

Transmission syntax for the data type INTEGERn

Octet number	1.	2.	3.	4.	5.	6.	7.	8.
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

3.3.2 Mixed data types

Mixed data types combine basic data types (INTEGERn, UNSIGNEDn, REAL). Two types of mixed data are distinguished:

- STRUCT
  This data type is composed of elements with different data types.
- ARRAY
  This data type is composed of elements of the same data type.

3.3.3 Extended data types

Extended data types are derived from basic data types and mixed data types. The types of extended data that are supported are defined below.

3.3.3.1 Octet String

The data type *OCTET_STRING* is defined with the data type *ARRAY*. *Length* is the length of the octet string.

ARRAY[length] OF UNSIGNED8	OCTET_STRINGlength

3.3.3.2 Visible String

The data type *VISIBLE_STRING* can be defined with the data type *UNSIGNED8* or the data type *ARRAY*. Permissible values are 00_h and the range from 20_h to $7E_h$. The data are interpreted as 7 bit ASCII code (as per ISO 646-1973(E)). *Length* is the length of the visible string.

UNSIGNED8	VISIBLE_CHAR
ARRAY[length] OF VISIBLE_CHAR	VISIBLE_STRINGlength

3.4 Communication Objects

Communication Objects are described with the help of service elements and protocols. Two basic types of service elements are distinguished:

- Unconfirmed services
- Confirmed services

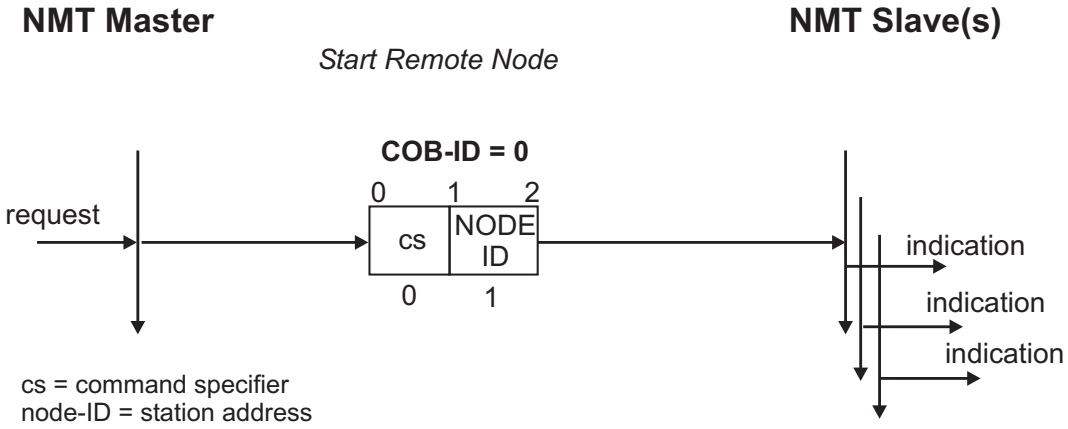
All services require faultless operation of the *Data Link* and *Physical Layer*.

SERVOSTAR 400/600 supports Communication Objects that are described in detail in the following sections:

- Network Management Objects (NMT)
- Synchronization Object (SYNC)
- Time Stamp Object (TIME)
- Emergency Object (EMCY)
- Process Data Object (PDO)
- Service Data Object (SDO)
- Nodeguard

3.4.1 Network Management Objects (NMT)

The NMT telegram looks like this:



The drive supports the following network management functions:

cs = 129, reset node: causes a cold-start of the drive.
This deletes all parameters saved in the RAM and loads the values stored in the EEPROM or the default values.

cs = 1, start remote node: starts the CAN node.
I.e. the PDOs of the drive are enabled for operation.
From this moment, transmit-PDOs will be transmitted under event-control, and cyclical process data operation can commence.

cs = 2, stop remote node: stops the CAN node, I.e. the drive no longer responds to any received PDOs or transmits any PDOs.

3.4.2 Synchronization Object (SYNC)

This is a periodic *Broadcast Object* and provides the basic clock for the bus. SYNC has a high priority, to ensure constant time intervals. The usage of this protocol is explained in the application section of this manual.

3.4.3 Time-Stamp Object (TIME)

This communication Object is not supported by SERVOSTAR.

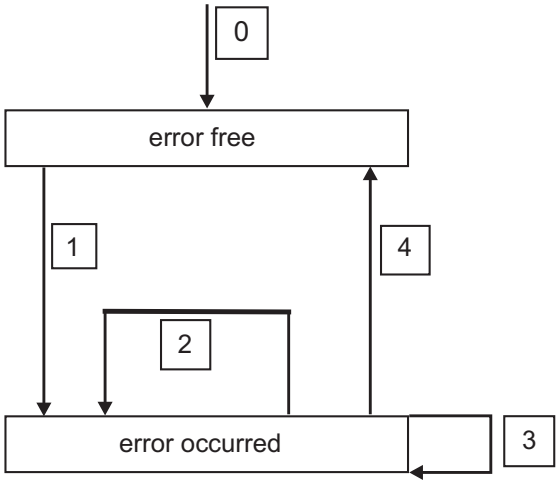
3.4.4 Emergency Object (EMCY)

EMCY is event-triggered and generated by an internal fault/error situation. This Object is transmitted afresh for every error. Since the error codes are device-dependent, they are described in the Chapter *CANopen Drive Profile* (⇒ chapter 4).

3.4.4.1 Application of the Emergency Object

The reaction in the event of an error or fault depends on the error class and is therefore variable. For this reason, the reaction is described with the aid of an error status machine. The error conditions *error- free* and *error occurred* are distinguished. The following transitions are defined:

0. After initialization, the error-free status is taken up if no errors are detected.
   No error signal is generated in this condition.
1. The SERVOSTAR detects an internal error and indicates this in the first three bytes of the emergency telegram (*error code* in Bytes 0,1 and *error register* in Byte 2).
   Since the SERVOSTAR can distinguish between different types of error, Byte 3 of the manufacturer-specific error field is used to indicate the error category.
2. One error has been reset, but not all. The *EMCY* telegram contains error code 0000_h and the error register indicates the remaining errors that are present.
   The manufacture-specific area is set to zero.
3. A new error has occurred. The SERVOSTAR remains in the *error status* and transmits an *EMCY* Object with the corresponding error code. The new error code is entered in Bytes 0 and 1.
4. All errors have been reset. The *EMCY* telegram contains the error code 0000_h, the error register does not indicate any other errors.
   The manufacture-specific area is set to zero.



3.4.4.2 Composition of the Emergency Object

The Emergency Object is composed of 8 bytes, divided as follows:

Byte	0	1	2	3	4	5	6	7
Content	Emergency error code (⇒ 4.1)		Error register (Object 1001_h)	Category	Reserved			

If an Emergency Object is generated, the error condition is then signaled to the status machine (error free / error occurred) by the generation of a second Emergency Object. Only the first four bytes are relevant in this case (*Emergency Error code , Error register, Category*). Byte 0/1 contains the *Error Reset code* (0000_h) and Byte 2 indicates if a possible further error is present. If the error register contains 00_h, the error status is *error-free*.
Byte 3 contains the category. The interpretations of the error numbers (*error code*) and the error categories are described in the section *Emergency Messages* (⇒ 4.1). The error register is defined through Object 1001_h *Error register*.

3.4.5 Service Data Objects (SDO)

SDOs are used to implement access to the Object Dictionary. The SDOs are required for parametrerization and for status polling. Access to an individual Object is made with a multiplexer via the Index and Sub-index of the Object Dictionary. The following communication protocols are supported by SERVOSTAR 400/600:

- Initiate SDO Download Protocol
- Download SDO Segment Protocol
- Initiate SDO Upload Protocol
- Upload SDO Segment Protocol
- Abort SDO Transfer Protocol

The definitions of the individual communication services and protocols can be found in DS301. Examples of the usage of SDOs can be found in the application section of this manual.

NOTE Since an SDO is a confirmed service, the system must always wait for the SDO response telegram before it is allowed to transmit a new telegram.

3.4.5.1 Composition of the Service Data Object

An SDO is consists of the following components:

Byte	1	2	3	4	5	6	7	8
Content	rw	Index		Sub-index	Data			

1. The control byte (Byte 1):
   The control byte determines whether the SDO has write or read access to the entry in the Object Dictionary. A description of the complete Object Dictionary for SERVOSTAR 400/600 can be found in Section 6.3.
   Data exchange with the SERVOSTAR 400/600 is governed by the *CMS multiplexed domain protocols* standard, as described in the CAN standard DS 202.
   To read data, the control byte must be written in the manner shown below:

Bit	7	6	5	4	3	2	1	0
Content	ccs*=2			0	0	0	0	0

* ccs ⇒ client command specifier (ccs = 2 ⇒ initiate download request)

So a value of 0100 0000 (binary) or 40h has to be transmitted in the control byte.

The servo amplifier sends back a corresponding response byte:

Bit	7	6	5	4	3	2	1	0
Content	scs*=2			X	n		e	s

* scs ⇒ server command specifier (scs = 2 ⇒ initiate upload response)
n ⇒ only valid for e = s = 1
if this is so, n contains the number of bytes that do not contain data
X ⇒ free data

If reading is successfull, the response byte always has set the bits 0 and 1 (e = s = 1).
Encoded byte length in the SDO response:

0x43 - 4 bytes
0x47 - 3 bytes
0x4B - 2 bytes
0x4F - 1 byte.

If an error occurs, scs is set to 4, the response byte is 0x80 and the error information is in the four byte data field. The decoding of the error can be found in Section 3.4.5.6.

To write data, the control byte must be written in the manner shown below:

Client	*Initiate Domain Download*														Server
	Byte 1								**2**	**3**	**4**	**5**	**6**	**7**	**8**	
request	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**							indication
⇒	ccs=1			X	n		e	s	m			d			⇒

⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒

⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐

⇐	**Byte 1**								**2**	**3**	**4**	**5**	**6**	**7**	**8**	⇐
confirm	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**							response
	scs=3			X					m			reserved			

2. Index (Bytes 2 and 3):
   The Index is the main entry in the Object Dictionary, and divides the parameters into groups. (Example: Index 2022_h is position data for the *Position* mode).
   As for all CAN data, the Index is stored with the bytes in reverse order.
   E.g. Index 6040_h means Byte 2 = 40_h, Byte 3 = 60_h)

3. Sub-index (Byte 4):
   The Sub-index divides the parameters within a group of parameters.

4. Data field (Bytes 5 ... 8):
   These components are used for the exchange of user data. In read-request telegrams to the SERVOSTAR 400/600 they are set to 0. They have no content in a write confirmation from the SERVOSTAR 400/600 if the transfer was successful, but if the write operation was faulty they contain an error code (⇒ 3.4.5.6).

3.4.5.2 Initiate SDO Download Protocol

The *Initiate SDO Download* protocol is used for write access to Objects with up to 4 bytes of user data (*expedited transfer*) or to initiate a segment transfer (*normal transfer*).

Expedited transfer is also used for Objects that only have the character of a command (e.g. ASCII: *SAVE)* and thus do not require any further user data.

3.4.5.3 Download SDO Segment Protocol

The *Download SDO Segment* protocol is used for write access to Objects with more than 4 bytes of user data (*normal transfer*). This service is not supported by SERVOSTAR 400/600 at present, since there are no Objects that make use of more than 4 bytes of user data.

3.4.5.4 Initiate SDO Upload Protocol

The *SDO Upload* protocol is used for read access to Objects with up to 4 bytes of user data (*expedited transfer*) or to initiate a segment transfer (*normal transfer*).

3.4.5.5 Upload SDO Segment Protocol

The *Upload SDO Segment* protocol is used for read access to Objects with more than 4 bytes of user data (*normal transfer*). This service is not supported by SERVOSTAR 400/600 at present, since there are no Objects that make use of more than 4 bytes of user data.

3.4.5.6 Abort SDO Protocol

The Abort SDO protocol breaks off SDO transmission, and indicates the error that caused the break in transmission through an abort code (error code). The error code is in the format of an UNSIGNED32 value. The following table shows possible reasons for an abort SDO.

Abort Code	Description
0503 0000$_h$	*Toggle bit* was not toggled
0504 0000$_h$	*Timeout* for SDO protocol
0504 0001$_h$	Client/server command - invalid or unknown Identifier
0504 0002$_h$	Unrecognized block size (block mode only)
0504 0003$_h$	Unrecognized block number (block mode only)
0504 0004$_h$	CRC error (block mode only)
0504 0005$_h$	Out of memory
0601 0000$_h$	Access to this Object is not supported
0601 0001$_h$	Attempted read access to a write-only Object
0601 0002$_h$	Attempted write access to a read-only Object
0602 0000$_h$	Object does not exist in Object Dictionary
0604 0041$_h$	Object cannot be mapped to a PDO
0604 0042$_h$	Size and number of mapped Objects exceed permissible PDO length
0604 0043$_h$	General parameter incompatibility
0604 0047$_h$	General device incompatibility
0606 0000$_h$	Access infringement caused by hardware error
0607 0010$_h$	Data type incompatible, length of service parameter is incompatible
0607 0012$_h$	Data type incompatible, length of service parameter is too long
0607 0013$_h$	Data type incompatible, length of service parameter is too short
0609 0011$_h$	Sub-index does not exist
0609 0030$_h$	Outside value range for the parameter (only for write access)
0609 0031$_h$	Parameter value too high
0609 0032$_h$	Parameter value too low
0609 0036$_h$	Maximum value is lower than minimum value
0800 0000$_h$	General error/fault
0800 0020$_h$	Data cannot be transmitted or saved
0800 0021$_h$	Data cannot be transmitted or saved because device is under local control
0800 0022$_h$	Data cannot be transmitted or saved because of device status
0800 0023$_h$	Dynamic generation of the Object Dictionary not possible or already available (e.g Object Dictionary is created from a file, and an error occurs because of a defect in the file)

Abort Codes not listed above are reserved.

3.4.6 Process Data Object (PDO)

PDOs are used for real-time data communication. PDOs can, for instance, be used to set up controllers similar to analog drives. Instead of +/-10VDC setpoints and ROD feedback, digital speed setpoints and position feedback are attained via PDOs in this case.
Transmission is carried out unconfirmed without a protocol “overhead”. This communication Object uses the unconfirmed communication service.
PDOs are defined via the Object Dictionary for the SERVOSTAR 400/600, whereby pre-defined PDOs can be selected (mapping of pre-defined PDOs) or composed by the user (mapping of variables). Mapping is made during the configuration phase, with the help of SDOs. The lengths and mapping numbers for the PDOs are defined by the drive profile DS 402.

The definition of the PDO service and protocol can be found in DS301. Examples of the usage of PDOs can be found in the application section of this documentation.
Basically, two types of PDOs can be distinguished, depending on the direction of transmission:

- Transmit-PDOs (TPDOs) (SERVOSTAR ⇒ Master)
  The TPDOs transmit data from SERVOSTAR to control system
  (e.g actual value Objects, amplifier status).
- Receive-PDOs (RPDOs) (Master ⇒ SERVOSTAR)
  The RPDOs receive data from control system to SERVOSTAR
  (e.g setpoints).

SERVOSTAR 400/600 supports two independent PDO channels for each direction of transmission. The channels are labeled by the channel numbers 1 to 4.
There are three parameters each for the configuration of each of the four possible PDOs, and they can be set up through the corresponding SDOs:

1. Selection parameters, to select the two PDOs in each direction to be used for process data operation from a number of possible PDOs (Objects 2600_h to 2603_h, $2A00_h$ to $2A03_h$).
2. Mapping parameter, to determine which data are available (mapped) in the selected PDO, and for entering the mapping of the PDO for the freely configured PDOs (Nos. 37 to 40).
3. Communication parameters, that define whether the PDOs operate in synchronized mode, or event-driven (Objects 1400_h to 1403_h, 1800_h to 1803_h).

Furthermore, individual bits of the TPDOs can be masked, so that an automatic generation of the transmit-trigger can be controlled by individual bit event in TPDOs.

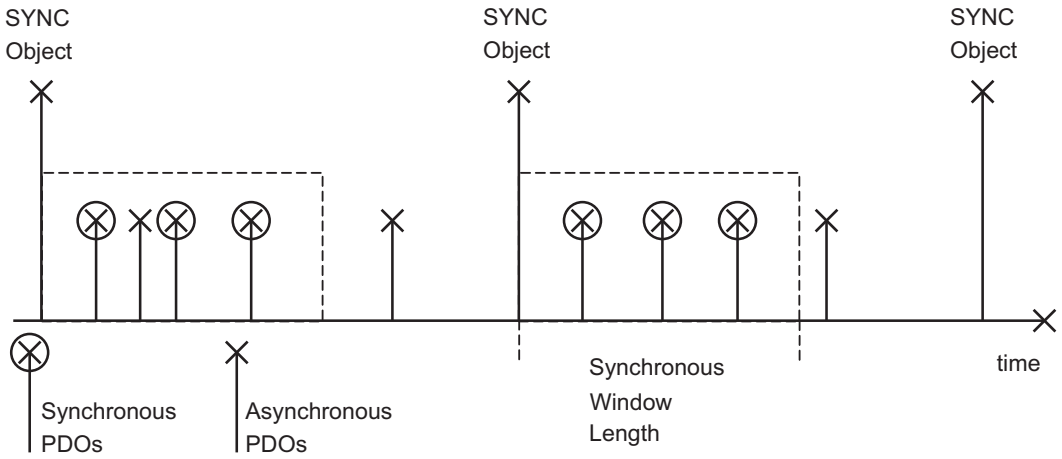
3.4.6.1 Transmission modes

The following PDO transmission modes are distinguished:

- Synchronous transmission
- Asynchronous transmission

The pre-defined SYNC Object is transmitted periodically (bus clock), to synchronize the drives. Synchronous PDOs are transmitted within a pre-defined time window immediately following the SYNC Object.

The transmission modes are set up with the aid of the PDO communication parameters.



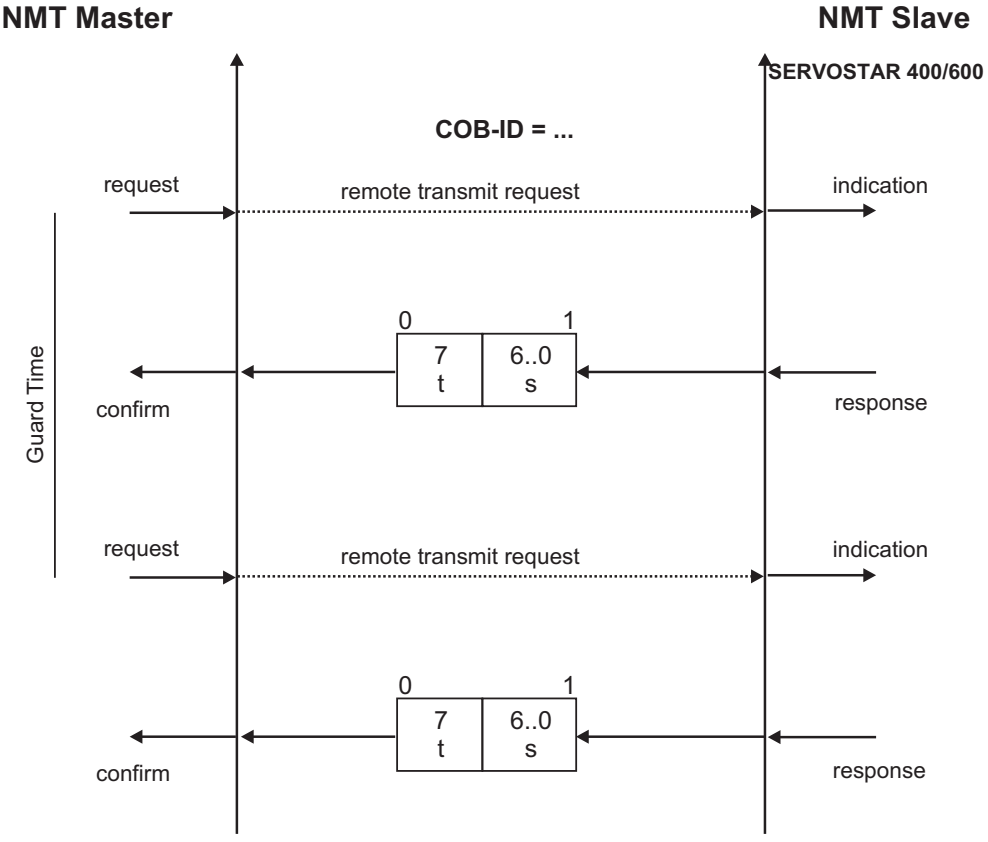
3.4.6.2 Trigger modes

Three different trigger modes are distinguished:

- **Event driven**
  The transmission of the telegrams is triggered by an Object-specific event. There is also the option of masking individual bits (regardless of the Object) so that the automatic generation of telegrams is restricted, and thus reduce the bus loading (⇒ 4.3.2.2.4 f).
- **Time driven**
  If event driven signals put a high strain on the bus, you can determine the period of time after which a PDO can be transmitted again via the *inhibit time* (Communication parameter, sub-index 03_h)

3.4.7 Nodeguard

The Node Guarding protocol is a functional monitoring for the drive. It requires that the drive is accessed at regular intervals by the CANopen master.
The maximum time interval that is permitted between two Nodeguard telegrams is given by the product of the *Guard Time* (Object $100C_h$, ⇒ 4.2.1.12) and the *Life Time Factor* (Object $100D_h$, ⇒ 4.2.1.13). If one of these two values is 0, then the response monitoring is de-activated.
Node guarding is only activated when the output stage is enabled. If the drive is not accessed within the time defined by Objects $100C_h$ und $100D_h$, then Warning N04 (response monitoring) appears on the drive, the drive is braked to a stop with the Quickstop ramp, and any other movement is prevented.
The time sequence for node guarding is as shown below:



t = toggle Bit, changes its status with every slave telegram
s = status of the NMT slave status machine

Node guarding is carried out by the Master through RTR telegrams with the COB-ID, which can be set for the slave by Object $100E_h$ (⇒ 4.2.1.14).
The default value is 700_h + slave node address.

This page has been deliberately left blank.

4 CANopen Drive Profile

4.1 Emergency Messages

Emergency messages are triggered by internal equipment errors. They have a high ID-priority, to ensure quick access to the bus. An *Emergency message* contains an error field with pre-defined error/fault numbers (2 bytes), an error register (1byte), the error category (1 byte) and additional information (⇒ chapter 3). The higher-value byte of the error number describes the error category, and the lower-value byte provides the error number in this category.

Error numbers from $xx00_h$ to $xx7F_h$ are defined in the communication or drive profile. Error numbers from $xx80_h$ to $xxFF_h$ have manufacturer-specific definitions. The error category can be used to classify the significance of any errors that occur. The following error categories are defined:

1: Errors that can only be cleared by a reset (*COLDSTART* command, or Bit 7 in the control word ⇒ 4.4.2.1). These errors are also indicated by blinking of the LED display in the front panel. (Fxx, xx = error number)
2: Errors that can be cleared by Bit 11 in the control word (⇒ 4.4.2.1).
3: Error messages that may appear when a PDO is processed.
4: Faults, that **cannot** be cleared by the user.
5: Operating errors/warnings.

The following table describes the various *Error Codes*:

Error Code	Category	Description
0000_h	—	Error reset or no error (mandatory)
1000_h	—	Generic error (mandatory)
1080_h	5	No BTB/RTO (status *not ready for operation*)
2330_h	2	Earth short (F22)
3100_h	1	No mains/line-BTB (F16)
3110_h	1	Overvoltage in DC-bus/DC-link (F02)
3120_h	1	Undervoltage in DC-bus/DC-link (F05)
3130_h	1	Supply line phase missing (with PMODE = 2) (F19)
4110_h	1	Ambient temperature too high (F13)
4210_h	1	Heat sink temperature too high (F01)
4310_h	1	Motor temperature too high (F06)
5111_h	1	Fault in ±15V auxiliary voltage (F07)
5380_h	1	Fault in A/D converter (F17)
5400_h	1	Fault in output stage (F14)
5420_h	1	Regen (ballast, chopper) (F18)
5441_h	1	Operating error for AS-option (F27)
5530_h	1	Serial EEPROM (F09)
5581_h	1	Flash EEPROM (F10)
6010_h	4	Watchdog (software reset, F32)
6181_h	4	BCC error (table)
6182_h	4	BCC error (system macro)
6183_h	4	BCC error (serial EEPROM)
6184_h	4	FPGA error
6185_h	4	Fault/error (table)
6281_h	4	User software BCC (macro, F32)
6282_h	4	Faulty user software (macro, F32)
6320_h	3	Parameter error
7111_h	1	Braking error/fault (F11)
7122_h	1	Commutation error (F25)
7181_h	5	Could not enable SERVOSTAR
7182_h	5	Command only possible in *disabled* status
7303_h	1	Feedback device error (F04)
8053_h	1	Handling error (F21)
8181_h	2	Response monitoring activated
8182_h	1	CAN *bus off* (F23)
8281_h	5	Status machine not in *operation enable* condition
8282_h	5	wrong mode setting
8331_h	1	I^2t (torque fault, F15)
8480_h	1	Overspeed (F08)
8611_h	2	Lag/following error
8681_h	5	Invalid motion task number
8682_h	2	External trajectory error (F28) (only with Sercos)
$FF01_h$	4	Serious exception error (F32)
$FF02_h$	3	Error in PDO elements
$FF03_h$	5	Operating mode
FF04	1	Slot error (F20)
FF06	2	Warning display as error (F24)
FF07	2	Homing error (drove onto HW limit switch) (F26)
FF08	2	Sercos error (F29)
FF10	1	Motor phase error (F12)
FF11	1	Sercos

4.2 General definitions

This chapter describes Objects with a general validity (e.g. Object 1000_h *Device Type*). The next section explains the free configuration of Process Data Objects ("free mapping").

4.2.1 General Objects

4.2.1.1 Object 1000h: Device Type (DS301)

This Object describes the device type (servo drive) and device functionality (DS402 drive profile). It is laid out as follows:

MSB																LSB
additional information																device profile number
mode bits								type								402_d
31							24	23							16	15 0

Number of the device profile: 402_d (drive profile)
Type: 02 (servo drive)
Mode bits: 0 (manufacturer-specific)

Index	1000_h
Name	device type
Object code	VAR
Data type	UNSIGNED32
Category	M

Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	no

4.2.1.2 Object 1001h: Error register (DS301)

If an error bit is set in the manufacturer independent error register, then more detailed information is made available in Object 1003_h. This Object is part of the Error Object (Emergency Message).

Index	1001_h
Name	Error register
Object code	VAR
Data type	UNSIGNED8
Category	M

Access	ro
PDO mapping	not possible
Value range	UNSIGNED8
Default value	no

The following describes the bit assignment for the error register. A bit that is set indicates an error/fault event.

Bit	Description	Bit	Description
0	generic error	4	communication error
1	current	5	device profile specific
2	voltage	6	reserved
3	temperature	7	manufacturer-specific

4.2.1.3 Object 1002h: Manufacturer Status Register (DS301)

This Object contains the manufacturer-specific status register, which also contains SERVOSTAR warnings (see also ASCII Object DRVSTAT).

Index	1002_h
Name	Manufacturer Status Register
Object code	VAR
Data type	UNSIGNED8
Category	M

Access	ro
PDO mapping	not possible
Value range	UNSIGNED8
Default value	no

The following table shows the bit assignment for the status register:

Bit	Description
0	Warning 1: I²t-threshold exceeded
1	Warning 2: full regen power reached
2	Warning 3: lag/following error
3	Warning 4: response monitoring activated
4	Warning 5: power supply phase missing
5	Warning 6: software limit-switch 1 has been activated
6	Warning 7: software limit-switch 2 has been activated
7	Warning 8: faulty motion task started
8	Warning 9: no reference point set at start of motion task
9	Warning 10: PSTOP activated
10	Warning 11: NSTOP activated
11	Warning 12: motor default values were loaded
12	Warning 13: expansion board not functioning correctly
13	Warning 14: motor phase
14	Warning 15: erroneous VCT entry selected
15	Warning 16: reserve
16	motion task active
17	reference point set
18	actual position = home position
19	In Position
20	position latch made (positive edge)
21	reached Position 0
22	reached Position 1
23	reached Position 2
24	reached Position 3
25	reached Position 4
26	finished initialization
27	reached Position 5
28	speed = 0
29	safety relay has been activated
30	output stage enabled
31	error present

4.2.1.4 Object 1003h: Pre-Defined Error Field (DS301)

This Object contains the last error event that was reported by an error telegram. Only the most recent error is displayed.

1. The value in Sub-index 00_h shows the number of recorded errors.
2. The most recent error is shown in Sub-index 01_h.
3. If the value 0 is written to Sub-index 02_h, the error is deleted from the error list (caution: error will not be acknowledged!).
4. The error number has the data type UNSIGNED32 and is composed of a 16-bit error number and a manufacturer-specific information field.
   The manufacturer-specific information field is not used by SERVOSTAR 400/600.
   The possible error numbers are described in the chapter “Error Message” (⇒ 4.1).
   The data field is composed as follows:

MSB	LSB
Information field	Error number

Index	1003_h
Name	Error Field
Object code	ARRAY
Data type	UNSIGNED32
Category	O

Sub-indexes:

Sub-index	00_h
Description	Number of errors
Category	M
Access	rw
PDO mapping	not possible
Value range	0 ... 254
Default value	no

Sub-index	01_h
Description	Standard error field (⇒ 4.1)
Category	M
Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	no

4.2.1.5 Object 1004h: Number of supported PDOs (DS301)

This Object is only supported to provide compatibility with older versions of the CANopen Standards DS 301.

The structure of the parameter is as follows:

Sub-index	MSB	LSB
0	no. of supported RPDOs	no. of supported TPDOs
1	no. of supported synchronous RPDOs	no. of supported synchronous TPDOs
2	no. of supported asynchronous RPDOs	no. of supported asynchronous TPDOs

Index	1004_h
Name	no. of supported PDOs
Object code	ARRAY
Data type	UNSIGNED32
Category	O

Sub-indexes:

Sub-index	00_h
Description	no. of supported PDOs
Category	O
Access	ro
PDO mapping	not possible
Value range	00_h ... $1FF01FF_h$
Default value	00020002_h

Sub-index	01_h
Description	no. of synchronous PDOs
Category	O
Access	ro
PDO mapping	not possible
Value range	00_h ... $1FF01FF_h$
Default value	00_h

Sub-index	02_h
Description	no. of asynchronous PDOs
Category	O
Access	ro
PDO mapping	not possible
Value range	00_h ... $1FF01FF_h$
Default value	00020002_h

This only shows the assignment when the SERVOSTAR 400/600 has started up. Afterwards, the PDOs can be configured through the communication parameters 1400_h to 1403_h, 1800_h to 1803_h.

4.2.1.6 Object 1005h: COB-ID of the SYNC Message (DS301)

This Object can be used to change the COB-ID for the SYNC message.

Index	1005_h
Name	COB-ID for the SYNC message
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	80000080_h

The structure of the parameter is as follows:

MSB				LSB
X	0/1	0	00_h	11-bit identifier
31	30	29	28 ... 11	10 ... 0

Only Bits 0 ... 10, 29 and 30 have any significance:

Bit	Value	Meaning
31	X	—
30	0	Device not generating SYNC message
	1	Device generating SYNC message
29	0	11 Bit ID
	1	29 Bit ID
28 ... 11	X	Bit 11 ... 28 of 29-bit SYNC COB-ID
10 ... 0	X	Bit 0 ... 10 of SYNC COB-ID

4.2.1.7 Object 1006h: Communication Cycle Period (DS301)

This Object can be used to define the period (in µs) for the transmission of the SYNC telegram.

Index	1006_h
Name	Period of the communication cycle
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	00_h

4.2.1.8 Object 1007h: Synchronous window length (DS301)

This Object defines the length of the time window (in µs) for synchronous PDOs.

Index	1007_h
Name	Time window for synchronous messages
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	00_h

4.2.1.9 Object 1008h: Manufacturer Device Name (DS301)

The device name consists of four ASCII characters in the form *SDxx*, whereby xx stands for the current in the output stage (e.g. SD06).

Index	1008_h
Name	Manufacturer Device Name
Object code	VAR
Data type	Visible String
Category	Optional

Access	const
PDO mapping	not possible
Value range	no
Default value	no

4.2.1.10 Object 100Ah: Manufacturer Software Version (DS301)

The interface software version consists of four ASCII characters (e.g. 3.00).

Index	$100A_h$
Name	Manufacturer Software Version
Object code	VAR
Data type	Visible String
Category	Optional

Access	const
PDO mapping	not possible
Value range	no
Default value	no

4.2.1.11 Object 100Bh: Node-ID (DS301)

The station address of the SERVOSTAR 400/600 can be read out through the Object *Node-ID*. The station address can be altered through the ASCII-Object *ADDR*. The address range (value range) that has to be set depends on the ASCII-Object *MDRV*, and the station address can be set within the range 1 ... 63 or 1 ... 127.

NOTE

If the expanded address range is used, then the setup software can no longer be networked over the CAN-bus.

Index	$100B_h$
Name	Node-ID
Object code	VAR
Data type	UNSIGNED32
Category	Optional

Access	ro
PDO mapping	not possible
Value range	1 ... 127 (1 ... 63)
Default value	1

4.2.1.12 Object 100Ch: Guard Time (DS301)

The arithmetical product of the Objects $100C_h$ *Guard Time* and $100D_h$ *Lifetime Factor* is the response monitoring time. The Guard Time is given in milliseconds. The response monitoring is activated with the first *Nodeguard* Object (⇒ 3.4.7). If the value of the Object *Guard Time* is set to zero, then the response monitoring is inactive.

Index	$100C_h$
Name	Guard Time
Object code	VAR
Data type	UNSIGNED16
Category	Value description:

Access	ro
PDO mapping	not possible
Value range	UNSIGNED16
Default value	0

4.2.1.13 Object 100Dh: Lifetime Factor (DS301)

The arithmetical product of the Objects $100C_h$ *Guard Time* and $100D_h$ *Lifetime Factor* is the response monitoring time. The response monitoring is activated with the first *Nodeguard* Object (⇒ 3.4.7).
If the value of the Object *Guard Time* is set to zero, then the response monitoring is inactive.

Index	$100D_h$
Name	Lifetime Factor
Object code	VAR
Data type	UNSIGNED8
Category	Mandatory

Access	rw
PDO mapping	not possible
Value range	0 ... 255
Default value	0

4.2.1.14 Object 100Eh: COB-ID Nodeguarding (DS301)

This Object defines the COB-ID for the Node Guarding protocol.

Index	$100E_h$
Name	Node Guarding identifier
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	700 + node-ID

4.2.1.15 Object 100Fh: Number of the supported Objects (DS301)

This Object describes the number of Objects that are supported by the device.

Index	$100F_h$
Name	Number of supported Objects
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	01_h

4.2.1.16 Object 1010h: Store Parameters (DS301)

This object can be used to save communication- and mapping parameter.

Index	1010_h
Name	store parameters
Object code	ARRAY
Data type	UNSIGNED32
Category	O

Sub-indexes:

Sub-index	0_h
Description	Number of entries
Access	ro
PDO Mapping	not possible
Value range	2
Default value	2

Sub-index	1_h
Description	save all parameters
Access	rw
PDO Mapping	not possible
Value range	UNSIGNED32
Default value	0

this sub-index is reserved

Sub-index	2_h
Description	save communication parameters
Access	rw
PDO Mapping	not possible
Value range	UNSIGNED32
Default value	0

this sub-index must be used to save communication- and mapping parameters. A writing access with the signature *save* must be made, thus the value below has to be written:

MSB			**LSB**
[e]	[v]	[a]	[s]
65_h	76_h	61_h	73_h

Parameters can only be saved if this function is enabled by DRVCNFG (Object 6372, Sub-index 1), Bit 3.

4.2.1.17 Object 1012h: COB-ID for Time-Stamp Message (DS301)

This Object can be used to define the COB-ID for the *Time Stamp* message.

Index	1012_h
Name	COB-ID time stamp message
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	100_h

4.2.1.18 Object 1013h: High Resolution Time-Stamp (in preparation) (DS301)

This Object can be used to read or write a time value.

Index	1013_h
Name	high-resolution time stamp
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	00_h

4.2.1.19 Object 1014h: COB-ID for Emergency Message (DS301)

This Object can be used to define the COB-ID for the *Emergency* message.

Index	1014_h
Name	COB-ID emergency message
Object code	VAR
Data type	UNSIGNED32
Category	O

Access	rw
PDO mapping	not possible
Value range	UNSIGNED32
Default value	80_h + Node - ID

4.2.1.20 Object 1018h: Identity Object (DS301)

The *Identity* Object contains general device information.

Index	1018_h
Name	Identity Object
Object code	RECORD
Data type	Identity

Sub-indexes:

Sub-index	00_h
Description	Number of entries
Access	ro
PDO mapping	not possible
Value range	1 ... 4
Default value	4

Sub-index	01_h
Description	Vendor ID*
Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	$6A_H$

* 00 00 00 $6A_h$

Sub-index	02_h
Description	Product Code*
Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	no

* Rated current of the amplifier

Sub-index	03_h
Description	Revision Number*
Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	no

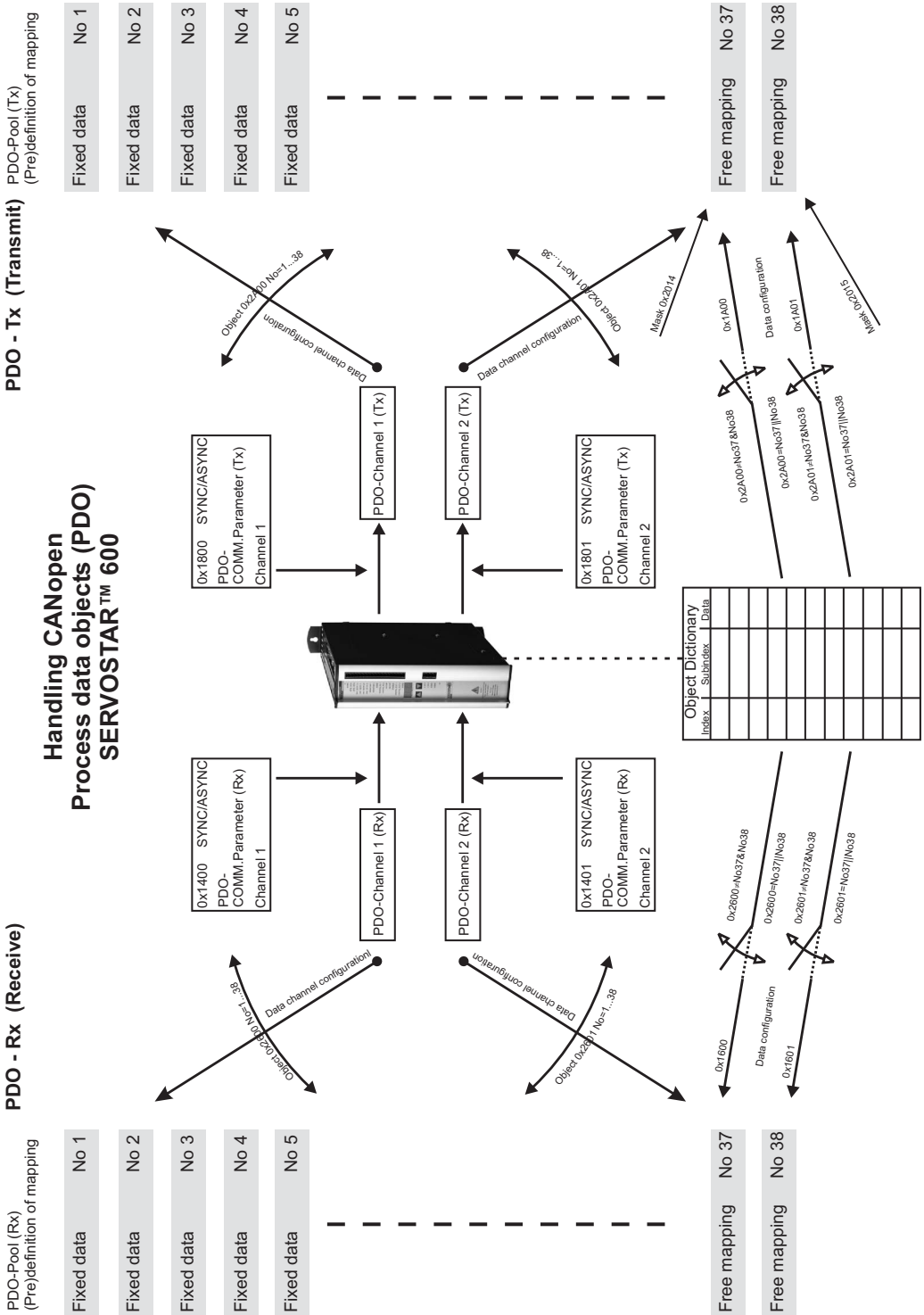
* Bits 16 ... 31 = firmware version, Bits 15 ... 0 = CANopen

Sub-index	04_h
Description	Serial Number
Access	ro
PDO mapping	not possible
Value range	UNSIGNED32
Default value	no

4.3 PDO Mapping

Since SERVOSTAR 400/600 supports more than one operating mode, different PDOs are required for the transmission and received directions, depending on the application.
The first method of handling of freely configurable PDOs was *more-or-less free mapping*. With this type of free mapping, predefined PDOs are selected with the aid of an index from the PDO library. The contents of these PDOs cannot be reconfigured at a later stage. However, *more-or-less free mapping* turned out to be too rigid for the increasing number of extremely variegated applications. As a result, *completely free mapping* was introduced additionally for PDOs, whereby the user can also alter the contents of the PDO (⇒ 3.4.6).

The following diagram “Handling CANopen Process Data Objects” illustrates the use of PDOs.



4.3.1 Receive-PDOs (RPDO)

The Objects 2600_h to 2603_h can be used to select RPDOs. The configuration of the RPDOs is made through the Objects 1400_h to 1403_h (communication parameter) and 1600_h to 1603_h (mapping parameter). These Objects have channel-linked operation, i.e.

- RPDO Channel 1: 2600_h, 1400_h, 1600_h
- RPDO Channel 2: 2601_h, 1401_h, 1601_h
- RPDO Channel 3: 2602_h, 1402_h, 1602_h
- RPDO Channel 4: 2603_h, 1403_h, 1603_h

The Objects 1600_h to 1603_h can only be read, in order to disclose the Object configuration within the PDOs.

NOTE

Exception: If one or more PDOs 37 to 40 are selected, then the Object configuration for the relevant RPDO channel must be made through the Objects 1600_h to 1603_h (write access).

The following PDOs have been defined:

PDO Number	Mapping Object Index	Sub-index	Mapping Object Name
1 (01h)	6040_h	—	control word
2..20 (02h..14h)	—	—	reserved
21 (15h)	3100_h	$01_h..08_h$	ASCII-Object
22 (16h)	6040_h	—	control word
	2060_h	00_h	current or speed setpoint
23..31 ($17_h..1F_h$)	—	—	reserved
32 (20_h)	2060_h	00_h	current or speed setpoint
33 (21_h)	2022_h	04_h	incremental position setpoint
	2022_h	04_h	incremental position setpoint
34 (22_h)	2022_h	01_h	position
	2022_h	02_h	velocity
	2022_h	03_h	type of motion task
35 (23_h)	2022_h	05_h	motion block number (Caution: in the device status *Operation Enable* it will start immediately)
36 (24_h)	—	—	reserved
37..40 ($25_h..28_h$)	$xxxx_h$	xx_h	freely definable PDO
41 (29_h)	2022h	04h	master position for CAN slaves
42..64 ($30_h..40_h$)	—	—	reserved

4.3.1.1 Description of predefined Receive-PDOs

4.3.1.1.1 PDO controlword (1) (DS402)

The PDO control word (Default-PDO) consists of the control word (UNSIGNED16). This PDO can only be used to operate the status machine (⇒ 4.4.1), and is available for use in all modes.
After switch-on, this PDO is mapped to RPDO 1.
The table shows the mapping of the PDO control word:

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	60400010_h	control word

4.3.1.1.2 PDO Receive ASCII Channel (21)

With the help of the ASCII channel (Default-PDO), all the parameters and commands can be transmitted to the SERVOSTAR. Up to 8 ASCII characters can be sent in one PDO. Commands or parameters that require more than 8 characters must be segmented. All commands and parameters are terminated by the ASCII code *CR LF* ($0xD_h$, $0xA_h$). The unused bytes in the PDO are filled with the ASCII code *NUL* ($0x0_h$), because otherwise every surplus character would be interpreted as a new command. After switch-on, this PDO is mapped to RPDO 2.
The table describes the mapping of the PDO *Receive ASCII channel*:

Sub-index	Value	Description
00_h	1_h	number of entries
$01..08_h$	31000208_h	ASCII chars. 1 to 8

This Object **only** supports transmission type 255 (asynchronous).

4.3.1.1.3 PDO Current/Speed Setpoint (22)

The PDO current or speed setpoint is formed from the control word (UNSIGNED16) and the setpoint (SIGNED16). This PDO must only be used in the *Digital speed* or *Digital current* mode. It will be recognized as a speed or current setpoint, depending on the mode that is set (digital current or digital speed). The PDO is executed immediately. A repeated transmission of the PDO with various setpoint values does not require an intermediate halt of the drive.

Current normalization: 3280 = peak current of the controller
1640 = rated current of the controller
e.g.. rated current = 3A, setpoint = 1.0A ⇒ 547 increments

speed normalization:
e.g. speed = 3000 rpm ⇒ setpoint = 419430

The table shows the mapping for the *Setpoint* PDO:

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	60400010_h	control word
02_h	20600020_h	current/speed setpoint

4.3.1.1.4 PDO Setpoint 2 (32)

The PDO *Setpoint 2* is a time- and data-optimized PDO. It contains just one 32-bit setpoint. This setpoint can only be used in the *Digital speed* or *Digital current* mode. It will be recognized as a speed or current setpoint, depending on the mode that is set (digital current or digital speed).
The PDO is executed immediately. A repeated transmission of the PDO with various setpoint values does not require an intermediate halt of the drive.

The table shows the mapping for the PDO *Setpoint 2*:

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	20600020_h	current/speed setpoint

4.3.1.1.5 PDO Trajectory (33)

The *Trajectory* PDO is a time- and data-optimized PDO. This PDO must only be used in the Trajectory mode. The *Trajectory* PDO must always be transmitted at constant time intervals (to be set with the *PTBASE* command), otherwise there may be irregularities in the speed characteristic. This PDO consists of two components: the incremental actual position values for two axes. The assignment of the data for the axes, which must both be set to the same COB-ID for this RPDO (Sub-index 01_h of the corresponding communication parameters), is made through the Object 2721_h.

Example of the calculation of the absolute position:

$$\text{Position} = \frac{\text{incremental position value}}{2^{20}}$$

The maximum difference between two incremental positions is given by the final limit speed that is set (ASCII-command VLIM) – see example.

Example of the maximum incremental position difference:

$$\text{max. achievable final speed} / 1000 \frac{\text{rev}}{\text{min}} = 0.016667 \frac{\text{rev}}{\text{ms}}$$
$$|\text{incr.pos.}(t_2) - \text{incr.pos.}(t_1)| \le 2^{20} \times 0.016667 = 17475$$

Depending on the amplifier parameters that have been set, there may be a larger or smaller lag/following error. If the error message *contouring error* appears and the axis is stopped with the emergency ramp, there are several possible reasons:

- The selected incremental position difference is too large (see above).
- The contouring error window has been set too small (Object 2020_h Sub-index 03_h)
- The amplifier parameters have not been set up optimally.

The table shows the mapping of the *Trajectory* PDO:

Sub-index	Value	Description
00_h	2_h	number of entries
01_h	20220420_h	incremental position
02_h	20220420_h	incremental position

This Object does **not** support transmission type 255 (asynchronous).

4.3.1.1.6 PDO Motion Block (34)

The *Motion Block* PDO is put together from the position (SIGNED32, weighted), speed (UNSIGNED16) and the type of motion tasks (UNSIGNED16).
The PDO starts a motion block from the volatile motion block memory (motion block number = 0) and can only be used in the Position mode.

The table shows the mapping for the *Motion Block* PDO:

Sub-index	Value	Description
00_h	3_h	number of entries
01_h	20220120_h	position
02_h	20220210_h	speed
03_h	20220310_h	motion task type (absolute/relative)

This Object **only** supports transmission type 255 (asynchronous).

4.3.1.1.7 PDO Start Motion Block (35)

The *Start Motion Block* PDO is formed by the motion task number (UNSIGNED16).
The PDO starts a motion block from the volatile memory (motion block number = 0, 192 ... 255) or from the permanent memory (motion block number = 1 ... 180).
This PDO can only be used in the *Position* mode.

The table shows the mapping of the *Start Motion Block* PDO:

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	20220510_h	motion task number

This Object **only** supports transmission type 255 (asynchronous).

4.3.1.1.8 PDO Free Definition (37 to 40)

If these PDOs are selected, then Objects can be freely added. The Objects 1600_h to 1603_h (mapping parameters) are used for this purpose. Up to 8 individual Objects can be mapped into a single PDO.

An example of PDO mapping and communication configuration can be found in the appendix.
(⇒ 6.1)

4.3.1.1.9 PDO master position for CAN slaves (41)

CAN slaves receive position setpoints from a master via PDO 41; they follow the master in exactly the same way as in the case of the electronic gear. These position setpoints must be transmitted using a fixed time base.

The following settings are required for this type of operation:

– The clock pulse must be set via Object 60C2h (-> PTBASE).
– The synchronization source is the CAN bus -> SYNCSRC = 3
– FPGA = 3 (SR600 only)
– CAMMCTRL = 2
– CANopen mode 0xFA (Trajectory mode)

The table shows how the Receive PDO "Master position over CAN" is mapped.

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	20220420_h	internal position setpoint

4.3.1.2 Object Decription

4.3.1.2.1 Object 1400-1403h: 1st-4th Receive-PDO communication parameter (DS301)

Sub-index	Description
00_h	number of entries
01_h	PDO COB-ID*
02_h	transmission type**
03_h	inhibit time
04_h	CMS priority group

* After selecting a PDO higher than 2600_h to 2603_h , here you can define the COB-identifier that the amplifier responds to (if a different setting is required from the default value).

** For selecting SYNC, event-triggering etc.

4.3.1.2.2 Object 1600-1603h: 1st-4th Receive-PDO mapping parameter (DS301)

Sub-index	Description
00_h	number of configured Objects
01_h ... 08_h	description of the configured Objects

4.3.1.2.3 Object 2600-2603h: 1st-4th Receive PDO select

This Object is used to select a predefined receive-PDO. The Objects 1400_h to 1403_h *1st to 4th receive-PDO parameter* and 1600_h to 1603_h *1st to 4th receive-PDO mapping* can then be used to define the characteristics of this PDO.

This Object enables variable mapping of predefined PDOs. The possible PDOs for selection are listed in the table in Chapter 4.3.1.

Index	2600_h ... 2603_h
Name	1st ... 4th receive-PDO select
Object code	VAR
Data type	UNSIGNED8

Access	rw
PDO mapping	not possible
Value range	0 ... 255
Default value	0

4.3.1.2.4 Object 2721h: Configuration of Receive-PDO 33

This Object influences the processing of the eight data bytes received for the PDO *Trajectory* (No. 33). The bytes of the PDO are interpreted as an incremental setpoint for the next movement, as follows: with the value *LOW* (= 0), the bytes 0 – 3 (Sub-index 01_h of the PDO mapping) are used, and with the value *HIGH* (= 1), the bytes 4 – 7 (Sub-index 02_h of the PDO mapping) are used.

Index	2721_h
Name	PDO 33 data select
Object code	VAR
Data type	UNSIGNED8

Access	rw
PDO mapping	not possible
Unit	—
Value range	0, 4
EEPROM	no
Default value	0

4.3.2 Transmit-PDOs (TPDO)

The Objects $2A00_h$ to $2A03_h$ can be used to select TPDOs. The configuration of the PDOs is made with the Objects 1800_h to 1803_h (communication parameter) und $1A00_h$ to $1A03_h$ (mapping parameter). These Objects have channel-linked operation, i.e.

- TPDO Channel 1: $2A00_h$, 1800_h, $1A00_h$
- TPDO Channel 2: $2A01_h$, 1801_h, $1A01_h$
- TPDO Channel 3: $2A02_h$, 1802_h, $1A02_h$
- TPDO Channel 4: $2A03_h$, 1803_h, $1A03_h$

The Objects $1A00_h$ to $1A03_h$ can only be read, in order to disclose the Object configuration within the PDOs.

NOTE

Exception: If one or more PDOs 37 to 40 are selected, then the Object configuration for the relevant TPDO channel must be made through the Objects $1A00_h$ to $1A03_h$ (write access).

The following PDOs have been defined:

PDO Number	Mapping Object Index	Sub-index	Mapping Object Name
1 (01_h)	6041_h	—	status word
2..20 (02_h..14_h)	—		reserved
21 (15_h)	3102_h	01_h..08_h	ASCII Object
22 (16_h)	6041_h	—	status word
	2070_h	01_h	actual position (20 bits/turn)
	2070_h	02_h	speed (1 bit = 1875/262144 [revs/min])
23 (17_h)	6041_h	—	status word
	1002_h	—	manufacturer-specific status register
24..31 (18_h..$1F_h$)	—	—	reserved
32 (20_h)	2070_h	01_h	actual position (20 bits/turn)
	2070_h	02_h	speed (1 bit = 1875/262144 [revs/min])
33 (21_h)	2070_h	03_h	incremental actual position value
	2070_h	11_h	2nd enhanced status register
34..35 (22_h..23_h)	—	—	—
36 (24_h)	—	—	reserved
37..40 (25_h..28_h)	$xxxx_h$	—	freely definable TPDOs
41 (29_h)	2070_h	19_h	internal master position setpoint
42..64 (30_h..40_h)	—	—	reserved

4.3.2.1 Description of the predefined Transmit-PDOs

4.3.2.1.1 PDO Status Word (1) (DS402)

The *status word* PDO (default PDO) is formed by the status word (UNSIGNED16).
This PDO can only be used to disclose the state of the status machine (⇒ 4.4.1). The PDO is mode-independent.
After switch-on, this PDO is mapped to TPDO 1.

The table shows the mapping for the *status word* PDO.

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	60410010_h	status word

4.3.2.1.2 PDO Transmit ASCII Channel (21)

As soon as ASCII characters have been transferred to the ASCII transmission buffer, they are transmitted to the master (control system) with the help of this PDO (default PDO). This always takes place when commands or parameters have been transmitted with the aid of the PDO *Receive ASCII channel* (⇒ 4.3.1.1.2). After switch-on, this PDO is mapped to TPDO 2.

The table shows the mapping for the PDO *Transmit ASCII channel*.

Sub-index	Value	Description
00_h	8_h	number of entries
$01_h..08_h$	31020208_h	ASCII chars. 1 ... 8

This Object **only** supports transmission type 255 (asynchronous).

4.3.2.1.3 PDO Actual Position (22)

The *Actual Position* PDO is formed from the status word (UNSIGNED16), actual position (UNSIGNED24) and speed in revs/minute (UNSIGNED24). This PDO can be used to disclose the position in the *Digital speed* or *Digital current* mode.

The table shows the mapping for the PDO *Actual Position*:

Sub-index	Value	Description
00_h	3_h	number of entries
01_h	60410010_h	status word
02_h	20700118_h	actual position (20 bits/turn)
03_h	20700218_h	speed *

* Resolution: 1 bit = 1875 / 262144 revs/min

4.3.2.1.4 PDO Enhanced Status (23)

The Enhanced Status PDO is formed by the status word (UNSIGNED16) and the manufacturer specific status register (UNSIGNED32). This PDO can additionally be triggered by an event in the status register area. The PDO is mode-independent.

The table shows the mapping for the PDO *Enhanced Status*:

Sub-index	Value	Description
00_h	2_h	number of entries
01_h	60410010_h	status word
02_h	10020020_h	manufacturer specific status register

4.3.2.1.5 PDO Actual Position 2 (32)

The PDO *Actual Position 2* is a time- and data-optimized PDO (see PDO 21). It contains the actual position (UNSIGNED24) and the revs/minute (UNSIGNED 24). This PDO can be used to disclose the position in the *Digital speed* or *Digital current* mode.
This PDO can **only** be requested with the **SYNC Object.**

The table shows the mapping for the PDO *Actual Position 2*:

Sub-index	Value	Description
00_h	2_h	number of entries
01_h	20700118_h	actual position (20 bits/turn)
02_h	20700218_h	speed *

* Resolution: 1 bit = 1875 / 262144 revs/min

This Object **only** supports transmission types 1 to 240 (cyclically synchronous).

4.3.2.1.6 PDO Incremental Actual Position (33)

The PDO *Incremental Actual Position* is a data-optimized Object that can **only** be requested with a **SYNC Object**.

Calculation of the absolute position:

$$\text{Position} = \frac{\text{incremental actual position}}{2^{20}}$$

The table shows the mapping for the PDO *Incremental Actual Position*:

Sub-index	Value	Description
00_h	2_h	number of entries
01_h	20700320_h	incremental actual position
02_h	20701120_h	enhanced status register for PDO 33

This Object **only** supports transmission types 1 to 240 (cyclically synchronous).

4.3.2.1.7 PDO Position Treshold (34)

PDO *position threshold* is used to recognize an exceeding of the fast position registers P1..P16. They are configured by Objects 2051 - 2053.

The table shows the mapping for the PDO *position threshold*:

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	20860010_h	Display of position threshold

4.3.2.1.8 PDO Free Definition (37 to 40)

If these PDOs are selected, then Objects can be freely added. The Objects $1A00_h$ to $1A03_h$ (mapping parameters) are used for this purpose. Up to 8 individual Objects can be mapped into a single PDO.

NOTE

Even if Objects are “event-triggered” and not called over the CAN bus, they will still be monitored. 16 mapped Objects can therefore place a heavy processing load on the CPU. To avoid this, only map those objects which are really needed, or increase the “inhibit time” accordingly (e.g. to 400 ms). An excessive CPU loading can cause a long response time for an SDO data service (average response time > 40 ms).

4.3.2.1.9 PDO Internal Master Position Setpoint (41)

The CAN master can use Transmit PDO 41 to notify the bus of its internal trajectory position setpoint (e.g. from travel-block mode). These values are output using a time base determined by PTBASE.

The following settings are required for this type of operation:

- The clock pulse must be set via Object 60C2 (->PTBASE)
- SYNCSRC = 3
- FPGA = 3 (S600 only)
- CAMMCTRL = 1

The table shows how the PDO "Internal master position setpoint" is mapped.

Sub-index	Value	Description
00_h	1_h	number of entries
01_h	20701920_h	internal position setpoint

4.3.2.2 Object Description

4.3.2.2.1 Object 1800-1803h: 1st-4th Transmit-PDO communcation param. (DS301)

Sub-index	Description
00_h	number of entries
01_h	PDO COB-ID *
02_h	transmission type
03_h	inhibit time **
04_h	CMS priority group

* After selecting a PDO higher than $2A00_h$ to $2A03_h$, here you can define the COB-identifier that the amplifier responds to (if a different setting is required from the default value).

** This sets a waiting time that must be observed after a PDO has been transmitted, before a new transmission can be made (only important for event-triggered PDOs).

4.3.2.2.2 Object 1A00-1A03h: 1st-4th Transmit-PDO mapping parameter (DS301)

Sub-index	Description
00_h	number of configured Objects
01_h ... 08_h	description of the configured Objects

4.3.2.2.3 Object 2A00-2A03h: 1st-4th Transmit-PDO select (DS301)

This Object is used to select a predefined transmit-PDO.
The Objects 1800_h to 1803_h *1st to 4th transmit-PDO parameter* and $1A00_h$ to $1A03_h$ *1st to 4th transmit-PDO mapping* can then be used to define the characteristics of this PDO.

This Object enables variable mapping of predefined PDOs. The possible PDOs for selection are listed in the table in Chapter 4.3.2.

Index	$2A00_h$ to $2A03_h$
Name	1st to 4th transmit-PDO select
Object code	VAR
Data type	UNSIGNED8

Access	rw
PDO mapping	not possible
Value range	0 ... 255
Default value	0

4.3.2.2.4 Object 2014-2017h: 1st-4th Mask 1 to 4 for Transmit-PDO

In order to reduce the bus loading with event-triggered PDOs, masking can be used to switch off the monitoring for individual bits in the PDO. In this way it can be arranged, for instance, that actual position values are only signaled once per turn.
This Object masks the PDO-channels 1 to 4. If only two bytes have been defined in a PDO, then it masks just two bytes, although 4 bytes of mask information have been transmitted.
An activated bit in the mask means that monitoring is active for the corresponding bit in the PDO.

Index	$0x2014_h$ to $0x2017_h$
Name	tx_mask 1 to 4
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Sub-Indexes:

Sub-index	01_h
Brief description	tx_mask1 to 4_low
Mode	independent
Access	rw
PDO mapping	not possible
Unit	—
Value range	UNSIGNED32
EEPROM	no
Default value	$FFFFFFFF_h$

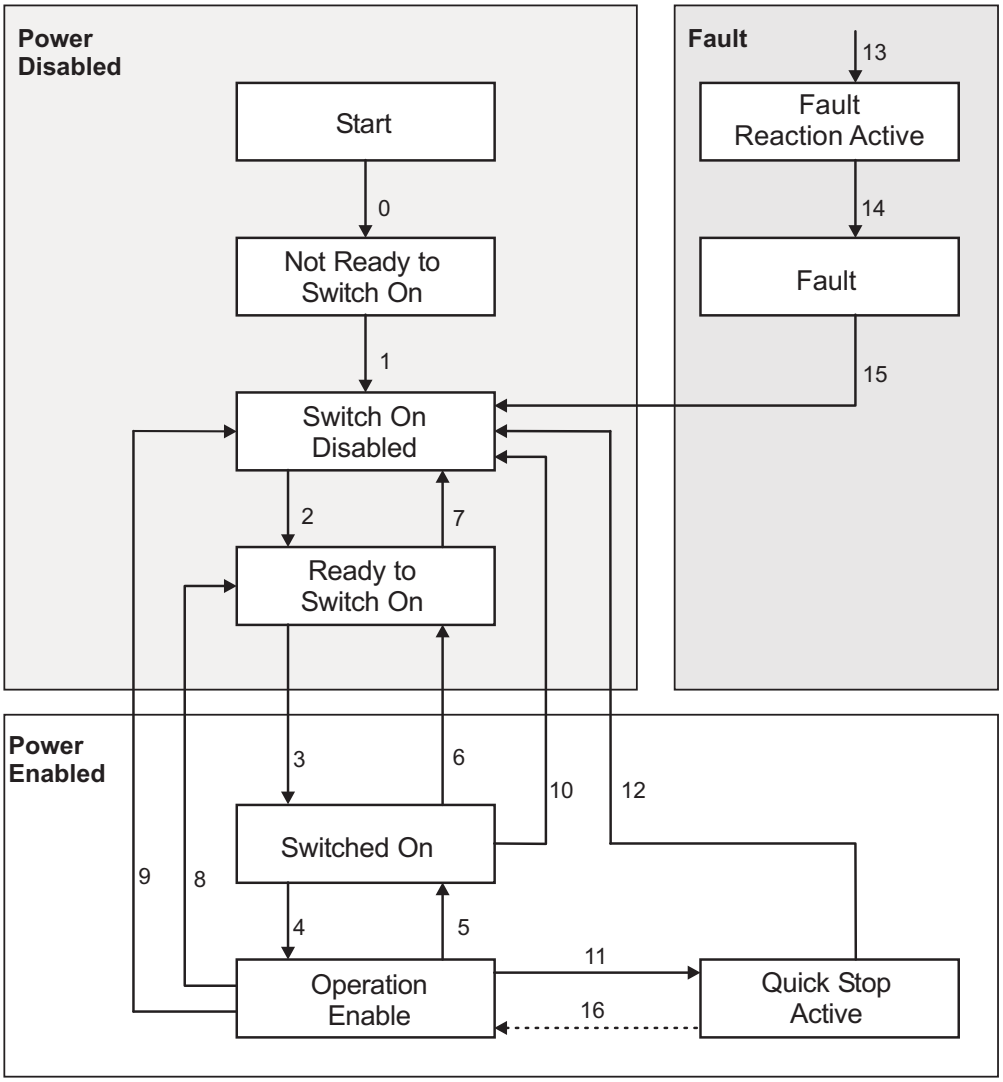
Sub-index	02_h
Brief description	tx_mask1 to 4_high
Mode	independent
Access	rw
PDO mapping	not possible
Unit	—
Value range	UNSIGNED32
EEPROM	no
Default value	$FFFFFFFF_h$

4.4 Device Control (dc)

The device control of the SERVOSTAR can be used to carry out all the motion functions in the corresponding modes. The control of the SERVOSTAR 400/600 is implemented through a mode-dependent status machine. The status machine is controlled through the control word (⇒ 4.4.2.1).
The mode setting is made through the Object “Modes of Operation” (⇒ 6.3). The states of the status machine can be revealed by using the status word (⇒ 4.4.2.2).

4.4.1 Status Machine (DS402)

4.4.1.1 States of the Status Machine



State	Description
Not Ready for Switch On	SERVOSTAR is not ready to switch on, there is no operational readiness (BTB/RTO) signaled from the controller program.
Switch On Disable	SERVOSTAR is ready to switch on, parameters can be transferred, the DC-link voltage can be switched on, motion functions cannot be carried out yet.
Ready to Switch On	DC-link voltage may be switched on, parameters can be transferred, motion functions cannot be carried out yet.
Switched On	DC-link voltage must be switched on, parameters can be transferred, motion functions cannot be carried out yet, output stage is switched on (enabled).
Operation Enable	No fault present, output stage is enabled, motion functions are enabled.
Quick Stop Active	Drive has been stopped with the emergency ramp, output stage is enabled, motion functions are enabled, response depends on Object $605A_h$ (⇒ 6.3)
Fault Reaction Active	not supported at present
Fault	not supported at present

4.4.1.2 Transitions of the status machine

The state transitions are affected by internal events (e.g. switching off the DC-link voltage) and by the flags in the control word (bits 0,1,2,3,7).

Transition	Event	Action
0	Reset	Initialization
1	Initialization completed successfully. SERVOSTAR is ready to operate.	none
2	Bit 1 *Disable Voltage* and Bit 2 *Quick Stop* are set in the control word (*Shut-down* command). DC-link voltage may be present.	none
3	Bit 0 is also set (*Switch On* command)	Output stage is switched on (enabled), provided that the hardware enable is present (logical AND). Drive has torque.
4	Bit 3 is also set (*Enable Operation* command)	Motion function is enabled, depending on the mode that is set.
5	Bit 3 is canceled (*Disable Operation* command)	Motion function is inhibited. Drive is stopped, using the relevant ramp (mode-dependent). The present position is maintained.
6	Bit 0 is canceled (*Shutdown* command)	Output stage is disabled. Drive has no torque.
7	Bits ½ are canceled (*Quick Stop / Disable Voltage* command)	none
8	Bit 0 is canceled (*Shutdown* command)	Output stage is disabled. Drive has no torque.
9	Bit 1 is canceled (*Disable Voltage* command)	Output stage is disabled. Drive has no torque.
10	Bits ½ are canceled (*Quick Stop / Disable Voltage* command)	Output stage is disabled. Drive has no torque.
11	Bit 2 is canceled (*Quick Stop* command)	Drive is stopped with the emergency braking ramp. The output stage remains enabled. Setpoints are canceled (motion block number, digital setpoint, speed for jogging or homing). Bit 2 must be set again before any further motion tasks can be performed.
12	Bit 1 is canceled ('Disable Voltage' command)	Output stage is disabled. Drive has no torque.
13	not supported at present	none
14	not supported at present	none
15	not supported at present	none
16	Bit 2 is set	Motion function is enabled again.

NOTE

If the servo amplifier is operated through the control word / status word, then no control commands may be sent through another communication channel (RS232, CANopen, ASCII channel, Option board).

4.4.2 Object description

4.4.2.1 Object 6040h: Controlword (DS402)

The control commands are built up from the logical combination of the bits in the control word and external signals (e.g enable output stage). The definitions of the bits are shown below:

Index	$0x6040_h$
Name	control word
Object code	VAR
Data type	UNSIGNED16

Mode	all
Access	rw
PDO mapping	possible
Unit	—
Value range	0 ... 65535
EEPROM	no
Default value	0

Bit assignment im control word

Bit	Name	Bit	Name
0	Switch on	8	Pause/halt
1	Disable Voltage	9	reserved
2	Quick Stop	10	reserved
3	Enable Operation	11	Acknowledge lag/following error and response monitoring
4	Operation mode specific	12	Reset position
5	Operation mode specific	13	Manufacturer-specific
6	Operation mode specific	14	Manufacturer-specific
7	Reset Fault (only effective for faults)	15	Manufacturer-specific

Commands in the control word

Command	Bit 7 Fault Reset	Bit 3 Enable Operation	Bit 2 Quick Stop	Bit 1 Disable Voltage	Bit 0 Switch on	Transitions
Shutdown	X	X	1	1	0	2, 6, 8
Switch on	X	X	1	1	1	3
Disable Voltage	X	X	X	0	X	7, 9, 10, 12
Quick Stop	X	X	0	1	X	7, 10, 11
Disable Operation	X	0	1	1	1	5
Enable Operation	X	1	1	1	1	4, 16
Fault Reset	1	X	X	X	X	15

Bits marked by an X are irrelevant.

Mode-dependent bits im control word

The following table shows the mode-dependent bits in the control word. Only manufacturer-specific modes are supported at present. The individual modes are set by Object 6060_h *Modes of operation*.

Operation mode	No.	Bit 4	Bit 5	Bit 6
Position	FF_h	reserved	reserved	reserved
Digital speed	FE_h	reserved	reserved	reserved
Digital current	FD_h	reserved	reserved	reserved
Analog speed	FC_h	reserved	reserved	reserved
Analog current	FB_h	reserved	reserved	reserved
Trajectory	FA_h	reserved	reserved	reserved
Homing	$F9_h$	Start homing	reserved	reserved
Jogging	$F8_h$	reserved	reserved	reserved
Profile Position Mode (pp)	01_h	new_set_point	change_set_immediately	absolute / relative
Profile Velocity Mode (pv)	03_h	reserved	reserved	reserved
Homing Mode (hm)	06_h	homing_operation_start	reserved	reserved

Description of the remaining bits in the control word

The remaining bits in the control word are described below.

Bit 8 Pause If Bit 8 is set, then the drive halts (pauses) in all modes. The setpoints (speed for homing or jogging, motion task number, setpoints for digital mode) for the individual modes are retained.
Bit 9,10 These bits are reserved for the drive profile (DS402).
Bit 11 Acknowledge error Setting Bit 11 acknowledges the response monitoring and/or the contouring error.
Bit 12 Reset the position, taking account of the reference offset. (See also Homing type 6 in Object 2024_h, Sub-index 01_h)

Bit 13, 14, 15 These bits are manufacturer-specific, and reserved at present.

4.4.2.2 Object 6041h: Statusword (DS402)

The momentary state of the status machine can be read out with the aid of the status word (⇒ 4.3.2.1.1).

Index	$0x6041_h$
Name	Status word
Object code	VAR
Data type	UNSIGNED16

Mode	all
Access	rw
PDO mapping	possible
Unit	—
Value range	0 ... 65535
EEPROM	yes
Default value	0

Bit assignment in the status word

Bit	Name	Bit	Name
0	Ready to switch on	8	Manufacturer-specific (reserved)
1	Switched on	9	Remote (not supported)
2	Operation enable	10	Target reached
3	Fault (in preparation)	11	Internal limit active
4	Disable voltage	12	Operation mode specific (reserved)
5	Quick stop	13	Operation mode specific (reserved)
6	Switch on disabled	14	Manufacturer-specific (reserved)
7	Warning	15	Manufacturer-specific (reserved)

States of the status machine

State	Bit 6 switch on disable	Bit 5 quick stop	Bit 3 fault	Bit 2 operation enable	Bit 1 switched on	Bit 0 ready to switch on
Not ready to switch on	0	X	0	0	0	0
Switch on disabled	1	X	0	0	0	0
Ready to switch on	0	1	0	0	0	1
Switched on	0	1	0	0	1	1
Operation enabled	0	1	0	1	1	1
Fault	not supported at present					
Fault reaction active	not supported at present	X	X	X	X	15
Quick stop active	0	0	0	1	1	1

Bits marked by X are irrelevant

Description of the remaining bits in the status word

Bit 4: voltage_disable The DC-link voltage is present if this bit is canceled.

Bit 7: warning There are several possible reasons for Bit 7 being set and this warning being produced. The reason for this warning can be revealed by using the Object 1002_h *manufacturer-specific status register*. (⇒ 4.2.1.3)

Bit 8: Toggle–bit mode Jogging, Homing ,Position, Positioning (pp), Homing (hm)
The toggle-bit is always changed (i.e. set or reset) when a motion block has been successfully executed (incrementally precise in the target position!). The bit is not toggled if a motion block is canceled (e.g. through the STOP command or a lag/following error). The evaluation of the toggle-bit can be performed in combination with Bit 10 *target reached* (Object 6041_h) and Bit 16 *motion block active* (Object 1002_h). Evaluation of this bit makes sense if no change in Bit 10 or 16 would be visible, because of the motion block data (very short or identical motion blocks).

Bit 9: remote is not supported at present

Bit 10: target_reached This is set when the drive has reached the target position.

Bit 11: internal_limit_active If bit 20 is set in parameter DRVCNFG, it will be set in the event of the following warnings:
- Response Monitoring (n04)
- SW Limit Switch 1 (n06)
- SW Limit Switch 2 (n07)
- no reference point set (n09)
- P-STOP (n10)
- N-STOP (n11)
- Slot Warning (n13)

Bit 12: profile position mode setpoint acknowledge
digital speed modes velocity 0 detection (1 = speed = 0)

4.4.2.3 Object 6060h: modes_of_operation (DS402)

This Object is used to set the mode, which can be read out by Object 6061_h. Two types of operating mode can be distinguished:

manufacturer-specific operating modes
These modes of operation have been optimized to the functionality of the equipment.

operating modes as per CANopen drive profile DSP402
These operating modes are defined in the CANopen drive profile DSP402.

After the mode has been changed, the corresponding setpoint must be set once more (for instance, the homing velocity in the mode homing_setpoint). If the position or jogging mode is stored, then the Homing mode is set after a RESET of the servo amplifier.

NOTE

An operating mode only becomes valid when it can be read by Object 6061_h.

⚠ WARNING Unexpected Start!

Risk of death or serious injury for humans working in the machine.

- Never change the mode while the motor is running!
- When the servo amplifier is enabled, a mode change is only permissible at zero speed.
- Set the speed setpoint to 0 before changing over.

Index	$0x6060_h$
Name	mode_of_operation
Object code	VAR
Data type	INTEGER8

Mode	all
Access	wo
PDO mapping	possible
Unit	—
Value range	—
EEPROM	no
Default value	—

Mode	decimal	hex.	Comments
	-10 ... -128	$F8_h$ ... 80_h	reserved
Electrical gearing	-9	$F7_h$	—
Jogging	-8	$F8_h$	—
Homing	-7	$F9_h$	—
Trajectory	-6	FA_h	—
Analog current	-5	FB_h	—
Analog speed	-4	FC_h	—
Digital current	-3	FD_h	—
Digital speed	-2	FE_h	—
Position	-1	FF_h	mode required for motion tasks
—	0	0	reserved as per DSP402
Positioning (pp)	1	1_h	as per DSP402
Speed (vl)	2	2_h	not supported
Speed (pv)	3	3_h	as per DSP 402
Torque (tq)	4	4_h	as per DSP 402 (not supported at present)
—	5	5_h	reserved
Homing (hm)	6	6_h	as per DSP 402
Interpolation	7	7_h	as per DSP 402 (not supported)
—	8 ... 127	8_h ... $7F_h$	reserved

4.4.2.4 Object 6061h: modes_of_operation_display (DS402)

This Object can be used to read the mode that is set by Object 6060_h. An operating mode only becomes valid when it can be read by Object 6061_h (see also Object 6060_h).

Index	$0x6061_h$
Name	mode_of_operation_display
Object code	VAR
Data type	INTEGER8

Mode	all
Access	wo
PDO mapping	possible
Unit	—
Value range	—
EEPROM	no
Default value	—

4.5 Factor Groups (fg) (DS402)

4.5.1 General Information

4.5.1.1 Factors

There is a possibility to convert between physical dimensions and sizes, and the internal units used in the device (increments). Several factors can be implemented. This chapter describes how these factors influence the system, how they are calculated and which data are necessary to build them.
Normalized parameters are denoted by an asterisk *.

4.5.1.2 Relationship between physical and internal units

The factors defined in the factor group set up a relationship between device-internal units (increments) and physical units.

The factors are the result of the calculation of two parameters called dimension index and notation index. The dimension index indicates the physical dimension, the notation index indicates the physical unit and a decimal exponent for the values. These factors are directly used to normalize the physical values.

The notation index can be used in two ways:

- For a unit with decimal scaling and notation index < 64, the notation index defines the exponent/decimal place of the unit.
- For a unit with non-decimal scaling and notation index > 64, the notation index defines the sub-index of the physical dimension of the unit.

4.5.2 Object Description

4.5.2.1 Object 608Bh: velocity_notation_index (DS402)

Index	$0x608B_h$
Name	velocity_notation_index
Object code	VAR
Data type	INTEGER8

Mode	all
Access	rw
PDO mapping	not possible
Unit	—
Value range	-128 ... 127
EEPROM	no
Default value	0

This Object determines the notation for Object 6081_h *profile_velocity*. In combination with Object $608C_h$ *velocity_dimension_index*, the following *basic units* can be represented:

physical dimension	units	velocity_dimension_index	velocity_notation_index
manufacturer-specific	incr/sec	0	0
revolution	revs/min	11	73

4.5.2.2 Object 608Ch: velocity_dimension_index (DS402)

Index	$0x608C_h$
Name	velocity_dimension_index
Object code	VAR
Data type	UNSIGNED8

Mode	all
Access	rw
PDO mapping	not possible
Unit	—
Value range	0 ... 255
EEPROM	no
Default value	0

This Object determines the dimension for Object 6081_h *profile_velocity*. In combination with Object $608B_h$ velocity_notation_index, the following *basic units* can be represented:

physical dimension	units	velocity_dimension_index	velocity_notation_index
manufacturer-specific	incr/sec	0	0
revolutions	revs/min	11	73

4.5.2.3 Object 6093h: position_factor (DS402)

The position factor converts the target position into the internal data format of the SERVOSTAR 400/600 (increments).
The position controller can be run with the resolution set to 20 bits/turn or 16 bits/turn (see Object $35D1_h$ and the ASCII command PRBASE). The numerator and the feed constant can be used to set up any value of scaling.

$$\text{position\_ factor} = \frac{\text{position\_ encoder\_ resolution} \times \text{gear\_ ratio}}{\text{feed\_ constant}}$$

- **position_encoder_resolution**
  resolution of the position controller is 2^{20} or 2^{16}
- **gear_ratio**
  transmission ratio for the gearing that is used
- **feed_constant**
  the feed constant of the output side of the drive gearing

Index	$0x6093_h$
Name	position_factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Sub-Indexes:

Sub-index	01_h
Brief description	numerator
Mode	pp
Access	rw
PDO mapping	not possible
Unit	increments [incr]
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	2^{20}

Sub-index	02_h
Brief description	feed_constant
Mode	pp
Access	rw
PDO mapping	not possible
Unit	—
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	10000

Example: One turn is to be equivalent to 10000 increments. The gear ratio is 1.

$$\text{position\_ factor} = \frac{2^{20}\ \text{incr}}{10000\ \text{incr}}$$

⇒ **Numerator**: 2^{20} **Feed constant**: 10000
⇒ Setpoint provision in [incr/turn] for Object $607A_h$ (target_position).

The Numerator corresponds to the ASCII - parameter PGEARO, feed_constant to the parameter PGEARI.

4.5.2.4 Object 6094h: velocity_encoder_factor (DS402)

The velocity_encoder_factor converts the target speed (revs/min) or velocity (incr/sec) into the internal data format of the SERVOSTAR 400/600 (increments).

The velocity_encoder_factor is calculated as:

$$\text{velocity\_ encoder\_ factor} = \frac{\text{velocity\_ encoder\_ resolution} \times \text{gear\_ ratio} \times \text{position\_ unit} \times \text{Fvelocity(notation\_ index)}}{\text{feed\_ constant} \times \text{velocity\_ unit} \times \text{sec} \times \text{Fposition(notation\_ index)}}$$

- **velocity_encoder_resolution** the resolution of the speed 2^{20}
- **gear_ratio** the transmission ratio for the gearing that is used
- **position_unit** in meters
- **Fposition(notation_index)** in dimension_index = 1, notation_index = 0 [m]
- **feed_constant** the feed constant for the output side of the drive gearing
- **velocity_unit** in [m/s]
- **Fvelocity (notation_index)** in dimension_index = 13, notation_index = 0 [m / s]

(see also Object $606B_h$ *velocity_notation_index* and Object $606C_h$ *velocity_dimension_index*)

Index	$0x6094_h$
Name	velocity_encoder_factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Sub-Indexes:

Sub-index	01_h
Brief description	numerator
Mode	pv
Access	rw
PDO mapping	not possible
Unit	increments [incr]
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	0

Sub-index	02_h
Brief description	divisor
Mode	pv
Access	rw
PDO mapping	not possible
Unit	seconds [s]
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	0

Example: The speed-setpoint provision is to be made in revolutions per minute (revs/min). The transmission ratio and feed constant are 1.

$$\text{velocity\_encoder\_factor} = \frac{2^{20}}{1} \quad \frac{1/1}{1} \quad \frac{\text{incr}}{\text{s}}$$

Expanded for revs/min:

$$\text{velocity\_encoder\_factor} = \frac{2^{20}}{1} \quad \frac{1/1}{1} \quad \frac{[\text{m}]}{[\text{m}/\text{s}]} \quad \frac{\text{incr}}{\text{s}} \times \frac{1}{60[\text{s}/\text{min}]} = \frac{2^{20}}{60} \quad \frac{\text{incr}}{\text{s}}$$

⇒ **Numerator** : 2^{20} **Divisor** : 60

⇒ Setpoint provision in [revs/min] for Object $60FF_h$ *target_velocity / velocity units*

⇒ Setpoint provision in [incr/s] for Object 6081_h *Profile_velocity / speed units*

Note: Since the speed controller operates internally with a resolution of 2^{20} bits/turn, regardless of the resolution of the encoder system, the following expression is used to calculate the operating mode pv (for revolutions per minute):

$$\text{increments} = \frac{262144}{1875} \times \text{speed setpoint}[\text{min}^{-1}]$$

This incremental setpoint provision should be used for cyclical applications (for example, position control, 4ms cycle). The advantages are: no rounding error, lower CPU loading.

The above calculation is valid if **divisor or numerator is set to 0**.

The *velocity_encoder_factor* also affects Object 6081_h *profile_velocity*. In order to be able to use this factor for the position mode (pp) as well, the internal gearing factors PGEARI and PGEARO must be equal (PGEARI =PGEARO; Object 2020_h Sub-index: 08_h,09_h). If the **divisor or numerator is set to 0**, then the internal scaling is used: *increments per cycle (250 µs)*.

4.5.2.5 Object 6097h: acceleration_factor (DS402)

The *acceleration_factor* converts the acceleration [unit: /s^2] into the internal format of the SERVOSTAR 400/600.
At present, the numerator and divisor are read-only. The values are set to 1. If the *acceleration_factor* is 1, then the ramp settings (Object 6083_h *profile_acceleration* and Object 6084_h *profile_deceleration*) will be provided as acceleration times [ms] required to reach the target speed (Object 6081_h *profile_velocity*).

Index	$0x6097_h$
Name	acceleration_factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Sub-Indexes:

Sub-index	01_h
Brief description	numerator
Mode	pp, pv
Access	ro
PDO mapping	not possible
Unit	increments [incr]
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	0

Sub-index	02_h
Brief description	divisor
Mode	pp, pv
Access	ro
PDO mapping	not possible
Unit	—
Value range	0 ... (2^{32}-1)
EEPROM	no
Default value	0

4.6 Manufacturer-specific Current and Speed-Mode

4.6.1 Object 2060h: Digital current or speed setpoint

Index	2060_h
Brief description	digital current/speed setpoint
Unit	A or /min
Access	rw
PDO mapping	possible (pre-mapped to selectable RPDO 22)
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

This Object is used for the transfer of digital setpoint values which are evaluated according to the digital mode that has been set (mode FD_h = digital current, mode FE_h = digital speed, adjustable via Object 6060_h). The normalizations are as follows:

Current: $I[A] = {\text{digital current setpoint}}/{1640} \times I_{max}$

Speed: $n[min^{-1}] = {1875}/{262144} \times \text{digital speed setpoint}$

A new setpoint only becomes valid after a fresh *Enable Operation* (via Object 6040_h, control word)

NOTE The SERVOSTAR 400/600 position controller is switched off while speed or current control is activated.

4.6.2 Objekt 2061h: Current limitation

This Object is used rapid current limitation in Speed mode (0x3, 0xFE, 0xFC). A value of 3280 corresponds to the maximum device current that can be requested via DIPEAK.

Index	2061_h
Brief description	Current limitation
Unit	A (referred to DIPEAK)
Access	wo
PDO mapping	possible
Data type	UNSIGNED16
Value range	0...3280
Default value	0

This Object affects ASCII parameter DPRILIMIT. In order for it to become effective, configuration parameter DILIM must be set to 1.

4.7 Setup data for manufacturer-specific Jogging/Homing mode

4.7.1 Object 2024h: Setup operation for position mode (SERVOSTAR)

Index	2024_h
Brief description	parameters for homing and jogging
Object code	RECORD
Number of elements	7

Used to enter parameters which are important for the homing and jogging modes of operation.

Sub-index	01_h
Brief description	homing type
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0..6
Default value	0

Used to set the type of homing movement. The following settings are possible:

Value	Meaning
0	The reference mark is set to be at the present position. The actual position that is signaled is then the same as the preset reference offset.
1	Homing to reference switch, followed by search of the resolver zero mark.
2	Homing to limit switch, followed by search of the resolver zero mark.
3	Homing to reference switch, without subsequent search of the resolver zero mark.
4	Homing to limit-switch, without subsequent search of the resolver zero mark.
5	Homing within a single turn of the motor to the resolver zero mark. The direction of movement is given by the Sub-index 02_h, whereby: 0: negative direction of motion 1: positive direction of motion 2: motor turns in the direction which gives the shortest path to the resolver zero mark within a single turn.
6	The reference mark is set to the value of the reference offset at the present setpoint position of the position controller. The new actual position retains the same distance to the setpoint position as before.

NOTE

For homing types 1 and 3, a digital input must be configured as the zero-position input (Home position).
For homing types 2 and 4, a digital input must be configured as a hardware limit-switch.
For homing types 1 to 5, the setting of the index pulse offset for the ROD output is taken into account (ASCII command ENCZERO), i.e. the zero point is positioned so that both the output of the index pulse and the display of the 0-position take place at the point of the index pulse offset.

Sub-index	02_h
Brief description	homing direction
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0 ... 2
Default value	0

Used to define the movement direction for homing types 1 to 5.
The values have the following interpretation:

0: negative direction of motion
1: positive direction of motion
2: the motor turns in the direction which gives the shortest path to the resolver zero mark within a single turn (only relevant for homing type 5).

Sub-index	03_h
Brief description	velocity for homing
Unit	µm/s
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to define the velocity for homing movements.

Sub-index	04_h
Brief description	acceleration ramp for homing/jogging
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 32767
Default value	10

Used to set the acceleration ramp for homing and jogging. The ramp has a trapezoidal form. The time that is set refers to the preset velocities homing and jogging movements.

Sub-index	05_h
Brief description	braking ramp for homing/jogging
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 32767
Default value	10

Used to set the acceleration ramp for homing and jogging. The ramp has a trapezoidal form. The time that is set refers to the preset velocities homing and jogging movements. The emergency stop ramp (ASCII parameter DECSTOP) is used as the braking ramp for homing movements to a hardware limit-switch.

Sub-index	06_h
Brief description	reference offset
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to set the reference offset, i.e. the actual position value displayed when homing has been completed (Object 2070_h, Sub-index 06_h).

Sub-index	07_h
Brief description	velocity for jogging t
Unit	µm/s
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to define the jogging velocity.

4.8 Positioning data for position mode (SERVOSTAR)

4.8.1 Object 2020h: Position controller

Index	2020_h
Brief description	parameter for position controller
Object code	RECORD
Number of elements	10

Used to enter all the general parameters for the *Position* mode.

Sub-Indexes:

Sub-index	01_h
Brief description	axis type
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0, 1
Default value	0

Describes the type of mechanical axis.

Value 0: Linear axis. Positions are counted from a defined reference mark. This must be defined by making a homing movement or setting a reference point. The movement of the axis will be restricted by software limit-switches, if they have been configured.

Value 1: Rotary axis. A reference position is not required. The position is set to 0 at the start of each motion block or jogging mode. Movement is not limited by software limited switches.

Sub-index	02_h
Brief description	In-Position window
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	4000_h

Defines a target window for positioning. Bit 19 of the manufacturer-specific status register is set when the window limits are reached, and the selected output goes *High* (if the corresponding configuration was implemented).

Sub-index	03_h
Brief description	maximum contouring error
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	40000_h

Defines a maximum value for the contouring (lag) error. The drive will be stopped if the contouring error exceeds this value. Bit 2 of the manufacturer-specific status register is used to indicate an excessive contouring error. The contouring error will not be monitored if the value is set to 0.

Sub-index	04_h
Brief description	position register 1
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Depending on the configuration, if the position goes above or below the preset position value, either a threshold bit (Bit 22 of the manufacturer-specific status register) will be set or the axis will be stopped. (going below software-limit switch 1 : Bit 5 = 1 in the manufacturer-specific status register).

Sub-index	05_h
Brief description	position register 2
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Depending on the configuration, if the position goes above or below the preset position value, either a threshold bit (Bit 23 of the manufacturer-specific status register) will be set or the axis will be stopped. (going above software-limit switch 2 : Bit 6 = 1 in the manufacturer-specific status register).

Sub-index	06_h
Brief description	position register 3
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Depending on the configuration, if the position goes above or below the preset position value, a threshold bit (Bit 24 of the manufacturer-specific status register) will be set.

Sub-index	07_h
Brief description	Position register 4
Unit	µm
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Depending on the configuration, if the position goes above or below the preset position value, a threshold bit (Bit 25 of the manufacturer-specific status register) will be set.

Sub-index	08_h
Brief description	resolution / denominator of the gearing ratio
Unit	revs (turns)
Access	rw
PDO mapping	not possible
Data type	UNSIGNED32
Value range	1 ... $(2^{32}-1)$
Default value	1

The ratio of the values of Sub-indices 08_h and 09_h provides the mechanical resolution of the axis in µm/turn.

Sub-index	09_h
Brief description	resolution / numerator of the gearing ratio
Unit	µm
Access	rw
PDO mapping	not possible
Data type	UNSIGNED32
Value range	1 ... $(2^{32}-1)$
Default value	1

The ratio of the values of Sub-indices 08_h and 09_h provides the mechanical resolution of the axis in µm/turn.

Sub-index	$0A_h$
Brief description	count direction
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0, 1
Default value	1

The value gives the count direction for current, speed, and position control. A value of 1 means that the positive count direction has been selected. If a positive setpoint is provided, the motor shaft rotates in a clockwise direction (looking at the shaft end).

4.8.2 Object 2022h: Position data for position mode

Index	2022_h
Brief description	motion task parameter
Object code	RECORD
Number of elements	12

Used to enter all the parameters that refer to a direct motion task or a motion task which is stored in the controller. (See ASCII command *ORDER*)

Sub-Indexes:

Sub-index	01_h
Brief description	position
Mode	position
Access	rw
PDO mapping	possible (pre-mapped to selectable RPDO 34)
Data type	INTEGER32
Unit	increments or µm
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
EEPROM	no
Default value	0

Used to enter the target position (absolute motion task) or the distance to be moved (relative motion task). The choice is defined by Bit 0 of the motion task type. Bit 13 of the motion task type defines whether the given value is to be interpreted as an increment or as an SI value.

Sub-index	02_h
Brief description	weighted velocity setpoint
Mode	position
Access	rw
PDO mapping	possible (pre-mapped to selectable RPDO 34)
Data type	INTEGER16
Unit	increments/s or µm/s
Value range	-32768 ... 32767
EEPROM	no
Default value	0

Used to enter the target velocity for motion tasks. It is weighted by Sub-index $0D_h$.
If the value is defined as an SI unit by motion task type Bit 13 = 1, then the incremental velocity vi is given by

$$v_i = v_{SI} \times \frac{PGEARO}{PGEARI \times 4000},$$

whereby PGEARO (Object 2020_h, Sub-index 08_h) contains the number of increments that must be moved to cover a distance of PGEARI (Object 2020_h, Sub-index 09_h). It must be noted here that one revolution of the motor corresponds to 2^{20} increments = 1048576.

Sub-index	03_h
Brief description	motion task type
Mode	pp
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Unit	—
Value range	0 ... $FFFF_h$
EEPROM	no
Default value	0

Used to set motion parameters for the motion task. The interpretation of the bits is shown in the following tables.

Bit	Value	Meaning
0	$0x0001_h$	Bit for type of relative/absolute motion block (see table 2)
1	$0x0002_h$	Bit for type of relative motion block (see table 2)
2	$0x0004_h$	Bit for type of relative motion block (see table 2)
3	$0x0008_h$	=0 No following (next) motion block available. Drive stops when target position has been reached. =1 Following motion block available. Next motion block starts automatically when target position has been reached. The number of the next motion block is given by the command *O_FN*.
4	$0x0010_h$	Bit for type of next motion block (see table 3)
5	$0x0020_h$	Bit for type of next motion block (see table 3)
6	$0x0040_h$	Bit for type of next motion block (see table 3)
7	$0x0080_h$	Bit for type of next motion block (see table 3)
8	$0x0100_h$	Bit for type of next motion block (see table 3)
9	$0x0200_h$	reserved
10	$0x0400_h$	reserved
11	$0x0800_h$	reserved
12	$0x1000_h$	=0 Acceleration and deceleration are given as acceleration/deceleration time (in msec) from 0 to target velocity (or reversed). =1 Acceleration and deceleration are given mm/sec². (see also commands: *O_ACC1, O_ACC2, O_DEC1, O_DEC2*).
13	$0x2000_h$	=0 Target position and target velocity are interpreted as increments. There is no conversion of the values. =1 Before the start of the motion block, the target position and target velocity are converted into increments. The conversion uses the parameters PGEARI and PGEARO. (see also commands: *O_S, O_V, PGEARI, PGEARO*)
14	$0x4000_h$	=0 When the motion block starts, the velocity for the motion block is taken as the target velocity. =1 The target velocity is provided as an analog value (SW1) at the start of the motion block. The analog SW1 value is read at the start of the motion block, and taken as the target velocity. (scaling: 10V=VSCALE1). The sign of the SW1 voltage is ignored.
15	$0x8000_h$	Bit 3 for type of relative motion block (see separate table)

Type of relative/absolute motion block

Bit 0	Bit 1	Bit 2	Bit 15	Meaning
0	x	x	x	Absolute motion block: the position given in the motion block is taken as the target position.
1	0	0	x	Relative motion block: the position given in the motion block is taken as the distance to be moved. The target is calculated according to the status of the IN-POSITION signal: IN-POSITION=1: new target position = last target position + movement distance IN-POSITION=0: new target position = actual position + movement distance
1	1	0	x	Relative motion block: the position given in the motion block is taken as the distance to be moved. new target position = last target position + movement distance
1	0	1	x	Relative motion block: the position given in the motion block is taken as the distance to be moved. new target position = actual position + movement distance
1	1	1	0	Relative motion block: the position given in the motion block is taken as the distance to be moved. new target position = positive latch position + movement distance (see also *LATCH32* command)
1	1	1	1	Relative motion block: the position given in the motion block is taken as the distance to be moved. new target position = negative latch position + movement distance (see also *LATCH32N* command)

Type of next motion block

Bit 4 NOBRAKE	Bit 5 FOL_IO	Bit 6 HI/LO	Bit 7 FTIME	Bit 8 VTARG	Meaning
0	0	0	0	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. Afterwards, the next motion block is started.
1	0	0	0	0	Flying changeover to the next motion task at the target position. The drive moves with the target velocity up to the target position of the first motion block. It then changes over to the next motion block at full speed.
1	0	0	0	1	Flying changeover to the next motion task at the target position. The changeover point for the next motion task is advanced, so that the drive already has the target velocity for the next motion block when it reaches the target position of the first motion block.
0	1	0	0	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. The next motion block is started when the pre-defined input (function INxMODE=15) goes *LOW*.
0	1	1	0	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. The next motion block is started when the pre-defined input (function INxMODE=15) goes *HIGH*.
0	0	0	1	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. The next motion block is started when the pre-programmed delay time (O_FT) has elapsed.
0	1	0	1	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. The next motion block is started when the pre-defined input (function INxMODE=15) goes *LOW*, or the pre-programmed delay time (O_FT) has elapsed.
0	1	1	1	0	Changeover to next motion task with braking. The drive brakes to the target position of the first motion block. The next motion block is started when the pre-defined input (function INxMODE=15) goes *HIGH* or the pre-programmed delay time (O_FT) has elapsed.

Sub-index	04_h
Brief description	trajectory
Unit	—
Access	rw
PDO mapping	possible (pre-mapped to selectable RPDO 33)
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Under development

Sub-index	05_h
Brief description	motion task number
Unit	—
Access	rw
PDO mapping	possible (pre-mapped to selectable RPDO 35)
Data type	UNSIGNED16
Value range	1 ... 180, 129 ... 255
Default value	0

This Index gives the number of the selected motion task. Motion tasks 1 to 180 are EEPROM motion blocks, and motion tasks 192 to 255 are RAM motion blocks. The RAM motion blocks are loaded with the first 64 EEPROM motion blocks when the servo amplifier is switched on or reset. Motion block 0 is also a RAM motion block, which is used as a copy buffer for motion tasks, and also to hold the motion task data for a direct motion task (RPDO 34).

Sub-index	06_h
Brief description	run-up time (acceleration)
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 65535
Default value	0

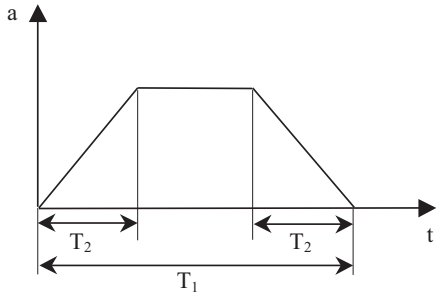
Used to define the total time taken to run up to the target velocity for the motion task. The value chosen for Sub-index 08_h sets the form of the acceleration ramp.

Sub-index	07_h
Brief description	braking time (deceleration)
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 65535
Default value	0

Used to define the total time taken to brake down to velocity 0 (standstill) at the target position. The value chosen for Sub-index 09_h sets the form of the deceleration ramp.

Sub-index	08_h
Brief description	jolt (acceleration) limiting (under development)
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 65535
Default value	0

This Index sets the form of the acceleration ramp. The value chosen must be smaller than half of the run-up time (Sub-index 06_h). The following diagram illustrates the relationships:



T_1 corresponds to the Sub-index 06_h and T_2 to Sub-index 08_h.
If $T_2 = 0$, then a trapezoidal acceleration ramp is used.
If $T_2 = \frac{T_1}{2}$ then an approximately $\sin^2$ ramp is used.

Sub-index	09_h
Brief description	jolt (deceleration) limiting (under development)
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	0 ... 65535
Default value	0

This Index defines the form of the braking/deceleration ramp. The value chosen must be smaller than half of the braking time (Sub-index 07_h). Jolt (deceleration) limiting functions in the same way as jolt (acceleration)

Sub-index	$0A_h$
Brief description	number of the next motion task
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	0 ... 180, 192 ... 255
Default value	0

Used to set the number of the next motion task. The setting of Sub-index 03_h, Bit 3, decides whether this is used to continue.

Sub-index	$0B_h$
Brief description	start delay for next motion task
Unit	ms
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	1 ... 65535
Default value	0

This Object is used to set a delay time for the start of the next motion task. To do this, the function must be enabled through Sub-index 03_h, Bit 7.

Sub-index	$0C_h$
Brief description	copy a motion task
Unit	—
Access	wo
PDO mapping	not possible
Data type	2 x UNSIGNED16
Value range	0 ... 180, 192... 255
Default value	0, 0

This Object can be used to copy motion tasks. The number that appears first in the CAN telegram defines the source motion task, the following number defines the target motion task.

Sub-index	$0D_h$
Brief description	velocity weighting factor
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED16
Value range	0 ... 65535
Default value	1

This Object sets a multiplying factor for the velocity given in the RPDO motion block.

Sub-index	$0E_h$
Brief description	velocity for direct motion task
Unit	increments / 250µs, or dependent on resolution
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

This Object defines the velocity for the direct motion task (motion block 0). The type of motion task then determines whether the velocity is evaluated in increments or in SI units.

Sub-index	$0F_h$
Brief description	Position setpoint CAN-Master-Slave
Unit	Increments / 250µs or depending on resolutio
Access	rw
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}-1)..(2^{31}-1)$
Default value	0

This Object is used to specify the target position in position controller increments (2^{20} increments/motor revolution) in Electronic Gear mode. This Object is located within Rx-PDO 41, whose definition is fixed.

4.9 Object 2050h: Auxiliary variable for digital inputs

Index	2050_h
Brief description	Auxiliary variable for digital inputs
Object code	ARRAY
Number of elements	6

Sub-Indexes:

Sub-index	$01_h...04_h$
Brief description	Trigger variable inputs 1...4
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

Sub-index	05_h
Brief description	Auxiliary trigger variable for electronic gear
Unit	—
Access	wo
PDO mapping	possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

In Electronic Gear mode (0xF7), this Object is used to specify the distance for the purpose of synchronizing the slave drive with the master speed, if the ENGAGE parameter is set to a value of 3. The distance is specified in increments (one motor revolution corresponds to 2^{20} increments).

4.10 Latch function

4.10.1 Object 2026h: Latch enable

Index	2026_h
Brief description	enable the latch function for CAN
Object code	RECORD
Number of elements	1

Sub-index:

Sub-index	01_h
Brief description	enable the latch function for CAN
Unit	—
Access	rw
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0, 1
Default value	1

SERVOSTAR 400/600 provides the option of acquiring an actual position with high precision (acquisition time < 1 µs) via a latch input (input 2, IN2MODE 26, can also be configured through Object 3565_h, Sub-index 01_h). Object 2026_h can be used to decide whether a latching pulse is reported via CAN. A value of 0 means it is inhibited.

4.10.2 Object 2082h: 32/24-bit Latch positive

On a positive edge at digital input 1, these Objects supply the latched position in 32-bit format.

Index	2082_h
Brief description	32/24-bit Latch positive
Object code	RECORD
Number of elements	3

Sub-index:

Sub-index	01_h
Brief description	32 Bit Latch positive
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

Sub-index	02_h
Brief description	24 Bit Latch positive
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER24
Value range	INTEGER24
Default value	0

4.10.3 Object 2083h: 32/24-bit Latch negative

On a negative edge at digital input 1, these Objects supply the latched position in 32-bit format.

Index	2083_h
Brief description	32/24-bit Latch negative
Object code	RECORD
Number of elements	3

Sub-index:

Sub-index	01_h
Brief description	32 Bit Latch negative
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

Sub-index	02_h
Brief description	24 Bit Latch negative
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER24
Value range	INTEGER24
Default value	0

4.10.4 Object 2084h: 16-bit Latch positive

On a positive edge at digital input 1, these Objects supply the latched position in 16-bit format.

Index	2084_h
Brief description	16-bit Latch positive
Object code	RECORD
Number of elements	2

Sub-index:

Sub-index	01_h
Brief description	16 Bit Latch positive
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER16
Value range	INTEGER16
Default value	0

4.10.5 Object 2085h: 16-bit Latch negative

On a negative edge at digital input 1, these Objects supply the latched position in 16-bit format.

Index	2085_h
Brief description	16-bit Latch negative
Object code	RECORD
Number of elements	2

Sub-index:

Sub-index	01_h
Brief description	16-bit Latch negative
Unit	—
Access	rw
PDO mapping	not possible
Data type	INTEGER16
Value range	INTEGER16
Default value	0

4.10.6 Object 2087h: Latch Positions Digital Input 1

The Objects covered by Index 2087 are used for reading out the latch positions from digital input 1 and for enabling the latch functions of digital inputs 1 and 2.

Index	2087_h
Brief description	Latch Positions Digital Input 1
Object code	RECORD
Number of elements	4

Sub-index:

Sub-index	01_h
Brief description	Latch Positions Digital Input 1, positive
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

Sub-index	02_h
Brief description	Latch Positions Digital Input 1, negative
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

Sub-index	03_h
Brief description	Latch Enable for Inputs 1 and 2
Unit	—
Access	rw
PDO mapping	possible
Data type	UNSIGNED8
Value range	0...15
Default value	0

This mappable Object is used to enable the positive/negative edges of latch inputs 1/2. For this purpose, the latch function (INxMODE = 26) must be configured for these inputs. The enable bits are assigned to the function as shown below:

Bit	Enable
0	Enable positive latch event at digital input 1
1	Enable negative latch event at digital input 1
2	Enable positive latch event at digital input 2
3	Enable negative latch event at digital input 2

4.11 Manufacturer specific values

4.11.1 Object 2070h: Actual values

Index	2070_h
Brief description	actual values
Object code	RECORD
Number of elements	16

This Index makes relevant actual values available from the SERVOSTAR 400/600.

Sub-Indexes:

Sub-index	01_h
Brief description	actual position
Unit	—
Access	ro
PDO mapping	possible (pre-mapped to selectable TPDO 22, TPDO 32)
Data type	UNSIGNED32
Value range	0 ... 16777215
Default value	0

Used to read the motor position within 16 turns/revolutions. One turn is resolved with 20 bits, in increments.

1 turn $\Rightarrow 2^{20}$ increments $\Rightarrow$1048576 increments

Sub-index	02_h
Brief description	actual speed
Unit	min^{-1}
Access	ro
PDO mapping	possible (pre-mapped to selectable TPDO 22, TPDO 32)
Data type	UNSIGNED32
Value range	0 ... 1677215
Default value	0

Used to read the motor speed. The actual speed is given by:

$n[min^{-1}] = {}^{1875}\!/_{262144} \times$(speed value as read)

Sub-index	03_h
Brief description	incremental actual position
Unit	—
Access	ro
PDO mapping	possible (pre-mapped to selectable TPDO 33)
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

This Index can be used to read the incremental value for the actual position. One turn is resolved with 20 bits, in increments.

1 turn $\Rightarrow 2^{20}$ increments $\Rightarrow$1048576 increments

Sub-index	$04l_h$
Brief description	read the 16-bit position latch
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER16
Value range	$-(2^{15})$ ... $(2^{15}-1)$
Default value	0

Used to read the 16-bit position stored in the latch. The position is given as increments within a single turn. The output is not affected by the gearing factor or the Factor Groups.

Sub-index	05_h
Brief description	read the 32-bit position latch
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}) ... (2^{31}-1)$
Default value	0

Used to read the 32-bit position stored in the latch. The position is given as increments within a single turn. The output is not affected by the gearing factor or the Factor Groups.

Sub-index	06_h
Brief description	SI actual position
Unit	µm
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}-1)..(2^{31}-1)$
Default value	0

This Index can be used to read the actual position in SI units. The ratio of distance actually traveled to motor revolutions is given by

$$S_{SI} = S_{Inkr} \times PGEARI / PGEARO$$

whereby PGEARO (Object 2020_h, Sub-index 08_h) contains the number of increments that must moved to cover a distance of PGEARI (Object 2020_h, Sub-index 09_h). It must be noted that one turn of the motor corresponds to 2^{20} = 1048576 increments.

Sub-index	07_h
Brief description	SI actual velocity
Unit	µm/s
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1) ... (2^{31}-1)$
Default value	0

This Index can be used to read the actual speed in SI units.

Sub-index	08_h
Brief description	lag/following error
Unit	µm
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1) ... (2^{31}-1)$
Default value	0

This Index can be used to read out the momentary lag error, measured in SI units.

Sub-index	09_h
Brief description	rms (effective) current
Unit	mA
Access	ro
PDO mapping	not possible
Data type	UNSIGNED32
Value range	0 ... 2 * rated current [mA]
Default value	0

The Index can be used to read out the momentary rms (effective) current.

Sub-index	$0A_h$
Brief description	speed
Unit	min^{-1}
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the momentary speed measurement.

Sub-index	$0B_h$
Brief description	heat sink temperature
Unit	°C
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the heat sink temperature.

Sub-index	$0C_h$
Brief description	internal temperature
Unit	°C
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read the internal temperature of the servo amplifier.

Sub-index	$0D_h$
Brief description	DC-link (DC-bus) voltage
Unit	V
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the momentary DC-bus voltage.

Sub-index	$0E_h$
Brief description	regen power
Unit	W
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the momentary regen power.

Sub-index	$0F_h$
Brief description	I^2t loading
Unit	%
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the I^2t loading.

Sub-index	10_h
Brief description	operating time
Unit	min
Access	ro
PDO mapping	not possible
Data type	INTEGER32
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	0

Used to read out the operating time (hours) counter of the servo amplifier.

Sub-index	11_h
Brief description	enhanced status for TPDO 33
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED32
Value range	0 ... $(2^{32}-1)$
Default value	—

Description:

Bit	Value	Description
0	0x00000001	status input 1
1	0x00000002	status input 2
2	0x00000004	status input 3
3	0x00000008	status input 4
4	0x00000010	reserved
5	0x00000020	reserved
6	0x00000040	reserved
7	0x00000080	reserved
8	0x00000100	reserved
9	0x00000200	reserved
10	0x00000400	reserved
11	0x00000800	reserved
12	0x00001000	reserved
13	0x00002000	reserved
14	0x00004000	reserved
15	0x00008000	reserved
16	0x00010000	task active (position control)
17	0x00020000	homing/reference point set
18	0x00040000	home position
19	0x00080000	In-Position
20	0x00100000	position latched
21	0x00200000	free
22	0x00400000	signal position 1
23	0x00800000	signal position 2
24	0x01000000	signal position 3
25	0x02000000	signal position 4
26	0x04000000	initialization finished
27	0x08000000	free
28	0x10000000	motor standstill
29	0x20000000	safety relay
30	0x40000000	output stage enabled
31	0x80000000	error present

Sub-index	12_h
Brief description	position absolut encoder in SI units
Unit	user units
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}-1)..(2^{31}-1)$
Default value	—

Assuming that an absolute encoder has been configured in the device (EXTPOS 1, 2 or 3), this Object can be used to determine the encoder position. Scaling is defined by parameters PGEARI, PGEARO, ENCIN, and EXTMUL.

Sub-index	13_h
Brief description	Internal Position Setpoint
Unit	internal position counts
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}-1)..(2^{31}-1)$
Default value	—

This Object can be used to read the position setpoint used for internal position control. The value is output in internal position counts.

Sub-index	14_h
Brief description	Analog Input 1 actual value
Unit	mV (scaled)
Access	ro
PDO mapping	possible
Data type	INTEGER16
Value range	$-(2^{15}-1)..(2^{15}-1)$
Default value	—

This Object supplies a scaled value for the input voltage at analog input 1. In this context, 10 V corresponds to 8192 increments.

Sub-index	15_h
Brief description	Analog Input 2 actual value
Unit	mV (scaled)
Access	ro
PDO mapping	possible
Data type	INTEGER16
Value range	$-(2^{15}-1)..(2^{15}-1)$
Default value	—

This Object supplies a scaled value for the input voltage at analog input 2. In this context, 10 V corresponds to 8192 increments.

Sub-index	16_h
Brief description	Error register
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED32
Value range	$0..(2^{32}-1)$
Default value	0

This Object supplies the content of error register ERRCODE. Below is a list of possible error messages:

Bit	Number	Designation	Explanation
0	F01*	heat sink temperature	heat sink temperature too high limit is set by manufacturer to 80°C (176°F)
1	F02*	overvoltage	overvoltage in DC bus link limit depends on the mains supply voltage
2	F03*	following error	message from the position controller
3	F04	feedback	cable break, short circuit, short to ground
4	F05*	undervoltage	undervoltage in DC bus link limit is set by manufacturer to 100V
5	F06	motor temperature	motor temperature too high limit is set by manufacturer to145°C (293°F)
6	F07	aux. voltage	internal aux. voltage not OK
7	F08*	overspeed	motor running away, speed is too high
8	F09	EEPROM	checksum error
9	F10	Flash-EPROM	checksum error
10	F11	brake	cable break, short circuit, short to ground
11	F12	motor phase	motor phase missing (cable break or similar)
12	F13*	internal temperature	internal temperature too high
13	F14	output stage	fault in the output stage
14	F15	I²t max	I²t max. value exceeded
15	F16*	supply - BTB/RTO	2 or 3 phases missing in the supply feed
16	F17	A/D converter	error in the analog-digital conversion, usually caused by excessive EMI
17	F18	regen	regen circuit faulty or incorrect setting
18	F19*	supply phase	a supply phase is missing (can be switched off for 2-phase operation)
19	F20	Slot fault	cable break, short circuit, short to ground
20	F21	Handling fault	motor phase missing (cable break or similar)
21	F22	Short circuit to earth (ground)	SERVOSTAR 640/670 only: short circuit to earth (ground)
22	F23	CAN Bus off	CAN Bus total communication error
23	F24	Warning	Warning displays as error
24	F25	Commutation error	Encoder systems only
25	F26	Limit switch	Homing error (hardware limit switch reached)
26	F27	AS-option	operational error with -AS- option, input for AS-Enable and ENABLE have been set at the same time
27	F28	reserved	reserved
28	F29	Fieldbus-Sync	Fieldbus not synchronized
29	F30	Emerg. Stop Timeout	Emerg. Stop Timeout
30	F31	reserved	reserved
31	F32	System error	system software not responding correctly

* = These error messages can be cancelled by the ASCII command CLRFAULT, without executing a reset. If only these errors are present, and the RESET button or the I/O-function RESET is used, the CLRFAULT command is also all that is carried out.

Sub-index	17_h
Brief description	internal velocity setpoint
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	$-(2^{31}-1)..(2^{31}-1)$
Default value	—

This Object can be used to read out the internal speed setpoint. In this context, an internal speed count corresponds to a position count/250 microseconds. Thus, the speed in rpm is calculated as follows:

$$v_{rpm} = \frac{v_{incr} \cdot 4000}{2^{20}} \cdot 60$$

Sub-index	18_h
Brief description	Master velocity (electronic gear)
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED32
Value range	$0..(2^{32}-1)$
Default value	—

This Object supplies the speed resulting from the master-encoder pulses read in on the electronic gear (Mode 0xF7).

Sub-index	19_h
Brief description	control variable (electronic gear)
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER8
Value range	$-(2^7-1)..(2^7-1)$
Default value	0

This Object is used to display the status/transition associated with synchronization/decoupling within the electronic gear. The following values are defined (only when ENGAGE = 1 or 3):

Value	Description
0	No synchronization
1	Synchronization with master speed using ramp ACCR
2	Uncoupling using ramp DECR
3	Travel synchronized with master
4	Synchronization over distance defined by Object 2050 sub 5

4.11.2 Objekt 6077h: Torque actual value

The torque actual value refers to the current torque in the motor. The scaling equates to 1/1000 of the rated torque.

Index	6077_h
Brief description	Torque actual value
Object code	VAR
Number of elements	1

Access	ro
PDO mapping	possible
Value range	INTEGER16
Default value	0

4.11.3 Objekt 60C2h: Interpolation time period

The interpolation time period is used for the PLL (phase-locked loop) synchronized positioning mode.

The time unit (Subindex 1) is determined on the basis of $10^{\text{interpolation time index}}$s; only multiples of 1 ms are permitted. The two values define the internal ASCII parameter PTBASE (multiples of 250 µs).

In order to make use of PLL synchronized mode, SYNCSRC must be set to 3 (synchronization over CAN) or FPGA set to 3 (SERVOSTAR 600 only).

Index	$60C2_h$
Brief description	Interpolation time period
Object code	RECORD
Number of elements	3

Sub-Indexes:

Sub-index	01h
Brief description	Interpolation time units
Access	rw
PDO mapping	possible
Value range	UNSIGNED8
Default value	1

Sub-index	02h
Brief description	Interpolation time index
Access	rw
PDO mapping	possible
Value range	-123 ... 63
Default value	-3

4.12 Freely available, mappable PLC variables, Objects 2030h / 2090h

Parameters DPRVAR1 .. DPRVAR16 enable status variables to be exchanged between the CAN bus and internal PLC sequential program. Two sets of eight Objects - one for write-only (WO) and one for read-only - are provided for this purpose (these Objects can also be mapped in PDOs).

4.12.1 Object 2030h: DP-Ram-Variables 9-16 (write only)

Index	2030_h
Brief description	DP-Ram Variables, write only (PDO)
Object code	ARRAY
Number of elements	9

Sub-Indexes:

Sub-index	01_h...08h
Brief description	DP-Ram Variable 9...16
Unit	—
Access	rw
PDO mapping	possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

4.12.2 Object 2090h: DP-Ram-Variables 1-8 (read only)

Index	2090_h
Brief description	DP-Ram Variablen, nur lesbar (PDO)
Object code	ARRAY
Number of elements	9

Sub-Indexes:

Sub-index	01_h...08h
Brief description	DP-Ram Variable 1...8
Unit	—
Access	ro
PDO mapping	possible
Data type	INTEGER32
Value range	INTEGER32
Default value	0

4.13 Dummy Variables, Objects 2031h / 2071h

These Objects can be used for gaps in Process Data Objects (PDOs). Possible applications include a setpoint PDO at multiple nodes or a situation where actual values have to be located at specific points in the PDO and the other PDO sections have to be executed quickly. These Objects can be used to plug gaps that are 1 to 4 bytes wide.

4.13.1 Object 2031h: Dummy variables for mapping (WO)

Index	2031_h
Brief description	Dummy variables for mapping
Object code	RECORD
Number of elements	5

Sub-Indexes:

Sub-index	01_h
Brief description	Write-only dummy 8 bit
Unit	—
Access	wo
PDO mapping	possible
Data type	UNSIGNED8
Value range	UNSIGNED8
Default value	0

Sub-index	02_h
Brief description	Write-only dummy 16 bit
Unit	—
Access	wo
PDO mapping	possible
Data type	UNSIGNED16
Value range	UNSIGNED16
Default value	0

Sub-index	03_h
Brief description	Write-only dummy 24 bit
Unit	—
Access	wo
PDO mapping	possible
Data type	UNSIGNED24
Value range	UNSIGNED24
Default value	0

Sub-index	04_h
Brief description	Write-only dummy 32 bit
Unit	—
Access	wo
PDO mapping	possible
Data type	UNSIGNED32
Value range	UNSIGNED32
Default value	0

4.13.2 Object 2071h: Dummy variables for mapping (RO)

Index	2027_h
Brief description	Dummy variables for mapping
Object code	RECORD
Number of elements	5

Sub-Indexes:

Sub-index	01_h
Brief description	Read-only dummy 32 bit
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED8
Value range	UNSIGNED8
Default value	0

Sub-index	02_h
Brief description	Read-only dummy 32 bit
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED16
Value range	UNSIGNED16
Default value	0

Sub-index	03_h
Brief description	Read-only dummy 32 bit
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED24
Value range	UNSIGNED24
Default value	0

Sub-index	04_h
Brief description	Read-only dummy 32 bit
Unit	—
Access	ro
PDO mapping	possible
Data type	UNSIGNED32
Value range	UNSIGNED32
Default value	0

4.14 Profile Velocity Mode (pv) (DS402)

4.14.1 General Information

The *profile velocity* mode enables the processing of velocity setpoints and the associated accelerations.

4.14.2 Objects that are defines in this section

Index	Object	Name	Type	Access
$606C_h$	VAR	velocity_actual_value	INTEGER32	ro
$60FF_h$	VAR	target_velocity	INTEGER32	rw

4.14.3 Objects that are defines in other sections

Index	Object	Name	Type	Section
6040_h	VAR	control word	INTEGER16	dc (⇒ 4.4)
6041_h	VAR	status word	UNSIGNED16	dc (⇒ 4.4)
6063_h	VAR	position_actual_value*	INTEGER32	pc (⇒ 4.15)
6083_h	VAR	profile_acceleration	UNSIGNED32	pp (⇒ 4.17)
6084_h	VAR	profile_deceleration	UNSIGNED32	pp (⇒ 4.17)
6086_h	VAR	motion_profile_type	INTEGER16	pp (⇒ 4.17)
6094_h	ARRAY	velocity_encoder_factor	UNSIGNED32	fg (⇒ 4.5)

4.14.4 Object Description

4.14.4.1 Object 606Ch: velocity_actual_value* (DS402)

The Object *velocity_actual_value* represents the actual speed. The scaling of the value depends on the factor *velocity_encoder_resolution* (Object 6096_h).

Index	$0x606C_h$
Name	velocity_actual_value
Object code	VAR
Data type	INTEGER32
Mode	pv
Access	ro
PDO mapping	possible
Unit	velocity units
Value range	(-2^{31}) ... $(2^{31}-1)$
Default value	—
EEPROM	no

4.14.4.2 Object 60FFh: target_velocity (DS402)

The speed setpoint (*target_velocity*) represents the setpoint for the ramp generator.
The scaling of this value depends on the factor *velocity_encoder_resolution* (Object 6096_h).

Index	$0x60FF_h$
Name	target_velocity
Object code	VAR
Data type	INTEGER32
Mode	pv
Access	rw
PDO mapping	possible
Unit	increments
Value range	(-2^{31}) ... $(2^{31}-1)$
Default value	—
EEPROM	no

4.15 Position Control Function (pc) (DS402)

4.15.1 General Information

This section describes the actual position values that are associated with the position controller of the drive. They are used for the *profile position* mode.

4.15.2 Objects that are defined in this section

Index	Object	Name	Type	Access
6063_h	VAR	position_actual_value*	INTEGER32	ro
6064_h	VAR	position_actual_value	INTEGER32	ro

4.15.3 Objects that are defined in other sections

Index	Object	Name	Type	Section
$607A_h$	VAR	target_position	INTEGER32	pp (⇒ 4.17)
$607B_h$	VAR	position_range_limit	INTEGER32	pp (⇒ 4.17)
$607C_h$	VAR	home-offset	INTEGER32	hm (⇒ 4.16)
6093_h	VAR	position_factor	UNSIGNED32	fg (⇒ 4.5)
6094_h	ARRAY	velocity_encoder_factor	UNSIGNED32	fg (⇒ 4.5)
6096_h	ARRAY	acceleration_factor	UNSIGNED32	fg (⇒ 4.5)
6040_h	VAR	control word	INTEGER16	dc (⇒ 4.4)
6041_h	VAR	status word	UNSIGNED16	dc (⇒ 4.4)

4.15.4 Object Decription

4.15.4.1 Object 6063h: position_actual_value* (DS402)

The Object *position_actual_value* provides the momentary actual position in increments. The resolution is 16 bits or 20 bits per turn (see *PRBASE* command).

Index	$0x6063_h$
Name	position_actual_value
Object code	VAR
Data type	INTEGER32
Mode	pc, pp
Access	rw
PDO mapping	possible
Unit	increments (1 turn = 16 bit / 20 bit)(see *PRBASE*)
Value range	(-2^{31}) ... $(2^{31}-1)$
Default value	—
EEPROM	no

4.15.4.2 Object 6064h: position_actual_value (DS402)

The Object *position_actual_value* provides the actual position. The resolution (manufacturer-specific units: see Objects 2020_h 08_h/09_h or as per drive profile DSP402: see Object $607A_h$) can be altered by the gearing factors of the position controller.

Note: this Object should not be defined (mapped) in a synchronous TPDO.

Index	$0x6064_h$
Name	position_actual_value
Object code	VAR
Data type	INTEGER32
Mode	pc, pp
Access	rw
PDO mapping	possible
Unit	position units
Value range	(-2^{31}) ... $(2^{31}-1)$
Default value	—
EEPROM	no

4.16 Homing Mode (hm) (DS402)

4.16.1 General Information

This section describes the various parameters which are required to define a homing mode.

4.16.2 Objects that are defined in this section

Index	Object	Name	Type	Access
$607C_h$	VAR	home_offset	INTEGER32	rw
6098_h	VAR	homing_method	INTEGER8	rw
6099_h	ARRAY	homing_speeds	UNSIGNED32	rw
$609A_h$	VAR	homing_acceleration	UNSIGNED32	rw

4.16.3 Objects that are defined in other sections

Index	Object	Name	Type	Section
6040_h	VAR	control word	INTEGER16	dc (⇒ 4.4)
6041_h	VAR	status word	UNSIGNED16	dc (⇒ 4.4)

4.16.4 Object Description

4.16.4.1 Object 607Ch: home_offset (DS402)

The reference offset (*home_offset*) is the difference between the zero position for the application and the zero point of the machine. All subsequent absolute motion tasks take account of the reference offset.

Index	$0x607C_h$
Name	home_offset
Object code	VAR
Data type	INTEGER32
Mode	hm
Access	rw
PDO mapping	possible
Unit	user-defined
Value range	(-2^{31}) ... $(2^{31}-1)$
Default value	0
EEPROM	yes

4.16.4.2 Object 6098h: homing_method (DS402)

Index	$0x6098_h$
Name	homing_method
Object code	VAR
Data type	INTEGER8
Mode	hm
Access	rw
PDO mapping	possible
Unit	position units
Value range	-128 ... 127
Default value	0
EEPROM	yes

The following homing methods are supported:

Method as per DSP402	Brief description: Homing	ASCII command
-128..-4	reserved	—
-3	move to mechanical stop, with zeroing	NREF = 7
-2	set reference point at present position, allowing for lag/following error	NREF = 6
-1	homing within a single turn (direction of rotation depends on distance)	NREF = 5, DREF= 2
0	reserved	—
1	homing to negative limit switch, with zeroing, negative direction of motion	NREF = 2, DREF= 0
2	homing to positive limit switch, with zeroing, positive direction of motion	NREF = 2, DREF= 1
3..7	not supported	—
8	homing to reference switch, with zeroing, positive direction of motion	NREF = 1, DREF= 1
9..11	not supported	—
12	homing to reference switch, with zeroing, negative direction of motion	NREF = 1, DREF= 0
13..14	not supported	—
15..16	reserved	—
17	homing to negative limit switch, without zeroing, negative direction of motion	NREF = 4, DREF= 0
18	homing to negative limit switch, without zeroing, positive direction of motion	NREF = 4, DREF= 1
19..23	not supported	—
24	homing to reference switch, without zeroing, positive direction of motion	NREF = 3, DREF= 1
25..27	not supported	—
28	homing to reference switch, without zeroing, negative direction of motion	NREF = 3, DREF= 0
29..30	not supported	—
31..32	reserved	—
33	homing within a single turn negative direction of rotation	NREF = 5, DREF= 0
34	homing within a single turn positive direction of rotation	NREF = 5, DREF= 1
35	set reference point at present position	NREF = 0
36..127	reserved	—

4.16.4.2.1 Description of the homing methods

Choosing a homing method by writing a value to homing_method (Object 6098_h) will clearly establish:

- the homing signal (P-Stop, N-Stop, reference switch)
- the direction of actuation

and where appropriate

- the position of the index pulse.

The reference position is give by the reference offset (Object $607C_h$). The manufacturer-specific parameter *ENCZERO* (Object 3537_h, Sub-index 01_h) can be used to adapt the initial position of the motor for homing to match the index pulse for homing with zeroing.

A detailed description of the types of homing movement can be found in the description of the setup software DRIVE.EXE.

4.16.4.3 Object 6099h: homing_speeds (DS402)

Index	$0x6099_h$
Name	homing_speeds
Object code	ARRAY
Number of elements	1
Data type	UNSIGNED32

Sub-index	01_h
Brief description	speed_during_search_for_switch
Mode	hm
Access	rw
PDO mapping	possible
Unit	velocity units
Value range	0 ... (2^{32}-1)
EEPROM	yes
Default value	2^{20}

4.16.4.4 Object 609Ah: homing_acceleration (DS402)

Index	$0x609A_h$
Name	homing_acceleration
Object code	VAR
Data type	UNSIGNED32
Mode	hm
Access	rw
PDO mapping	possible
Unit	acceleration units
Value range	0 ... (2^{32}-1)
Default value	0
EEPROM	yes

4.16.5 Homing Mode Sequence

The homing movement is started by setting Bit 4 (positive edge). The successful conclusion is indicated by Bit 12 in the status word (see Object 6041_h). Bit 13 indicates that an error occurred during the homing movement. In this case, the error code must be evaluated (error register: Objects 1001_h, 1003_h, manufacturer status: Object1002_h).

Bit 4	Meaning
0	homing inactive
0 ⇒ 1	start homing movement
1	homing active
1 ⇒ 0	interruption of homing movement

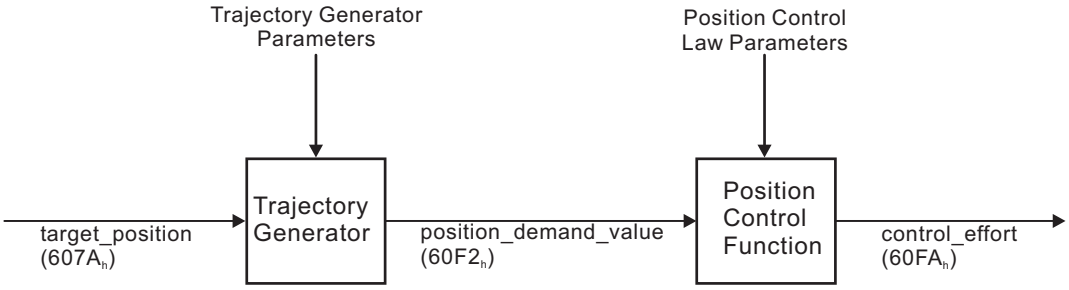
Bit 13	Bit 12	Meaning
0	0	reference point not set, or homing movement not yet finished
0	1	reference point set, homing movement finished
1	0	homing movement could not be successfully concluded (lag error)
1	1	impermissible state

4.17 Profile Position Mode (pp)

4.17.1 General Information

The overall structure for this mode is shown in this figure.:

The special handshake procedure for the control word and status word is described in Section 4.17.4.1.



4.17.2 Objects that are defined in this section

Index	Object	Name	Type	Access
$607A_h$	VAR	target_position	INTEGER32	rw
$607B_h$	ARRAY	position_range_limit	INTEGER32	rw
6081_h	VAR	profile_velocity	UNSIGNED32	rw
6083_h	VAR	profile_acceleration	UNSIGNED32	rw
6084_h	VAR	profile_deceleration	UNSIGNED32	rw
6085_h	VAR	quick_stop_deceleration	UNSIGNED32	rw
6086_h	VAR	motion_profile_type	INTEGER16	rw

4.17.3 Objects that are defined in other sections

Index	Object	Name	Type	Section
6040_h	VAR	control word	INTEGER16	dc (⇒ 4.4)
6041_h	VAR	status word	UNSIGNED16	dc (⇒ 4.4)
$605A_h$	VAR	quick_stop_option_code	INTEGER16	dc (⇒ 4.4)
6093_h	ARRAY	position_factor	UNSIGNED32	fg (⇒ 4.5)
6094_h	ARRAY	velocity_encoder_factor	UNSIGNED32	fg (⇒ 4.5)
6097_h	ARRAY	acceleration_factor	UNSIGNED32	fg (⇒ 4.5)

4.17.4 Object Description

4.17.4.1 Object 607Ah: target_position (DS402)

The Object *target_position* defines the target position for the drive. The target position is interpreted as a relative distance or an absolute position, depending on Bit 6 of the control word.
The type of relative movement can be further defined by the manufacturer-specific parameter 2022_h Sub-index 03_h.
The mechanical resolution is set by the gearing factors Object 6093_h Sub-index 01_h and 02_h.

Index	$0x607A_h$
Name	target_position
Object code	VAR
Data type	INTEGER32
Mode	pp
Access	rw
PDO mapping	possible
Unit	user-defined
Value range	$-(2^{31}-1)$ ... $(2^{31}-1)$
Default value	—
EEPROM	no

4.17.4.2 Object 607Bh: position_range_limit (DS402)

The Object *position_range_limit* is used to define the start and end of the range of movement for a modulo axis. The start of the range is defined by Sub-index 01_h *min_position_range_limit* (ASCII *SRND*) and the end by Sub-index 02_h *max_position_range_limit* (ASCII *ERND*). This functionality can only be used after a re-configuration of the amplifier. To do this, Object 2020_h Sub-index 01_h must have the value 2 applied, and then the configuration procedure can be started (⇒ 6.4).

Index	$0x607B_h$
Name	position_range_limit
Object code	ARRAY
Number of elements	2
Data type	INTEGER32

Sub-Indexes:

Sub-index	01_h
Brief description	min_position_range_limit
Mode	pp, pc
Access	rw
PDO mapping	not possible
Unit	position units
Value range	(-2^{31}) ... $(2^{31}-1)$
EEPROM	yes
Default value	-2^{31}

Sub-index	02_h
Brief description	max_position_range_limit
Mode	pp, pc
Access	rw
PDO mapping	not possible
Unit	position units
Value range	(-2^{31}) ... $(2^{31}-1)$
EEPROM	yes
Default value	$2^{31}-1$

4.17.4.3 Object 6081h: profile_velocity (DS402)

The *profile_velocity* is the final velocity that should be reached after the acceleration phase of a motion task. The scaling used depends on the setting of the *velocity_encoder_factor* (Object 6094_h).
The application of the setpoint depends on the operation mode (pp, pv) that is set.

Index	$0x6081_h$
Name	profile_velocity
Object code	VAR
Data type	UNSIGNED32
Mode	pp, pv
Access	rw
PDO mapping	possible
Unit	speed units
Value range	0 ... $(2^{32}-1)$
Default value	10
EEPROM	no

4.17.4.4 Object 6083h: profile_acceleration (DS402)

The acceleration ramp (*profile_acceleration*) is given in units that are defined by the user. The processing and interpretation of the acceleration setpoint can be made in two ways:

- **Internal gearing factors PGEARI = PGEARO (see Objects 2020_h 08_h/09_h, ASCII command PGEARI/PGEARO)**
  The acceleration ramp is interpreted as the acceleration time [ms] or rate of acceleration [incr/s^2] referred to the target velocity (Object 6081_h *profile_velocity*). The scaling for this value depends on Object 6097_h *acceleration_factor*.
  Note: only the acceleration time is supported as a unit at present!
- **Internal gearing factors PGEARI <> PGEARO (see Objects 2020_h 08_h/09_h, ASCII command PGEARI/PGEARO)**
  The acceleration ramp is interpreted as the acceleration time [ms] or rate of acceleration [length units/s^2] referred to the target velocity. The scaling for this value depends on the gearing factors that are set (see description of the ASCII commands *PGEARI* and *PGEARO*) and the basic unit that is set [ms] or [length unit/s^2]. The selection of the basic unit is made through Bit 12 in the control word for the motion task (Object 2022_h03_h, ASCII command *O_C*).

The type of acceleration ramp can be selected as a linear ramp or a sin^2 ramp (see Object 6086_h).

Index	$0x6083_h$
Name	profile_acceleration
Object code	VAR
Data type	UNSIGNED32
Mode	pp
Access	rw
PDO mapping	possible
Unit	acceleration units
Value range	0 ... (2^{15}-1)
Default value	0

4.17.4.5 Object 6084h: profile_adeceleration (DS402)

The braking/deceleration ramp is handled in the same way as the acceleration ramp (see Object 6083_h).

Index	$0x6084_h$
Name	profile_deceleration
Object code	VAR
Data type	UNSIGNED32
Mode	pp
Access	rw
PDO mapping	possible
Unit	acceleration units
Value range	0 ... (2^{15}-1)
Default value	0

4.17.4.6 Object 6086h: motion_profile_type (DS402)

The type of acceleration ramp can be selected by this Object as a linear or as sin^2 ramp.

Index	$0x6086_h$
Name	motion_profile_type
Object code	VAR
Data type	INTEGER16
Mode	pp
Access	rw
PDO mapping	possible
Unit	none
Value range	$(-2^{15})..(2^{15}-1)$
Default value	—
EEPROM	yes

profile code	profile type
-32768 ... -1	manufacturer-specific (not supported)
0	linear (trapezoidal)
1	sin²
2 ... 32767	profile-specific extensions (not supported)

4.17.5 Functional Description

Two different ways to apply *target_positions* to a drive are supported by this device profile.

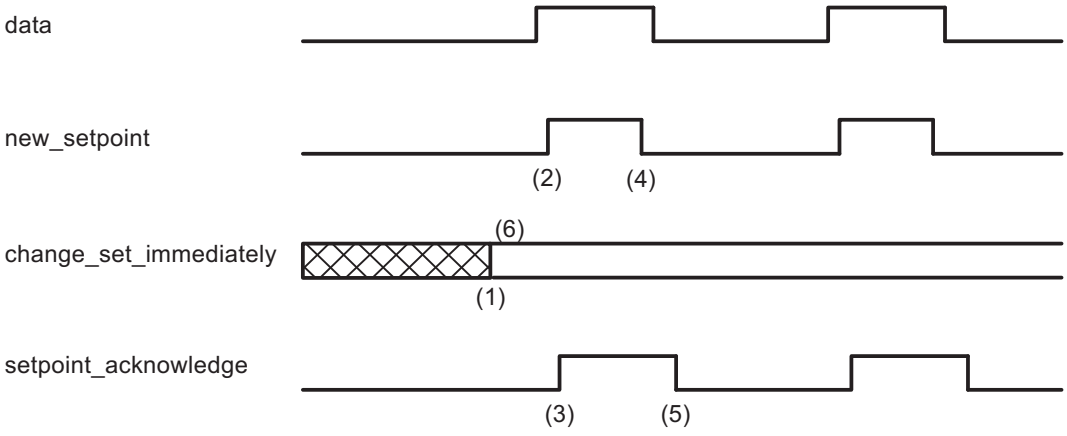
Set of setpoints:

After reaching the *target_position*, the drive device immediately processes the next *target_position*, which results in a move where the velocity of the drive normally is not reduced to zero after achieving a setpoint. With SERVOSTAR 400/600, this is only possible if trapezoidal ramps are used.

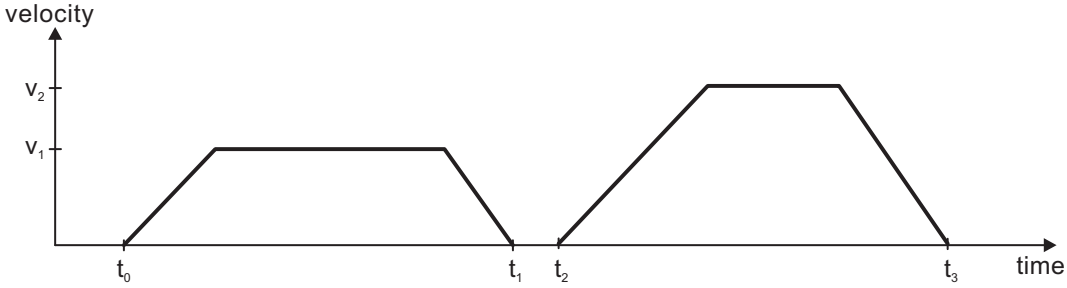
Single setpoints:

After reaching the *target_position*, the drive device signals this status to a host computer and then receives a new setpoint. After reaching a *target_position*, the velocity is normally reduced to zero before starting a move to the next setpoint.

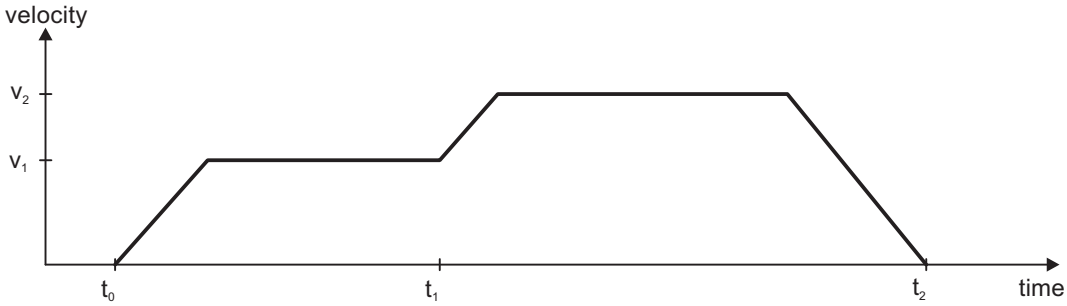
The two modes are controlled by the timing of the bits for *new_setpoint* and *change_set_immediately* in the control word, and *setpoint_acknowledge* in the status word. These bits allow the setting up of a request-response mechanism in order to prepare a set of setpoints while another set is still being processed in the drive unit. This minimizes reaction times within a control program on a host computer.



The figures show the difference between the *set of setpoints* mode and the *single setpoint* mode. The initial status of the bit *change_set_immediately* in the control word determines which mode is used. To keep these examples simple, only trapezoidal moves are used. If the bit *change_set_immediately* is “0”(continuously drawn line in Figure 1) a single setpoint is expected by the drive (1). After data is applied to the drive, a host signals that the data is valid by changing the bit *new_setpoint* to “1" in the control word (2). The drive responds with *setpoint_acknowledge* set to “1" in the status word (3) after it has recognized and buffered the new valid data. Now the host can release *new_setpoint* (4) and subsequently the drive will signal through *setpoint_acknowledge* = “0" its ability to accept new data again (5). In Figure 2 this mechanism results in a velocity of zero after ramping down to reach a target_position X_1 at t_1. After signaling to the host, that the setpoint has been reached as described above, the next target_position is processed at t_2 and reached at t_3.



With *change_set_immediately* set to “1" (6), symbolized by the dashed line in Figure 1, the host instructs the drive to apply a new setpoint immediately after reaching the previous one. The relative timing of the other signals is unchanged. This behavior causes the drive to process the next setpoint X_2 in advance, and to hold its velocity when it reaches the target_position X_1 at t_1. The drive then moves immediately to the next target_position X_2 that has already been calculated.



Bits im controlword:

Bit 4 new_set_point (positive Flanke!)
Bit 5 change_set_immediatly
Bit 6 absolut / relativ

Bits im statusword:

Bit 12 setpoint acknowledge
Bit 13 following error

Notes on motion task type *relative*:

If Bit 6 is set, then the motion task type is *relative*, and activated according to the last target position or actual position. If other types of relative motion are required, these must be activated in advance through Object 2022_h Sub-index 03_h *position data for position mode* (see also Object 2022_h Sub-index 03_h or ASCII Object *O_C*).

Notes on *profile position* mode:

Functional description for the *profile position* mode
The drive profile DSP402 distinguishes between two methods of moving to a target position. These two methods are controlled by the bits for *new_setpoint* and *change_set_immediately* in the control word, and *setpoint_acknowledge* in the status word. These bits can be used to prepare a motion task while another is still being carried out (handshake).

- **Moving to several target positions without an intermediate halt**
  After the target position has been reached, the drive moves immediately to the next target position. This requires that new setpoints are signaled to the drive. This is done through a positive transition of the *new_setpoint* bit. In this case, the *setpoint_acknowledge* bit must not be set (see also *Handshake* DSP402).
  The velocity is not reduced to zero when the first setpoint is reached.
- **Moving to a single target position**
  The drive moves to the target position, whereby the velocity is reduced to zero. Reaching the target position is signaled by the bit for *target_reached* in the status word.

This page has been deliberately left blank.

5 The Object Channel

5.1 Object Description

5.1.1 Object > 3500h Manufacturer Specific Object Channel

The Object Dictionary has been expanded beyond Index 3500_h (reserved Object range 3500_h ... 3900_h) for all Device Objects that can be described in up to 4 bytes of user data.
This range can be dynamically extended, i.e. if extensions are made, new device parameters that fulfil the above-mentioned format are **automatically** added to the table for the core firmware.
Object 3500_h (Sub-index 01_h, read) can be used to show the total number of Objects in the Object Channel (⇒ 6.3).

NOTE The objects in the Object Channel (ASCII parameters) cannot be mapped in a PDO!

Each Object in this range is described with the aid of 8 Sub-indices. Object structure:

Index	> 3500_h
Name	Object-dependent
Object code	VAR
Data type	RECORD

Sub-Indexes:

Sub-index	00_h
Description	number of entries
Unit	—
Access	—
PDO mapping	not possible
Data type	UNSIGNED8
Value range	0 ... 2^8-1
EEPROM	—
Default value	—

Sub-index	01_h
Description	read/write a parameter
Unit	see corresponding ASCII command
Access	see corresponding ASCII command
PDO mapping	not possible
Data type	see corresponding ASCII command
Value range	see corresponding ASCII command
EEPROM	see Sub-index 04_h
Default value	see corresponding ASCII command

Sub-index	02_h
Description	read lower limit value
Unit	see corresponding ASCII command
Access	ro
PDO mapping	not possible
Data type	see corresponding ASCII command
Value range	see corresponding ASCII command
EEPROM	—
Default value	—

Sub-index	03_h
Description	read upper limit value
Unit	see corresponding ASCII command
Access	ro
PDO mapping	not possible
Data type	see corresponding ASCII command
Value range	see corresponding ASCII command
EEPROM	—
Default value	—

Sub-index	04_h
Description	read the default value
Unit	see corresponding ASCII command
Access	ro
PDO mapping	not possible
Data type	see corresponding ASCII command
Value range	see corresponding ASCII command
EEPROM	—
Default value	—

Sub-index	05_h
Description	read the parameter format
Unit	—
Access	ro
PDO mapping	not possible
Data type	see corresponding ASCII command
Value range	see corresponding ASCII command
EEPROM	—
Default value	—

Description:

The following parameter formats are possible:

0	Function (no parameter)
1	Function (INTEGER32 parameter)
2	Function (INTEGER32 parameter with weighting 3)
3	INTEGER8
4	UNSIGNED8
5	INTEGER16
6,13	UNSIGNED16
7	INTEGER32
8,12	UNSIGNED32
9,10	INTEGER32 (weighting 3)

NOTE Parameters with parameter format 0 are read-only!

Sub-index	06_h
Description	read the parameter check data
Unit	—
Access	ro
PDO mapping	not possible
Data type	UNSIGNED32
Value range	0 ... 2^{32}-1
EEPROM	—
Default value	—

Description:

0x00010000 After an alteration the variable must be saved and the controller must be reset.

0x00020000 Variable is saved in the serial EEPROM.

0x00200000 Variable is read-only, must not be written to over the bus.

Sub-index	07_h / 08_h
Description	reserved
Unit	—
Access	ro
PDO mapping	not possible
Data type	UNSIGNED32
Value range	0 ... 2^{32}-1
EEPROM	—
Default value	—

Number	ASCII command	Data format	Weighting	Status	EEPROM
3500	MAXSDO	INTEGER32	—	—	—
3501	ACC	INTEGER16	—	—	yes
3502	ACCR	INTEGER16	—	—	yes
3503	ACTFAULT	INTEGER8	—	Disabled + Reset	yes
3504	ACTIVE	INTEGER8	—	—	no
3505	ADDR	UNSIGNED8	—	—	yes
3506	AENA	INTEGER8	—	—	yes
3507	ANCNFG	INTEGER8	—	Disabled + Reset	yes
3508	ANDB	INTEGER32	☒	—	yes
3509	ANIN1	INTEGER32	—	—	no
350A	ANIN2	INTEGER32	—	—	no
350B	ANOFF1	INTEGER16	—	—	yes
350C	ANOFF2	INTEGER16	—	—	yes
350D	ANOUT1	INTEGER8	—	Disabled + Reset	yes
350E	ANOUT2	INTEGER8	—	Disabled + Reset	yes
350F	ANZERO1	—	—	—	—
3510	ANZERO2	—	—	—	—
3511	AVZ1	INTEGER32	☒	—	yes
3512	CALCHP	—	—	Enabled	—
3513	CALCRK	—	—	Enabled	—
3514	CALCRP	—	—	Disabled + Reset	—
3515	CBAUD	INTEGER16	—	Disabled + Reset	yes
3516	reserved				
3517	CDUMP	—	—	—	—
3518	CLRFAULT	—	—	—	—
3519	CLRHR	—	—	—	—
351A	CLRORDER	INTEGER16	—	Disabled	—
351B	CLRWARN	UNSIGNED8	—	Disabled + Reset	yes
351C	CONFIG	—	—	—	—
351D	CONTINUE	—	—	Enabled	—
351E	CTUNE	—	—	Enabled	—
351F	CUPDATE	—	—	Disabled	—
3520	DAOFFSET1	INTEGER16	—	—	yes
3521	DAOFFSET2	INTEGER16	—	—	yes

Number	ASCII command	Data format	Weighting	Status	EEPROM
3522	DEC	INTEGER16	—	—	yes
3523	DECDIS	INTEGER16	—	—	yes
3524	DECR	INTEGER16	—	—	yes
3525	DECSTOP	INTEGER16	—	—	yes
3526	DEVICE	—	—	—	no
3527	DICONT	INTEGER32	☒	—	no
3528	DIFVAR	—	—	—	—
3529	DIPEAK	INTEGER32	☒	—	no
352A	DIR	INTEGER8	—	Disabled + Reset	yes
352B	DIS	—	—	Enabled	—
352C	DREF	INTEGER8	—	—	yes
352D	DRVSTAT	INTEGER32	—	—	no
352E	DR_TYPE	INTEGER16	—	—	no
352F	DUMP	—	—	—	—
3530	EN	—	—	Disabled	—
3531	ENCCAPT	INTEGER8	—	Disabled	yes
3532	ENCIN	INTEGER32	—	Disabled + Reset	yes
3533	ENCLINES	INTEGER16	—	Disabled + Reset	yes
3534	ENCMODE	INTEGER8	—	—	yes
3535	ENCOUT	INTEGER16	—	—	yes
3536	reserved				
3537	ENCZERO	INTEGER16	—	—	yes
3538	EXTMUL	INTEGER16	—	—	yes
3539	EXTPOS	INTEGER8	—	Disabled + Reset	yes
353A	EXTWD	INTEGER32	—	—	yes
353B	FBTYPE	INTEGER8	—	Disabled + Reset	yes
353C	FILTMODE	UNSIGNED8	—	Disabled + Reset	yes
353D	FOLDMODE	INTEGER8	—	Disabled + Reset	yes
353E	GEARI	INTEGER16	—	—	yes
353F	GEARMODE	INTEGER8	—	Disabled + Reset	yes
3540	GEARO	INTEGER16	—	—	yes
3541	GET	—	—	—	—
3542	GP	INTEGER32	☒	—	yes
3543	GPFBT	INTEGER32	☒	—	yes
3544	GPFFT	INTEGER32	☒	—	yes
3545	GPFFV	INTEGER32	☒	—	yes
3546	GPTN	INTEGER32	☒	—	yes
3547	GPV	INTEGER32	☒	—	yes
3548	GV	INTEGER32	☒	—	yes
3549	GVFBT	INTEGER32	☒	—	yes
354A	GVFILT	INTEGER8	—	—	yes
354B	GVFR	INTEGER32	☒	—	yes
354C	GVT2	INTEGER32	☒	—	yes
354D	GVTN	INTEGER32	☒	—	yes
354E	HACOFFS	INTEGER16	—	—	Encoder
354F	HAFACT1	INTEGER16	—	—	Encoder
3550	HASOFFS	INTEGER16	—	—	Encoder
3551	HDUMP	—	—	—	—
3552	HICOFFS	INTEGER16	—	—	yes
3553	HIFACT1	INTEGER16	—	—	Encoder
3554	HISOFFS	INTEGER16	—	—	Encoder
3555	HRESET	—	—	—	—
3556	HSAVE	—	—	—	—
3557	HVER	—	—	—	no
3558	I	INTEGER32	☒	—	no
3559	I2T	INTEGER32	—	—	no
355A	I2TLIM	INTEGER8	—	—	yes
355B	ICMD	INTEGER32	☒	—	no
355C	ICONT	INTEGER32	☒	—	yes
355D	ID	INTEGER32	☒	—	no
355E	IDUMP	—	—	—	—
355F	IMAX	INTEGER32	☒	—	no

Number	ASCII command	Data format	Weighting	Status	EEPROM
3560	IN	—	—	—	—
3561	IN1	INTEGER8	—	—	—
3562	IN1MODE	INTEGER8	—	Disabled + Reset	yes
3563	IN1TRIG	INTEGER32	—	—	yes
3564	IN2	INTEGER8	—	—	no
3565	IN2MODE	INTEGER8	—	Disabled + Reset	yes
3566	IN2TRIG	INTEGER32	—	—	yes
3567	IN3	INTEGER8	—	—	no
3568	IN3MODE	INTEGER8	—	Disabled + Reset	yes
3569	IN3TRIG	INTEGER32	—	—	yes
356A	IN4	INTEGER8	—	—	no
356B	IN4MODE	INTEGER8	—	Disabled + Reset	yes
356C	IN4TRIG	INTEGER32	—	—	yes
356D	INPOS	—	—	—	—
356E	IPEAK	INTEGER32	☒	—	yes
356F	IPEAKN	INTEGER32	☒	—	yes
3570	IQ	INTEGER32	☒	—	no
3571	ISCALE1	INTEGER32	☒	—	yes
3572	ISCALE2	INTEGER32	☒	—	yes
3573	K	—	—	Enabled	no
3574	KC	INTEGER32	☒	—	yes
3575	KEYLOCK	INTEGER8	—	—	yes
3576	reserved				
3577	L	INTEGER32	☒	—	yes
3578	LATCH16	INTEGER16	—	—	—
3579	LATCH16N	INTEGER16	—	—	no
357A	LATCH32	INTEGER32	—	—	—
357B	LATCH32N	INTEGER32	—	—	no
357C	LATCHX32	INTEGER32	—	—	no
357D	LATCHX32N	INTEGER32	—	—	no
357E	LED1	INTEGER8	—	—	—
357F	LED2	INTEGER8	—	—	—
3580	LED3	INTEGER8	—	—	—
3581	LEDSTAT	INTEGER16	—	—	—
3582	LIST	—	—	—	no
3583	LOAD	—	—	—	no
3584	MAXTEMPE	INTEGER16	—	—	yes
3585	MAXTEMPH	INTEGER16	—	—	yes
3586	MAXTEMPM	INTEGER32	☒	—	yes
3587	MBRAKE	INTEGER8	—	Disabled + Reset	yes
3588	MDBCNT	—	—	—	—
3589	MDBGET	—	—	—	—
358A	MDBSET	INTEGER16	—	—	—
358B	MDUMP	—	—	—	—
358C	reserved				
358D	MH	—	—	Enabled	—
358E	MICONT	INTEGER32	☒	—	yes
358F	MIPEAK	INTEGER32	☒	—	yes
3590	reserved				
3591	MJOG	—	—	Enabled	—
3592	MVANGLP	INTEGER16	—	—	yes
3593	MKT	INTEGER32	☒	—	yes
3594	reserved				
3595	MLGC	INTEGER32	☒	—	yes
3596	MLGD	INTEGER32	☒	—	yes
3597	MLGP	INTEGER32	☒	—	yes
3598	MLGQ	INTEGER32	☒	—	yes
3599	MNUMBER	INTEGER16	—	Disabled	yes
359A	MONITOR1	INTEGER16	—	—	no
359B	MONITOR2	INTEGER16	—	—	no
359C	MPHASE	INTEGER16	—	Disabled	yes
359D	MPOLES	INTEGER8	—	Disabled	yes

Number	ASCII command	Data format	Weighting	Status	EEPROM
359E	MRD	—	—	Enabled	—
359F	reserved				
35A0	MRESBW	INTEGER16	—	—	yes
35A1	MRESPOLES	INTEGER8	—	Disabled	yes
35A2	MSG	INTEGER8	—	—	yes
35A3	MSPEED	INTEGER32	☒	—	yes
35A4	reserved				
35A5	MTANGLP	INTEGER16	—	—	yes
35A6	reserved				
35A7	MVANGLB	INTEGER32	—	—	yes
35A8	MVANGLF	INTEGER16	—	—	yes
35A9	M_RESET	—	—	Disabled	—
35AA	NONBTB	INTEGER8	—	—	yes
35AB	NOTCHBW	INTEGER32	☒	—	yes
35AC	NOTCHHZ	INTEGER32	☒	—	yes
35AD	NREF	INTEGER8	—	—	yes
35AE	O1	INTEGER8	—	—	no
35AF	O1MODE	INTEGER8	—	Disabled + Reset	yes
35B0	O1TRIG	INTEGER32	—	—	yes
35B1	O2	INTEGER8	—	—	no
35B2	O2MODE	INTEGER8	—	Disabled + Reset	yes
35B3	O2TRIG	INTEGER32	—	—	yes
35B4	OPMODE	INTEGER8	—	—	yes
35B5	OPTION	INTEGER16	—	—	no
35B6	OVRIDE	INTEGER8	—	—	yes
35B7	O_ACC1	INTEGER16	—	—	no
35B8	O_ACC2	INTEGER16	—	—	no
35B9	O_C	INTEGER16	—	—	no
35BA	O_DEC1	INTEGER16	—	—	no
35BB	O_DEC2	INTEGER16	—	—	no
35BC	O_FN	INTEGER16	—	—	no
35BD	O_FT	INTEGER16	—	—	no
35BE	O_P	INTEGER32	—	—	no
35BF	O_V	INTEGER32	—	—	no
35C0	PBAL	INTEGER32	—	—	no
35C1	PBALMAX	INTEGER32	—	—	yes
35C2	PBALRES	INTEGER8	—	—	yes
35C3	PBAUD	INTEGER32	☒	—	no
35C4	PDUMP	—	—	—	—
35C5	PE	INTEGER32	—	—	no
35C6	PEINPOS	INTEGER32	—	—	yes
35C7	PEMAX	INTEGER32	—	—	yes
35C8	PFB	INTEGER32	—	—	no
35C9	PFB0	INTEGER32	—	—	no
35CA	PGEARI	INTEGER32	—	Disabled + Reset	yes
35CB	PGEARO	INTEGER32	—	Disabled + Reset	yes
35CC	PIOBUF	—	—	—	no
35CD	PMODE	INTEGER32	—	Disabled + Reset	yes
35CE	PNOID	INTEGER32	—	—	no
35CF	POSCNFG	INTEGER8	—	Disabled + Reset	yes
35D0	PPOTYP	INTEGER8	—	—	yes
35D1	PRBASE	INTEGER8	—	Disabled + Reset	yes
35D2	PRD	INTEGER32	—	—	no
35D3	PROMPT	INTEGER16	—	—	no
35D4	PSTATE	—	—	—	no
35D5	PTBASE	INTEGER8	—	—	yes
35D6	PTMIN	INTEGER16	—	—	yes
35D7	PV	INTEGER32	—	—	no
35D8	PVMAX	INTEGER32	—	—	yes
35D9	PVMAXN	INTEGER32	—	—	yes
35DA	reserved				
35DB	reserved				

Number	ASCII command	Data format	Weighting	Status	EEPROM
35DC	READNIMP	—	—	—	—
35DD	READY	INTEGER8	—	—	no
35DE	RECDONE	INTEGER8	—	—	no
35DF	RECING	INTEGER8	—	—	no
35E0	RECOFF	—	—	—	—
35E1	RECRDY	INTEGER8	—	—	no
35E2	REFIP	INTEGER32	☒	—	yes
35E3	REFPOS	INTEGER32	—	—	no
35E4	REMOTE	INTEGER8	—	—	no
35E5	RESPHASE	INTEGER16	—	—	yes
35E6	RK	INTEGER16	—	—	yes
35E7	ROFFS	INTEGER32	—	—	yes
35E8	RS232T	INTEGER16	—	—	yes
35E9	RSTVAR	—	—	Disabled	no
35EA	S	—	—	—	—
35EB	SAVE	—	—	—	—
35EC	SBAUD	INTEGER8	—	—	yes
35ED	SCAN	—	—	—	—
35EE	SDUMP	—	—	—	—
35EF	SERIALNO	INTEGER32	—	—	no
35F0	SETREF	—	—	—	—
35F1	SETROFFS	—	—	—	—
35F2	SLEN	INTEGER8	—	—	yes
35F3	SLOTIO	INTEGER32	—	—	no
35F4	SPHAS	INTEGER8	—	—	no
35F5	SPSET	INTEGER16	—	—	yes
35F6	SSIGRAY	INTEGER8	—	Disabled	yes
35F7	SSIINV	INTEGER8	—	Disabled	yes
35F8	SSIMODE	INTEGER8	—	—	yes
35F9	SSIOUT	INTEGER8	—	Disabled	yes
35FA	SSTAT	—	—	—	no
35FB	STAT	INTEGER16	—	—	no
35FC	STATIO	—	—	—	no
35FD	STATUS	—	—	—	no
35FE	STOP	—	—	Enabled	—
35FF	STOPMODE	INTEGER8	—	Disabled + Reset	yes
3600	SWCNFG	UNSIGNED16	—	Disabled + Reset	yes
3601	SWCNFG2	UNSIGNED16	—	Disabled + Reset	yes
3602	SWE0	INTEGER32	—	—	yes
3603	SWE0N	INTEGER32	—	—	yes
3604	SWE1	INTEGER32	—	—	yes
3605	SWE1N	INTEGER32	—	—	yes
3606	SWE2	INTEGER32	—	—	yes
3607	SWE2N	INTEGER32	—	—	yes
3608	SWE3	INTEGER32	—	—	yes
3609	SWE3N	INTEGER32	—	—	yes
360A	SWE4	INTEGER32	—	—	yes
360B	SWE4N	INTEGER32	—	—	yes
360C	SWE5	INTEGER32	—	—	yes
360D	SWE5N	INTEGER32	—	—	yes
360E	T	INTEGER32	☒	Enabled	—
360F	TASK	—	—	—	no
3610	TEMPE	INTEGER32	—	—	no
3611	TEMPH	INTEGER32	—	—	no
3612	TEMPM	INTEGER32	—	—	no
3613	TRJSTAT	INTEGER32	—	—	no
3614	TRUN	—	—	—	yes
3615	reserved				
3616	UID	INTEGER16	—	—	yes
3617	UVLTMODE	INTEGER8	—	Disabled + Reset	yes
3618	V	INTEGER32	—	—	no
3619	reserved				

Number	ASCII command	Data format	Weighting	Status	EEPROM
361A	VBUS	INTEGER32	—	—	no
361B	VBUSBAL	INTEGER16	—	—	yes
361C	VBUSMAX	INTEGER32	—	—	yes
361D	VBUSMIN	INTEGER16	—	—	yes
361E	VCMD	INTEGER32	☒	—	no
361F	VDUMP	—	—	—	—
3620	VELO	INTEGER32	☒		yes
3621	VJOG	INTEGER32	—	—	yes
3622	VLIM	INTEGER32	☒	—	yes
3623	VLIMN	INTEGER32	☒	—	yes
3624	VMAX	INTEGER32	☒	—	no
3625	VMIX	INTEGER32	☒	—	yes
3626	VMUL	INTEGER32	—	—	yes
3627	VOSPD	INTEGER32	☒	—	yes
3628	VREF	INTEGER32	—	—	yes
3629	VSCALE1	INTEGER16	—	—	yes
362A	VSCALE2	INTEGER16	—	—	yes
362B	\	UNSIGNED8	—	—	—
362C	DILIM	INTEGER8	—	Disabled + Reset	yes
362D	DENA	INTEGER8	—	—	yes
362E	IN2PM	INTEGER8	—	—	yes
362F	KTN	INTEGER32	☒	—	yes
3630	INPT	INTEGER16	—	—	yes
3631	UCOMP	INTEGER32	—	—	yes
3632	COLDSTART	—	—	Disabled	—
3633	reserved				
3634	UID1	INTEGER32	—	—	yes
3635	SETVCT	INTEGER16	—	—	no
3636	WPOS	INTEGER8	—	Disabled + Reset	no
3637	SRND	INTEGER32	—	—	yes
3638	ERND	INTEGER32	—	—	yes
3639	MDRV	INTEGER8	—	—	yes
363A	BCC	INTEGER16	—	—	no
363B	FPGA	INTEGER8	—	Disabled + Reset	yes
363C	REFMODE	INTEGER8	—	—	yes
363D	VLO	INTEGER32	☒	—	yes
363E	WMASK	INTEGER32	—	—	no
363F	WPOSE	INTEGER32	—	—	no
3640	WPOSP	INTEGER32	—	—	no
3641	WPOSX	INTEGER32	—	—	no
3642	MOVE	INTEGER16	—	Enabled	—
3643	POSRSTAT	INTEGER32	—	—	no
3644	P1	INTEGER32	—	—	yes
3645	P2	INTEGER32	—	—	yes
3646	P3	INTEGER32	—	—	yes
3647	P4	INTEGER32	—	—	yes
3648	P5	INTEGER32	—	—	yes
3649	P6	INTEGER32	—	—	yes
364A	P7	INTEGER32	—	—	yes
364B	P8	INTEGER32	—	—	yes
364C	P9	INTEGER32	—	—	yes
364D	P10	INTEGER32	—	—	yes
364E	P11	INTEGER32	—	—	yes
364F	P12	INTEGER32	—	—	yes
3650	P13	INTEGER32	—	—	yes
3651	P14	INTEGER32	—	—	yes
3652	P15	INTEGER32	—	—	yes
3653	P16	INTEGER32	—	—	yes
3654	PTARGET	INTEGER32	—	—	yes
3655	ACTRS232	INTEGER8	—	—	no
3656	ROFFS2	INTEGER32	—	—	yes
3657	FW	INTEGER32	☒	—	no

Number	ASCII command	Data format	Weighting	Status	EEPROM
3658	reserved				
3659		INTEGER32	—	—	yes
365A	VCOMM	INTEGER32	—	—	yes
365B	MTMUX	INTEGER16	—	—	no
365C	ROFFS0	INTEGER32	—	—	yes
365D	REFLS	INTEGER32	—	—	yes
365E	BOOT	—	—	—	yes
365F		INTEGER32	—	—	yes
3660		INTEGER32	—	—	yes
3661	reserved				
3662	reserved				
3663	reserved				
3664	reserved				
3665	reserved				
3666	reserved				
3667	reserved				
3668	reserved				
3669	reserved				
366A	reserved				
366B	reserved				
366C	reserved				
366D	reserved				
366E	TBRAKE	INTEGER16	—	—	yes
366F	TBRAKE0	INTEGER16	—	—	yes
3670	CMDDLY	INTEGER16	—	—	yes
3671	reserved				
3672	DRVCNFG	INTEGER32	—	—	yes

This page has been deliberatey left blank.

6 Appendix

6.1 Setup examples

All values are hexadecimal. The axis-related values always refer to Station1.

6.1.1 Basic testing of the connection control<->SERVOSTAR

When the SERVOSTAR 400/600 is switched on, an Emergency Message is transmitted over the bus with 0 or 8 data bytes (contents 0), depending on the setting of Bit 2 of the DRVCNFG parameter. The telegram continues to be transmitted, as long as it has not yet found a suitable receiver in the bus system.
If a CAN master is unable to recognize this message, then the following measures can be taken to test communication:

- Check the bus cable: correct characteristic impedance, correct termination resistors at both ends?
- With a multimeter: check the quiescent level of the bus cables CAN-H and CAN-L against CAN-GND (approx. 2.5 V).
- With an oscilloscope: check the output signals on CAN-H and CAN-L at the SERVOSTAR 400/600. Are signals being transmitted on the bus? The voltage difference between CAN-H and CAN-L for a logical "0" is approx. 2-3 V.
- Does signal transmission stop if the master is connected? Check the master hardware.
- Check the master software!

6.1.2 Example of operating the Status Machine

After the SERVOSTAR 400/600 is switched on and the boot-up message has been detected, communication via SDOs can be initiated. For example: all the parameters can be read out or written to, or the status machine for the drive can be controlled.

In the other examples it is assumed that functions are available for reading and writing SDOs, which appear as follows:

SDO-read (UINT Index, USHORT Sub-index);

SDO-write (UINT Index, USHORT Sub-index, ULONG value);

The state of the status machine can be obtained through the following query:

SDO-read (6041_h, 00_h)

Directly after switch-on, a value will then be returned, such as 0040_h. This corresponds to the status *Switch on disabled* (⇒ 4.4.1.1).

The following data would then be visible on the CAN bus: (der Aufbau des SDO-Telegramms ist in Kapitel 3.4.5.1 beschrieben):

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	40	41	60	00_h	40 00 00 00	
581	4B	41	60	00_h	40 00 00 00	reply telegram
	2 bytes of data				status	

If the supply power is present and the hardware enable is at the *High* level (24 V to DGND) then SDO-write (6040_h, 00_h, 0x7) can be used to switch the drive to the state *Switched on*. If this is successful, there will be a positive acknowledgement in the SDO reply (control byte 0 in the data field = 60_h).

Switch on

The messages then appear as follows:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	40	60	00_h	07 00 00 00	control word
581	60	40	60	00_h	00 00 00 00	response telegram

control word = 0x0007 Meaning: Bit 0, Bit 1, Bit 2 set ⇒ Switch On, *Disable Voltage* off, *Quick Stop* off

Status query 2

The new status can then be queried again, and returns the following result:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	40	41	60	00_h	—	query status
581	4B	41	60	00_h	23 00 00 00	response telegram

Status = 0x0023 Meaning: Bit 0, Bit 1, Bit 5 set ⇒ ready to Switch On, Switched On, Quick Stop

6.1.3 Example of PDO usage

Using all four possible PDOs in operation:

1. RPDO: PDO trajectory for one axis
2. RPDO: PDO control word and mode changeover
1. TPDO: PDO enhanced status
2. TPDO: PDO with incremental actual position, speed and operating mode display

Procedure:

Since the first RPDO is not available from the drive in the pre-defined form as required (⇒ 4.3.1) it must be assembled. First, it is necessary to check whether the entries for the incremental setpoint provision are available in mappable form.
This is the case with Object 2022_h Sub-index 04_h. So the 1st RPDO is selected:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	00	26	00_h	25 00 00 00	select PDO 37 as 1st RPDO
581	60	00	26	00_h	00 00 00 00	response telegram

The freely mappable RPDO 37 has thus been selected. In the next step, data must be attached to this PDO. This is done through the mapping parameter for the first RPDO:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	00	16	00_h	00 00 00 00	delete data for 1st RPDO
581	60	00	16	00_h	00 00 00 00	response telegram
601	23	00	16	01_h	20 04 22 20	data for first entry of the 1st RPDO Object 2022_h Sub-index 04_h, data length: 32 bit
581	60	00	16	01_h	00 00 00 00	response telegram

The data content of the first PDO has now been defined, and it includes 4 bytes of user data.

Next, the communication parameters can be defined:

The system must react to COB-ID 201_h as standard. So Sub-index 01_h must remain set to the default value. But the drive must react to every SYNC Object, so a value of 1 must be applied to Sub-index 2:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	00	14	02_h	01 00 00 00	reaction to every SYNC
581	60	00	14	02_h	00 00 00 00	response telegram

The 2nd RPDO is to have two components: the CANopen control word (Object 6040_h Sub-index 00_h) and the Object for changing the operating mode (Object 6060_h Sub-index 00_h).

The selection of the 2nd RPDO is made as follows:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	26	00_h	26 00 00 00	select PDO 38 as 2nd RPDO
581	60	01	26	00_h	00 00 00 00	response telegram

The mapping is defined next:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	01	16	00_h	00 00 00 00	delete data for 2nd RPDO
581	60	01	16	00_h	00 00 00 00	response telegram
601	23	01	16	01_h	10 00 40 60	data for first entry of the 2nd RPDO Object 6040_h Sub-index 00_h, data length: 16 bit
581	60	01	16	01_h	00 00 00 00	response telegram
601	23	01	16	02_h	08 00 60 60	data for second entry of the 2nd RPDO Object 6060_h Sub-index 00_h, data length: 8 bit
581	60	01	16	02_h	00 00 00 00	response telegram

This Object has to be evaluated immediately, so the communication parameters can remain at their default values. The first TPDO is already available in the drive, it just has to be selected.

PDO 23 is selected:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	00	2A	00_h	17 00 00 00	select PDO 23 as 1st TPDO
581	60	00	2A	00_h	00 00 00 00	response telegram

The corresponding mapping can be read out by Object $1A00_h$. The PDO contains 2 bytes for the CANopen status word and 4 bytes for the manufacturer status register.
The second TPDO is assembled again:
1. Selection via Object 2A01:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	2A	00_h	25 00 00 00	select PDO 37 as 2nd TPDO
581	60	01	2A	00_h	00 00 00 00	response telegram

2. The mapping for the three components required:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	01	1A	00_h	00 00 00 00	delete data for 2nd TPDO
581	60	01	1A	00_h	00 00 00 00	response telegram
601	23	01	1A	01_h	20 03 70 20	data for first entry of the 2nd TPDO, Object 2070_h Sub-index 03_h, data length: 32 bit
581	60	01	1A	01_h	00 00 00 00	response telegram
601	23	01	1A	02_h	18 02 70 20	data for second entry of the 2nd TPDO, Object 2070_h Subindex 02_h, length: 24 bit
581	60	01	1A	02_h	00 00 00 00	response telegram
601	23	01	1A	03_h	08 00 61 60	data for third entry of the 2nd TPDO, Object 6061_h Subindex 00_h, data length: 8 bit
581	60	01	1A	03_h	00 00 00 00	response telegram

Now the communication parameters can be defined. The drive must react to every SYNC Object, so a value of 1 must be applied to Sub-index 2:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	18	02_h	01 00 00 00	2nd TPDO: reaction to every SYNC
581	60	01	18	02_h	00 00 00 00	response telegram

6.1.4 Example of Homing

When the SERVOSTAR 400/600 is operated as a linear axis, a reference/homing point must be defined before positioning tasks can be executed. This can be done simply through *Set reference point* (control word Bit 12 = 0 -> 1 -> 0 or homing type 35 in *Homing* mode) or by starting a homing movement, either in the manufacturer-specific mode *Homing/reference* (0xF9) or in the *Homing* mode (0x6).

This example shows the procedure in the *Homing* mode.

First switch over to the *Homing movement* mode:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	60	60	00_h	F9 00 00 00	*Homing* mode
581	60	60	60	00_h	00 00 00 00	O.K. signal

In the following example, all parameters that affect the homing movement are set via the bus. If you can be absolutely certain that no-one has altered the parameters in the amplifier, then this part can be omitted, since the amplifiers save the data in non-volatile memory. (The inputs must have been previously configured as limit switches.)

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	24	20	01_h	17 00 00 00	homing to limit switch & resolver zero mark
581	60	24	20	01_h	00 00 00 00	OK
601	2F	24	20	02_h	00 00 00 00	negative direction
581	60	24	20	02_h	00 00 00 00	OK
601	2B	24	20	03_h	10 27 00 00	v = 10mm/s
581	60	24	20	03_h	00 00 00 00	OK
601	2B	24	20	04_h	32 00 00 00	accel. ramp 50ms
581	60	24	20	04_h	00 00 00 00	OK
601	2B	24	20	05_h	32 00 00 00	decel. ramp 50ms
581	60	24	20	05_h	00 00 00 00	OK
601	23	24	20	06_h	30 75 00 00	reference offset 30000µm
581	60	24	20	06_h	00 00 00 00	OK

To check the bit signals that are important for the homing movement, the PDO for enhanced status (PDO 23) must be used. PDO 23 is selected as TPDO 1:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	00	2A	00_h	17 00 00 00	select PDO 23 as 1^{st} TPDO
601	60	00	2A	00_h	00 00 00 00	response telegram

TPDO1 is thus composed of 6 bytes, whereby the first two bytes contain the CANopen status word (Object 6041), and the other four bytes contain the manufacturer-specific status register (Object 1002).

This assignment can be queried through the Objects for the PDO mapping (Object $1A00_h$, Sub-index 00_h-02_h). The PDOs are subsequently enabled through an NMT Object:

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

From now on, status changes for TPDO1 will be automatically reported.

The homing movement can now be started by Bit 4 of the CANopen control word:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2B	40	60	00_h	1F 00 00 00	mode
581	60	40	60	00_h	00 00 00 00	homing runs until home conditions are fulfilled

A positive edge transition of the *Reference point set* bit in the manufacturer-specific status register can be used to detect that the servo drive has calibrated (zeroed) its position system. The homing procedure is terminated by the reset of the *Motion task active* bit.

A TPDO1 could therefore look like this:

COB-ID	Data	Comment
181	05 27 00 00 0A 54	mode

The status of the homing movement can be seen from the enhanced status register, Bit 17 (*Reference point set*).

6.1.5 Example of Motion Block Processing

Switch on position control

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	60	60	00_h	FF 00 00 00	position control mode
581	60	60	60	00_h	00 00 00 00	position control switched on

Map the second receive-PDO:

(Start motion block, motion blocks for the example are already defined. 1st receive-PDO is set to the standard setting: control word.)

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	26	00_h	23 00 00 00	start motion block Object
581	60	01	26	00_h	00 00 00 00	OK, mapped

Switch NMT status machine to *operational*:

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

Access second receive-Object

COB-ID	Motion block number: Low	Motion block number: High
301	01	00

Response: none, the given motion block 1 is being processed.

Motor Quick Stop

COB-ID	Control: Low	Control: High
201	03	00

Response: none, motor is stopped with *t_not*.

Disable controller:

COB-ID	Control: Low	Control: High	Motion block number
201	03	00	1

Response: none, drive is without torque.

6.1.6 Example of Profile Positioning Mode usage

This section shows the operation of the *Profile position* mode. For this, the PDOs are set as follows:

First RPDO: PDO control word (No. 1)
Second RPDO: freely mappable PDO 2 (No. 38)
First TPDO: freely mappable PDO 1 (No. 37)
Second TPDO: freely mappable PDO 2 (No. 38)

The telegrams have an analogous appearance to the example for PDO operation (⇒ 6.1.3).

Data are placed in the freely mappable PDOs as shown in the following telegram examples:

1) second RPDO:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	01	16	00_h	00 00 00 00	2nd RPDO: delete mapping
581	60	01	16	00_h	00 00 00 00	
601	23	01	16	01_h	20 00 7A 60	2nd RPDO, entry 1: target_position
581	60	01	16	01_h	00 00 00 00	
601	23	01	16	02_h	20 00 81 60	2nd RPDO, entry 2: profile_velocity
581	60	01	16	02_h	00 00 00 00	

2) first TPDO:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	00	1A	00_h	00 00 00 00	1st TPDO: delete mapping
581	60	00	1A	00_h	00 00 00 00	
601	23	00	1A	01_h	10 00 41 60	1st TPDO, entry 1: profile status word
581	60	00	1A	01_h	00 00 00 00	
601	23	00	1A	02_h	08 09 80 20	1st TPDO. entry 2: TRJSTAT 3rd byte = manuf. status 3rd byte
581	60	00	1A	02_h	00 00 00 00	

3) second TPDO:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	01	1A	00_h	00 00 00 00	2nd TPDO: delete mapping
581	60	01	1A	00_h	00 00 00 00	
601	23	01	1A	01_h	20 00 64 60	2nd TPDO, entry 1: position_actual_value
581	60	01	1A	01_h	00 00 00 00	
601	23	01	1A	02_h	20 00 6C 60	1st TPDO, entry 2: velocity_actual_value
581	60	01	1A	02_h	00 00 00 00	

The first TPDO is to be transmitted under event-control. Since this corresponds to the default value for the communication parameters, nothing has to be changed in this case. The second TPDO is to be transmitted with every SYNC from the drive:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	01	18	02_h	01 00 00 00	transmit 2nd TPDO with every SYNC
581	60	01	18	02_h	00 00 00 00	

After the PDOs have been defined, they can be enabled through the NMT:

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

If the mechanical resolution is to be defined, this can only be written through Object 6093_h, Sub-index 01_h and 02_h. The pre-setting after switching on the drive corresponds to the drive-specific factors PGEARI and PGEARO:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	93	60	01_h	00 00 10 00	2^{20} increments
581	60	93	60	01_h	00 00 00 00	
601	23	93	60	02_h	A0 8C 00 00	36000 user-units
581	60	93	60	02_h	00 00 00 00	

You could use this example, for instance, for operating a rotary table with 0.1° angular resolution.

After these settings, a homing movement can be set up and initiated:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2B	40	60	00_h	0F 00 00 00	control word: enable operation
581	60	40	60	00_h	00 00 00 00	
601	2F	60	60	00_h	06 00 00 00	mode: set *Homing* mode
581	60	60	60	00_h	00 00 00 00	
601	2F	98	60	00_h	0C 00 00 00	homing type 12, negative direction
581	60	98	60	00_h	00 00 00 00	
601	23	99	60	01_h	40 19 01 00	homing velocity 72000 units/s = 2 s^{-1}
581	60	99	60	01_h	00 00 00 00	
601	2B	40	60	00_h	1F 00 00 00	start homing movement
581	60	40	60	00_h	00 00 00 00	

After the start of the homing movement, the following telegrams could come from the 1st TPDO:

COB-ID	Byte 0	Byte 1	Byte 2
181	27	00	21
181	27	10	27
181	27	10	23
181	27	14	2B
181	27	15	2A

The following bits from Byte 2 are required to detect the end of the homing movement:
Bit 0 = 0 : homing movement completed, Bit 1 = 1 : reference point set, Bit 3 = 1 In Position. Afterwards, the homing movement is also ended by the control word, in this case through the 1st RPDO:

COB-ID	Byte 0	Byte 1
201	0F	00

Now you can change over to the *Profile position* mode and set the ramps to be used for positioning:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	23	83	60	00_h	32 00 00 00	50ms acceleration time
581	60	83	60	00_h	00 00 00 00	
601	23	84	60	00_h	32 00 00 00	50ms deceleration time
581	60	84	60	00_h	00 00 00 00	
601	2F	60	60	00_h	01 00 00 00	*Profile position* mode
581	60	60	60	00_h	00 00 00 00	

A positioning movement can only be initiated by providing the setpoints through the 1st RPDO, followed by a start through the 2nd RPDO. This uses handshaking with *New setpoint* (control word) and *Setpoint acknowledge* (status word).

a) Setpoint:

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
301	F4	01	00	00	E8	03	00	00

b) Control word with *new setpoint* bit (Bit 4) set:

COB-ID	Byte 0	Byte 1
201	1F	00

c) Wait until the CANopen status word signals *Setpoint acknowledge* (Bit 12): e.g.

COB-ID	Byte 0	Byte 1	Byte 2
181	27	15	03

d) Control word is reset by the *New setpoint* bit (Bit 4) immediately:

COB-ID	Byte 0	Byte 1
201	0F	00

e) Drive removes *Setpoint acknowledge*:

COB-ID	Byte 0	Byte 1	Byte 2
181	27	01	03

Wait for the positioning to be completed:

COB-ID	Byte 0	Byte 1	Byte 2
181	27	05	0A

6.1.7 ASCII Communication

It makes sense to use ASCII communication through PDOs, since they can be used more efficiently in this way. This requires that the NMT status machine is in the *operational* status.

Example: Read the parameter T-tacho (see Setup Software Online Help).
(All data are hexadecimal, with the ASCII characters below, in square brackets.)

Direction	COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Master ⇒ SERVOSTAR	301	47_h [G]	56_h [V]	46_h [F]	42_h [B]	54_h [T]	$0D_h$ [CR]	$0A_h$ [LF]	0_h [NUL]
SERVOSTAR ⇒ Master	281	30_h [0]	$2E_h$ [.]	36_h [6]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]
SERVOSTAR ⇒ Master	281	$0D_h$ [CR]	$0A_h$ [LF]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]
SERVOSTAR ⇒ Master	281	$2D_h$ [-]	$2D_h$ [-]	$3E_h$ [>]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]	0_h [NUL]

Explanation: In telegram 1, the master queries the *GVFBT* parameter, terminated by the ASCII code *CR LF*. The last free byte is filled by *NUL*.
The response of the SERVOSTAR is made in telegram 2, with the value *0.6*, the terminating code *CR LF*, and the prompt "—>" for the next parameter or command. The segmentation of the response into three telegrams is not mandatory, but depends on the transmission rate that has been set and the internal synchronization mechanism.

6.1.8 Test for SYNC-telegrams

Aims:

1. Assign/set *Start motion block* to the PDO (1st receive-PDO)
2. Assign *Actual position* (PDO21) to PDO (1st transmit-PDO), generated with every 2nd SYNC.
3. Assign *Status word* (PDO1) to PDO (2nd transmit-PDO), generated with every 3rd SYNC.

Telegrams with the corresponding responses:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	00	26	00_h	23 00 00 00	set PDO *Start motion block* to 1st RPDO
581	60	00	26	00_h	00 00 00 00	
601	2F	00	2A	00_h	16 00 00 00	set PDO Actual position to 1st TPDO
581	60	00	2A	00_h	00 00 00 00	
601	2F	01	2A	00_h	17 00 00 00	set PDO Enhanced status word to 2nd TPDO
581	60	01	2A	00_h	00 00 00 00	
601	2F	00	18	02_h	02 00 00 00	1st TPDO triggered for every 2nd SYNC
581	60	00	18	02_h	00 00 00 00	
601	2F	01	18	02_h	03 00 00 00	2nd TPDO triggered for every 3rd SYNC
581	60	01	18	02_h	00 00 00 00	

6.1.9 SYNC-Object

COB-ID
080

Meaning: Object 181 (TPDO 1) appears at every 2nd SYNC,
Object 281 (TPDO 2) appears at every 3rd SYNC.

6.1.10 Emergency-Object

If, for instance, the resolver connector is disconnected, a serious error will be caused in the controller. This results in an *Emergency* telegram.

COB-ID	Emergency error		Error register	Data	Comment
	Low	High			
081	10	43	08	00 00 00 00	motor temperature, temperature, manufacturer specific
081	00	00	88	00 00 00 00	

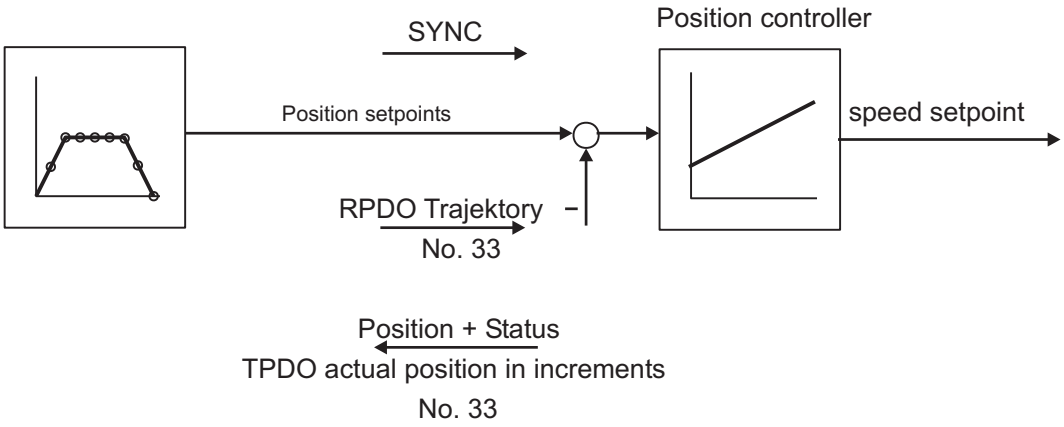
6.2 Special Applications

6.2.1 External Trajectory

6.2.1.1 Position controller in the servo amplifier

This example shows the situation where 2 axes each receive position setpoints through RPDO 33 *Trajectory*.

Controller structure for the position controller within the servo amplifier:



Description

All data are hexadecimal. In the example, the two axes in the system have the station addresses 1 and 2.

Examples of telegrams and responses:

Conditions:

- The resolution that is to be used within one motor turn must be defined for both axes: *PRBASE* set to 16 or 20 bit resolution for 1 turn.
- The time raster for trajectory inputs must be set through the *PTBASE* parameter.
  Here one unit has a value of 250 microseconds: for example, PTBASE = 8 produces a 2 ms time raster.
- The parameters must be saved in the EEPROM.
- The saved values only become available after a reset.

The *Trajectory* PDO contains 2 trajectory setpoints, and can be transmitted simultaneously to several stations, whereby each station can extract the relevant trajectory data.

Map the second receive-PDO for both axes to RPDO 33 *Trajectory* ($33_d = 21_h$):

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	26	00_h	21 00 00 00	set the *Trajectory* PDO for 1st axis
581	60	01	26	00_h	00 00 00 00	
602	2F	01	26	00_h	21 00 00 00	set the *Trajectory* PDO for 2nd axis
582	60	01	26	00_h	00 00 00 00	

After the *Trajectory* PDO has been mapped to the two axes, the communication parameters for both must be set so that they respond to the same Communication Object Identifier (COB-ID). The COB-ID for the first station can keep its default value of 301, the one for the second station can then be re-mapped to this setting:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
602	23	01	14	01_h	01 03 00 00	map 2nd RPDO for 2nd axis to 301
582	60	01	14	01_h	00 00 00 00	

Both stations will now respond to the same COB-ID 301.

Object 2721_h Sub-index 00_h can then be used to define which portion of the 8 byte data field is used by each axis for its trajectory information. The value 0 selects Bytes 0 ... 3 of the data, and the value 1 selects bytes 4 ... 7.

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	21	27	00_h	00 00 00 00	1st axis takes data from Bytes 0 ... 3
581	60	21	27	00_h	00 00 00 00	
602	2F	21	27	00_h	04 00 00 00	2nd axis takes data from Bytes 4 ... 7
582	60	21	27	00_h	00 00 00 00	

The actual positions of the axes are to be returned as incremental actual positions to the controls. The second transmit-PDO in each case is therefore mapped to TPDO 33 *Incremental actual position* ($33_d = 21_h$):

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	2A	00_h	21 00 00 00	set the *Trajectory* PDO for 1st axis
581	60	01	2A	00_h	00 00 00 00	
602	2F	01	2A	00_h	21 00 00 00	set the *Trajectory* PDO for 2nd axis
582	60	01	2A	00_h	00 00 00 00	

Here it is assumed that both amplifiers accept new trajectory values with every SYNC command, and have to return their incremental position values. The communication parameters must be set accordingly:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	01	14	02_h	01 00 00 00	RPDO 2, axis 1, reaction for every SYNC
581	60	01	14	02_h	00 00 00 00	
602	2F	01	14	02_h	01 00 00 00	RPDO 2, axis 2, reaction for every SYNC
582	60	01	14	02_h	00 00 00 00	
601	2F	01	18	02_h	01 00 00 00	TPDO 2, axis 1, reaction for every SYNC
581	60	01	18	02_h	00 00 00 00	
602	2F	01	18	02_h	01 00 00 00	TPDO 2, axis 2, reaction for every SYNC
582	60	01	18	02_h	00 00 00 00	

In order to be able to make trajectory movements, both servo amplifiers must be operating in the appropriate mode. This is set through Index 6060_h:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2F	60	60	00_h	FA 00 00 00	set trajectory mode for axis 1
581	60	60	60	00_h	00 00 00 00	
602	2F	60	60	00_h	FA 00 00 00	set trajectory mode for axis 2
582	60	60	60	00_h	00 00 00 00	

To start up the axes, the servo amplifiers must be put into the operational status (*operation enable*) and the network management functions must be started.

The network management functions enable the application of the Process Data Objects (PDOs) and are initialized by the following telegram for both axes:

Switch the NMT (**N**etwork **M**anagemen**t**) status machine to *operation enable*:

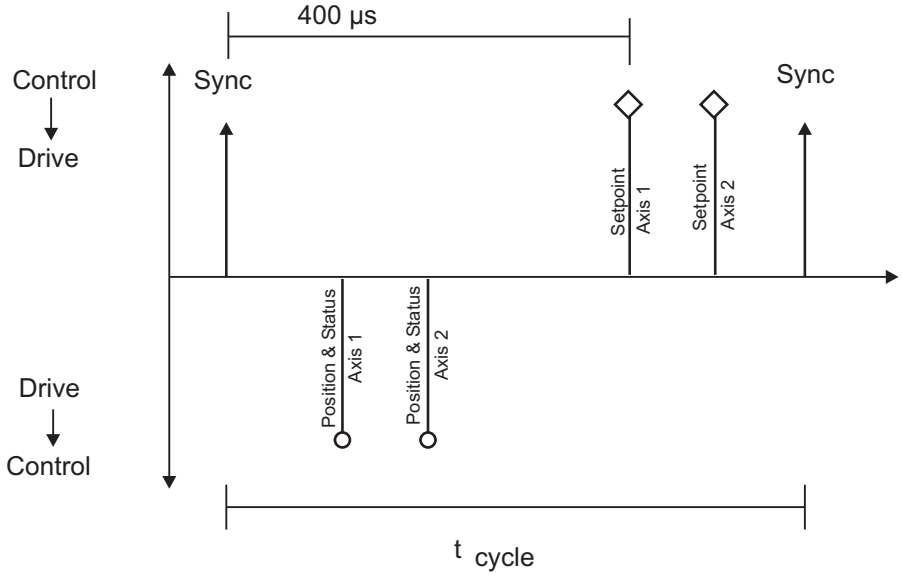
COB-ID	Command specifier (CS)	Node-ID	Comment
0	1	1	NMT enable for all axes

Next, power is applied to each servo amplifier, and they are put into the *operation enable* condition.

Control word for *operation enable*:

COB-ID	Control byte	Index		Sub-index	Data	Comment
		Low byte	High byte			
601	2B	40	60	00_h	0F 00 00 00	control word for axis 1
581	60	40	60	00_h	00 00 00 00	
602	2B	40	60	00_h	0F 00 00 00	control word for axis 2
582	60	40	60	00_h	00 00 00 00	

The configuration above now enables a cyclical sequence, as shown in the following diagram:



e.g. 2 axes

t_{cycle} 1 ms per axis at 1 MBaud

RPDO 2 can now be used to supply trajectory data for both axes, for instance:

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
301	F4	01	00	00	E8	03	00	00

In this example, the first axis receives a trajectory value of 500 increments (Bytes 0 ... 3) and the second axis receives a trajectory value of 1000 increments.

The axes accept these values, and the positioning is made when the next SYNC telegram is received.

The SYNC telegram looks like this:

COB-ID
080

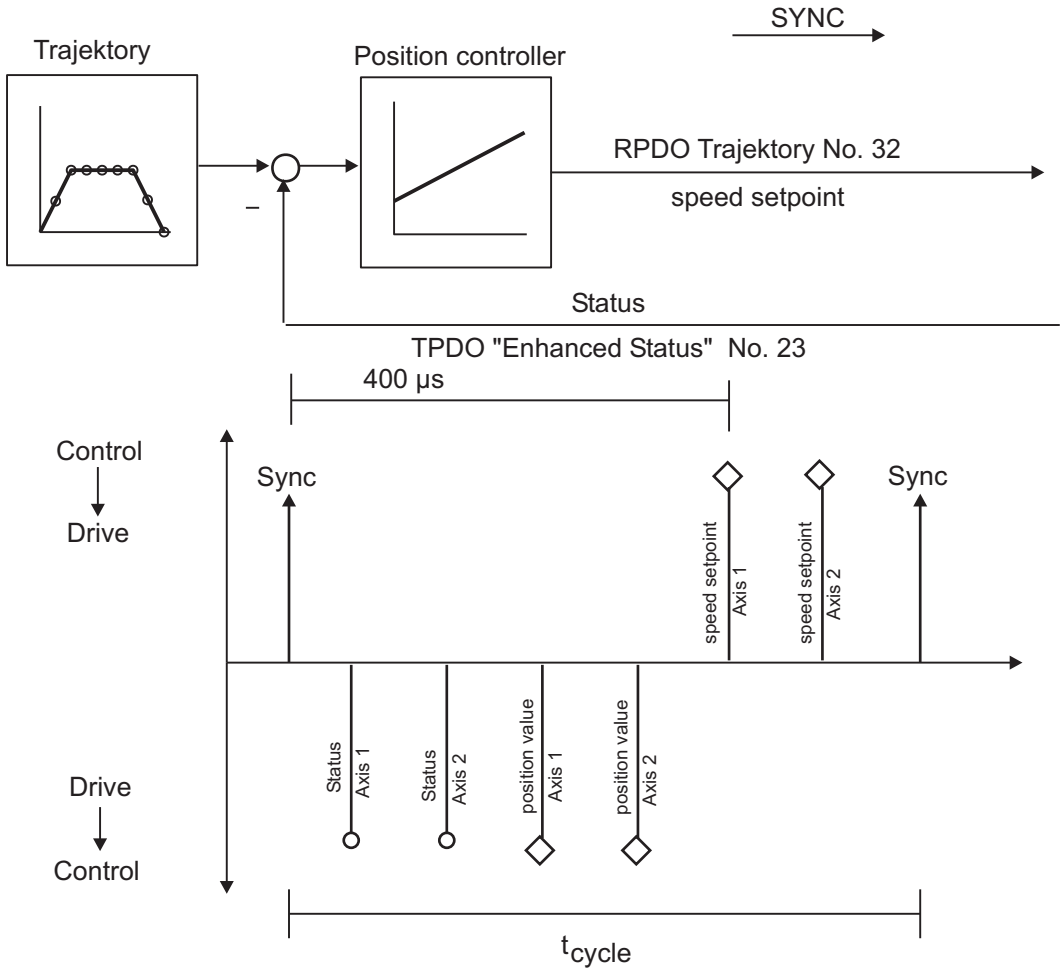
Afterwards, both axes send back their incremental positions and the contents of their status registers when the SYNC Object with the COB-ID for the 2nd TPDO is received:

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Comment
181	23	01	00	00	00	00	03	44	position + manufacturer status register for axis1
182	A5	02	00	00	00	00	03	44	position + manufacturer status register for axis2

If an error occurs during operation, the axis concerned transmits an *Emergency* message, which could appear as follows:
Emergency Object

COB-ID	Emergency error		Error register	Category		
	Low	High				
081	10	43	08	01	00 00 00 00	motor temperature, tempera-
081	00	00	08	00	00 00 00 00	ture, manufacturer-specific

6.2.1.2 Position controller in the control system



E.g. 2 axes

t_{cycle} 1 ms per Axis at 1 MBaud

6.3 Object Dictionary

The column *DEF* shows the corresponding profile:
S = SERVOSTAR **3** = DS301 **4** = DS402

Column *Access* means:
ro - read only, wo - write only
rwr - read / write on process input (mappbar für Rx-PDOs)
rww - read 7 write on process output (mappbar für Tx-PDOs)

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
1000_h	—	3	UNSIGNED32	ro	—	Device type	—
1001_h	—	3	UNSIGNED8	ro	—	Error register	—
1002_h	—	3	UNSIGNED32	ro	—	Manufacturer-specific status register	—
1003_h	00_h	3	UNSIGNED8	ro	—	Predefined error field (number of entries)	—
	01_h	3	UNSIGNED32	rw	—	Last reported error	—
1004_h	00_h	3	UNSIGNED32	ro	—	Number of supported PDOs	—
	01_h	3	UNSIGNED32	ro	—	Number of synchronous PDOs	—
	02_h	3	UNSIGNED32	ro	—	Number of asynchronous PDOs	—
1005_h	—	3	UNSIGNED32	ro	—	COB-ID SYNC message	—
1006_h	—	3	UNSIGNED32	rw	—	Cycle time for communication	—
1007_h	—	3	UNSIGNED32	rw	—	Time window for synchronous CAN messages	—
1008_h	—	3	Visible String	ro	—	Device name	VER*
$100A_h$	—	3	Visible String	ro	—	Software version	ADDR
$100B_h$	—	3	UNSIGNED32	ro	—	Node address	—
$100C_h$	—	3	UNSIGNED16	rw	—	Guard time	—
$100D_h$	—	3	UNSIGNED8	rw	—	Lifetime factor	—
$100E_h$	—	3	UNSIGNED32	rw	—	Node Guarding COB Identifier	—
$100F_h$	—	3	UNSIGNED32	rw	—	Number of supported Objects	—
1010_h	00_h	3	RECORD	ro	—	Store parameters	—
	01_h	3	UNSIGNED32	rw	—	Save all parameters	—
	02_h	3	UNSIGNED32	rw	—	Save communication parameters	—
1012_h	—	3	UNSIGNED32	rw	—	COB-ID for the time stamp message (in pre.)	—
1013_h	—	3	UNSIGNED32	rw	—	High resolution time stamp (in preparation)	—
1014_h	—	3	UNSIGNED32	rw	—	COB-ID for the Emergency message	—
1018_h	00_h	3	RECORD	ro	—	Identity Object	—
	01_h	3	UNSIGNED32	ro	—	Vendor ID	—
	02_h	3	UNSIGNED32	ro	—	Product Code	—
	03_h	3	UNSIGNED32	ro	—	Revision number	—
	04_h	3	UNSIGNED32	ro	—	Serial number	SERIALNO
1400_h	00_h	3	RECORD	ro	—	1^{st} receive-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB - ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time (not useful for RPDOs)	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1401_h	00_h	3	RECORD	ro	—	2^{nd} receive-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB-ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time (not useful for RPDOs)	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1402_h	00_h	3	RECORD	ro	—	3^{rd} receive-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB-ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time (not useful for RPDOs)	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1403_h	00_h	3	RECORD	ro	—	4^{th} receive-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB-ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time (not useful for RPDOs)	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1600_h	00_h	3	RECORD	rw	—	1^{st} receive-PDO mapping parameter	—
	01_h-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th Object	—
1601_h	00_h	3	RECORD	rw	—	2^{nd} receive- PDO mapping parameter	—
	01_h-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th Object	—

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
1602_h	00_h	3	RECORD	rw	—	3rd receive- PDO mapping parameter	—
	01_h-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th Object	—
1603_h	00_h	3	RECORD	rw	—	4th receive- PDO mapping parameter	—
	01_h-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th Object	—
1800_h	00_h	3	RECORD	ro	—	1st transmit-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB-ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1801_h	00_h	3	RECORD	ro	—	2nd transmit-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB - ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1802_h	00_h	3	RECORD	rw	—	3rd transmit-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB - ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
1803_h	00_h	3	RECORD	ro	—	4th transmit-PDO communication parameter	—
	01_h	3	UNSIGNED32	rw	—	PDO COB - ID	—
	02_h	3	UNSIGNED8	rw	—	Transmission type	—
	03_h	3	UNSIGNED16	rw	—	Inhibit time	—
	04_h	3	UNSIGNED8	rw	—	Compatibility entry (CMS priority group)	—
$1A00_h$	00_h	3	RECORD	rw	—	1st transmit-PDO mapping parameter	—
	01_h-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th object	—
$1A01_h$	00_h	3	RECORD	rw	—	2nd transmit-PDO mapping parameter	—
	01-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th object	—
$1A02_h$	00_h	3	RECORD	rw	—	3rd transmit-PDO mapping parameter	—
	01-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th object	—
$1A03_h$	00_h	3	RECORD	rw	—	4th transmit-PDO mapping parameter	—
	01-08_h	3	UNSIGNED32	rw	—	PDO mapping parameter for the n-th object	—
2014_h	00_h	S	ARRAY	ro	—	Mask PDOs 37..40 Channel 1	—
	01_h	S	UNSIGNED32	rw	—	Mask (Byte 0 ... 3)	—
	02_h	S	UNSIGNED32	rw	—	Mask (Byte 4 ... 7)	—
2015_h	00_h	S	ARRAY	ro	—	Mask PDOs 37..40 Channel 2	—
	01_h	S	UNSIGNED32	rw	—	Mask (Byte 0 ... 3)	—
	02_h	S	UNSIGNED32	rw	—	Mask (Byte 4 ... 7)	—
2016_h	00_h	S	ARRAY	ro	—	Mask PDOs 37..40 Channel 3	—
	01_h	S	UNSIGNED32	rw	—	Mask (Byte 0 ... 3)	—
	02_h	S	UNSIGNED32	rw	—	Mask (Byte 4 ... 7)	—
2017_h	00_h	S	ARRAY	ro	—	Mask PDOs 37..40 Channel 4	—
	01_h	S	UNSIGNED32	rw	—	Mask (Byte 0 ... 3)	—
	02_h	S	UNSIGNED32	rw	—	Mask (Byte 4 ... 7)	—
2020_h	00_h	S	RECORD	ro	—	Position controller	—
	01_h	S	UNSIGNED8	rw	—	Axis type	POSCNFG
	02_h	S	INTEGER32	rw	—	In-Position window	PEINPOS
	03_h	S	INTEGER32	rw	—	Lag/following error window	PEMAX
	04_h	S	INTEGER32	rw	—	Position register 1	SWE1
	05_h	S	INTEGER32	rw	—	Position register 2	SWE2
	06_h	S	INTEGER32	rw	—	Position register 3	SWE3
	07_h	S	INTEGER32	rw	—	Position register 4	SWE4
	08_h	S	UNSIGNED32	rw	—	Denominator for resolution	PGEARO
	09_h	S	UNSIGNED32	rw	—	Numerator for resolution	PGEAR
	$0A_h$	S	UNSIGNED8	rw	—	Count direction	DIR

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
2022h	00h	S	RECORD	ro	—	Position data for *Position* mode	—
	01h	S	INTEGER32	rww	☒	Position (motion block 0)	O_P
	02h	S	UNSIGNED16	rww	☒	Weighted velocity, see ASCII VMUL (motion block 0, see also Object 2020h Sub-index02h)	O_V
	03h	S	UNSIGNED16	rww	☒	Motion task type (motion block 0)	O_C
	04h	S	INTEGER32	rww	☒	Position for external trajectory	—
	05h	S	UNSIGNED16	rww	☒	Motion task number	—
	06h	S	UNSIGNED16	rww	☒	Acceleration time (motion block 0)	O_ACC1
	07h	S	UNSIGNED16	rww	☒	Braking time (motion block 0)	O_DEC1
	08h	S	UNSIGNED16	rww	☒	Jolt limit for acceleration time (motion block 0)	O_ACC2
	09h	S	UNSIGNED16	rww	☒	Jolt limiting for braking time (motion block 0)	O_DEC2
	0Ah	S	UNSIGNED16	rww	☒	Number of following task	O_FN
	0Bh	S	UNSIGNED16	rww	☒	Start delay for following task	O_FT
	0Ch	S	2 x UNSIGNED16	wo	—	Copy a motion task	OCOPY
	0Dh	S	UNSIGNED16	rw	—	Velocity weighting factor (Object 2020h Sub-index 02h)	VMUL
	0Eh	S	UNSIGNED32	rww	☒	Speed/velocity (motion block 0)	O_V
	0Fh	S	INTEGER32	rww	☒	Setpoint position CAN-Master-Slave	
2024h	00h	S	RECORD	r	—	Setup operation for *Position* mode	—
	01h	S	UNSIGNED8	rw	—	Homing type	NREF
	02h	S	UNSIGNED8	rw	—	Homing direction	DREF
	03h	S	INTEGER32	rw	—	Velocity for homing movement	VREF
	04h	S	UNSIGNED16	rw	—	Acceleration ramp [jogging and homing]	ACCR
	05h	S	UNSIGNED16	rw	—	Braking ramp [jogging and homing]	DECR
	06h	S	INTEGER32	rw	—	Reference offset	ROFFS
	07h	S	INTEGER32	rw	—	Jogging velocity	VJOG
2026h	00h	S	RECORD	r	—	Special Object for *Position* mode	—
	01h	S	UNSIGNED8	rw	—	Activate internal evaluation unit for saving actual position value	—
2030h	00h	S	ARRAY	ro	—	DP-Ram Variables, write only (PDO)	—
	01h	S	INTEGER32	rww	☒	DP-Ram Variable 9	DPRVAR9
	02h	S	INTEGER32	rww	☒	DP-Ram Variable 10	DPRVAR10
	03h	S	INTEGER32	rww	☒	DP-Ram Variable 11	DPRVAR11
	04h	S	INTEGER32	rww	☒	DP-Ram Variable 12	DPRVAR12
	05h	S	INTEGER32	rww	☒	DP-Ram Variable 13	DPRVAR13
	06h	S	INTEGER32	rww	☒	DP-Ram Variable 14	DPRVAR14
	07h	S	INTEGER32	rww	☒	DP-Ram Variable 15	DPRVAR15
	08h	S	INTEGER32	rww	☒	DP-Ram Variable 16	DPRVAR16
2031h	00h	S	RECORD	ro	—	Dummy-Variables for mapping	—
	01h	S	UNSIGNED8	wo	☒	Write-only dummy 8 bit	—
	02h	S	UNSIGNED16	wo	☒	Write-only dummy 16 bit	—
	03h	S	UNSIGNED24	wo	☒	Write-only dummy 24 bit	—
	04h	S	UNSIGNED32	wo	☒	Write-only dummy 32 bit	—
2050h	00h	S	ARRAY	ro	—	Aux. variable for the digital inputs	—
	01h	S	INTEGER32	rw	—	Trigger variable input 1	IN1TRIG
	02h	S	INTEGER32	rw	—	Trigger variable input 2	IN2TRIG
	03h	S	INTEGER32	rw	—	Trigger variable input 3	IN3TRIG
	04h	S	INTEGER32	rw	—	Trigger variable input 4	IN4TRIG
	05h	S	INTEGER32	wo	☒	Trigger help variable for electronic gearing	InxTRIG 51
2051h	00h	S	ARRAY	ro	—	Configuration for threshold register	—
	01h	S	UNSIGNED32	rw	—	Monitoring (deactivate / activate)	WPOSE
	02h	S	UNSIGNED32	rw	—	Signaling type (repeated / once)	WPOSX
	03h	S	UNSIGNED32	rw	—	Polarity position signaling	WPOSP

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
2052h	00h	S	ARRAY	ro	—	Position threshold register ABSOLUTE (refresh time < 1ms)	POSRSTAT
	01h	S	INTEGER32	rww	☒	Position register	P1
	02h	S	INTEGER32	rww	☒	Position register	P2
	03h	S	INTEGER32	rww	☒	Position register	P3
	04h	S	INTEGER32	rww	☒	Position register	P4
	05h	S	INTEGER32	rww	☒	Position register	P5
	06h	S	INTEGER32	rww	☒	Position register	P6
	07h	S	INTEGER32	rww	☒	Position register	P7
	08h	S	INTEGER32	rww	☒	Position register	P8
	09h	S	INTEGER32	rww	☒	Position register	P9
	0Ah	S	INTEGER32	rww	☒	Position register	P10
	0Bh	S	INTEGER32	rww	☒	Position register	P11
	0Ch	S	INTEGER32	rww	☒	Position register	P12
	0Dh	S	INTEGER32	rww	☒	Position register	P13
	0Eh	S	INTEGER32	rww	☒	Position register	P14
	0Fh	S	INTEGER32	rww	☒	Position register	P15
	10h	S	INTEGER32	rww	☒	Position register	P16
2053h	00h	S	INTEGER8	ro	—	Position threshold RELATIVE (refresh time < 1ms) see also ASCII	POSRSTAT
	01h	S	INTEGER32	rww	☒	Position register	P1
	02h	S	INTEGER32	rww	☒	Position register	P2
	03h	S	INTEGER32	rww	☒	Position register	P3
	04h	S	INTEGER32	rww	☒	Position register	P4
	05h	S	INTEGER32	rww	☒	Position register	P5
	06h	S	INTEGER32	rww	☒	Position register	P6
	07h	S	INTEGER32	rww	☒	Position register	P7
	08h	S	INTEGER32	rww	☒	Position register	P8
	09h	S	INTEGER32	rww	☒	Position register	P9
	0Ah	S	INTEGER32	rww	☒	Position register	P10
	0Bh	S	INTEGER32	rww	☒	Position register	P11
	0Ch	S	INTEGER32	rww	☒	Position register	P12
	0Dh	S	INTEGER32	rww	☒	Position register	P13
	0Eh	S	INTEGER32	rww	☒	Position register	P14
	0Fh	S	INTEGER32	rww	☒	Position register	P15
	10h	S	INTEGER32	rww	☒	Position register	P16
2060h	00h	S	INTEGER32	rw	—	Speed or current setpoint	—
2061h	00h	S	UNSIGNED16	wo	☒	Current limitation	DPRILIMIT

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
2070h	00h	S	RECORD	r	—	Manufacturer-specific actual values	—
	01h	S	INTEGER24	r	☒	Actual position (rot. angle, 20-bit res./turn)	PRD
	02h	S	INTEGER24	r	☒	Speed in incr. (1[incr.] = 1875/262144 [rpm])	—
	03h	S	INTEGER32	r	☒	Incr. act. position (res. depends on PRBASE)	—
	04h	S	UNSIGNED16	r	☒	Stored position (HW - LATCH positive, 16-bit resolution/turn)	LATCH16
	05h	S	INTEGER32	r	☒	Stored position (HW - LATCH positive, 32-bit resolution/turn)	LATCH32
	06h	S	INTEGER32	r	☒	Position, dependent on gearing factors (PGEARI, PGEARO)	PFB
	07h	S	INTEGER32	r	—	Velocity, dependent on gearing factors (PGEARI, PGEARO)	PV
	08h	S	INTEGER32	r	☒	Lag error, dependent on gearing factors (PGEARI, PGEARO)	PE
	09h	S	UNSIGNED32	r	—	Effective (r.m.s.) current	I
	0Ah	S	INTEGER32	r	☒	Speed, revs/min	V
	0Bh	S	INTEGER32	r	—	Heat sink temperature	TEMPH
	0Ch	S	INTEGER32	r	—	Internal temperature	TEMPE
	0Dh	S	INTEGER32	r	—	DC-link (DC-bus) voltage	VBUS
	0Eh	S	INTEGER32	r	—	Regen power (ballst power)	PBAL
	0Fh	S	INTEGER32	r	—	I²t loading	I2T
	10h	S	INTEGER32	r	—	Operating time	TRUN
	11h	S	INTEGER32	r	☒	Enhanced status for TPDO 33	—
	12h	S	INTEGER32	ro	☒	Encoder position (SI-units)	PFB0
	13h	S	INTEGER32	ro	☒	Internal position setpoint	S_SETL (user-defined)
	14h	S	INTEGER16	ro	☒	Analog input value 1	ANIN1
	15h	S	INTEGER16	ro	☒	Analog input value 2	ANIN2
	16h	S	UNSIGNED32	ro	☒	Error register	ERRCODE
	17h	S	INTEGER32	ro	☒	Speed setpoint	PSPEED (user-defined)
	18h	S	UNSIGNED32	ro	☒	Read master speed (electronic gearing)	MASTSPEED (user-defined)
	19h	S	INTEGER8	ro	☒	Control variable (electronic gearing)	ENGEDGE (user-defined)
2071h	00h	S	RECORD	ro	—	Dummy-variables for mapping	—
	01h	S	UNSIGNED8	ro	☒	Read-only dummy 8 bit	—
	02h	S	UNSIGNED16	ro	☒	Read-only dummy 16 bit	—
	03h	S	UNSIGNED24	ro	☒	Read-only dummy 24 bit	—
	04h	S	UNSIGNED32	ro	☒	Read-only dummy 32 bit	—
2080h	00h	S	ARRAY	ro	—	Status information	—
	01h	S	UNSIGNED8	ro	☒	Input/Output/Latch	—
	02h	S	UNSIGNED8	ro	☒	Collective info (Warning/Error/Reserve)	—
	03h	S	UNSIGNED8	ro	☒	DRVSTAT LSB	DRVSTAT
	04h	S	UNSIGNED8	ro	☒	"	DRVSTAT
	05h	S	UNSIGNED8	ro	☒	"	DRVSTAT
	06h	S	UNSIGNED8	ro	☒	DRVSTAT MSB	DRVSTAT
	07h	S	UNSIGNED8	ro	☒	TRJSTAT LSB	TRJSTAT
	08h	S	UNSIGNED8	ro	☒	"	TRJSTAT
	09h	S	UNSIGNED8	ro	☒	"	TRJSTAT
	0Ah	S	UNSIGNED8	ro	☒	TRJSTAT MSB	TRJSTAT
2081h	00h	S	INTEGER8	ro	—	Incr. actual position (bytewise/24-bit)	—
	01h	S	INTEGER8	ro	☒	Incr. position LSB	—
	02h	S	INTEGER8	ro	☒	Incremental position	—
	03h	S	INTEGER8	ro	☒	Incremental position	—
	04h	S	INTEGER8	ro	☒	Incremental position MSB	—
	05h	S	INTEGER24	ro	☒	Incremental position 24-bit	—
	06h	S	INTEGER32	ro	☒	Incremental position 32-bit	—
2082h	00h	S	INTEGER8	ro	—	32/24-bit positive latch	—
	01h	S	INTEGER32	ro	☒	32-bit latch	LATCH32
	02h	S	INTEGER24	ro	☒	24-bit latch	—

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
2083h	00h	S	INTEGER8	ro	—	32/24-bit negative latch	—
	01h	S	INTEGER32	ro	☒	32-bit latch	LATCH32N
	02h	S	INTEGER24	ro	☒	24-bit latch	—
2084h	00h	S	INTEGER8	ro	—	16-bit positive latch	—
	01h	S	INTEGER16	ro	☒	16-bit latch	LATCH16
2085h	00h	S	INTEGER8	ro	—	16-bit negative latch	—
	01h	S	INTEGER16	ro	☒	16-bit latch	LATCH16N
2086h	00h	S	UNSIGNED16	ro	☒	Position trigger word	POSRSTAT
2087h	00h	S	RECORD	ro	—	Latch positions Input 1	—
	01h	S	INTEGER32	ro	☒	Latch position input 1, positive edge	LATCHX32
	02h	S	INTEGER32	ro	☒	Latch position input 1, negative edge	LATCHX32N
	03h	S	UNSIGNED8	rw	☒	Latch enable for inputs 1 and 2	—
2090h	00h	S	ARRAY	ro	☒	DP-Ram variables, write only (PDO)	-
	01h	S	INTEGER32	ro	☒	DP-Ram Variable 1	DPRVAR1
	02h	S	INTEGER32	ro	☒	DP-Ram Variable 2	DPRVAR2
	03h	S	INTEGER32	ro	☒	DP-Ram Variable 3	DPRVAR3
	04h	S	INTEGER32	ro	☒	DP-Ram Variable 4	DPRVAR4
	05h	S	INTEGER32	ro	☒	DP-Ram Variable 5	DPRVAR5
	06h	S	INTEGER32	ro	☒	DP-Ram Variable 6	DPRVAR6
	07h	S	INTEGER32	ro	☒	DP-Ram Variable 7	DPRVAR7
	08h	S	INTEGER32	ro	☒	DP-Ram Variable 8	DPRVAR8
2600h	00h	S	INTEGER8	rw	—	1st receive-PDO select	—
2601h	00h	S	INTEGER8	rw	—	2nd receive-PDO select	—
2602h	00h	S	INTEGER8	rw	—	3rd receive-PDO select	—
2603h	00h	S	INTEGER8	rw	—	4th receive-PDO select	—
2721h	00h	S	INTEGER8	rw	—	Configuration receive-PDO 33	—
2A00h	00h	S	INTEGER8	rw	—	1st transmit-PDO select	—
2A01h	00h	S	INTEGER8	rw	—	2nd transmit-PDO select	—
2A02h	00h	S	INTEGER8	rw	—	3rd transmit-PDO select	—
2A03h	00h	S	INTEGER8	rw	—	4th transmit-PDO select	—
3100h	00h	S	Visible String	rw	—	ASCII character direction	—
3500h	00h	S	RECORD	ro	—	Start of Object Channel	—
	01h	S	UNSIGNED16	ro	—	Total number of Objects in Object Channel	MAXSDO
	02h	S	UNSIGNED16	ro	—	Lower limit value	—
	03h	S	UNSIGNED16	ro	—	Upper limit value	—
	04h	S	UNSIGNED16	ro	—	Default value	—
	05h	S	UNSIGNED8	ro	—	Data format of Object	—
	06h	S	UNSIGNED32	ro	—	Check data	—
	07h	S	—	—	—	reserved	—
	08h	S	—	—	—	reserved	—
3501h	00h	S	RECORD	ro	—	Acceleration ramp: speed control	—
	01h	S	UNSIGNED16	ro	—	Value	ACC
	02h	S	UNSIGNED16	ro	—	Lower limit value	—
	03h	S	UNSIGNED16	ro	—	Upper limit value	—
	04h	S	UNSIGNED16	ro	—	Default value	—
	05h	S	UNSIGNED8	ro	—	Data format of Object	—
	06h	S	UNSIGNED32	ro	—	Check data	—
	07h	S	—	—	—	reserved	—
	08h	S	—	—	—	reserved	—
6040h	00h	4	INTEGER16	wo	☒	DS402 control word	—
6041h	00h	4	INTEGER16	ro	☒	DS402 Status	—
605Ah		4	INTEGER16	rw	—	Quick Stop option code	—
6060h	00h	4	INTEGER8	wo	☒	Mode of Operation	—
6061h	00h	4	INTEGER8	ro	☒	Display Mode of Operation	—
6063h	00h	4	INTEGER32	ro	☒	Incr. actual position	—
6064h	00h	4	INTEGER32	ro	☒	Actual position (taking account of gearing factors)	—
606Ch	00h	4	INTEGER32	ro	☒	Actual speed (mode: pv)	—
6077h	00h	4	INTEGER16	ro	☒	Torque actual value	IQ
607Ah	00h	4	INTEGER32	rww	☒	Target position (mode: pp)	—
607Bh	00h	4	ARRAY	ro	—	position_range_limit	—
	01h	4	INTEGER32	rw	—	min_position_range_limit	—
	02h	4	INTEGER32	rw	—	max_position_range_limit	—

Index	Sub-index	DEF	Data format	Access	PDO mapp.	Description	ASCII Object
$607C_h$	00_h	4	INTEGER32	rww	☒	Reference offset	—
6081_h	00_h	4	UNSIGNED32	rww	☒	Velocity (Mode: pp)	—
6083_h	00_h	4	UNSIGNED32	rww	☒	Acceleration (Mode: pp pv)	—
6084_h	00_h	4	UNSIGNED32	rww	☒	Deceleration (Mode: pp pv)	—
6086_h	00_h	4	INTEGER16	rww	☒	Type of motion block ramp (motion_profile_type)	—
6089_h	00_h	4	INTEGER8	rw	—	Notation index: position (not supported)	—
$608A_h$	00_h	4	UNSIGNED8	rw	—	Dimension index: position (not supported)	—
$608B_h$	00_h	4	INTEGER8	rw	—	Notation index: speed (not supported)	—
$608C_h$	00_h	4	UNSIGNED8	rw	—	Dimension index: speed (not supported)	—
$608D_h$	00_h	4	INTEGER8	rw	—	Notation index: acceleration (not supported)	—
$608E_h$	00_h	4	UNSIGNED8	rw	—	Dimension index: acceleration (not supported)	—
6093_h	00_h	4	UNSIGNED32	rw	—	Position factor	—
	01_h	4	UNSIGNED32	rww	☒	Numerator	PGEARO
	02_h	4	UNSIGNED32	rww	☒	feed_constant	PGEARI
6094_h	00_h	4	UNSIGNED32	rw	—	Velocity factor: encoder	—
	01_h	4	UNSIGNED32	rw	—	Numerator	—
	02_h	4	UNSIGNED32	rw	—	Divisor	—
6097_h	00_h	4	UNSIGNED32	rw	—	Acceleration factor	—
	01_h	4	UNSIGNED32	rw	—	Divisor	—
	02_h	4	UNSIGNED32	rw	—	Acceleration factor	—
6098_h	00_h	4	INTEGER8	rww	☒	Homing type	—
6099_h	00_h	4	ARRAY	ro	—	Homing velocity	—
	01_h	4	INTEGER32	rw	☒	Homing velocity (homing to references switch)	VREF
	02_h	4	INTEGER32	ro	☒	Homing velocity (homing to resolver zero mark)	VREF
$609A_h$	00_h	4	UNSIGNED32	rw	☒	Homing acceleration	—
$60C2_h$	00_h	4	RECORD	ro	—	Interpolation time period	PTBASE
	01_h	4	UNSIGNED8	rw	—	Interpolation time units	—
	02_h	4	INTEGER16	rw	—	Interpolation time index	—
$60FF_h$	00_h	4	INTEGER32	rw	☒	Speed setpoint	—

6.4 New configuration of SERVOSTAR 400/600

In SERVOSTAR 400/600 there are a number of parameters, that can only be integrated into the current firmware program and produce their proper effect for the first time during the initialization phase, when the amplifier is switched on. As a result, a change of one of these parameters means that all parameters have to be saved, and the amplifier must be restarted afterwards. This can be carried out via the CAN-bus, by using the following Objects:

1. Write Object $35EB_h$ Sub-index 01_h, value 0, save command
2. Write Object 3632_h Sub-index 01_h, value 0, restart amplifier

6.5 Index

! 1000h . . . 31
1001h . . . 31
1002h . . . 32
1003h . . . 33
1004h . . . 34
1005h . . . 35
1006h . . . 35
1007h . . . 36
1008h . . . 36
100Ah . . . 36
100Bh . . . 37
100Ch . . . 37
100Dh . . . 37
100Eh . . . 38
100Fh . . . 38
1010h . . . 39
1012h . . . 40
1013h . . . 40
1014h . . . 40
1018h . . . 41
1400-1403h . . . 47
1600-1603h . . . 47
1800-1803h . . . 51
1A00-1A03h . . . 51
2014-2017h . . . 52
2020h . . . 68
2022h . . . 71
2024h . . . 66
2026h . . . 78
2030h . . . 90
2031h . . . 91
2050h . . . 77
2060h . . . 65
2061h . . . 65
2070h . . . 82
2071h . . . 92
2082h . . . 78
2083h . . . 79
2084h . . . 79
2085h . . . 80
2087h . . . 81
2090h . . . 90
2600-2603h . . . 47
2721h . . . 47
2A00-2A03h . . . 51
3500h . . . 107
6040h . . . 55
6041h . . . 56
6060h . . . 58
6061h . . . 59
6063h . . . 95
6064h . . . 95
606Ch . . . 93
6077h . . . 89
607Ah . . . 100
607Bh . . . 101
607Ch . . . 96
6081h . . . 101
6083h . . . 102
6084h . . . 102
6086h . . . 103
608Bh . . . 60
608Ch . . . 60
6093h . . . 61
6094h . . . 62
6097h . . . 64
6098h . . . 97
6099h . . . 98
609Ah . . . 98
60C2h . . . 89
60FFh . . . 94

A Abbreviations . . . 9
Acknowledge lag/contouring error . 56
Acknowledge response monitoring . 56
Actual values . . . 82
Additional documentation . . . 7

B Basic data types . . . 19
Basic features . . . 9
Bus cable . . . 13

C COB-ID . . . 18
Cable length . . . 13
Communication Objects . . . 20
Communication faults . . . 10
Communication profile . . . 17
Configuration parameter . . . 15
Control word . . . 55

D Data Frame . . . 17
Data transfer functions . . . 9
Data types . . . 18
Device control . . . 52
Digital setpoint . . . 65
Drive profile . . . 29

E Emergency Message . . . 29
Emergency Object . . . 21
Extended data types . . . 20

F Factor Groups . . . 59

H Homing Mode . . . 96

I Inhibit time . . . 51
Installation . . . 11

L Latch function . . . 78

M Mapping . . . 42
Mixed data types . . . 19

N Network Management Object . . . 21
New configuration . . . 137
Nodeguard . . . 27

O Object Channel 107
Object Dictionary 131
Operating mode 58
P Permitted use 8
Position Control Function. 94
Positioning functions. 9
Process Data Object 25
Profile Position Mode. 99
Profile Velocity Mode. 93
R Receive-PDOs 43
Remote Frame 17
Response monitoring. 37
S SDO abort codes. 25
Service Data Object 23
Setup. 14
Setup examples. 117
Setup functions 9
Station address. 12
Status machine. 53
Status word. 56
Synchronization Object. 21
System requirements. 10
T Target group. 7
Termination resistance 13
Time Stamp Object. 21
Transmission modes 26
Transmission procedure 10
Transmission rate 10
Transmission rate setting. 12
Transmit-PDOs. 48
Trigger modes 27
U Use as directed 8

Service

We are committed to quality customer service. In order to serve in the most effective way, please contact your local sales representative for assistance.
If you are unaware of your local sales representative, please contact the Customer Support.

Europe

KOLLMORGEN
Internet www.kollmorgen.com/en-gb
Archiv www.wiki-kollmorgen.eu
Support https://kdn.kollmorgen.com/
E-Mail technik@kollmorgen.com
Tel.: +49 (0)2102 - 9394 - 0
Fax: +49 (0)2102 - 9394 - 3155

KOLLMORGEN
EU Website

European
File Archive

North America

KOLLMORGEN
Internet www.kollmorgen.com/en-us
Support https://kdn.kollmorgen.com/
E-Mail support@kollmorgen.com
Tel.: +1 - 540 - 633 - 3545
Fax: +1 - 540 - 639 - 4162

KOLLMORGEN
US Website

KOLLMORGEN
Developer Network

South America

KOLLMORGEN
Internet www.kollmorgen.com/pt-br
Support https://kdn.kollmorgen.com/
E-Mail contato@kollmorgen.com
Tel.: +55 11 4615 - 6300

KOLLMORGEN
Brazil Website

Asia

KOLLMORGEN
Internet www.kollmorgen.cn
Support https://kdn.kollmorgen.com/
E-Mail sales.china@kollmorgen.com
Tel: +86 - 400 661 2802

KOLLMORGEN
CN Website