Programming manual

Language	**English**
	Translation
Document No.	5.02065.04
Part No.	372860
Status	02.05.2012

be in motion be in motion

BAUMÜLLER

CANopen Slave

Programming manual

Version 03.12

E	5.02065.04

Read the Operating Manual before starting any work!

Preliminary information **Caution**: The following applies if this document is identified as preliminary information:
This version contains preliminary technical information which the users of the devices and functions described are intended to receive in advance, in order to be able to make arrangements for any changes or functional enhancements that may be required.
This information should be considered to be provisional, as it has not yet been subjected to the final Baumüller internal review process. In particular this information is still subject to change, so that no legal obligation can be deduced on the basis of this preliminary information. Baumüller accepts no liability for damages that may arise from this possibly erroneous or incomplete version.
Should you detect or suspect errors of content and/or serious technical errors in this preliminary information, we ask you to contact the Baumüller support person responsible for you and inform us of your findings and comments so that they might be considered and possibly incorporated when the preliminary information becomes finalized (reviewed by Baumüller).
The conditions noted in the following section under "Liability" do not apply in the case of preliminary information.

Obligation This Programming manual is a part of the device/machine. This Programming manual must be accessible to the operator at all times and be in a legible condition. When the device/machine is sold/relocated, this Programming manual must be passed on together with the device/machine by the owner.
After the device/machine is sold, this original and all copies must be handed over to the purchaser. After disposal or other end of service life, this original and all copies must be destroyed.

When this Programming manual is handed over, the corresponding Operating Manuals with earlier issue dates become invalid.
Please note that specifications/data/information are the **current values on the date of printing**. These specifications are **not legally binding** for measuring, computation and costing.
Baumüller Nürnberg GmbH reserves the right to change the technical data and operation of Baumüller products within the framework of its own further development of the products.

However no guarantee can be provided regarding the freedom from errors of this Programming manual, unless otherwise described in the General Conditions for Sales and Supply.

© **Baumüller Nürnberg GmbH**

Ostendstr. 80 - 90
90482 Nürnberg
Germany

Tel. +49 9 11 54 32 - 0
Fax: +49 9 11 54 32 - 1 30

E-Mail: mail@baumueller.de
Internet: www.baumueller.de

Contents

1

INTRODUCTION

The program manual is an important part of your b maXX® 4400 device. Therefore please, completely read this manual, before starting operation, last but not least on behalf of your own security. In this documentation you will learn how Baumüller Nürnberg GmbH has implemented the CANopen interface on the option module CANopen slave for the b maXX®4400 device series.

The introduction contains general information on the CANopen slave option module.

1.1 General information

The CANopen option module connects the b maXX® 4400 via the CAN bus with other CANopen nodes (e. g. PC, SPS, further b maXX® 4400, I/O modules).

Information according option and function modules for the device series bmaXX®4400 is to be found in the documentation 5.01040.

Information according the programming of the b maXX®4400 controller is to be found in the parameter manual 5.02017.

1.2 Mounting and installation

The mounting of option module CANopen slave we describe in the documentation 5.02014.

1.3 Address setting

The address setting and the setting of the baudrate of the option module CANopen slave we describe in the manual 5.02014.

1.4 EDS file

The EDS file is an ASCII file and is used in describing the functional range of a CANopen device. It is an electronic data sheet of the CANopen device.
The EDS file is used by the CANopen masters or bus configurators. The EDS file contains information about the supported objects, baud rates and other features of the slave.

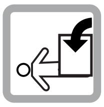
The extension of name of the EDS file is *.eds.

2

FUNDAMENTAL SAFETY INSTRUCTIONS

In this chapter we describe the dangers, which can occur during parameterization of the Baumüller b maXX®4400 controller unit and we explain the meaning of the information sign.

2.1 Safety notes and instructions



(WARNING)

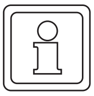
The following **can occur**, if you disregard this warning instruction:

- serious personal injury
- death

The danger is: **mechanical and electrical cause.** *The modification of parameters affects the action of the Baumüller unit and consequently the action of the installation and its components. If you change the adjustments of the parameters, you may cause dangerous actions to the construction and/or of its components.*

After each modification of the parameter settings, execute a commissioning with consideration to all safety instructions and safety regulations.

2.2 Information sign



NOTE

This note is a very important information.

2.2 Information sign

3

BASIC PRINCIPLES OF CAN/ CANOPEN

3.1 Literature on CAN

On behalf of basic information with reference to CAN we recommend the following literature:

- 'Etschberger'
  CAN Controller Area Network
  Konrad Etschberger
  Carl Hauser Verlag München Wien

- 'Lawrenz'
  CAN Controller Area Network. From Theory to Practical Applications
  Wolfhard Lawrenz
  Hüthig Verlag

- 'Zeltwanger'
  CANopen
  Holger Zeltwanger
  VDE-Verlag

- www.can-cia.de
  CAN in Automation e.V.
  Am Weichselgarten 26
  D-91058 Erlangen

3.2 Basic principles CAN

The CAN field bus is implemented using a line structure. As a physical fundamental principle of data transfer a triple wire is used, which has the connections CAN_High, CAN_Low and CAN_Ground. CAN uses a ground-symmetric transfer, in order to suppress common-mode interferences. Therefore differential inputs are evaluated.

Network

CAN is a multi master network. Every user can have access to and be active on the bus, with equal priority. CAN uses object-oriented addressing, i. e. the transferred message is identified by an identifier defined network-wide. The identifier shows the coded message name.

Bus access

The bus access is made via the CSMA/CA procedure (carrier sense multiple access/collision avoidance). As each user is entitled to begin with the sending of the message after recognition of the necessary bus quiescence, collisions can occur. This is avoided by the bitwise arbitration of the messages to be sent. Thereby it is differed between two bus levels, a dominating level, logical bit value 0, and a recessive level, logical bit value 1. The worst case would be that all users, which are willing to send simultaneously start the message sending on the bus. If a recessive bit of a user is overwritten by a dominant bit of another one, then the „recessive“ node draws back from the bus and after detecting bus quiescence, attempts once more to transmit its message. Therewith is guaranteed, that the most important message (with the lowest identifier) is free from collisions and is transmitted without delay. For this reason it is necessary, that each identifier must be assigned at the CAN bus once only.

Identifier

There are different identifiers available in the CAN specification CAN 2.0A 2032 (CiA). Each user can transmit unrequested (multi-master-capability). A transmitter sends its message to all CAN nodes (broadcast) and each then decides on the basis of the identifier whether they will continue processing the message or not.

Error

In a CAN data telegram up to eight bytes of service data can be transmitted. For error or overload signaling, a CAN node can send error or overload telegrams. This occurs on layer 2 of the OSI/ISO reference model (the data link layer), that means independent of the application. Due to high quality error detection and handling on layer 2, a Hamming distance (unit of error detection) of HD = 6 is achieved, i. e. a maximum of five simultaneously-occurring bit errors within a telegram are reliably recognized as error.

3.3 Basic principles of CANopen

CANopen is an open and hence manufacturer-independent field bus system defining layers 1 and 2 of the CAN standard.

CAL specification

The CANopen protocol is based on the CAL specification (layer 7 protocol).
With CANopen, profiles are differentiated. The communication profile (DS 301) defines the method of data exchange and general definitions applicable for all devices.

Device profile

Device profiles contain application- and device-specific definitions describing the contents-related meaning of data and device functionality. Device profiles exist for drives, I/O modules, encoders or programmable devices. The CANopen slave option module for the b maXX ®4400 controller is implemented in accordance with the device profile (drives and motion control).

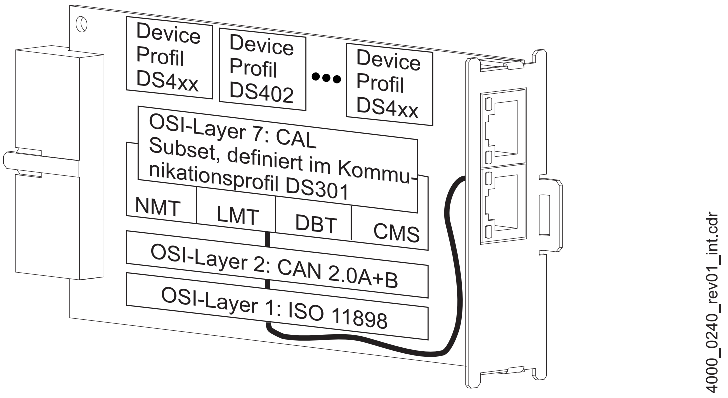


Figure 1: CANopen profile structure

Object directory

The central element of every CANopen device is its object directory .CANopen-device.

Index (hex)	Object
0000	Not used
0001_{hex} - $001F_{hex}$	Static data types
0020_{hex} - $003F_{hex}$	Complex data types
0040_{hex} - $005F_{hex}$	Manufacturer-specific data types
0060_{hex} - $007F_{hex}$	Device profile-specific static data types
0080_{hex} - $009F_{hex}$	Device profile-specific dynamic data types
$00A0_{hex}$ - $0FFF_{hex}$	Reserved
1000_{hex} - $1FFF_{hex}$	Range for the communication profile
2000_{hex} - $5FFF_{hex}$	Range for manufacturer-specific objects
6000_{hex} - $9FFF_{hex}$	Range for the device profile
$A000_{hex}$ - $AFFF_{hex}$	Control objects for devices programmable in accordance with IEC 61131-3 (DSP 405)

The objects are always addressed via an index (16 bit) and additionally via a subindex (8 bit).

CANopen differentiates between 4 types of messages:

- Administrative messages (e. g. network-management NMT, layer-management LMT)
- Service data (SDO)
- Process data (PDO)
- Predefined messages (e. g. synchronization, time stamp, emergency)

NMT

The communication states of the device are controlled and monitored by means of NMT (network management) services.

SDO

SDOs are used for the transfer of greater volume of data with low priority (service data) In addition, a data block with more than 4 bytes of user data is segmented and distributed across several SDOs by means of the CANopen protocol (SDO segmented transfer). Data volumes of 4 bytes maximum are transmitted with one SDO (SDO expedited transfer). Typically, SDOs are used for device configuration. SDOs are transmitted asynchronous and are accepted by the receiver. All entries in the object directory can be accessed by means of SDOs.

PDO

The function of PDOs is the exchange of process data (data with high priority). PDOs can be transmitted both synchronously and asynchronously. They have broadcast character and are not confirmed by the receiver.

Synchronous means that transmission depends on the synchronization object. The content of PDOs must be established by the user via SDOs (variable PDO mapping). This mapping must be completed before beginning process data communication. Default mapping is specified in the device profiles.

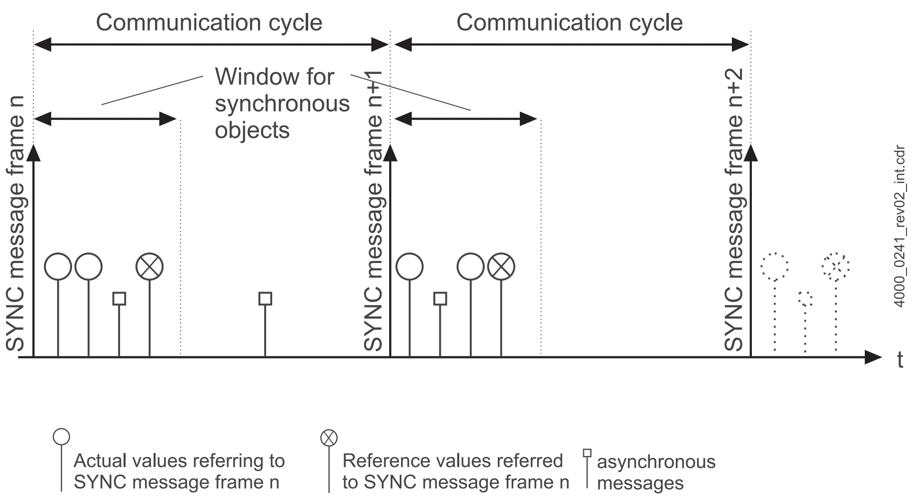


Figure 2: PDO transmission types

PDO communication is triggered either by the occurrence of certain events (e. g. reception of a SYNC telegram or value modification) or time-controlled.

To be able to establish peer-to-peer communication between master and slave directly after boot-up, predefined identifier assignment is available. This identifier assignment can be reconfigured by the user.

Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Function code				Module ID						

For the 7 bits for the module IDs, a maximum number of 127 nodes results per CANopen network.

Object	Binary function code	Resultant COB ID	Object
NMT	0000	0	
SYNC	0001	128	1005_{hex}, 1006_{hex}
EMERGENCY	0001	129 - 255	1014_{hex},1015_{hex}
PDO1 (TX)	0011	385 - 511	1800_{hex}
PDO1 (RX)	0100	513 - 639	1400_{hex}
PDO2 (TX)	0101	641 - 767	1801_{hex}
PDO2 (RX)	0110	769 - 895	1401_{hex}
PDO3 (TX)	0111	896 - 1023	1802_{hex}
PDO3 (RX)	1000	1025 - 1151	1402_{hex}
PDO4 (TX)	1001	1153 - 1279	1803_{hex}
PDO4 (RX)	1010	1281 - 1407	1403_{hex}
SDO (TX)	1011	1409 - 1535	1200_{hex}
SDO (RX)	1100	1537 - 1663	1200_{hex}
Nodeguard	1110	1793 - 1919	$100C_{hex}$, $100D_{hex}$

CANopen defines a network boot up. The simple boot up contains 4 communication states:

- INITIALIZATION
- PRE-OPERATIONAL
- STOPPED
- OPERATIONAL

The individual state transitions are triggered by NMT commands. After initializing, the CANopen slave option module switches automatically to the PRE-OPERATIONAL state. Additional data is available in ▹Communication state machine◃ on page 43.

3.4 Operating modes supported by device profile DSP 402

3.4.1 Brief overview

The following operating modes are supported, i. e. all mandatory objects are existent via the CANopen slave option module.	
Device control	Optional objects are completely existent
Homing objects	Optional objects are completely existent
Objects of position mode profiles	Optional objects are partly existent
Position control function	Optional objects are partly existent
Velocity mode objects	Optional objects are partly existent
Velocity mode profile objects	Optional objects are partly existent
Common entries in the object dictionary (no mandatory objects existent)	Optional objects are partly existent
Interpolated position mode	Optional objects are completely existent

The following operating modes are not supported, i. e. at least one mandatory object is not existent, also optional objects can be existent.	
Profile torque mode	Two objects

3.4.2 Operating modes and field bus objects

Operating mode	
Field bus objects	Field bus name

Homing mode objects **all mandatory objects and all optional objects are supported** **(homing)**		
6098_{hex}	mandatory	homing_method
6099_{hex}	mandatory SIX 0 = 2	homing_speed
$607C_{hex}$	optional	home_offset
$609A_{hex}$	optional	homing_acceleration

Device control **all mandatory objects and all optional objects are supported** **(device control)**		
6040_{hex}	mandatory	control word
6041_{hex}	mandatory	statusword
6060_{hex}	mandatory	modes_of_operation
6061_{hex}	mandatory	modes_of_operation_display
$605A_{hex}$	optional	quick_stop_option_code
$605B_{hex}$	optional	shutdown_option_code
$605C_{hex}$	optional	disable_operation_option_code
$605D_{hex}$	optional	halt_reaction_option_code
$605E_{hex}$	optional	fault_reaction_option_code

Torque mode profile objects **one optional object is supported** **(torque control)**		
6072_{hex}	optional	max_torque
6077_{hex}	optional	torque_actual_value

Objects of position mode profiles all mandatory objects, partially optional objects are supported (positioning)			
$607A_{hex}$	mandatory		target_position
$607D_{hex}$	optional	SIX 0 = 2	software_position_limit
$607F_{hex}$	optional		max_profile_velocity
6080_{hex}	optional		max_motor_speed
6081_{hex}	mandatory		profile_velocitiy
6083_{hex}	mandatory		profile_acceleration
6084_{hex}	mandatory		profile_deceleration
6085_{hex}	optional		quick_stop_deceleration
6086_{hex}	mandatory		motion_profile _type

Velocity mode profile objects all mandatory objects, partially optional objects are supported (speed control)		
$606A_{hex}$	mandatory	sensor_selection_code
6069_{hex}	mandatory	velocity_sensor_actual_value
$606B_{hex}$	mandatory	velocity_demand_value
$606C_{hex}$	mandatory	velocity_actual_value
$606F_{hex}$	optional	velocity_threshold
$60FF_{hex}$	mandatory	target_velocity
$60F_{hex}$	optional	max_slippage

Interpolated position mode all objects, objects are supported (positioning control)			
$60C0_{hex}$	optional		Interpolation sub mode select
$60C1_{hex}$	optional	SIX 0 = 1	Interpolation data record
$60C2_{hex}$	mandatory	SIX 0 = 2	Interpolation time period
$60C3_{hex}$	optional	SIX 0 = 2	Interpolation sync definition
$60C4_{hex}$	optional	SIX 0 = 6	Interpolation data configuration

Position control function **all mandatory objects, partially optional objects are supported** **(positioning control)**			
6067_{hex}	optional		position_window
6068_{hex}	optional		position_window_time
6064_{hex}	mandatory		position_actual_value
6063_{hex}	optional		position_actual_value*
6062_{hex}	optional		position_damand_value
6066_{hex}	optional		following_error_time_out
$60FB_{hex}$	optional	SIX 0 = 28	position_control_parameter_set

Velocity mode objects **all mandatory objects, partially optional objects are supported** **(speed control)**			
6042_{hex}	mandatory		vl_target_velocity
6043_{hex}	mandatory		vl_velocity_demand
6044_{hex}	mandatory		vl_control_effort
6045_{hex}	optional		vl_manipulated_velocity
6048_{hex}	mandatory	SIX 0 = 2	vl_velocity_acceleration
6049_{hex}	mandatory	SIX 0 = 2	vl_velocity_deceleration
6046_{hex}	mandatory	SIX 0 = 2	vl_velocity min_max_amount
$604C_{hex}$	optional	SIX 0 = 2	vl_manipulated_velocity
$604D_{hex}$	optional		vl_pole_number
$60F_{hex}$	optional		vl_ramp_function_time
6050_{hex}	optional		vl_slow_down_time
6051_{hex}	optional		vl_quick_stop_time

Common entries in object dictionary **No mandatory objects available, partially optional objects are supported** **(general inputs in object directory)**			
$60FD_{hex}$	optional		digits_inputs
6007_{hex}	optional		abort_connection_option_code
6510_{hex}	optional	SIX 0 = 08	drive_date

Factor group No mandatory objects available, partially optional objects are supported (user units group)			
6092_{hex}	optional	SIX 0 = 2	feed_constant

4

COMMUNICATION TO THE b maXX®CONTROLLER

In this chapter we describe the data communication between the CANopen slave option module and the b maXX®4400 device.

4.1 Communication flow

The CANopen option module exchanges data with the bmaXX®4400 controller via a dual port RAM. This data exchange takes place within a certain sampling time via the BACI interface (Baumüller bus).

The CANopen slave option module initiates communication with the b maXX®4400 controller. During communication, two different types of data are transferred:

- Process data
- Service data

Process data is always transferred cyclically. In the remaining time, service data is transferred. Process data transfer takes place within an adjustable time period, the SYNC interval. At the same time the setpoint and the actual values are transferred during the SYNC interval, each with a varying offset.

The cycle time of the SYNC telegram must agree with the BACI cycle time at the operating modes position control and synchronous operation. At other operating modes the BACI cycle time can differ from the cycle time of the SYNC telegram. The SYNC telegram is perceived as a trigger condition here.

NOTE

Cyclic communication is active only in the CANopen OPERATIONAL communication state.

4.2 Parameterizing the BACI communication times

Between the option module CANopen slave and the b maXX®controller 8 setpoints (exception: FBO $607A_{hex}$target position is 2 FBOs, because it is shown on 2 controller parameters) and 8 actual values can be exchanged as process data in a communication cycle. Which setpoints and which actual values they exchange is established in the mapping objects in the CANopen slave option module (setting via SDO through the CANopen master or default setting: see the chapter ▷Data exchange and parameterization◁ from page 31). How to parameterize communication is revealed in this chapter.

Communication between the CANopen slave option module and b maXX® controller is parameterized preferably via ProDrive/WinBASS II. It is also possible to parameterize via field bus.

The communication cycle time (rate setpoints, actual values), the setpoint cycle offset and the actual value cycle offset is set via the ProDrive/WinBASS II page ’BACI’ (option module 1).

The b maXX® controller initiates a communication time slice every 125 µs, in which process data setpoints or process data actual values are transferred.

The communication cycle time is a multiple of the communication time slice (every 125 µs). In the ’setpoint rates, actual value rates’ edit box, only the factor is specified, i. e. the value in the edit box „rate setpoint values, rate actual values“ is calculated as follows:

$$\text{Cycle time setpoint values, actual values (P800)} = \frac{\text{communication cycle time (in µs)}}{125\ \text{µs}}$$

Examples:

Communication cycle time = 2000 µs ⇒ setpoint, actual values = 16 (P800)

Communication cycle time = 4000 µs ⇒ setpoint, actual values = 32 (P800)

It is recommended to set the cycle time of the BACI (P800) to 1000 µs is equivalent to 8. Excepted from this are the operating modes position control and synchronous operation. The settings see ▷Parameterizing the of BACI communication times at the operation modes position control, synchronous operation and IP mode◁ from page 21.

- Cycle offset set values (P818) = 7
- Cyclle offset actual values = 0

NOTE

When establishing BM_u_Baci1M1Period note the following: BACI can only be accessed every 125 µs.

The process data setpoints and the process data actual values are transferred in various communication time slices. Therefore there is a different cycle offset for the setpoints than for the actual values. Cycle offset is nothing else than the number of the communication time slice in which the data is transferred.

NOTE

Other settings for the cycle offset of the setpoints and actual values are possible, but, due to this the time response changes.

You will find the parameter numbers of the setpoints and actual values (P801 to P816) on the ProDrive/WinBASS II page 'BACI' (option module 1). These are only for display, because the setting of the parameter numbers for the process data exchange in the mapping objects are defined on the option module CANopen slave. The preselected settings for the offsets must be saved in the data set and the controller must be booted again.

NOTE

If cyclic communication is interrupted, e. g. at transition from OPERATIONAL to PRE-OPERATIONAL the error/warning Alive Counter or the error cyclic communication can occur.

NOTE

CANopen State Machine

At transition from PRE-OPERATIONAL to OPERATIONAL the parameter numbers are assigned to mapping. This assignment is time-consuming and can last several milliseconds (up to 11 ms, according to time setting of BACI also longer), during this time no PDO is send and also no RX-PDO is processed.

4.2.1 Parameterizing the of BACI communication times at the operation modes position control, synchronous operation and IP mode

At the operation modes position control, synchronous operation and IP mode the cycle times of BACI must comply with the cycle times of the field bus (sync-telegram), because the synchronization of the controller is effected by means of the sync-telegram.

The smallest value for the "BACI Cycle Time" is 2 ms, which means that the smallest communication time slice is 2 ms for this operation modes. The greatest value for the "BACI Cycle Time" is 8 ms.

The following considerations were made, which can help to get a sense of knowing, how many telegrams are possible at which baudrate in CANopen.

At a 1000 Kbit baudrate, e.g., the transfer of an eight byte telegram takes 130 - 140 µs; at 500 Kbit, the transmission time doubles.

Consideration of a 1 Mbit-baudrate (these are approximate values only, because, if service data is transferred also, the CAN-bus is reaches its limits and accordingly synchronization-trouble may occur).

- At 2 ms and at a maximum of 4 slaves in the ring, one PDO can be processed per direction.
- At 4 ms and at a maximum of 7 slaves in the ring, one PDO can be processed per direction.

- At 8 ms and at a maximum of 14 slaves in the ring, one PDO can be processed per direction.

The process data setpoints and the process data actual values are transferred in different communication time slices to the controller. Therefore, for the setpoints another BACI-cycle offset than for the actual values is defined. The BACI-cycle offset is the number of the communication time slice, in which the data are transferred.

At a cycle time 2000 µs	Setpoints = 15	Actual value = 0
At a cycle time 4000 µs	Setpoints = 31	Actual value = 0
At a cycle time 8000 µs	Setpoints = 63	Actual value = 0

The settings must be saved in the data set and the controller must be rebooted.

Other settings for the cycle-offset of the setpoints and actual values are possible, but it can happen, that the data acceptance slides into the next cycle and accordingly the data-acceptance-time behavior is changed, thereby.

NOTE

The settings are valid for the operating modes position control and synchronous operation, only.

The parameter numbers of the setpoints and of the actual values (P801-P816) are found on the WinBASS II-page "BACI" (option module 1). These are for display only, because the setting of the parameter numbers for the process data exchange is specified in the mapping objects on the option module CANopen-slave.

4.3 Configuration possibilities of CANopen option card in ProDrive/WinBASS II

ProDrive/WinBASS II „option module G/H - configuration 1“ (P830 / P840).

Dependent upon which slot the CANopen option card is plugged in.

NOTE

Settings result in a modified behavior!

4.3.1 Differentiation of the boot-up telegrams

Standard is the boot-up behavior according to DS 301 V4.

For differentiation of the boot-up telegram according to DS 301 V3 or DS 301 V4 the parameter ’option module G - configuration 1’ or ’option module H - configuration 1’ in the b maXX® controller can be set. Dependent upon the slot in which the CANopen option card is plugged in.

Bit 0 ⇒ 0: acc. to DS 301 V4

Bit 0 ⇒ 1: acc. to DS 301 V3

1 Boot-up acc. to DS 301 V4, boot-up telegram with ID = 700_{hex} + node-ID
DLC = 1 byte 0 is filled with data = 0.

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
701_{hex}	01_{hex}	00_{hex}							

DSP 301 V4

2 Boot-up according to DS 301 V3, ID = 80_{hex} + node-ID,
DLC = 0 and following ID = 80 $_{hex}$ + node-ID,
DLC = 8 byte 0-7 filled with data = 0 (reset of the error register).

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
81_{hex}	00_{hex}								
81_{hex}	08_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

DSP 301 V3

4.3.2 Initialization of the CAN after to bus off

Bit 1 ⇒ 0: bus off remains

Bit 1 ⇒ 1: initialization of the CAN after a waiting time of 200 ms, until the bus off has been removed.

4.3.3 Settings from controller version V03.00 (LC2) on

(Changes of units of a few FBOs, e. g. units 1/10 RPM)

Bit 2 ⇒ 0: resolution for speed 1 RPM

Bit 2 ⇒ 1: new functions, innovations are specified in the further procedure
resolution for speed 1/10 RPM
The following objects are concerned by this:
6042_{hex}, 6043_{hex}, 6045_{hex}, 6048_{hex} SIX 1, 6049_{hex} SIX 1, $606B_{hex}$, $606C_{hex}$, $60FF_{hex}$

4.3.4 Default mapping according to CANopen Standard DSP 402 changeable

Bit 3 ⇒ 0: default mapping according to DSP 402.
As well as FBO 6007_{hex} Node Guarding reaction = 0 (CANopen standard).

Bit 3 ⇒ 1: default mapping where all FBO objects are set to zero, trigger types remain according to DSP 402 on default.
FBO 6007_{hex} Node guarding reaction then is set to 3.

4.3.5 Node guarding reaction

Bit 4 ⇒ 0: CANopen state does not change to PRE-OPERATIONAL, but remains in current state (standard).

Bit 4 ⇒ 1: up to now, CANopen state always changes to PRE-OPERATIONAL after a node guard event has occurred.

4.3.6 CANopen offset

Bit 5 ⇒ 0: conversion of numerical scale from U32 to INT 32, at positioning an offset of 2^{31} according to direction is added/subtracted on the associated FBOs..

Bit 5 ⇒ 1: no offset is used.

For further information see ▹CANopen offset◃ on page 29.

4.3.7 Reset after a SYNC telegram failure

Bit 6 ⇒ 0: no reset after a SYNC telegram failure

Bit 6 ⇒ 1: reset after a SYNC telegram failure

4.3.8 Deactivation of synchronize-mechanism to the controller

The mechanism is used for the compensation of greater jitter of the SYNC telegram. It is implemented in the FPGA on the option card.

Bit 8 ⇒ Mechanism activated

Bit 8 ⇒ Mechanism deactivated

4.3.9 EMCY error code

Not defined controller errors in the DSP 402 are added to the manufacturer-specific error code $FF00_{hex}$
e. g. Controller error number 167 (no release of brake), is then brake, then is issued with $FF00_{hex}$ + $00A7_{hex}$ (No. 167) = $FFA7_{hex}$.

Bit 9 ⇒ 0: new behavior as described above standard e. g.: $FFA7_{hex}$

Bit 9 ⇒ 1: prior behavior e. g.: $FF00_{hex}$

NOTE

CANopen State Machine

At transition of PRE-OPERATIONAL to OPERATIONAL the parameter numbers are assigned to the mapping. This assignment is time-consuming and can last several milliseconds (up to 11 ms according to time settings of BACI also longer), during this time no PDO is send and also no RX-PDO is processed.

4.4 General comments to the CANopen option card

Important: Modifications, which occur via Win BASS II, are not updated or noticed automatically on the CANopen option card. Therefore the access to the controller should, as far as possible, occur with FBO via CANopen.

Modifications via ProDrive/WinBASS II e. g. can not be noticed at switchover between relative and absolute positioning modes on the CANopen option card during positioning operation. Thereto also belong, e. g. modification of operation mode via ProDrive/ WinBASS II. The switchover/modification should only occur via the CANopen.

Following controller parameters are affected:

P0830/ P0840	no FBO according to DSP 402, access only via the manufacturer-specific object possible (433E$_{hex}$ /4348$_{hex}$)
P0304	(FBO 6060$_{hex}$),
P1031	(FBO6080$_{hex}$),
P3050	(FBO 6092$_{hex}$ SIX1),
P3051	(FBO 6092$_{hex}$ SIX2),
P0601	(internal switchover on the CANopen option card by the control word bit 6, operation mode positioning, relative and absolute modes),
P0531	(FBO 1006$_{hex}$)
P0532	(FBO 1006$_{hex}$),
P3314	(FBO 604C$_{hex}$ SIX1),
P3315	(FBO 604C$_{hex}$ SIX2)
P1190	(FBO 6086$_{hex}$).

If it is desired to make the modifications via ProDrive/WinBASS II, there is the possibility to update the parameters at transition of the CANopen state machine OPERATIONAL to PREOPERATIONAL or to STOP. Furthermore after saving the data set and making a power-down and power-up of the controller an updating occurs.

The following FBO can be entered via the field bus with SIX 1/2:

FBO 6048$_{hex}$ SIX1, SIX2, for the determination of acceleration (P1172),

FBO 6049$_{hex}$ SIX1, SIX2, for the determination of delay (P1173).

The scales of the FBO can deviate from those of ProDrive/WinBASS II.

e. g.: input of positioning speed via the FB in [m/S] and input via
ProDrive/WinBASS II in [INC/ms] accords to a difference of factor 1000.

4.4 General comments to the CANopen option card

4.4.1 Application parameters

The application parameters P3314, P3315, P3329, P3330 are used on the CANopen option card and must not be used otherwise.

4.4.2 Speed profile at positioning FBO 6086_{hex}

The changes are valid for the handling of SW/HW limit switches as well as for homing (e. g. activate limits/warnings). Saving in the data set and a re-start are necessary, if the settings occur via ProDrive/WinBASS II. The speed profile can be set via the FBO 6086 $_{hex}$, also during positioning. Thereby the present moving command is brought to an end and then the new moving command with the new profile is started.

4.4.3 Settable behavior, if new target outside the software limit switch

This is to be set in 'Drive manager 2 warning activated' in ProDrive/WinBASS II and it can be saved in the data set.

If new target outside ⇒ no movement;
There is a CAN emergency message code 8600 $_{hex}$ positioning controller (controller error No. 196 SW limit switch 1, controller error No. 197 SW limit switch 2). Behavior of drive via $605A_{hex}$. The error must be reset and a new positioning set then can be executed.

If the present position already is outside and the new target also is outside ⇒ no movement;
There is a CAN emergency message code 8600_{hex} positioning controller (controller error No. 196 SW limit switch 1, controller error No. 197 SW limit switch 2). Behavior of drive via 605A $_{hex}$ adjustable. The error must be reset and then a new positioning data set can be executed.

4.4.4 Error tripping at moving in hardware limit switch

The HW limit switch monitoring is settable ProDrive/WinBASS II via Drive manager 2 activate warning.

There is a CAN emergency message code 8600 $_{hex}$ positioning controller (controller error No. 198 negative HW limit switch, controller error No. 199 positive HW limit switch). The generated error does not lead to a pulse inhibit. It must be reset, before starting a new positioning.

4.4.5 User units UU

The user units can be specified via ProDrive/WinBASS II ⇒ in ’Rescaling’. Store in the data set to save the user units after switching off.

If the required UUs are set, these should also be maintained with the following updates of the controller. To be on the safe side, once more check in ProDrive/WinBASS II.

Important: In the default data set for the user units 1 UU = 1 INC is set.

With CANopen the user units can be set via FBO 6092_{hex}.

6092_{hex}: feed constant = feed / driving shaft revolutions

„Driving shaft revolutions“ is multiplied internally on the CANopen option card with 65536. Maximum input for ’feed’ (UU) is 0 ... 2^{24}- 1.

SIX1 = feed [in user units e. g. 360.00 degrees, 1/100 degrees resolution]

Shown on P3050 in b maXX® controller and can be saved in data set.

SIX2 = driving shaft revolutions [1 revolution is internally multiplied on the CANopen option card with 65536 [Inc]].

Shown on P3051 in b maXX® controller and can be saved in data set.

The number of revolutions is limited to 255.

The input via the field bus 360.00 degrees is converted on the option card to 65536 increments for one revolution.

Example: position setpoint in UU = 36000; accords to 360.00 degrees (accords to 65536 Inc).

The conversion on the CANopen option card for the position setpoint FBO $607A_{hex}$: looks like follows:

Position setpoint [INC] in b maXX®

= FBO [BE] * driving shaft revolutions * 65536 [INC] / feed [BE]

= 36000 * 1 * 65536 / 36000 [UU * Inc / UU]

= 65536 [INC]

NOTE

The determination of user units is very time-consuming and should not be used at cyclic operation (position and synchronous operation). If the user units are set to 1:1 the calculation is to be dispensed with.

The UUs have an effect on the following FBOs:

6062_{hex}, 6063_{hex}, 6064_{hex}, 6067_{hex}, $607A_{hex}$, $607C_{hex}$, $607D_{hex}$ Sub1/2, 6081_{hex}, 6083_{hex}, 6084_{hex}, 6085_{hex}, 6099_{hex} Sub1/2, $609A_{hex}$

4.4.6 Gear factor

Additionally to the user units a new gear factor can be introduced, which is set with the field bus object 604C$_{hex}$, which newly has been introduced. With the gear factor it is now possible e. g. to take the gear ratio or other scalings into account, what from the required speed of the drive is calculated.

FBO 604C$_{hex}$:

vl_dimension_factor =
vl_dimension_factor_numerator / vl_dimension_factor_denominator

SIX1 = vl_dimension_factor_numerator

Is mapped on P3314 in b maXX® INT32 (-33000 ... 33000)

SIX2 = vl_dimension_factor_denominator

Mapped on P3315 in b maXX® INT32 (-33000 ... 33000)

The calculation is the following:

Speed setpoint motor in b maXX®:

For vl_dimension_factor_numerator = 10

and vl_dimension_factor_denominator = 5

Speed setpoint motor = FBO[RPM] * vl_dimension_factor
= 100 * 10 / 5 [RPM]
= 200 [RPM]

NOTE

The calculation of the gear factors is very time-consuming and should not at cyclical operation (position- and synchronous operation) be used. If the gear factors are set to 1:1 calculation can be dispensed with.

The gear factor has an affect on the following FBOs:

6042$_{hex}$, 6043$_{hex}$, 6045$_{hex}$, 6048$_{hex}$ Sub01/02, 6049$_{hex}$ Sub01/02, 606B$_{hex}$, 606C$_{hex}$, 60FF$_{hex}$

4.4.7 CANopen offset

Mapping of numerical scale USIGN32 to INT32 (CANopen mode). At changes of several FBOs an offset of 2^{31} is internally added or subtracted on the CANopen option card accordant to direction.

If the position actual values and the target position in ProDrive/WinBASS II shall be displayed also in the INT32 numerical scale, a checkbox for the offset can be activated on the page 'Rescaling' .

The CANopen offset has an effect on the following FBOs:

(6062_{hex}, 6064_{hex}, $607A_{hex}$, $607C_{hex}$, $607D_{hex}$ Sub ½) - 2^{31}

($607A_{hex}$, $607C_{hex}$, $607D_{hex}$ Sub ½) + 2^{31}

4.4.8 Homing for positioning is necessary

In ProDrive/WinBASS II on the page of 'Homing' with the provided checkbox activation can result from the drive permitting a positioning, if there was no first homing.

Deactivated: In order to operate in the operation mode positioning no homing is necessary.

Activated: If the drive is enabled in operation mode positioning, without taking place of homing, an error message (EMYC-telegram 8600_{hex}⇒controller error No. 200) is generated and the drive remains position-controlled in the actual position. Positioning requests are not executed. Not until homing has been executed (once after switching on), positioning requests are executed. The error message can only be acknowledged, if homing was executed. After homing a positioning can be initiated.

NOTE

Homing is necessary, if the CANopen mode standard is defined as standard!

4.4.9 Versions of positioning dependent on positioning mode (P601)

NOTE

It must be considered, that in ProDrive/WinBASS II under positioning 0 also the positioning data set 0 has to be set, otherwise positioning via CANopen cannot be correctly executed. The switching over between the positioning modes 'relative', 'negative/positive' and 'absolute' must be executed only via the control word. Homing always should precede before positioning in CANopen mode (standard).

There are the following positioning modes:

Positioning mode P0601	Description
'Absolute/relative' CANopen (standard value 9)	• Target is in **P0607** (INT32) • Switchover 'absolute/relative' only occurs via the control word
'Relative, positive and negative' (value 4)	• Target is in **P0607** (INT32) • No switchover 'absolute/relative' via the control word.
„Absolute relative“ (value 10)	• Target is in **P0600** (USIGN32) • Switchover 'absolute/relative' only occurs via the control word
All other modes	• Target is in **P0600** (USIGN32) • No switchover 'absolute/relative' via the control word. • No conversion data type = UINT32

Switchover absolute / relative. Via the control word bit 6
Bit 6 = 0 -> absolute
Bit 6 = 1 -> relative

The conversion of data type INT32 <-> UINT32 means, that an offset of 2^{31} is added or subtracted, according to the direction. This is necessary, in order to have a consistent representation of the field bus objects in the data type INT, because several controller parameters for the positioning (▹CANopen offset◃ on page 29) are implemented as data type USIGN. For the user, in the positioning, the existent FBOs are seen as the data type INT.

The consideration of the offset 2^{31} can be deactivated, if required. In this case in Prodrive/ WinBASS II under the "Option Module G, H - Configuration 1":

Bit 5 -> 0:
Numerical scale conversion from the data type U32 to the data type INT32. At the positioning an offset of 2^{31} is added to the related FBO, according to the direction.

Bit 5 -> 1:
there is no offset used.

NOTE

The conversion in the positioning mode to P607 "Absolute /Relative" CANopen is not deactivated.

In **P1190** with bit 9 it is possible to deactivate the automatical mode setting 'Absolute/relative CANopen' during the Init Phase of the CANopen option card.

P1190 Bit 9 = 0 -> activated

P1190 Bit 9 = 1 -> deactivated.

5

DATA EXCHANGE AND PARAMETERIZATION

The access to data or parameters occurs with CANopen **always** via objects.

Accordant to profile structure it is differed between objects for communication control (indices $1XXX_{hex}$) and user- or device-specific objects. The latter are divided into objects according to profile DSP 402 (indices $6XXX_{hex}$) and manufacturer-specific objects (indices $4XXX_{hex}$. A listing of the 6XXX and the 4XXX objects are found in ▷Appendix B - Quick reference◁ on page 95.

Important:
With manufacturer-specific objects ($4XXX_{hex}$) the object index results from

4000_{hex} + b maXX®4400 parameter number in hexadecimal,

e. g. if object $412C_{hex}$ is at b maXX®4400-parameter 300, the control word will be transposed. These objects only have subindex 00_{hex}.

Object = 4000_{hex} + parameter number $_{(hex)}$

5.1 Directory of objects for communication control

In this section all objects of the communication-specific area of the object directory are to be found, which are supported by the Baumüller CANopen option module in accordance with DS301.

Name	Index	Subindex	Data type	Default value
Device type	1000_{hex}	00_{hex}	U32	$XX020192_{hex}$

This object is read-only and contains information on the related device (drive in accordance with DSP402).

Bit 31 .. 24 manufacturer-specific objects:

Bit 25	Bit 24	Option card for:
0	0	BKF/BKD 7000
0	1	V-controller
1	0	b maXX®4400

Name	Index	Subindex	Data type	Default value
Error register	1001_{hex}	00_{hex}	U8	0

This object can only be read. The object 1001_{hex} contains an error bit string with the following meaning:

Bit 24	Meaning
0	Error has occurred, general error
1	Current error
2	Power error
3	Temperature error
4	CAN communication error
5	Device profile-specific error
6	Not used
7	Manufacturer-specific error

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80_{hex} + address	08_{hex}	Emergency error code		Error register	Manufacturer-specific error field				

EMCY telegram for error reset/No error

Name	Index	Subindex	Data type	Default value
Manufacturer status register	1002_{hex}	00_{hex}	U32	-

This object can only be read. The low byte contains the controller status word low byte from parameter **P0301**.

Name	Index	Subindex	Data type	Default value
Predefined error field	1003_{hex}	00_{hex}	U8	-
Latest error		01_{hex}	U32	-
:		:	:	:
First error		$0F_{hex}$	U32	-

This object can only be written to subindex 00_{hex}. This contains an error history of 15 errors at maximum, in which the last-occurred error is in subindex 01_{hex}.

Subindex 00_{hex} of this object holds the number of errors registered. By writing ’00 hex’ to subindex 00_{hex} the list will be cleared.

$XXXX_{hex}$:	$XXXX_{hex}$
Emergency code:	manufacturer-specific

The meaning of error number is to be found in ▹Conversion of error messages to DSP 402 V1.1◃ from page 80.

Name	Index	Subindex	Data type	Default value
Number of PDOs supported	1004_{hex}	00_{hex}	U32	00040004_{hex}
Synchronous PDOs		01_{hex}	U32	00040004_{hex}
Asynchronous PDOs		02_{hex}	U32	00040004_{hex}

This object is read-only. Subindex 00_{hex} where the number of receive PDOs is in the high word and the number of transmit PDO is in the low word. Subindex 01_{hex} contains the possible number of synchronous PDOs, subindex 02_{hex} the possible numbers of asynchronous PDOs.

The values entered mean that 4 receive PDOs and 4 transmit PDOs are available, whereby each PDO can be defined as synchronous or asynchronous.

Name	Index	Subindex	Data type	Default value
COB ID SYNC message	1005_{hex}	00_{hex}	U32	80_{hex}

This object contains information about the SYNC slave behavior. The slave is not a SYNC master, e. g. only SYNC telegrams can be received. The lower 11 bits in the low word specify the identifier of the SYNC telegram (80_{hex}), only readable.

Name	Index	Subindex	Data type	Default value
Communication cycle period	1006_{hex}	00_{hex}	U32	0

In case the SYNC telegram is active, the SYNC interval must be set to the time of SYNC telegram (1000, 2000, 4000 or 8000 µs). The set time takes effects on the parameter **P0532** (SYNC interval) of the b maXX® controller.

Name	Index	Subindex	Data type	Default value
Synchronous window length	1007_{hex}	00_{hex}	U32	0

This object is not evaluated.

Name	Index	Subindex	Data type	Default value
Manufacturer device name	1008_{hex}	00_{hex}	VString	-

This object is read-only. It contains the following character strings: „b maXX 4400“.

Name	Index	Subindex	Data type	Default value
Manufacturer hardware version	1009_{hex}	00_{hex}	VString	-

This object is read-only. It contains the current hardware version of the option module, e. g. the character string: „HV01.00“.

Name	Index	Subindex	Data type	Default value
Manufacturer software version	$100A_{hex}$	00_{hex}	VString	-

This object is read-only. It contains the current software version of the option module, e. g. the character string: „SV01.00“.

Name	Index	Subindex	Data type	Default value
Node ID	$100B_{hex}$	00_{hex}	U32	Address

This object is read-only. It contains the CANopen node address (node ID). Only the low byte is valid.

Name	Index	Subindex	Data type	Default value
Guard time	$100C_{hex}$	00_{hex}	U16	0

In this object the node guarding time basis is set in milliseconds. By writing the value ’0’, node guarding will be deactivated.

Name	Index	Subindex	Data type	Default value
Life time factor	$100D_{hex}$	00_{hex}	U8	0

The value of this object is multiplied by object 100C $_{hex}$ and from this the node guarding period results. By writing the value ’0’, node guarding will be deactivated.

Name	Index	Subindex	Data type	Default value
COB ID guarding protocol	$100E_{hex}$	00_{hex}	U32	700_{hex} + address

This object contains the identifier of the node guarding telegram. The identifier can be changed - cannot be saved.

Name	Index	Subindex	Data type	Default value
Number of SDOs supported	$100F_{hex}$	00_{hex}	U32	00000001_{hex}

This object is read-only. It contains the number of supported SDOs. In the high word is the client SDO number, which supports the device. The low word contains the number of server SDOs of the device.

Name	Index	Subindex	Data type	Default value
Save parameters	1010_{hex}	00_{hex}	U8	02_{hex}
Save all parameters (not supported)		01_{hex}	U32	00_{hex}
Save communication parameters		02_{hex}	U32	00_{hex}

This object supports the saving of parameters in non-volatile memory. Subindex 00_{hex} is read-only and contains the greatest subindex to be supported (here 2). Subindex 01_{hex} is currently not supported. The 01_{hex} in subindex 02_{hex} indicates that saving is supported, here in particular saving mapping and communication parameters. Saving, only possible in the PRE-OPERATIONAL state, is initiated with the value 65766173_{hex} as U32 or with the value ’save’ as string and lasts a few hundred milliseconds. The values last set for the mapping and communication parameters will be saved.

If SIX 2 is read, value 1 returns. This means that it is not saved automatically, but it must be initiated.

Name	Index	Subindex	Data type	Default value
Restore parameters	1011_{hex}	00_{hex}	U8	02_{hex}
Restore all parameters (not supported)		01_{hex}	U32	00_{hex}
Restore default communication parameters		02_{hex}	U32	00_{hex}

The communication parameters are set according to DSP402 to default values for this object. Subindex 00_{hex} is read-only and contains the greatest subindex to be supported. Subindex 01_{hex} is not supported. With the value 1_{hex} in subindex 02_{hex}, the mapping and communication parameter default values are set. This occurs only in the PRE-OPERATIONAL state by entering the value $64616F6C_{hex}$ (U32) or with the input ’load’ as string parameter.

The default values are accepted after power down/power up was executed.

WARNING

The following **can occur**, if you disregard this warning instruction:

- serious personal injury • death

The danger is: **changed mapping.** *Modified mapping causes parameters to be used other than those planned and consequently the drive can react unexpectedly.*

In your application, prevent mapping from being modified in an uncontrolled manner.

Name	Index	Subindex	Data type	Default value
COB ID emergency	1014_{hex}	00_{hex}	U32	80_{hex} + address

This object contains the identifier of the EMCY telegram.

Name	Index	Subindex	Data type	Default value
Producer heartbeat time	1017_{hex}	00_{hex}	U8	03_{hex}

With the object the cyclic time of the heartbeat telegram is set. If the time is zero, no heartbeat telegram is set. The resolution is 1 ms.

Name	Index	Subindex	Data type	Default value
Identity object	1018_{hex}	00_{hex}	U8	03_{hex}
Vendor ID		01_{hex}	U32	$15H_{hex}$
Product code		02_{hex}	U32	350442
Revision number		03_{hex}	U32	see below

This object contains some information on the device.

The revision number contains the current version of firmware e. g. 00030002 for FW 6.1294.03.02.

Name	Index	Subindex	Data type	Default value
Server SDO parameter	1200_{hex}	00_{hex}	U8	02_{hex}
COB ID client ⇒ server		01_{hex}	U32	NODEID + 600_{hex}
COB ID server ⇒ client		02_{hex}	U32	NODEID + 580_{hex}

This object is read-only.

At the same time default value here means default mapping acc. to DSP402

Name	Index	Subindex	Data type	Default value
1. receive PDO parameters	1400_{hex}	00_{hex}	U8	02_{hex}
COD-ID used by PDO		01_{hex}	U32	200_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}

This object contains the contents of receive-PDO1. Subindex 00_{hex} is read-only. The identifier of the receive PDO1 is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO.

Name	Index	Subindex	Data type	Default value
2. receive PDO parameters	1401_{hex}	00_{hex}	U8	02_{hex}
COD-ID used by PDO		01_{hex}	U32	300_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}

This object contains the according receive PDO2. Subindex 00_{hex} is read-only. The identifier of the receive PDO2 is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO.

Name	Index	Subindex	Data type	Default value
3. receive PDO parameters	1402_{hex}	00_{hex}	U8	02_{hex}
COD-ID used by PDO		01_{hex}	U32	400_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}

This object contains the contents of receive-PDO3. Subindex 00_{hex} is read-only. The identifier of the receive PDO3 is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO.

Name	Index	Subindex	Data type	Default value
4. receive PDO parameters	1403_{hex}	00_{hex}	U8	02_{hex}
COD-ID used by PDO		01_{hex}	U32	500_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}

This object contains the contents of receive-PDO4. Subindex 00_{hex} is read-only. The identifier of the receive PDO4 is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO.

Name	Index	Subindex	Data type	Default value
1. receive PDO mapping	1600_{hex}	00_{hex}	U8	1
		01_{hex}	U32	60400010_{hex}
		:	:	
		n_{hex}	U32	

This object contains the information of receive PDO1. The total number of the following entries is in subindex 00_{hex}. By default the control word (object 6040_{hex} subindex 00_{hex} length 10_{hex} bits) is entered in subindex 01_{hex}. The total number of mapped bytes may not exceed the CAN telegram limit of eight bytes max. (also see ▷PDO mapping◁ from page 61).

Name	Index	Subindex	Data type	Default value
2. receive PDO mapping	1601_{hex}	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60400010_{hex}
		02_{hex}	U32	60600008_{hex}
		:	:	
		n_{hex}	U32	

This object contains the information of receive PDO2. The total number of the following entries is in subindex 00_{hex}. By default the control word (object 6040_{hex} subindex 00_{hex}length 10_{hex}bits) is entered in subindex 01_{hex}. In subindex 02_{hex} by default the object of the specified operating mode (index 6060_{hex} subindex 00 length 08_{hex}bits). The total number of the mapped bytes may not exceed the CAN telegram limit of max. eight bytes (also see ▷PDO mapping◁ from page 61).

Name	Index	Subindex	Data type	Default value
3. receive PDO mapping	1602_{hex}	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60400010_{hex}
		02_{hex}	U32	$607A0020_{hex}$
		:	:	
		n_{hex}	U32	

This object contain the contents of receive-PDO3. The total number of the following entries is in subindex 00_{hex}. By default the control word (object 6040_{hex} subindex 00_{hex}length 10_{hex}bits) is entered in subindex 01_{hex}. By default the Target position 1 object is in subindex 02_{hex} (index $607A_{hex}$ subindex 00 length 20_{hex} bits). The total number of mapped bytes may not exceed the CAN telegram limit of max. 8 bytes (also see ▷PDO mapping◁ from page 61).

Name	Index	Subindex	Data type	Default value
4. receive PDO mapping	1603_{hex}	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60400010_{hex}
		02_{hex}	U32	$60FF0020_{hex}$
		:	:	
		n_{hex}	U32	

This object contain the contents of receive-PDO4. The total number of the following entries is in subindex 00_{hex}. By default the control word (object 6040_{hex} subindex 00_{hex}length 10_{hex}bits) is entered in subindex 01_{hex}. By default, the speed actuating value object is in subindex 02_{hex} (index $60FF_{hex}$ subindex 00 length 20_{hex} bits). The total number of mapped bytes may not exceed the CAN telegram limit of max. eight bytes (also see ▷PDO mapping◁ from page 61).

Name	Index	Subindex	Data type	Default value
1. transmit PDO parameters	1800_{hex}	00_{hex}	U8	05_{hex}
COD-ID used by PDO		01_{hex}	U32	180_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}
Inhibit time		03_{hex}	U16	00_{hex}
CMS priority group		04_{hex}	U8	03_{hex}
Event timer		05_{hex}	U16	00_{hex}

This object contains information on transmit PDO1. Subindex 00_{hex} is read-only. The transmit PD01 identifier is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 03_{hex}. The input value is defined as a multiplier of 100 µs. Subindex 04_{hex} is currently not used. Subindex 05_{hex} is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond.

Name	Index	Subindex	Data type	Default value
2. transmit PDO parameters	1801_{hex}	00_{hex}	U8	04_{hex}
COD-ID used by PDO		01_{hex}	U32	280_{hex} + address
Transmission type		02_{hex}	U8	FF_{hex}
Inhibit time		03_{hex}	U16	00_{hex}
CMS priority group		04_{hex}	U8	03_{hex}
Event timer		05_{hex}	U16	00_{hex}

This object contains information on transmit PDO2. Subindex 00_{hex} is read-only. The transmit PD01 identifier is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 03_{hex}. The input value is defined as a multiplier of 100 µs. Subindex 04_{hex} is currently not used. Subindex 05_{hex} is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond.

Name	Index	Subindex	Data type	Default value
3. transmit PDO parameters	1802_{hex}	00_{hex}	U8	04_{hex}
COD-ID used by PDO		01_{hex}	U32	380_{hex} + address
Transmission type		02_{hex}	U8	FE_{hex}
Inhibit time		03_{hex}	U16	00_{hex}
CMS priority group		04_{hex}	U8	03_{hex}
Event timer		05_{hex}	U16	00_{hex}

This object contains information on transmit PDO3. Subindex 00_{hex} is read-only. The transmit PD01 identifier is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 03_{hex}. The input value is defined as a multiplier of 100 µs. Subindex 04_{hex} is currently not used. Subindex 05_{hex} is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond.

Name	Index	Subindex	Data type	Default value
4. transmit PDO parameters	1803_{hex}	00_{hex}	U8	04_{hex}
COD-ID used by PDO		01_{hex}	U32	480_{hex} + address
Transmission type		02_{hex}	U8	FE_{hex}
Inhibit time		03_{hex}	U16	00_{hex}
CMS priority group		04_{hex}	U8	03_{hex}
Event timer		05_{hex}	U16	00_{hex}

This object contains information on transmit PDO4. Subindex 00_{hex} is read-only. The transmit PD01 identifier is entered in subindex 01_{hex}. Subindex 02_{hex} contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 03_{hex}. The input value is defined as a multiplier of 100 µs. Subindex 04_{hex} is currently not used. Subindex 05_{hex} is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond.

Name	Index	Subindex	Data type	Default value
1. transmit PDO mapping	$1A00_{hex}$	00_{hex}	U8	01_{hex}
		01_{hex}	U32	60410010_{hex}
		:	:	
		n_{hex}		

This object contains the contents of transmit PDO1. The total number of the following entries is in subindex 00_{hex}. By default the status word $_{hex}$(object 6041_{hex} subindex 00_{hex} length 10_{hex} bits) is entered in subindex 01. The total number of mapped bytes may not exceed the CAN message frame limit of eight bytes max. (also see ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
2. transmit PDO mapping	$1A01_{hex}$	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60410010_{hex}
		02_{hex}	U32	60610008_{hex}
		:	:	
		n_{hex}	U32	

This object contains the content of transmit PDO2. The total number of the following entries is in subindex 00_{hex}. By default the status word $_{hex}$(object 6041_{hex} subindex 00_{hex} length 10_{hex} bits) is entered in subindex 01. By default the object of actual operating mode (index 6061_{hex}subindex 00 length 08_{hex} bit) is entered in subindex 02_{hex}. The total number of mapped bytes may not exceed the CAN message frame limit of eight bytes max. (also see ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
3. transmit PDO mapping	$1A02_{hex}$	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60410010_{hex}
		02_{hex}	U32	60640020_{hex}
		:	:	
		n_{hex}	U32	

This object contains the contents of transmit PDO3. The total number of the following entries is in subindex 00_{hex}. By default the status word $_{hex}$(object 6041_{hex} subindex 00_{hex} length 10_{hex} bits) is entered in subindex 01. By default the object of actual positioning value (index 6064_{hex} subindex 00 length 20_{hex} bits) is entered in subindex 02_{hex}. The total number of mapped bytes may not exceed the CAN message frame limit of eight bytes max. (also see ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
4. transmit PDO mapping	$1A03_{hex}$	00_{hex}	U8	02_{hex}
		01_{hex}	U32	60410010_{hex}
		02_{hex}	U32	$606C0020_{hex}$
		n_{hex}	U32	

This object contains the content of transmit PDO4. The total number of the following entries is in subindex 00_{hex}. By default the status word $_{hex}$(object 6041_{hex} subindex 00_{hex} length 10_{hex} bits) is entered in subindex 01. By default the object of actual speed value (index $606C_{hex}$ subindex 00 length 20_{hex} bits) is entered in subindex 02_{hex}. The total number of mapped bytes may not exceed the CAN message frame limit of eight bytes max. (also see ▹PDO mapping◃ from page 61).

5.2 Network management (NMT)

Network management commands function primarily to control communication states in the CANopen network.

5.2.1 Communication state machine

Here the communication state diagram of the CANopen slaves is shown.

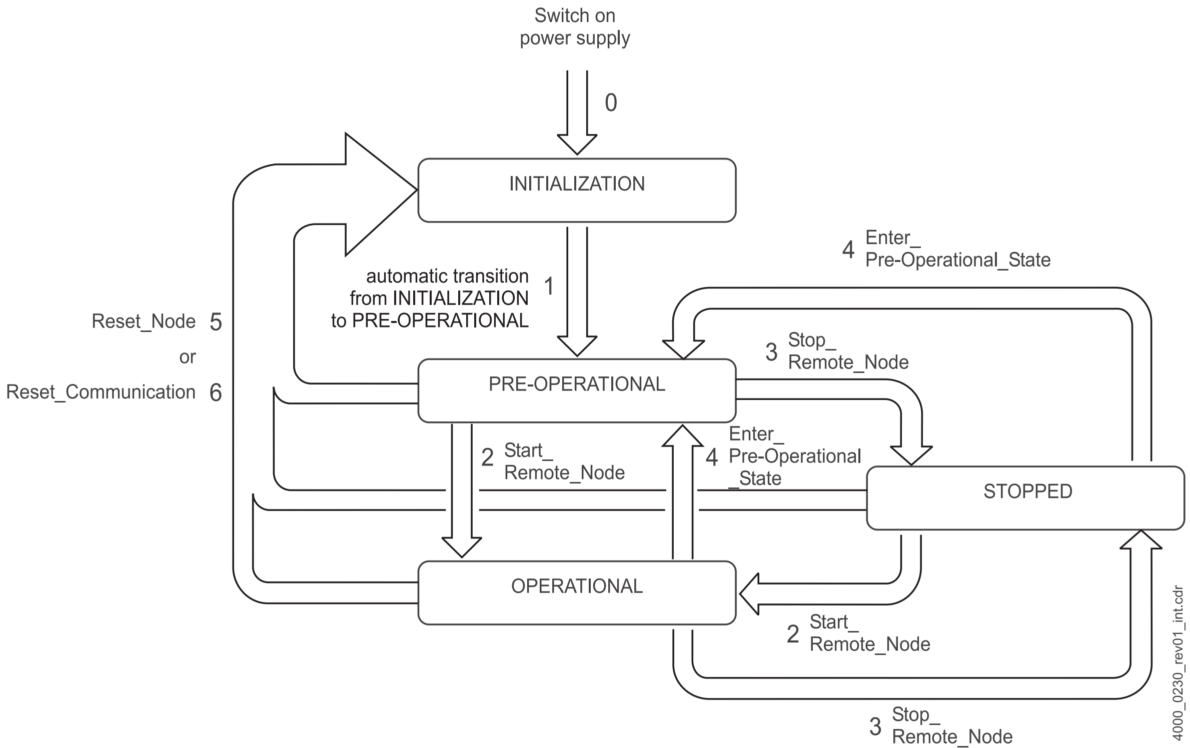


Figure 3: Communication state machine

After INITIALIZATION (triggered by switching on the device), the PRE-OPERATIONAL state will automatically be reached. If a slave is in this state, it can be configured via SDOs. Data exchange via PDOs is not possible.

In the STOPPED state, only node guarding is activated. Neither SDOs nor PDOs can be transmitted or received.

In the OPERATIONAL state (normal operating state), PDO and SDO data exchange as soon as node guarding is possible.

The individual state transitions are initiated by an NMT master. The Baumüller CANopen option module can process the following NMT commands:

1 Automatic transition from INITIALIZATION to PRE-OPERATIONAL
Note
At transition from PRE-OPERATIONAL to OPERATIONAL the parameter numbers are assigned to mapping. This assignment is time-consuming and can last several milliseconds (up to 11 ms, according to time setting of BACI also longer), during this time no PDO is send and also no RX-PDO is processed.
It is not able to work on 3 NMT commands within 15 ms.

2 Start_Remote_Node

3 Stop_Remote_Node

4 Enter_Pre-Operational_State

5 Reset_Node

6 Reset_Communication

It is not able to work on 3 NMT commands within 8 ms.

5.2.2 Telegrams

NMT telegrams for communication control have the default identifier ’0’ in accordance with the predefined connection set (also see ▷Basic principles CAN◁ from page 10).

5.2.2.1 State control

Two data bytes are transmitted per NMT-telegram. Data byte 0 contains the command specifier CS, data byte 1 contains the device address. If the address 0 is entered, then all nodes will be addressed with the appropriate command (broadcast).

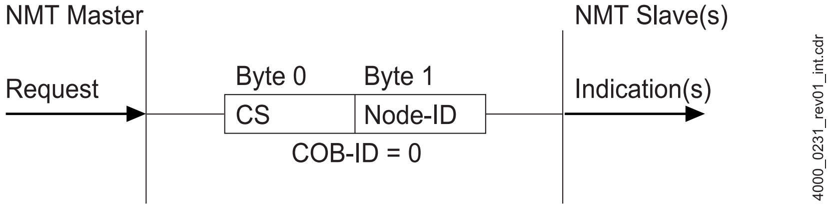


Figure 4: NMT telegram for controlling the communication states

CS	Identification	Effect
1	Start_Remote_Node	Starts normal operation
2	Stop_Remote_Node	Deactivates PDO and SDO communication
128	Enter_Pre-Operational_State	Transition to configuration mode
129	Reset_Node	Controlled reset of entire object directory to default values
130	Reset_Communication	Reset of the communication section of the object directory to default values

A telegram bringing node 16 into configuration mode has the following construction:

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00_{hex}	02_{hex}	80_{hex}	10_{hex}						

These telegrams are unconfirmed, i. e. no NMT slave acknowledges the correctly received message to the NMT master.

NOTE

The b maXX ®4400 currently does not include device reset by software.

After turning on supply voltage and a reset the CANopen slave signals a boot up telegram (also see ▷Boot up◁ from page 46). The entire reset sequence lasts a few seconds, starting with the receipt of the command reset_node to the acknowledgement using boot up telegram.

WARNING

The following **can occur**, if you disregard this warning instruction:

- serious personal injury
- death

The danger is: **mechanical and electrical cause.** *If a reset is triggered during a running cyclic operation, this can lead to undesired states in the application, because the boot record will be loaded in the controller and on the CANopen option card the default mapping* last saved in flash memory will be set (with object 1010_{hex} *in accordance with DS301).*

Check the mapping after each reset.

5.2.2.2 Boot up

Standard is the boot up behavior according to DS 301 V4

For differentiation of the boot-up telegram according to DS 301 V3 or V4 the parameter 'option module G - configuration 1' or 'option module H configuration 1' b maXX® controller can be set. Dependent in which slot the CANopen option card is plugged.

Bit 0 ⇒ 0: in accordance with DS 301 V4 (default)

Bit 0 ⇒ 1: acc. to DS 301 V3

Boot up according with DS 301 V4,
Boot up telegram with ID = 700_{hex} + node ID, DLC = 1 byte 0 filled with the data = 0.

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
701_{hex}	01_{hex}	00_{hex}							

DSP 301 V4

Boot up according to DS 301 V3, ID = 80_{hex} + node ID, DLC = 0 and following
ID = 80_{hex} + node ID, DLC = 8 bytes 0 - 7 filled up with the data = 0 (reset of error register).

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
81_{hex}	00_{hex}								
81_{hex}	08_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

DS 301 V3

5.2.3 Node guarding

The node guarding is used to monitor the slave by the master. Simultaneously the slave can monitor the master (life guarding).

The master scans the slaves in certain intervals by remote frames. Remote frames are special telegrams, which make it possible to request data telegrams. Remote frames possess the same COB ID as the corresponding data telegram, but show a data length of 1 byte. In order to differentiate between remote- and data telegram (telegram differentiation normally is carried out by the COB-ID), serves in the control field of the remote telegram the so-called RTR bit. In the remote frame the RTR bit is on „1“, in the data telegram on „0“.

The COB ID results from 700_{hex} + address according to the predefined connection set. This COB ID can also be changed. The object required for this is $100E_{hex}$.

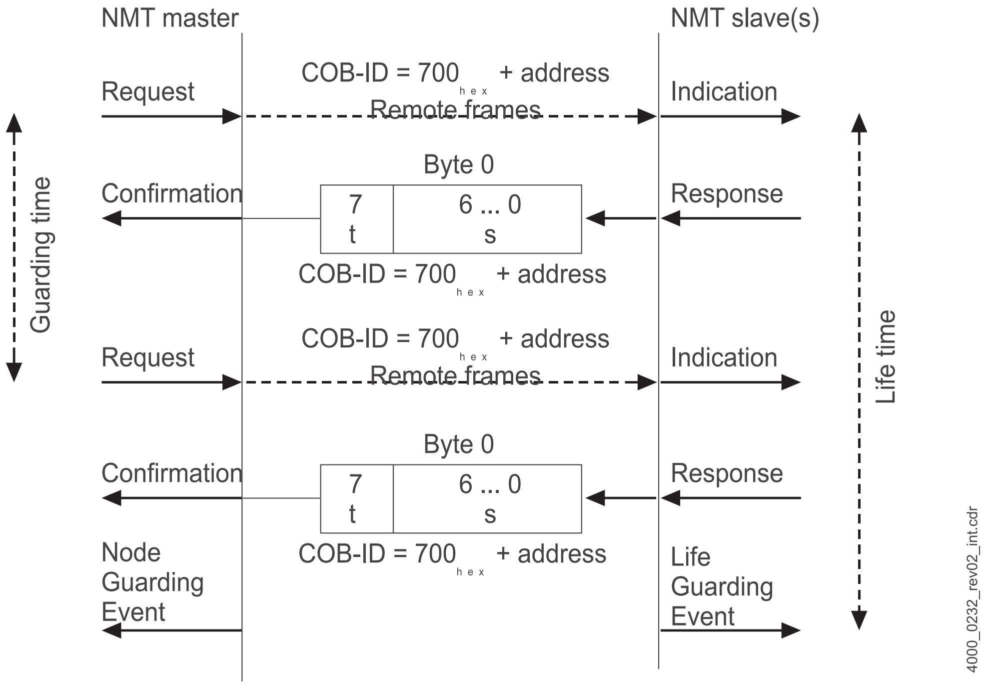


Figure 5: Node guarding protocol

The guarding time is set in objects 100C$_{hex}$ and 100D$_{hex}$. Within this time, the slave must have received a guarding request (remote telegram) from the master. Should this not be the case, the life guarding event occurs in the slave. Through this, the slave switches to the PRE-OPERATIONAL state and the reaction specified in object 6007$_{hex}$ is triggered in the controller.

If there is no response from the slave within a certain time, the node guarding event will be triggered in the master. If no time is set, the slave will respond to every RTR, but without monitoring lifetime.

The current communication state of the slave can be recognized from the response of the slave to a node guarding request from the master. The response telegram consists of one data byte (also see ▷Figure 5◁ on page 47). Field 's' differs according to the communication state. In addition, if there are two successive telegrams, toggle bit 't' will be changed.

Communication phase	**Identifier s**	**Resulting data with**	
		t = 0	t = 1
PRE-OPERATIONAL	7F$_{hex}$ (127)	7F$_{hex}$ (127)	FF$_{hex}$ (255)
OPERATIONAL	05$_{hex}$ (5)	05$_{hex}$	85$_{hex}$ (133)
STOPPED	04$_{hex}$ (4)	04$_{hex}$	84$_{hex}$ (132)

Node guarding is available in all communication phases. The toggle bit is only reset in phase INITIALIZATION to its default value. This means, that also at state changes the toggle mechanism is continued.

Node guarding is started in the slave after receipt of the first guarding request telegram. From the moment, the monitoring time parameterized in objects $100C_{hex}$ and $100D_{hex}$ runs in the slave.

NOTE

The node guarding time should be set at least 1.5 times greater than the remote telegram sent by the master.

5.2.4 Heartbeat protocol

The heartbeat protocol is for monitoring of the slave(s) by the master. The difference from node guarding is that there are no RTR frames, rather the slave transmits cyclic heartbeat messages. One or more heartbeat consumers receive or monitor the heartbeat messages. If the heartbeat message is not transferred within the set heartbeat time, the master (heartbeat consumer) releases a heartbeat event. The heartbeat time at the slave is set in the FB object 1017_{hex}. The resolution is 1 millisecond.

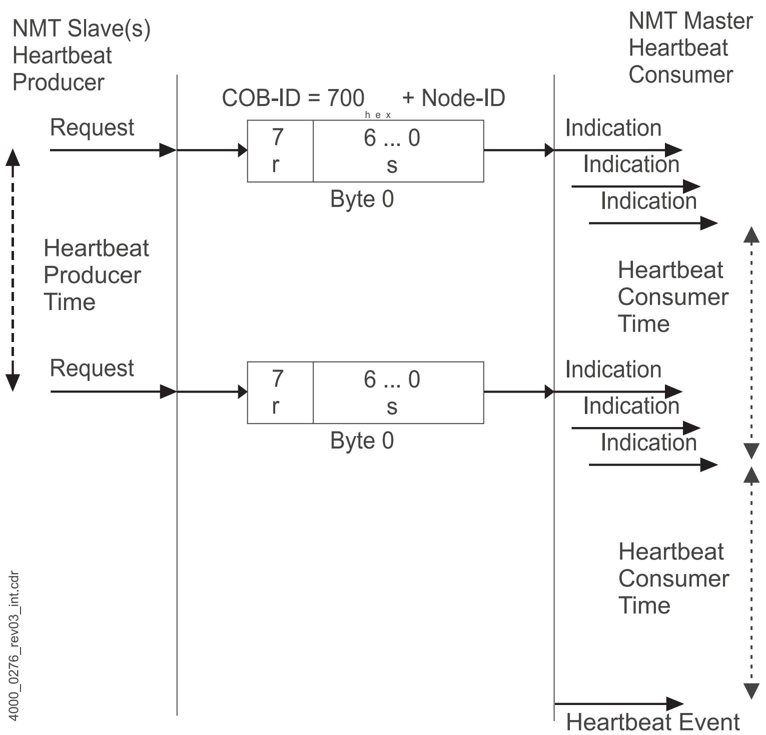


Figure 6: Heartbeat protocol

From the heartbeat message of the slave the current communication state of the slave is recognized. The heartbeat telegram consists of one data byte.

r: reserved (= 0)

s: field ’s’ differs according to the communication state „s“.

Communication phase	Identifier s
Boot up	00_{hex}
PRE-OPERATIONAL	$7F_{hex}$ (127)
OPERATIONAL	05_{hex} (5)
STOPPED	04_{hex} (4)

Either node guarding or the heartbeat is supported. If the heartbeat time is not equal ZERO, the heartbeat protocol is activated.

5.3 Service data (SDO)

Service data objects (SDO) are used in the exchange of messages without real time requirements. Therefore low-priority COB IDs are provided for this in the predefined connection set (also see ▹Basic principles CAN◃ from page 10). SDOs are used for parameterizing slaves and for setting the communication references for PDOs. Access on data occurs only via the object list. SDOs are always confirmed data, i. e. the transmitter receives an acknowledgement from the receiver. Data exchange via SDOs can only progress asynchronously (also see ▹Synchronization (SYNC)◃ from page 59).

SDOs follow the client-server model. The client initiates the communication and the server responds. A server cannot begin an SDO communication. The Baumüller CANopen option module supports one server SDO and no client SDOs.

5.3.1 Telegram structure

Die COB ID of the request SDO results from 600_{hex} + address, from the response SDOs from 580_{hex} + address. The data field of the CAN data telegram (8 bytes) for a SDO is divided into three parts, a command specifier CS (1 byte), a multiplexor M (3 bytes) and the actual user data D0 - D3 (4 bytes).

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
600_{hex} + address	08_{hex}	CS	M	M	M	D0	D1	D2	D3
580_{hex} + address	08_{hex}	CS	M	M	M	D0	D1	D2	D3

The multiplexor M exists of the 16 bit index of an object and of the associated eight bit wide subindex. At segmented telegrams the user data range is extended by the three bytes of the multiplexor, whereby seven bytes user data are transmitted per telegram. The command specifier CS classifies the different SDO types.

5.3.2 Types of SDO transfers

The Baumüller CANopen interface supports the expedited transfer and segmented transfer, in that the latter is only used for objects 1008_{hex}, 1009_{hex} and $100A_{hex}$ manufacturer device name.

Expedited transfer Objects can be written or read, with their data including a maximum of 4 bytes. Only two telegrams are required, a request and a response. All objects with the indices $1XXX_{hex}$, $4XXX_{hex}$, $6XXX_{hex}$ can be addressed via expedited SDOs with the exception of objects 1008_{hex}, 1009_{hex} and $100A_{hex}$.

Segmented transfer The segmented transfer is necessary for objects with data greater than 4 bytes. Thereby the user data is divided to several telegrams. This is only necessary when reading the objects 1008_{hex}, 1009_{hex} and $100A_{hex}$.

5.3.3 Writing object

In order to write objects at the Baumüller CANopen connection the expedited transfer is used. A SDO-client (master) transmits a write request to the slave (Baumüller CANopen interface). This slave carries out the request and acknowledges this with the response.

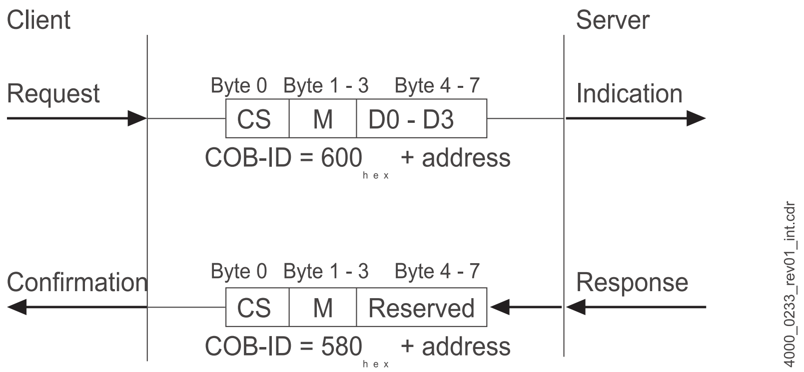


Figure 7: Initiate SDO download protocol

The command specifier CS for the request depends on the user data length. D0 is the LSB and D3 the MSB of the datum to be transmitted.

Data length in D0 - D3	Command specifier CS
1 byte	$2F_{hex}$
2 byte	$2B_{hex}$
4 byte	23_{hex}

The command specifier CS for the response is 60_{hex}, the multiplexor is identical to that of the request, the data field without meaning (reserved).

Example The value ’-3’ (FD_{hex}) to be written to object 6060_{hex}, subindex 00_{hex}, of the slave with the address 4. The data width of this object is 8 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	$2F_{hex}$	60_{hex}	60_{hex}	00_{hex}	FD_{hex}	00_{hex}	00_{hex}	00_{hex}

Basic address 600_{hex} + slave address 4_{hex}

Object 60 60_{hex}

Subindex 00_{hex}

Value -3

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	60_{hex}	60_{hex}	60_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The value ’12’ ($0C_{hex}$) is to be written to object $43E9_{hex}$, subindex 00_{hex}, of the slave with the address 4. The data width of this object is 16 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	$2B_{hex}$	$E9_{hex}$	43_{hex}	00_{hex}	$0C_{hex}$	00_{hex}	00_{hex}	00_{hex}

Basic address 600_{hex} + slave address 4_{hex}

Object 43 $E9_{hex}$

Subindex 00_{hex}

Value 12

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	60_{hex}	$E9_{hex}$	43_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The value „60610008$_{hex}$„ to be written to the object 1800$_{hex}$, subindex 02$_{hex}$, of the slave with the address 4. The data width of this object is 32 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604h	08h	23h	00h	18h	02h	08h	00h	61h	60h

Basic address 600$_{hex}$ + slave address 4$_{hex}$

Object 18 00$_{hex}$

Subindex 02$_{hex}$

Value 60 61 00 08$_{hex}$

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584$_{hex}$	08$_{hex}$	60$_{hex}$	00$_{hex}$	18$_{hex}$	02$_{hex}$	00$_{hex}$	00$_{hex}$	00$_{hex}$	00$_{hex}$

5.3.4 Reading object

With the Baumüller CANopen interface, expedited transfer is used to read objects; with objects 1008$_{hex}$, 1009$_{hex}$ and 100A$_{hex}$ segmented transfer is used.

5.3.4.1 Expedited transfer

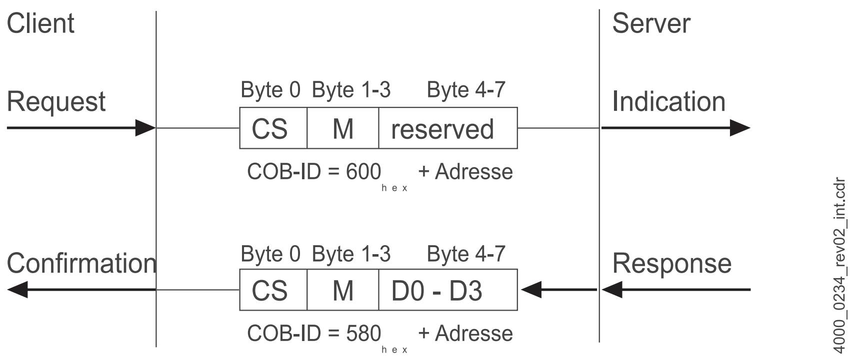


Figure 8: Initiate SDO upload expedited

A SDO client (master) transmits a read request to the slave (Baumüller CANopen interface). This slave carries out the request and sends the required data in the response telegram (response).

The command specifier CS for the request is always 40_{hex}. The command specifier CS for the response will depend on the length of the user data. D0 is the LSB and D3 the MSB.

Data length in D0 - D3	Command specifier CS
1 byte	$4F_{hex}$
2 byte	$4B_{hex}$
4 byte	43_{hex}

The request and response multiplexors agree.

Example

Object 6061_{hex}, subindex 00_{hex} of the slave with the address 4 is to be read . The data width of this object is 1 byte.

Request

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	40_{hex}	61_{hex}	60_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	$4F_{hex}$	61_{hex}	60_{hex}	00_{hex}	$D0_{hex}$	00_{hex}	00_{hex}	00_{hex}

Basic address 580_{hex} + slave address 4_{hex}
Object 60 61_{hex}
Subindex 00_{hex}
Value data

Object 6041_{hex}, subindex 00_{hex} of the slave with the address 4 is to be read . The data width of this object is 2 byte.

Request

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	40_{hex}	41_{hex}	60_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	$4B_{hex}$	41_{hex}	60_{hex}	00_{hex}	D0	D1	00_{hex}	00_{hex}

Basic address 580_{hex} + slave address 4_{hex}
Object 60 41_{hex}
Subindex 00_{hex}
Value DB high DB low

Object 1400_{hex}, subindex 01_{hex} of the slave with the address 4 is to be read. The data width of this object is 4 byte.

Request

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	40_{hex}	00_{hex}	14_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	43_{hex}	00_{hex}	14_{hex}	01_{hex}	D0	D1	D2	D3

Basic address 580_{hex} + slave address 4_{hex}

Object 14 00_{hex}

Subindex 00_{hex}

Value DB_3 DB_2 DB_1 DB_0

5.3.4.2 Segmented transfer

First of all a read request is sent to the slave with initiate SDO upload protocol. The slave responds with the command specifier CS 41_{hex}. The total number of the user data bytes to be transferred is returned in the data field (request 1, response 1). This user data will be transferred in the following cycles (request 2, response 2, request 3 and response 3).

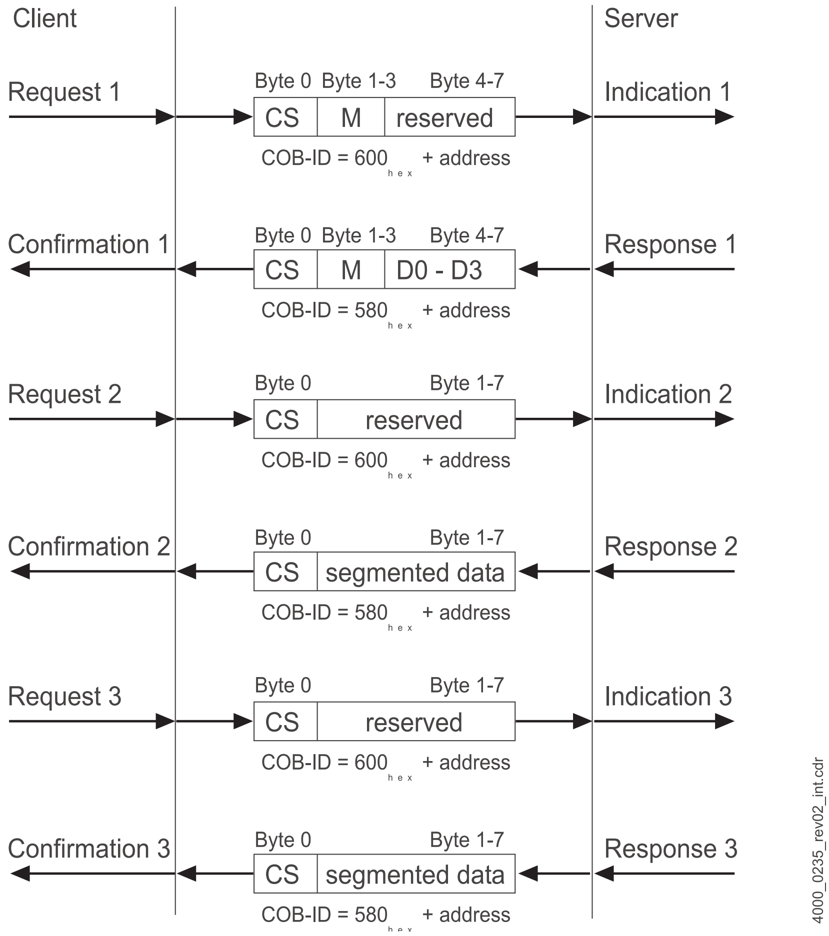


Figure 9: Upload SDO segmented protocol

The command specifiers contain a toggle bit, the value of which changes for each transfer.

for example to read object 1008_{hex} manufacturer device name of slave 4:

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	40_{hex}	08_{hex}	10_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response 1

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	41_{hex}	08_{hex}	10_{hex}	00_{hex}	6_{hex}	00_{hex}	00_{hex}	00_{hex}

Byte 0 in response 1 (command specifier 41_{hex}) signifies that the user data field contains the number of the user data bytes to be transferred (6).

Request

		CS							
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	60_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	D0	D1	D2	D3	D4	D5	D6
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	14_{hex}	62_{hex}	20_{hex}	$6D_{hex}$	61_{hex}	58_{hex}	58_{hex}	00_{hex}

Byte 0 In request 2 (command specifier 60_{hex}) means that the first segment (6 bytes) are to be transferred. Byte 0 in response 2 (command specifier, 14_{hex}) signifies that the user data field (6 bytes) contains valid data and that this segment is at the same time the last.

The result of the transfer is: b maXX.

5.3.5 Error reactions

Invalid SDO accesses are refused with abort codes. The structure of these abort telegrams is identical to the SDO telegram illustrated in ▷Figure 5.3.1◁ on page 49. The data field contains an abort code of 4 bytes.

With invalid accesses to communication-specific objects ($1XXX_{hex}$) the following messages are differentiated:

Abort code	Meaning
05_{hex} 03_{hex} 00_{hex} 00_{hex}	Inconsistent parameters (toggle bit has not changed)
05_{hex} 04_{hex} 00_{hex} 01_{hex}	Client/server command specific CS not valid or unknown.
06_{hex} 01_{hex} 00_{hex} 02_{hex}	Writing to write-protected object
06_{hex} 02_{hex} 00_{hex} 00_{hex}	Object does not exist
06_{hex} 04_{hex} 00_{hex} 41_{hex}	Data cannot be mapped (e. g. incorrect length indication)
06_{hex} 06_{hex} 00_{hex} 00_{hex}	Hardware access error (save/load from flash memory)
06_{hex} 07_{hex} 00_{hex} 10_{hex}	Incorrect length data value
06_{hex} 09_{hex} 00_{hex} 11_{hex}	Subindex does not exist
06_{hex} 09_{hex} 00_{hex} 30_{hex}	Value range exceeded (during write accesses)
06_{hex} 09_{hex} 00_{hex} 31_{hex}	Value too high (during write accesses)
08_{hex} 00_{hex} 00_{hex} 20_{hex}	Data cannot be transferred or saved to the application
08_{hex} 00_{hex} 00_{hex} 22_{hex}	Data cannot be mapped due to the current communication state (e. g. change mapping in the OPERATIONAL state).

Invalid accesses to all other objects ($4XXX_{hex}$ and $6XXX_{hex}$) are globally refused with the following codes:

Abort code	Meaning
06_{hex} 01_{hex} 00_{hex} 00_{hex}	Error in data format
06_{hex} 01_{hex} 00_{hex} 02_{hex}	Element cannot be changed
06_{hex} 02_{hex} 00_{hex} 00_{hex}	Element not present
06_{hex} 09_{hex} 00_{hex} 31_{hex}	Value too high (during write accesses)
06_{hex} 09_{hex} 00_{hex} 32_{hex}	Value too low (during write accesses)
08_{hex} 00_{hex} 00_{hex} 00_{hex}	General error occurred
08_{hex} 00_{hex} 00_{hex} 21_{hex}	Data not available at present

Example

Slave 4 object 1008_{hex} subindex 01_{hex} is to be read. Object *1008_{hex} manufacturer device name* has however only subindex 00_{hex}.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604_{hex}	08_{hex}	40_{hex}	08_{hex}	10_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584_{hex}	08_{hex}	80_{hex}	08_{hex}	10_{hex}	01_{hex}	11_{hex}	00_{hex}	09_{hex}	06_{hex}

Basic address 580_{hex} + slave address 4_{hex} — Object 10 08_{hex} — Subindex 01_{hex} — Code 06 09 00 11_{hex}

The command specifier CS (4 byte 0, 80_{hex}) in the response telegram specifies, that it is an abort telegram. The request and response multiplexors agree.

5.4 Synchronization (SYNC)

In order to synchronize the slaves a SYNC telegram is used. This telegram is unconfirmed (broadcast). It contains no data. The COB ID is stipulated in object *1005_{hex} COB ID SYNC*. By default, 80_{hex} is specified. The CANopen slave option module can receive SYNC telegrams. It is not a SYNC master!

Receipt of a SYNC telegram with the identifier set in object 1005_{hex} generates an interrupt to the CANopen option module, which is passed to the b maXX®controller. Because of this, this signal can be used to synchronize the b maXX® controller. All relevant telegrams must be transmitted to all configured slaves within one SYNC interval (communication cycle). Transfer rate, cable length, number of nodes, size of telegrams as well as processing times on the CANopen option card are to be taken into consideration during this setting the cycle time for the SYNC telegram is undertaken in object 1006_{hex}. For this, see ▷Directory of objects for communication control◁ from page 31. Furthermore, the communication cycle time in the ProDrive/WinBASS II must be matched on the page option module 1 BACI.

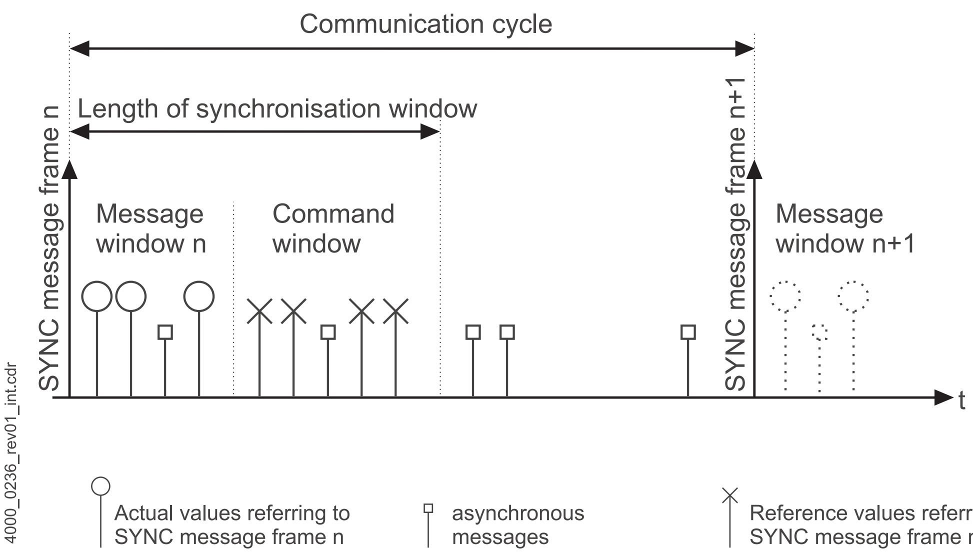


Figure 10: Communication cycle

After receipt of the SYNC telegram, the slaves first of all transmit their actual values by means of synchronous PDOs in the message window before the setpoints in the command window are transferred from the master to the slaves likewise by means of synchronous PDOs. The setpoints are accepted by the slaves with the next SYNC telegram (also see ▷Communication relationship via PDO◁ from page 65). Asynchronous messages (SDOs, PDOs, NMT) can occur at any time.

If the controller is not synchronized via the CANopen, no monitoring of the synchronization may occur in the controller. The SYNC telegram can continue to be used as trigger condition.

Settings for synchronization of the controller in ProDrive or WINBASS II

Additionally to the BACI settings the following settings must be done at the operating modes position control and synchronous operation on the page synchronization in ProDrive/WINBASS II.

Source for Sync signal:	Use Sync1 or Sync2 from BACI

The Sync interval must be set to the same time as the Sync telegram is sent (can be set also via FBO $0x1006_{hex}$)

Fault reaction at Sync error:	as required
Sync offset:	100
Sync tolerance:	max. 40 µs, otherwise depending on the jitter of the master

The Sync tolerance is the time in which the controller does not send a fault message, when the Sync telegram is received within this time. A mechanism (PLL) is implemented on the option module to compensate the jitter, so that the synchronization can take place a greater jitter. The „PLL“ is activated after the first received Sync telegram. The Sync telegram must be sent without interruption.

If the Sync telegram fails and switches on again during operation, it will take a half minute in the worst case until the „PLL“ is synchronous to the Sync telegram again, even if ProDrive/WinBASS II displays after a short time that the controller is running synchronous again.

The „PLL“ can be deactivated at the option module from FPGA version 0107 on. However this is only recommended if the master is able to send a jitter less than 1.6 µs for the Sync telegram.

The settings must be saved in the data set and the controller must be rebooted.

NOTE

If the cyclic communication is interrupted, e.g. at the transition from OPERATIONAL to PRE-OPERATIONAL the error/warning Alive Counter or the error cyclic communication can occur. The error cyclic communication occurs only if this monitoring is activated.

If the operating modes position control and synchronous operation are not used, there are no extra settings required for the BACI times and for the synchronization.

5.5 Process data (PDO)

Process data objects are unconfirmed telegrams with high-priority COB IDs. They are optimized for the exchange of data with real time requests. In the PDOs, the entire CAN data frame (8 bytes) can be used for user data transmission. The format of the data exchange via PDOs must therefore be defined before the start of communication between transmitter and receiver (mapping). Transmitting and receiving PDOs can be triggered in various ways (also see ▷Communication relationship via PDO◁ from page 65).

NOTE

All configured objects in the PDOs are transferred between the CANopen slave option module and the b maXX® controller as cyclic data (also see ▷Communication flow◁ from page 19). Because cyclic data transfer takes place only in the OPERATIONAL state, communication monitoring in Win BASS II BACI may be activated only in this state (timeout for cyclic communication **P0836** (BACI)).

5.5.1 PDO mapping

Mapping is a method of assigning variables/objects to PDOs. These variables/objects are moved across the CAN bus with the PDOs. Cyclic data exchange is configured by means of the mapping. SDOs are used for this parameterizing. Mapping is set in the object directory via addressable objects. There are four such objects for each PDO (also see ▷Directory of objects for communication control◁ from page 31). One of the objects determines the content of the PDO, the second one the communication relationship or triggering.

Process data object	Object for content	Object for the communication relationship
TX-PDO1	$1A00_{hex}$	1800_{hex}
TX-PDO2	$1A01_{hex}$	1801_{hex}
TX-PDO3	$1A02_{hex}$	1802_{hex}
TX-PDO4	$1A03_{hex}$	1803_{hex}
RX-PDO1	1600_{hex}	1400_{hex}
RX-PDO2	1601_{hex}	1401_{hex}
RX-PDO3	1602_{hex}	1402_{hex}
RX-PDO4	1603_{hex}	1403_{hex}

NOTE

Mapping cannot be changed in the OPERATIONAL state. New mapping will only be activated after switching to OPERATIONAL.

For the user data transmission a CAN data telegram provides a maximum of eight bytes. By mapping the logic content is determined by a maximum of eight bytes. For this determination certain information about the object, which is mapped is necessary: object index, subindex and length of datum. From the object index the according objects are entered in the mapping object. The sequence of this input, determined by the subindex of the mapping object, determines the data sequence in the CAN telegram. In the mapping objects (1600_{hex}, 1601_{hex}, 1602_{hex}, 1603_{hex}, $1A00_{hex}$, $1A01_{hex}$, $1A02_{hex}$, $1A03_{hex}$) the objects which are mapped, are written to the according subindices (start with 01_{hex}), e. g. to object 1600_{hex} subindex $_{hex}$ the value 60400010_{hex} is entered . This means, that the first two bytes of the data, which was received in RX-PD01 is written to the control word (object 6040_{hex} $subindex_{hex}$). The object 6040_{hex} is implemented to the b maXX®4400 parameter **P0300** control word (also see ▹Appendix C - Conversion tables◃ on page 101). Therewith the first word of the received telegram in the RX-PDO1 is written to the control word of the b maXX®4400. In subindex 00_{hex} the number of the objects, which must be mapped (number of assigned subindices with the valid objects) must be entered. In ▹Example for PDO mapping◃ from page 67 is a detailed example for mapping.

PDO mapping object		
00_{hex}	03_{hex}	
01_{hex}	Index b (10_{hex})	Object b
02_{hex}	Index N (08_{hex})	Object n
03_{hex}	Index a/ a2 (10_{hex})	Object a2

Object directory		
Index a	Subindex a1	
	Subindex a2	Object a2
Index b	Subindex b	Object b
Index n	Subindex n	Object n

PDO data field in the CAN telegram	Byte 0		Byte 4
	Object b	Object n	Object a2
	16 bit	8 bit	16 bit

Default mapping is described in ▹Directory of objects for communication control◃ from page 31.

In order to deactivate an existing mapping, the values in the subindices can be overwritten or the value '0' can be written to subindex 00_{hex} of the corresponding mapping object (1600_{hex}, 1601_{hex}, 1602_{hex}, 1603_{hex}, $1A00_{hex}$, $1A01_{hex}$, $1A02_{hex}$, $1A03_{hex}$). In this way, the entire mapping of the respective PDO will be deactivated, but the entry is preserved.

Furthermore deactivation of mapping objects via the associated communication objects 1400_{hex} to 1403_{hex}, 1800_{hex} to 1803_{hex} in subindex 1 with bit 31 set to 1 can take place. Note: therewith COBID must be written.

NOTE

At the setting of mapping in the (1600_{hex}, 1601_{hex}, 1602_{hex}, 1603_{hex}, $1A00_{hex}$, $1A01_{hex}$, $1A02_{hex}$, $1A03_{hex}$) is accordingly the subindex 00_{hex} with the correct number of the mapped objects is to be written at the end.

Setpoints: The permissible cyclical setpoints are marked in a table with the column 'PDO mapping' as 'RX'. The table is found in appendix B.2 (for the six thousands object numbers). The manufacturer-specific parameters (four thousands objects) must be checked up in the parameter manual b maXX®4400 basic unit (5.02017), chapter 6.1.4 attributes, for the b maXX®4400.

Actual values: The permissible cyclic actual values are marked in a table with the column 'PDO mapping' as 'TX'. The table is to be found in appendix B.2 (for the six thousands object numbers). At manufacturer-specific parameters (four thousands objects) it must be checked up in the parameter manual b maXX®4400 basic unit (5.02017), chapter 6.14 attributes, for the b maXX®4400. A detailed description of the b maXX® parameters are found in the parameter manual to b maXX®.

Incorrect mapping configuration (invalid mapping configurations in 1600_{hex}, 1601_{hex}, 1602_{hex}, 1603_{hex}, $1A00_{hex}$, $1A01_{hex}$, $1A02_{hex}$, $1A03_{hex}$) are signaled by abort codes via SDO.

The cyclic setpoints/actual values are initialized continuously in the BACI configuration, e.g. the first setpoint of PDO1 is in the first place in the BACI, the second setpoint of PDO1 in the second place and so on. Thereupon the setpoints of PDO2 follow. Analogously valid for the actual value initialization is the first actual value of PDO1 is in first place in the BACI, the second actual value of PDO1 in the second place and so on.

If the total range of PDO1 (max. four setpoints) is not used, the values of PDO2 move up. The cyclic data ranges of BACI are continuously.

NOTE

If the controller status word ($FB0641_{hex}$) is requested for the cyclic communication, then the status word must be entered in the first PDO in the first place!

If the status word is not requested, nevertheless it will be entered at BACI, but it is not considered at the field bus transfer in this case. The status word is required for internal samplings at this point. Only 7 more actual values can be used.

If a wrong parameter is mapped at the setpoints (e.g. an actual value parameter), the slave sends an EMYC telegram. Generally only the first error number is sent, when several error messages are present. This is not necessarily the error number which just triggers the error. (Therefore read out all present errors in ProDrive / WINBASS II if there are several error messages.)

In this case the controller reports the error number 54.4100 „Option module G/H wrong parameter number at setpoint parameter no. n".

Indeed the option module signalizes via the LED that it is in OPERATIONAL mode, but no cyclic setpoints are be handed on to the controller. The reason for this is that the controller has aborted the configuration of the BACI.

NOTE

A restart of the option module is required in this status.

Dummy mapping The option module CANopen slave provides 2 dummy objects: a 1 byte dummy object and a 2 byte dummy object, which also can be mapped into a PDO. These objects have the indices 0005_{hex} (1 byte dummy) and 0006_{hex} (2 byte dummy). The dummy object serves as a dummy, in order to use only certain objects within a CAN telegram (also see ▷Example for PDO mapping◁ from page 67).

NOTE

The present mapping, which was set gets lost after switching off or after a reset of the state machine. Thereupon the mapping, which was last saved via the object 1010_{hex} is set. If no mapping was saved, then the default mapping is used.

Description of equal field bus objects (FBO) via service data (SD) and process data (PD)

Generally PD write accesses overwrite cyclically SD write accesses in the same FBO. this is even then the case if the PD with the same FBO is not sent, but another FBO in another PD. The reason for this is that all listed BACI parameters are transfered from a therein contained parameter when a change is done.

In several cases it can happen that a write access via PD has been successfully, but this is not reliably.

NOTE

Avoid the access to the same field bus object via SD and PD in this context.

5.5.2 Communication relationship via PDO

In each mapping object an object exists for the setting of the communication.

The object index has an offset of -200$_{hex}$ for the corresponding mapping object.

Process data object mapping	Object for content	Object for communication relationship
TX-PDO1	1A00$_{hex}$	1800$_{hex}$
TX-PDO2	1A01$_{hex}$	1801$_{hex}$
TX-PDO3	1A02$_{hex}$	1802$_{hex}$
TX-PDO4	1A03$_{hex}$	1803$_{hex}$
RX-PDO1	1600$_{hex}$	1400$_{hex}$
RX-PDO2	1601$_{hex}$	1401$_{hex}$
RX-PDO3	1602$_{hex}$	1402$_{hex}$
RX-PDO4	1603$_{hex}$	1403$_{hex}$

The structure of these objects is described in ▹Directory of objects for communication control◃ from page 31.

The criterion for accepting a message transmitted onto the CAN bus into the CAN open slave option module is the matching COB ID. The COB ID is set in control objects 1400$_{hex}$ - 1403$_{hex}$, 1800$_{hex}$- 1803$_{hex}$ under subindex 01$_{hex}$. If the identifier parameterized here agrees with the message identifier transmitted via the CAN bus, the telegram will be accepted into its telegram buffer.

The PDOs can also be deactivated in this place, thereby the bit 31 is written to with 1.

e. g. 1400$_{hex}$ subindex 1

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
601$_{hex}$	08$_{hex}$	23$_{hex}$	00$_{hex}$	14$_{hex}$	01$_{hex}$	01$_{hex}$	00$_{hex}$	00$_{hex}$	**0**1$_{hex}$

Object 14 00$_{hex}$ Important Bit 31

Furthermore, triggering requirements for transmission and reception are defined for CANopen that group PDOs into synchronous and asynchronous. Triggering requirements are set in the objects 1400$_{hex}$- 1403$_{hex}$, 1800$_{hex}$- 1803$_{hex}$ accordingly in the subindex 02$_{hex}$.

NOTE

PDOs which are not needed should be deactivated, in order to exclude interferences. Deactivation occurs with the writing of the subindices 0 of the objects 1600$_{hex}$ to 1603$_{hex}$ and 1A00$_{hex}$ to 1A03$_{hex}$ with 0 or with bit 31 in SIX 2 of the objects 1401$_{hex}$ to 1403$_{hex}$ and 1801$_{hex}$ to 1803$_{hex}$.

Synchronous PDOs

Transmission and reception is linked with the SYNC telegram (also see ▷Synchronization (SYNC)◁ from page 59).

Asynchronous PDOs

Transmission and reception is linked with certain events.

Value in subindex 02_{hex}	Type	Effect: TX-PDO 1800_{hex}, 1801_{hex}, 1802_{hex}, 1803_{hex}	Effect: RX-PDO 1400_{hex}, 1401_{hex}, 1402_{hex}, 1403_{hex}
00_{hex} (0)	asynchronous	Transmission takes place after each SYNC telegram received **and** an event has occurred.	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
01_{hex} (1)	synchronous	Transmission occurs after each received SYNC telegram.	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
02_{hex} - $F0_{hex}$ (2 - 240)	synchronous	Transmission occurs after receiving the set number of SYNC telegrams	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
FC_{hex} (252)	RTR synchronous	Transmission takes place after receipt of the RTR telegram with matching COB ID updating the PDO takes place after receipt of the last SYNC telegram.	none
FD_{hex} (253)	RTR asynchronous	Transmission takes place after receipt of the RTR telegram with matching COB ID	none
FE_{hex} (254)	asynchronous	Transmission takes place in a time-controlled manner	none
FF_{hex} (255)	asynchronous	Transmission takes place in an event-controlled manner	Each PDO with a matching COB ID is accepted

Time-controlled transmission means that the transmission requirement is linked to a timer. This timer is set for the TX-PDO1 by means of subindex 05_{hex} in object 1800_{hex}(16 bit) is set in a similar manner. Analog the timer for TX-PDO2, TX-PDO3 and TX-PDO4 in the subindex 05_{hex} of object 1801_{hex}, 1802_{hex}, 1803_{hex} can be set. The resolution is 1 millisecond. The timer(s) is (are) started on change of state to OPERATIONAL. Transmission of the corresponding TX-PDO then takes place cyclically with the cycle time set in the timer. The timer will be cleared by writing the value '0' to subindices 05_{hex} of object 1800_{hex} - 1803_{hex}.

Time-controlled reception does not exist! The effect corresponds to event-controlled receipt.

Event-controlled transmission means that the transmission requirement is linked to the value change of the mapped objects. If for example 3 objects are mapped (status word, speed actual value, actual operating mode), the PDO transmits as soon as at least one of the 3 values changes. If the values remain constant, no PDO will be transmitted. Because of this, bus loading can be reduced (telegrams are only transmitted when they contain new information).

Event-controlled reception means that all PDOs with matching COB IDs will be accepted.

With transmission types synchronous RTR/asynchronous RTR (types 252 and 253), the PDO with matching COB ID will be transmitted after receipt of the RTR telegram. With type 252, the TX PDO is updated after each SYNC telegram received but not yet transmitted. With type 253, updating the PDO takes place after receipt of the RTR telegram (depending on the BACI cycle time). RTR telegrams are possible only for TX PDOs.

5.5.3 Example for PDO mapping

The CANopen slave option module with node 2 receives a speed setpoint from the master in RX PDO1. This speed setpoint must be written to the ramp-function generator input. The CANopen slave option module with node 7 should always exhibit a speed actual value identical to node 2. This value will be written to the ramp-function generator input of node 7. Implementation of this configuration is as follows:

The master transmits the speed setpoint to node 2. As soon as node 2 recognizes a change of this value, it transmits the actual value to node 7.

Furthermore, node 2 receives the control word from the master in its RX-PDO1. Node 7 likewise receives a control word from the master in RX-PDO 2. The configuration is shown in ▹Figure 11◃ on page 67. Object 6086$_{hex}$ is used in connection with dummy mapping.

The b maXX®4400 with the address 2 transmits its speed actual value and the status word every 10 ms. Node 7 transmits its status word only after receiving a SYNC telegram (from the master) 3 times.

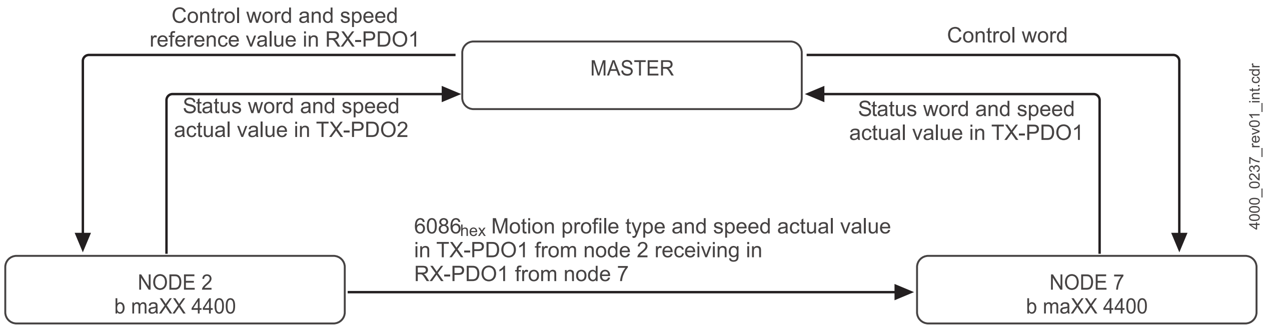


Figure 11: Example mapping with two b maXX®4400

1st step: determining the necessary objects

Ascertain the relevant object directory objects from the object list (see ▷Appendix C - Conversion tables◁ on page 101 and ▷Directory of objects for communication control◁ from page 31).

The following parameters are relevant for the devices that correspond with the specified objects:

P0301 Status word	⇔	6041_{hex} Status word
P0300 Control word	⇔	6040_{hex} Control word
P1171 Setpoint selection HLG input	⇔	6042_{hex} Speed setpoint at the HLG
P0353 Speed actual value	⇔	6044_{hex} Control effort
P1190 Positioning mode	⇔	6086_{hex} Motion profile type

The following objects are necessary for setting mapping:

Node 2

$1A00_{hex}$ (1. transmit PDO mapping), 1800_{hex} (1. transmit PDO parameters)

$1A01_{hex}$ (2. transmit PDO mapping), 1801_{hex} (2. transmit PDO parameters)

Node 7

1600_{hex} (1. receive PDO mapping), 1400_{hex} (1. receive PDO parameters)

1601_{hex} (2. receive PDO mapping), 1401_{hex} (2. receive PDO parameters)

2nd step: configure mapping

In order to set mapping the SDOs of the expedited transfers (also see ▷Service data (SDO)◁ from page 49) are used. These can be initiated via a master, a bus configurator or similar.

Mapping for slave 2

Write the first object to be mapped with index (6086_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 01_{hex} (TX-PDO 1). The object shall not be evaluated by slave 7.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	23_{hex}	00_{hex}	$1A_{hex}$	01_{hex}	10_{hex}	00_{hex}	86_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the first object to be mapped with index (6044_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 02_{hex} (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	23_{hex}	00_{hex}	$1A_{hex}$	02_{hex}	10_{hex}	00_{hex}	44_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	02_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Writing the number of the mapped objects (02_{hex}) to $1A00_{hex}$ subindex 00_{hex} (TXP-DO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	$2F_{hex}$	00_{hex}	$1A_{hex}$	00_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object $1A00_{hex}$ is as follows:

$1A00_{hex}$	**00_{hex}**	02_{hex}
	01_{hex}	60860010_{hex}
	02_{hex}	60440010_{hex}

The following mapping of object 6041_{hex} is shown for completeness, but need not be carried out because it is set in the default mapping.

Write the first object to be mapped with index (6041_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A01_{hex}$ subindex 01_{hex} (TX-PDO 2).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
602_{hex}	08_{hex}	23_{hex}	01_{hex}	$1A_{hex}$	01_{hex}	10_{hex}	00_{hex}	41_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
582_{hex}	08_{hex}	60_{hex}	01_{hex}	$1A_{hex}$	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Writing of the second object to be mapped with index (6044_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A01_{hex}$ subindex 02_{hex}.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
602_{hex}	08_{hex}	23_{hex}	01_{hex}	$1A_{hex}$	02_{hex}	10_{hex}	00_{hex}	44_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
582_{hex}	08_{hex}	60_{hex}	10_{hex}	$1A_{hex}$	02_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the number of the mapped objects (02_{hex}) to $1A01_{hex}$ subindex 00_{hex} (TX-PDO 2).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
602_{hex}	08_{hex}	$2F_{hex}$	01_{hex}	$1A_{hex}$	00_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
582_{hex}	08_{hex}	60_{hex}	01_{hex}	$1A_{hex}$	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object $1A01_{hex}$ is as follows:

$1A01_{hex}$	**00_{hex}**	02_{hex}
	01_{hex}	60410010_{hex}
	02_{hex}	60440010_{hex}

Write the first object to be mapped with index (6040_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 01_{hex} (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	23_{hex}	00_{hex}	16_{hex}	01_{hex}	10_{hex}	00_{hex}	40_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the second object to be mapped with index (6042_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 02_{hex} (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	23_{hex}	00_{hex}	16_{hex}	02_{hex}	10_{hex}	00_{hex}	42_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the number of the mapped objects (02_{hex}) to 1600_{hex} subindex 00_{hex} (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
602_{hex}	08_{hex}	$2F_{hex}$	00_{hex}	16_{hex}	00_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
582_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object 1600_{hex} is as follows:

1600$_{hex}$	**00**$_{hex}$	02_{hex}
	01$_{hex}$	60400010_{hex}
	02$_{hex}$	60420010_{hex}

Mapping for slave 7

In RX-PDO 1, slave 7 should only evaluate the speed setpoint of slave 2 (here the speed actual value). The speed setpoint is mapped to the second position of TX PDO 1 slave 2. For this reason, the dummy object must be used for the first position.

Write the first object to be mapped with index (6041_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 01_{hex} (TX-PDO 1).
However, this is also entered as the default mapping and need not necessarily be entered again.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	23_{hex}	00_{hex}	$1A_{hex}$	01_{hex}	10_{hex}	00_{hex}	41_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the first object to be mapped with index (6044_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 02_{hex} (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	23_{hex}	00_{hex}	$1A_{hex}$	02_{hex}	10_{hex}	00_{hex}	44_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	02_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the number of the mapped objects (02_{hex}) to $1A00_{hex}$ subindex 00_{hex} (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	$2F_{hex}$	00_{hex}	$1A_{hex}$	00_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	$1A_{hex}$	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object $1A00_{hex}$ is as follows:

$1A00_{hex}$	**00_{hex}**	02_{hex}
	01_{hex}	60410010_{hex}
	02_{hex}	60440010_{hex}

Write the first object to be mapped (16 bit dummy object) with index (0006_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 01_{hex} (RX PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	23_{hex}	00_{hex}	16_{hex}	01_{hex}	10_{hex}	00_{hex}	06_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the second object to be mapped with index (6042_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 02_{hex}.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	23_{hex}	00_{hex}	16_{hex}	02_{hex}	10_{hex}	00_{hex}	42_{hex}	60_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the number of the mapped objects (02_{hex}) to 1600_{hex} subindex 00_{hex}.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	$2F_{hex}$	00_{hex}	16_{hex}	00_{hex}	02_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	00_{hex}	16_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object 1600_{hex} is as follows:

1600_{hex}	**00_{hex}**	02_{hex}
	01_{hex}	00060010_{hex}
	02_{hex}	60420010_{hex}

Write the first object to be mapped with index (6040_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1601_{hex} subindex 01_{hex} (RX-PDO 2).

Request

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	23_{hex}	01_{hex}	16_{hex}	01_{hex}	10_{hex}	00_{hex}	40_{hex}	60_{hex}

Response

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	01_{hex}	16_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Write the number of the mapped objects (01_{hex}) to 1601_{hex} subindex 00_{hex}.

Request

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
607_{hex}	08_{hex}	$2F_{hex}$	01_{hex}	16_{hex}	00_{hex}	01_{hex}	00_{hex}	00_{hex}	00_{hex}

Response

		CS	**Multiplexor**			**D0**	**D1**	**D2**	**D3**
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
587_{hex}	08_{hex}	60_{hex}	01_{hex}	16_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

The content of object 1601_{hex} is as follows:

1601_{hex}	**00_{hex}**	01_{hex}
	01_{hex}	60400010_{hex}

Data exchange between the b maXX®4400 via the PDOs is shown in ▷Figure 12◁ on page 76. Example for a cross communication. The speed actual value of slave 2 becomes speed set value of slave 7.

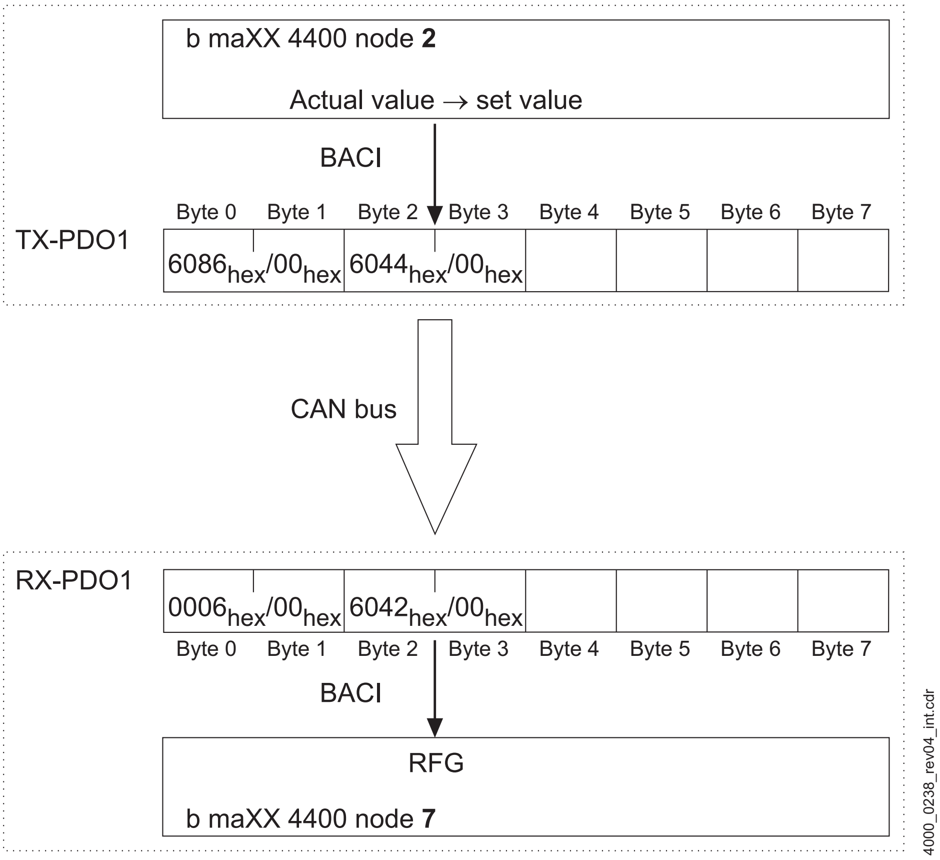


Figure 12: Telegram structure for example mapping

Only bytes 2 and 3 from TX-PDO1 of the b maXX®4400 node 2 are evaluated in b maXX®4400 node 7, because in RX-PDO1 only bytes 2 and 3 are validly linked with parameter numbers.

The communication parameters of both slaves are set via SDOs. So that a communication relationship can be built up, the COB-IDs of the transmitter and receiver must be in agreement. The COB-ID of the TX-PDO1 of slave 2 is by default set to 182_{hex}. The COB-ID of the TX-PDO1 of slave 7 is by default 207_{hex}. Which COB-ID must be used is the responsibility of the user. In the example, the COB-ID 207_{hex} is used.

Communication parameter for slave 2:

On object 1800_{hex} subindex 01_{hex} the value 207_{hex} is entered. On subindex 02_{hex} the value „FF_{hex}“ (event-controlled) is written.

1800_{hex}		
	01_{hex}	207_{hex}
	02_{hex}	FF_{hex}

The value FE_{hex} (timer triggered) is entered into object 1801_{hex} subindex 02_{hex}. The value ’$0A_{hex}$’ is written to subindex 05_{hex} (timer value = 10 ms).

1801_{hex}		
	02_{hex}	FE_{hex}
	05_{hex}	$0A_{hex}$

The remaining subindices contain their default values.

The type of trigger for RX-PDO1 is set to event-triggered (
(1400_{hex} subindex 02_{hex} = FF_{hex}).

Communication parameter for Slave 7:

On object 1400_{hex} subindex 02_{hex} the value „FF_{hex}“ (event controlled) is written. In subindex 01_{hex} is 207_{hex} by default.

1400_{hex}		
	01_{hex}	207_{hex}
	02_{hex}	FF_{hex}

Both of the TX-PDOs of node 7 keep their default COB IDs. The trigger type of TX-PDO1 is set to SYNC triggered with the value 03_{hex}. TX-PDO2 is parameterized as event triggered.

1800_{hex}		
	01_{hex}	187_{hex}
	02_{hex}	03_{hex}

1801_{hex}		
	01_{hex}	287_{hex}
	02_{hex}	FF_{hex}

5.5.4 Entry in BACI

A maximum of 8 cyclic setpoints and 8 cyclic actual values can be simultaneously exchanged between the CANopen slave option module and the b maXX® controller. All values are updated in one cycle. With CANopen, the setpoint and actual values can each be distributed across 4 PDOs.

For example if two setpoints each for TX PDO1 and TX PDO2 are to be mapped, in the course of this the following BACI configuration results:

BACI position	PDO	PDO position
1	TX-PDO1	1. Object
2	TX-PDO1	2. Object
3	TX-PDO2	1. Object
4	TX-PDO2	2. Object

The same method applies for RX PDOs.

The entries in the BACI are made continuously, beginning with the first object of PDO1.

NOTE

The dummy object is not taken into consideration in the BACI initialization.

Beginning with the first object of PDO1, the contents of the PDOs are alternately queried for their validity for the BACI configuration (not a dummy). If the object is valid, then this is entered into the next free BACI configuration position. If the PDO mapping is invalid (incorrect parameter numbers or the like), no cyclic communication between option card and b maXX ®4400 is started.

NOTE

If the same object number is mapped several times into the available PDOs of the same direction, then the object will appear only once in the BACI configuration.

Herewith it must be considered, that the data possibly interacts.

5.6 Error telegram (EMCY) according to DSP 402

Emergency telegrams serve as information for b maXX®4400-errors. This telegram is transmitted, as soon as b maXX®4400 recognizes an internal error. An emergency telegram is transmitted at each new error. There is no telegram repetition.

5.6.1 Telegram structure

The user data area of the emergency telegram is organized into three sections:

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80_{hex} + address	08_{hex}	Emergency error code		Error register	Manufacturer-specific error field				

In accordance with the predefined connection set, the COB ID results from 80_{hex} + node.

The emergency error code (4 bytes 0, 1) is defined in CANopen DSP 402. The conversion to b maXX®4400 error numbers is shown in ▷Conversion of error messages to DSP 402 V1.1◁ from page 80.

The error register corresponds to the content of object 1001h (also see ▷Directory of objects for communication control◁ from page 31).

The first two bytes of the manufacturer-specific error field contain the bmaXX®4400 error number.

Example

Slave 5 has recognized an encoder error at encoder 1 (cable break encoder 1).
The EMCY telegram then is the following type:

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
85_{hex}	08_{hex}	00_{hex}	73_{hex}	81_{hex}	0_{hex}	73_{hex}	00_{hex}	00_{hex}	00_{hex}

If there are several errors and an error is reset, the option module CANopen slave transmits the EMCY telegram with the next error number. If all errors are acknowledged, the telegram „Error reset / no error“ is transmitted.

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80_{hex} + address	08_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

5.6.2 Conversion of error messages to DSP 402 V1.1

The description of controller error messages and information on the rectification of faults can be found in the manual b maXX® 5.01040. The following table shows the conversion of controller error messages to CANopen error messages.

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0000_{hex}	Reserved	--
0001_{hex}	Watchdog-Error	7400_{hex}
0002_{hex}	Incorrect or unexpected interrupt has occurred	7400_{hex}
0003_{hex}	NMI interrupt has occurred - incorrect bus access	7400_{hex}
0010_{hex}	System boot error	5530_{hex}
0011_{hex}	Software error (e. g. switch)	5530_{hex}
0012_{hex}	Configuring error of the time-slice operating system	7400_{hex}
0013_{hex}	time slot - time error	7400_{hex}
0014_{hex}	No more free memory	7400_{hex}
0015_{hex}	Software error: invalid error code	7400_{hex}
0016_{hex}	Software error: invalid warning code	7400_{hex}
0017_{hex}	FPGA version is not compatible with firmware	7400_{hex}
0018_{hex}	Error at writing on target parameter by two-level controller output	
0019_{hex}	checksum error in external flash (system date, e.g. system time	
$001A_{hex}$	Power unit type (e.g. NWR) is not supported by firmware	
0020_{hex}	Timeout Proprog protocol	8100_{hex}
0021_{hex}	Protocol error	7400_{hex}
0022_{hex}	Incorrect module type	7400_{hex}
0023_{hex}	Too many data in the list and telegram	7400_{hex}
0024_{hex}	Too little data in the list and telegram	7400_{hex}
0025_{hex}	Invalid operand	7400_{hex}
0026_{hex}	Device supports only VARSTAT_MEMORY	7400_{hex}
0027_{hex}	Invalid operand address (log. address)	7400_{hex}
0028_{hex}	Value less than the minimum value	7400_{hex}
0029_{hex}	Value greater than the maximum value	7400_{hex}
$602A_{hex}$	Parameter is read-only	7400_{hex}
$002B_{hex}$	Parameter cannot be changed because of operational status	7400_{hex}
$002C_{hex}$	Invalid parameter value	7400_{hex}

Controller Error code	Description (of b maXX®controller)	CANopen Error code
002D$_{hex}$	ProDrive/WinBASS is not connected or inactive any-more	7400$_{hex}$
0030$_{hex}$	Error in SmallModule_A (to determine the related error number, parameter **P0240** in the b maXX® must be read, the error designation of **P0240** is described after this table)	FF00$_{hex}$
0031$_{hex}$	Error in SmallModule_B (to determine the related error number, parameter **P0240** in the b maXX® must be read; the error designation of **P0241**is described after this table)	FF00$_{hex}$
0032$_{hex}$	Error in SmallModule_C (to determine the related error number parameter **P0242** in the b maXX® must be read; the error designation of **P0240**is described after this table)	FF00$_{hex}$
0033$_{hex}$	Error in SmallModule_D (to determine the related error number, parameter **P0243** in the b maXX® must be read; the error designation of **P0240** is described after this table)	FF00$_{hex}$
0034$_{hex}$	Error in SmallModule_F (to determine the related error number , parameter **P0244** in the b maXX® must be read, the error designation of **P0240** is described after this table)	FF00$_{hex}$
0035$_{hex}$	Error in BigModule_G (to determine the related error number, the parameter **P0245** in the b maXX® must be read; the error designation of the **P0240** is described after this table)	FF00$_{hex}$
0036$_{hex}$	Error in BigModule_H (to determine the error number, the parameter **P0246** in the b maXX® must be read, the error designation of the **P0240** is described after this table)	FF00$_{hex}$
0037$_{hex}$	Error in the BigModule_J (to determine the error num-ber, the parameter **P0247** in the b maXX® must be read, the error designation of the **P0240** is described after this table)	FF00$_{hex}$
0038$_{hex}$	Error in the BigModule_K (to determine the related error number, parameter **P0248** in the b maXX® must be read; the error designation of **P0240** is described after this table)	FF00$_{hex}$
0039$_{hex}$	Error in BigModule_L (to determine the related error number, parameter **P0249** in the b maXX® must be read, the error designation of **P0240** is described after this table)	FF00$_{hex}$
003A$_{hex}$	Error in BigModule_M (to determine the related error number, the parameter **P0250** in the b maXX® must be read; the error designation of **P0240** is described after this table)	FF00$_{hex}$

Controller Error code	Description (of b maXX®controller)	CANopen Error code
$003B_{hex}$	Timeout at system initialization procedure	7400_{hex}
$003C_{hex}$	CRC error in SPI transmission module ⇒ controller	7400_{hex}
$003D_{hex}$	CRC error in SPI transmission controller ⇒ module	7400_{hex}
0040_{hex}	Mains failure	3100_{hex}
0041_{hex}	Phase error	3130_{hex}
0042_{hex}	Mains undervoltage	3120_{hex}
0043_{hex}	Mains overvoltage	3110_{hex}
0044_{hex}	24 V undervoltage	3100_{hex}
0050_{hex}	Communication errors in accordance with the hiperface specification (to determine the related error number, parameter **P0233** in the b maXX® must be read, the error designation is described after this table)	$FF00_{hex}$
0051_{hex}	Temperature threshold of heatsink exceeded	4210_{hex}
0052_{hex}	U DC link overvoltage	3210_{hex}
0053_{hex}	Overcurrent power unit	2310_{hex}
0054_{hex}	Ground current	2240_{hex}
0055_{hex}	Temperature threshold of inside air exceeded	4110_{hex}
0057_{hex}	Safety relay off (or defect)	5441_{hex}
0058_{hex}	Safety relay off (safety relay O.K., but there is no voltage)	5442_{hex}
0059_{hex}	Power unit not ready-to-operate	5400_{hex}
$005A_{hex}$	Phase error	3130_{hex}
$005B_{hex}$	Mains failure	3100_{hex}
$005C_{hex}$	Mains undervoltage	3120_{hex}
$005D_{hex}$	Mains overvoltage	3110_{hex}
$005E_{hex}$	Undervoltage U DC link	3100_{hex}
0060_{hex}	Temperature sensor of the motor short-circuited ($T_M \leq$ -30 °C)	4320_{hex}
0061_{hex}	Temperature sensor of the motor not connected ($T_M >$ +300 °C)	4310_{hex}
0062_{hex}	Error motor temperature - switch-off threshold exceeded	4310_{hex}
0063_{hex}	Error I^2t > 100% in the motor	7120_{hex}
0064_{hex}	Error CurrentDriveMax > MotorPeakCurrent	

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0070_{hex}	Communication error acc. to hiperface specification (to determine the error number the parameter **P0234**/ **P0235** in the b maXX® must be read, the error description is described subsequently to this table)	$FF00_{hex}$
0071_{hex}	Invalid module code	7300_{hex}
0072_{hex}	Error at writing of encoder position	7300_{hex}
0073_{hex}	Cable break encoder 1	7300_{hex}
0074_{hex}	Overspeed encoder 1	7310_{hex}
0075_{hex}	Amplitude limit exceeded	7300_{hex}
0076_{hex}	Encoder type unknown	7300_{hex}
0077_{hex}	Invalid data field for motor data	7120_{hex}
0078_{hex}	Incorrect motor data	7120_{hex}
0079_{hex}	Saving error of motor data	7120_{hex}
$007A_{hex}$	Motor data write-protected (not BM motors)	7120_{hex}
$007B_{hex}$	Error field angle	7300_{hex}
$007C_{hex}$	Encoder without temperature measuring	7300_{hex}
$007D_{hex}$	EEPROM capacity in encoder is insufficient	
0080_{hex}	Communication error according to hiperface specification (to determine the error number the parameter **P0235** in the b maXX® must be read, the error description is described subsequently to this table)	$FF00_{hex}$
0081_{hex}	Invalid module code	7300_{hex}
0082_{hex}	Error at writing of encoder position	7300_{hex}
0083_{hex}	Cable break encoder 2	7300_{hex}
0084_{hex}	Overspeed encoder 2	7310_{hex}
0085_{hex}	Amplitude limit exceeded	7300_{hex}
0086_{hex}	Encoder type unknown	7300_{hex}
0087_{hex}	Invalid data field for motor data	7120_{hex}
0088_{hex}	Incorrect motor data	7120_{hex}
0089_{hex}	Saving error of motor data	7120_{hex}
$008A_{hex}$	Motor data write-protected (not BM motors)	7120_{hex}
$008B_{hex}$	Field angle error	7300_{hex}
$008C_{hex}$	Encoder without temperature measuring	7300_{hex}
$008D_{hex}$	EEPROM capacity in encoder is insufficient	
0090_{hex}	Absolute position of the encoder unknown	7320_{hex}
0091_{hex}	Absolute position of the encoder unknown	7320_{hex}

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0092_{hex}	Encoder module 1 is missing	7300_{hex}
0093_{hex}	Encoder module 2 is missing	7300_{hex}
0094_{hex}	Measurement storage for encoder module is missing	7300_{hex}
0095_{hex}	At resolver no measured value storage possible	7300_{hex}
0096_{hex}	Triggering on zero pulse and encoder is no incremental encoder	7300_{hex}
0097_{hex}	Digital I/O module required and missing	7300_{hex}
0098_{hex}	Incremental encoder emulation module required and missing	7300_{hex}
0099_{hex}	Encoder module 1 required for incremental encoder emulation and missing	7300_{hex}
$009A_{hex}$	Encoder module 2 required for incremental encoder emulation and missing	7300_{hex}
$009B_{hex}$	Initialization error of the incremental encoder emulation module	7300_{hex}
$009C_{hex}$	Incremental encoder emulation module signals error	7300_{hex}
$009D_{hex}$	Incremental encoder emulation: Selecting the option ’start after first zero pulse’ for non-incremental encoder	7300_{hex}
$009E_{hex}$	SSI encoder emulation module is missing	7300_{hex}
$009F_{hex}$	Error in setpoint source encoder 1 or encoder 2	7300_{hex}
$00A0_{hex}$	Time monitoring Proprog communication	8100_{hex}
$00A1_{hex}$	Time monitoring BACI communication	8100_{hex}
$00A2_{hex}$	Time monitoring cyclic communication	8110_{hex}
$00A3_{hex}$	Time monitoring service data transmission	8100_{hex}
$00A4_{hex}$	Field bus error	8100_{hex}
$00A5_{hex}$	Controller not synchronous to external signal	8100_{hex}
$00A6_{hex}$	Error at brake control	8100_{hex}
$00A7_{hex}$	Brake does not open	
$00A8_{hex}$	Brake does not close	
$00A9_{hex}$	Cyclic monitoring of brake state messages error	
$00AA_{hex}$	Cyclic monitoring of brake lining messages error	
$00AB_{hex}$	DIO module for control/acknowledgement of the brake fails	
$00AC_{hex}$	Holding torque not reached before opening the brake	
$00B0_{hex}$	EEPROM copy error	5530_{hex}
$00B1_{hex}$	Timeout while writing to EEPROM	5530_{hex}

Controller Error code	Description (of b maXX®controller)	CANopen Error code
00B2hex	Checksum error in EEPROM	5530hex
00B3hex	No boot record	5530hex
00B4hex	Incompatible SW	5530hex
00B5hex	Data record switching: DS not present	5530hex
00B6hex	Checksum error in the PSI	5530hex
00B7hex	PSI is reset	5530hex
00B8hex	PSI data invalid	5530hex
00B9hex	Self-optimization tables are invalid. Execute self-optimization again	5530hex
00BAhex	A/D correction table invalid	5530hex
00BBhex	EEPROM is deleted	
00C0hex	Position deviation dynamic	8611hex
00C1hex	Position deviation static	8611hex
00C2hex	Encoder 1 for position control used, but not active	7300hex
00C3hex	Encoder 2 for position control used, but not active	7300hex
00C4hex	Software limit switch monitoring 1 active	8600hex
00C5hex	Software limit switch monitoring 2 active	8600hex
00C6hex	Hardware limit switch monitoring 1 active	8600hex
00C7hex	Hardware limit switch monitoring 2 active	8600hex
00C8hex	Positioning started without homing	8600hex
00C9hex	Setpoint in the mode set-of-setpoints didn't arrive in time	8600hex
00CAhex	Monitoring of modulo position active: target position > modulo position	8600hex
00CBhex	Spindle positioning: Error at initialization of the trigger	8600hex
00CChex	Spindle positioning: Timeout at trigger signal (zero pulse / switch input)	8600hex
00CDhex	Error at homing and process has broken	8600hex
00D0hex	Drive blocked	7121hex
00D1hex	Encoder 1 is used for motor control but inactive	7300hex
00D2hex	Encoder 2 is used for motor control but inactive	7300hex
00D3hex	Overspeed Open Loop	7300hex
00E0hex	Translation error	
00E1hex	Runtime error	
00EAhex	Error quick discharge (at the time only in BM41XX (NRW))	

Controller Error code	Description (of b maXX®controller)	CANopen Error code
$00EB_{hex}$	Slave has switched off because of error in master (consecutive reaction to error in master, if parameterized so)	
$00EC_{hex}$	No operating mode speed control, although drive is a slave	
$00ED_{hex}$	Common error return motion configuration not valid (e.g. operation of the motor is not allowed at mains failure)	
$00EE_{hex}$	Return motion positioning could not be completed	
$00EF_{hex}$	Application error (triggered by control word 2)	
The manufacturer-specific error codes 0030_{hex} to $003B_{hex}$, 0050_{hex}, 0070_{hex} and 0080_{hex} are issued together via the EMY telegram with the CANopen error code $FF00_{hex}$. The exact term for this can be read in the following parameters **P0233**, **P0234**, **P0235** and **P0240** to **P0250** and the description concerning the read error numbers is shown subsequently.		

Controller Error code	Description (of b maXX® controller)	CANopen Error code
The following b maXX® error codes are not issued via an EMY telegram. Error code (0050_{hex}) ⇒ **P0233** Communication error in accordance with hiperface specification (AmpHiperfaceError)		
06_{hex}	Data overflow	
07_{hex}	Bit frame error	
08_{hex}	Invalid command state	
09_{hex}	Parity error	
$0A_{hex}$	Incorrect checksum of transmitted data	
$0B_{hex}$	Unknown command code	
$0C_{hex}$	Number of the transmitted data is wrong	
$0D_{hex}$	Invalid argument	
$0E_{hex}$	Data field is write protected	
$0F_{hex}$	Invalid access code	
10_{hex}	Data field size cannot be changed	
11_{hex}	Specified word address outside data field	
12_{hex}	Access to non-existent data field	
24_{hex}	Incorrect PU data checksum	
25_{hex}	No response from PU	
42_{hex}	Invalid response	
Error code (0070_{hex}) ⇒ **P0234** Communication error in accordance with hiperface specification (Enc1HiperfaceError) Error code (0080_{hex}) ⇒ **P0235** Communication error in accordance with hiperface specification (Enc2HiperfaceError)		
01_{hex}	Analog signals outside specification	
02_{hex}	Error in internal angle offset	
03_{hex}	Data field partitioning table destroyed	
04_{hex}	Analog limit values not available	
05_{hex}	Internal I2C bus not operational	
06_{hex}	Internal checksum error	
07_{hex}	Internal watchdog error - encoder reset	
09_{hex}	Parity error	
$0A_{hex}$	Checksum of transferred data is incorrect	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
$0B_{hex}$	Unknown command code	
$0C_{hex}$	Number of the transmitted data is wrong	
$0D_{hex}$	Invalid argument	
$0E_{hex}$	Data field is write protected	
$0F_{hex}$	Invalid access code	
10_{hex}	Data field size cannot be changed	
11_{hex}	Specified word address outside data field	
12_{hex}	Access to non-existent data field	
$1C_{hex}$	Absolute monitoring of the analog signals	
$1D_{hex}$	Transmission current critical	
$1E_{hex}$	Encoder temperature critical	
$1F_{hex}$	Speed too high - no position generation possible	
20_{hex}	Invalid position singleturn	
21_{hex}	Multiturn position error	
22_{hex}	Multiturn position error	
23_{hex}	Multiturn position error	
24_{hex}	Incorrect MT data checksum	
40_{hex}	No answer from HIPERFACE encoder	
41_{hex}	No response from EnDat encoder	
42_{hex}	Useless answer to encoder command	
50_{hex}	CRC has determined an error	
51_{hex}	Invalid command	
52_{hex}	Address or MRS-code in reply telegram is wrong	
53_{hex}	Alarm bit of the encoder is set	
54_{hex}	Storage in encoder is occupied	
55_{hex}	Checksum error when reading the motor data	
56_{hex}	Motor data length and/or data version of encoder and controller firmware are not identical	
57_{hex}	Starting operation test has not determined an EnDat interface at the encoder	
58_{hex}	Exceeding of transmission format which is able to be evaluated	
59_{hex}	Exceeding of the measuring step length which is to be evaluated	
$5A_{hex}$	Signal period length < measuring step length	
60_{hex}	Error lighting	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
61_{hex}	Error signal amplitude	
62_{hex}	Error position value	
63_{hex}	Error overvoltage	
64_{hex}	Error undervoltage	
65_{hex}	Error overcurrent	
66_{hex}	Error battery	
Error Code (0030_{hex} 0034_{hex}) ⇒ **P0240...P0244** Error im SmallModule 1 to 5		
01_{hex}	Module not recognized	
02_{hex}	Recognized modules at invalid position	
03_{hex}	Digital output short-circuited	
04_{hex}	Invalid target parameter value by digital input	
05_{hex}	Direct PLC IO access for this module not permitted	
07_{hex}	Module in controller not permitted	
Error Code (0035_{hex} 0040_{hex}) ⇒ **P0245...P0250** Error in BigModule 1 to 6		
1000_{hex}	Wrong parameter No. at setpoint parameter 1	
1001_{hex}	Wrong parameter No. at setpoint parameter 2	
1002_{hex}	Wrong parameter No. at setpoint parameter 3	
1003_{hex}	Wrong parameter No. at setpoint parameter 4	
1004_{hex}	Wrong parameter No. at setpoint parameter 5	
1005_{hex}	Wrong parameter No. at setpoint parameter 6	
1006_{hex}	Wrong parameter No. at setpoint parameter 7	
1007_{hex}	Wrong parameter No. at setpoint parameter 8	
1008_{hex}	Wrong parameter No. at setpoint parameter 9	
1009_{hex}	Wrong parameter No. at setpoint parameter 10	
$100A_{hex}$	Wrong parameter No. at setpoint parameter 11	
$100B_{hex}$	Wrong parameter No. at setpoint parameter 12	
$100C_{hex}$	Wrong parameter No. at setpoint parameter 13	
$100D_{hex}$	Wrong parameter No. at setpoint parameter 14	
$100E_{hex}$	Wrong parameter No. at setpoint parameter 15	
$100F_{hex}$	Wrong parameter No. at setpoint parameter 16	
1010_{hex}	Wrong parameter No. at actual value parameter 1	
1011_{hex}	Wrong parameter No. at actual value parameter 2	
1012_{hex}	Wrong parameter No. at actual value parameter 3	

Controller Error code	**Description (of b maXX® controller)**	**CANopen Error code**
1013_{hex}	Wrong parameter No. at actual value parameter 4	
1014_{hex}	Wrong parameter No. at actual value parameter 5	
1015_{hex}	Wrong parameter No. at actual value parameter 6	
1016_{hex}	Wrong parameter No. at actual value parameter 7	
1017_{hex}	Wrong parameter No. at actual value parameter 8	
1018_{hex}	Wrong parameter No. at actual value parameter 9	
1019_{hex}	Wrong parameter No. at actual value parameter 10	
$101A_{hex}$	Wrong parameter No. at actual value parameter 11	
$101B_{hex}$	Wrong parameter No. at actual value parameter 12	
$101C_{hex}$	Wrong parameter No. at actual value parameter 13	
$101D_{hex}$	Wrong parameter No. at actual value parameter 14	
$101E_{hex}$	Wrong parameter No. at actual value parameter 15	
$101F_{hex}$	Wrong parameter No. at actual value parameter 16	
1020_{hex}	Invalid value at setpoint parameter No. 1	
1021_{hex}	Invalid value at setpoint parameter No. 2	
1022_{hex}	Invalid value at setpoint parameter No. 3	
1023_{hex}	Invalid value at setpoint parameter No. 4	
1024_{hex}	Invalid value at setpoint parameter No. 5	
1025_{hex}	Invalid value at setpoint parameter No. 6	
1026_{hex}	Invalid value at setpoint parameter No. 7	
1027_{hex}	Invalid value at setpoint parameter No. 8	
1028_{hex}	Invalid value at setpoint parameter No. 9	
1029_{hex}	Invalid value at setpoint parameter No. 10	
$102A_{hex}$	Invalid value at setpoint parameter No. 11	
$102B_{hex}$	Invalid value at setpoint parameter No. 12	
$102C_{hex}$	Invalid value at setpoint parameter No. 13	
$102D_{hex}$	Invalid value at setpoint parameter No. 14	
$102E_{hex}$	Invalid value at setpoint parameter No. 15	
$102F_{hex}$	Invalid value at setpoint parameter No. 16	
1030_{hex}	Invalid value for setpoint period	
1031_{hex}	Invalid value for actual value period	
1032_{hex}	Wrong value for cycle offset setpoints	
1033_{hex}	Wrong value for cycle offset actual values	
1034_{hex}	BACI timeout at cyclic data	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
1035_{hex}	BACI timeout at service data	
1036_{hex}	Checksum error during test	
1037_{hex}	Ramp-up timeout when waiting for the slave type or when waiting for the resetting of config-pending-flag	
1038_{hex}	Invalid data transfer structure type	
1039_{hex}	Internal error: wrong BACI status	
$103A_{hex}$	Access conflicts with slave by cyclic communication	
$103B_{hex}$	Error cyclic communication: parameter value wrong	
$103C_{hex}$	Error cyclic communication: alive-counter conflict	
$103D_{hex}$	Cmd interface: channel number wrong (0 or > 6)	
$103E_{hex}$	Cmd interface: the channel which was indicated does not exist	
$103F_{hex}$	Cmd interface: internal error - wrong pointer	
1040_{hex}	Cmd interface: internal error - wrong status	
1041_{hex}	Cmd interface: wrong package number	
1042_{hex}	Cmd interface: wrong command number	
1043_{hex}	Cmd interface: wrong status when handling the pack-age	
1044_{hex}	Cmd interface: timeout at command processing	
1045_{hex}	Cmd interface: wrong package length	
1046_{hex}	Cmd interface: descriptor not available (too little memory)	
1047_{hex}	Cmd interface: wrong package type	
1048_{hex}	Cmd interface: checksum error	
1049_{hex}	Module identification: PCI-error when reading	
$104A_{hex}$	Module identification: PCI-error when writing	
$104B_{hex}$	Module identification: general error at reading	
$104C_{hex}$	Module identification: general error at writing	
$104D_{hex}$	Internal error	
$104E_{hex}$	Configuration cyclic services: parameters are not or not cyclic writable	
$104F_{hex}$	Configuration cyclic services: invalid parameter number	
1050_{hex}	Incorrect option modules error code (settable with **P1007**)	
2000_{hex}	Error CANopen timeout on CAN bus (node guarding)	

A

APPENDIX A - ABBREVIATIONS

BACI	Baumüller drives serial interface
CA	Collision Avoidance
CSMA	Carrier Sense Multiple Access
DS	Draft Standard
DSP	Draft Standard Proposal
EMCY	Error frame message
HD	Hamming Distance
ID	Ident Number
LMT	Layer Management
M	Multiplexer
NMT	Network Management
PC	Personal Computer
PDO	Process data object
SDO	Service data object
SIX	Index
SPS	Programmable control
SYNC	Synchronization

APPENDIX B - QUICK REFERENCE

The following quick reference shows the connection between CANopen object numbers and the b maXX® controller parameter numbers (see manual b maXX® 5.02017).

B.1 ’4000’ object numbers (manufacturer-specific objects)

Manufacturer-specific objects result from
4000_{hex} + parameter $number_{hex}$.
Therefore the subindex for all 4000-target parameters is always 00_{hex}.

Example

Parameter **P0053** $\Rightarrow$ object index 4035_{hex} subindex 00_{hex}

B.2 '6000' object numbers (device profile DSP 402)

You can access numerous parameters with one '4000' **and** also with one or several '6000' objects.

There are only few parameters that can be accessed exclusively with a '6000' parameter ($606A_{hex}$, 6048_{hex} SIX 1, 6049_{hex} SIX 1, $604C_{hex}$ SIX 1 und SIX 2).

Please note the changed standardizations!

NOTE

Between the '4000' and '6000' parameters different standardizations can occur!

TX: Transmit; RX: Receive; r: read; w: write; ro: read only; wo: write only

CANopen object number		Parameter number	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	Subindex				
6007_{hex}	00_{hex}	**P0300**	TX / RX	rw	Common entries
6040_{hex}	00_{hex}	**P0300**	TX / RX	rw	Device control
6041_{hex}	00_{hex}	**P0301**	TX	ro	Device control
6042_{hex}	00_{hex}	**P1171**	TX / RX	rw	Velocity mode
6043_{hex}	00_{hex}	**P0351**	TX	ro	Velocity mode
6044_{hex}	00_{hex}	**P0353**	TX	ro	Velocity mode
6046_{hex}	01_{hex}	**P1041**	TX	ro	Velocity mode
6046_{hex}	02_{hex}	**P1041**, **P1042**	TX / RX	rw	Velocity mode
6048_{hex}	01_{hex}	**P3329**	TX / RX	rw	Velocity mode
6048_{hex}	02_{hex}	**P1172**	TX / RX	rw	Velocity mode
6049_{hex}	01_{hex}	**P3330**	TX / RX	rw	Velocity mode
6049_{hex}	02_{hex}	**P1173**	TX / RX	rw	Velocity mode
$604C_{hex}$	01_{hex}	**P3314**	TX / RX	rw	Velocity mode
$604C_{hex}$	02_{hex}	**P3315**	TX / RX	rw	Velocity mode
$604D_{hex}$	00_{hex}	**P0065**	TX	rw	Velocity mode
$605E_{hex}$	00_{hex}	**P1007**	TX		Device control
$604F_{hex}$	00_{hex}	**P1172**	TX / RX	rw	Velocity mode
6050_{hex}	00_{hex}	**P1173**	TX / RX	rw	Velocity mode

CANopen object number		Parameter number	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	Subindex				
6051_{hex}	00_{hex}	**P1174**	TX / RX	rw	Velocity mode
$605A_{hex}$	00_{hex}	**P1004**	TX	rw	Device control
$605B_{hex}$	00_{hex}	**P1005**	TX	rw	Device control
$605C_{hex}$	00_{hex}	**P1006**	TX	rw	Device control
$605D_{hex}$	00_{hex}	**P1003**	TX	rw	Device control
6060_{hex}	00_{hex}	**P1000**	- / RX	rw	Device control
6061_{hex}	00_{hex}	**P0304**	TX	ro	Device control
6062_{hex}	00_{hex}	**P0463**	TX	ro	Position control function
6063_{hex}	00_{hex}	**P0362**	TX	ro	Position control function
6064_{hex}	00_{hex}	**P0462**	TX	ro	Position control function
6066_{hex}	00_{hex}	**P1056**	TX	rw	Position control function
6067_{hex}	00_{hex}	**P1194**	TX / RX	rw	Position control function
6068_{hex}	00_{hex}	**P1195**	TX	rw	Position control function
6069_{hex}	00_{hex}	**P0362**	TX / RX	rw	Profile velocity mode
$606A_{hex}$	00_{hex}	-	-	ro	Profile velocity mode
$606B_{hex}$	00_{hex}	**P0352**	TX	ro	Profile velocity mode
$606C_{hex}$	00_{hex}	**P0353**	TX	ro	Profile velocity mode
$606F_{hex}$	00_{hex}	**P1073**	TX / RX	rw	Profile velocity mode
6072_{hex}	00_{hex}	**P0357**	TX / RX	rw	Profile torque mode
6077_{hex}	00_{hex}	**P0344**	TX	ro	Profile torque mode
$607A_{hex}$	00_{hex}	**P0600/ P0607**	TX / RX	rw	Profile position mode
$607C_{hex}$	00_{hex}	**P1200**	TX / RX	rw	Homing mode
$607D_{hex}$	01_{hex}	**P1196**	TX	rw	Profile position mode
$607D_{hex}$	02_{hex}	**P1197**	TX	rw	Profile position mode
$607F_{hex}$	00_{hex}	**P0057**	TX	rw	Profile position mode
6080_{hex}	00_{hex}	**P1031**	TX	rw	Profile position mode
6081_{hex}	00_{hex}	**P0602**	TX	rw	Profile position mode
6083_{hex}	00_{hex}	**P0603**	TX	rw	Profile position mode
6084_{hex}	00_{hex}	**P0604**	TX	rw	Profile position mode
6085_{hex}	00_{hex}	**P1213**	TX	rw	Profile position mode

CANopen object number Index	Subindex	Parameter number	PDO mapping	Access type	Operating mode acc. to DSP 402
6086_{hex}	00_{hex}	**P1190**	TX	rw	Profile position mode
6092_{hex}	01_{hex}	**P1193**	TX	rw	Factor group
6092_{hex}	02_{hex}	**P3050**	TX	rw	Factor group
6098_{hex}	00_{hex}	**P3051**	TX	rw	Homing mode
6099_{hex}	01_{hex}	**P1201**	TX / RX	rw	Homing mode
6099_{hex}	02_{hex}	**P1202**	TX / RX	rw	Homing mode
$609A_{hex}$	00_{hex}	**P1203**	TX / RX	rw	Homing mode
$60C0_{hex}$	00_{hex}			ro	Interpolated mode
$60C1_{hex}$	01_{hex}	**P369**	TX / RX	rw	Interpolated mode
$60C2_{hex}$	01_{hex}			ro	Interpolated mode
$60C2_{hex}$	02_{hex}	**P532**	TX / RX	rw	Interpolated mode
$60C3_{hex}$	01_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C3_{hex}$	02_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C4_{hex}$	01_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C4_{hex}$	02_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C4_{hex}$	03_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C4_{hex}$	04_{hex}	**P3331**	TX / RX	rw	Interpolated mode
$60C4_{hex}$	05_{hex}	**P3331**	RX	wo	Interpolated mode
$60C4_{hex}$	06_{hex}	**P3331**	RX	wo	Interpolated mode
$60FB_{hex}$	00_{hex}	**P1054**	TX / RX	rw	Profile velocity mode
$60FB_{hex}$	01_{hex}	**P0360**	TX	ro	Position control function
$60FB_{hex}$	02_{hex}	**P1050**	TX	rw	Position control function
$60FB_{hex}$	03_{hex}	**P1051**	TX	rw	Position control function
$60FB_{hex}$	04_{hex}	**P0364**	TX / RX	rw	Position control function
$60FB_{hex}$	05_{hex}	**P0363**	TX / RX	rw	Position control function
$60FB_{hex}$	06_{hex}	**P1053**	TX	rw	Position control function
$60FB_{hex}$	07_{hex}	**P0367**	TX	ro	Position control function
$60FB_{hex}$	08_{hex}	**P0362**	TX / RX	rw	Position control function
$60FB_{hex}$	09_{hex}	**P0392**	TX	ro	Position control function
$60FB_{hex}$	$0A_{hex}$	**P0391**	TX	ro	Position control function
$60FB_{hex}$	$0B_{hex}$	**P0365**	TX	ro	Position control function

CANopen object number Index	Subindex	Parameter number	PDO mapping	Access type	Operating mode acc. to DSP 402
$60FB_{hex}$	$0C_{hex}$	**P0460**	TX	ro	Position control function
$60FB_{hex}$	$0D_{hex}$	**P1191**	TX / RX	rw	Position control function
$60FB_{hex}$	$0E_{hex}$	**P1190**	TX	rw	Position control function
$60FB_{hex}$	$0F_{hex}$	**P1200**	TX	rw	Position control function
$60FB_{hex}$	10_{hex}	**P1208**	TX	rw	Position control function
$60FB_{hex}$	11_{hex}	**P0464**	TX	ro	Position control function
$60FB_{hex}$	12_{hex}	**P0605**	TX / RX	rw	Position control function
$60FB_{hex}$	13_{hex}	**P1198**	TX / RX	rw	Position control function
$60FB_{hex}$	14_{hex}	**P1199**	TX / RX	rw	Position control function
$60FB_{hex}$	15_{hex}	**P0601**	TX / RX	rw	Position control function
$60FB_{hex}$	16_{hex}	**P0608**	TX / RX	rw	Position control function
$60FB_{hex}$	17_{hex}	**P0370**	TX / RX	rw	Position control function
$60FB_{hex}$	18_{hex}	**P1209**	TX / RX	rw	Position control function
$60FB_{hex}$	19_{hex}	**P1204**	TX / RX	rw	Position control function
$60FB_{hex}$	$1A_{hex}$	**P0353**	TX	ro	Position control function
$60FB_{hex}$	$1B_{hex}$	**P0262** **P0263**	TX	ro	Position control function
$60FD_{hex}$	00_{hex}	**P0461**	TX	ro	Common entries
$60FF_{hex}$	00_{hex}	**P1171**	TX / RX	rw	Profile velocity mode
6510_{hex}	01_{hex}	**P0001**	TX	ro	Info
6510_{hex}	02_{hex}	**P0002**	TX	ro	Info
6510_{hex}	03_{hex}	**P0003**	TX	ro	Info
6510_{hex}	04_{hex}	**P0004**	TX	ro	Info
6510_{hex}	05_{hex}	**P0005**	TX	ro	Info
6510_{hex}	06_{hex}	**P0009**	TX	ro	Info
6510_{hex}	07_{hex}	**P0555**	TX	ro	Info
6510_{hex}	08_{hex}	**P0556**	TX	ro	Info

C

APPENDIX C - CONVERSION TABLES

This chapter contains tables specifying the conversion of CANopen communication objects into b maXX® controller communication parameters and vice versa. Conversion is performed by stating the value ranges ($x = x_{min} .. x_{max}$) and the mapping function $x = f(x)$ (in the most simple case, the value is just passed through: $y = x$).

The tables contain the following entries:

CANopen object: Identification of the CANopen object from DS402

Index ◂ P. No.: Mapping of CANopen object indices to b maXX®controller parameters

Controller parameters: Identification of the controller parameters

P. No. ▸ index: Conversion of the b maXX®controller parameter to CANopen object indices

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
abort_connection_option _code	6007_{hex}	►	**P0300**			**P0300**	►	6007_{hex}	The parameter 6007_{hex} affects the control word in the b maXX®, in the course of which local variables are used Note: Quickstop reactions via FBO $605A_{hex}$ to be set. For the mode malfunction via writing on **P3145** an error is activated, whose reaction can be set via FBO $605E_{hex}$. **Only at changes in ’Option module G/H configuration 1’ bit 3 = 1,** default value then is 3instead of 0.
	x = -32768 .. 32767	►	y = x			x =0 .. $FFFF_{hex}$	►	y = x	
No action	x = 0				not used	x = 0	►	y = x	
Malfunction	x = 1				Response adjustable	x = 1	►	y = x	
Device control command „disable_voltage“	x = 2	►	y = x		Inhibit voltage	x = 2	►	y = x	
Device control command ’quick_stop’	x = 3	►	y = 4		Quickstop	x = 3	►	y = 4	
reserved	x =4 .. 32767	►	y = x		not used	x =4 .. 32767	►	y = x	
Manufacturer specific	x = -32768 .. -1								
control word	6040_{hex}	►	**P0300**		Control word	**P0300**	►	6040_{hex}	Bit 6 in the control; Bit 6 = 0 : Positioning mode „absolute“ Bit 6 = 1: Positioning mode „relative, negative positive“ No other positioning mode is supported via the CANopen and the control word. In the operating mode = 7 (IP mode) Bit 4 is set to1 always.
	x =0 .. $FFFF_{hex}$	►	y = x			x =0 .. $FFFF_{hex}$	►	y = x	
Switch On	Bit 0	►	unchanged		Switch on	Bit 0	►	unchanged	
Disable voltage	Bit 1	►	unchanged		Inhibit voltage	Bit 1	►	unchanged	
Quickstop	Bit 2	►	unchanged		Quickstop	Bit 2	►	unchanged	
Enable Op.	Bit 3	►	unchanged		Operation enabled	Bit 3	►	unchanged	
Operation mode specific	Bit 4	►	unchanged		Depending on operation mode	Bit 4	►	unchanged	
Operation mode specific	Bit 5	►	unchanged		Depending on operation mode	Bit 5	►	unchanged	
Operation mode specific	Bit 6	►	unchanged		Depending on operation mode	Bit 6	►	unchanged	
Reset fault	Bit 7	►	unchanged		Reset error	Bit 7	►	unchanged	
Operation mode specific	Bit 8	►	unchanged		Depending on operation mode	Bit 8	►	unchanged	
reserved	Bit 9	►	unchanged		Reserved (always 0)	Bit 9	►	unchanged	
reserved	Bit 10	►	unchanged		Reserved (always 0)	Bit 10	►	unchanged	
Manufacturer specific	Bit 11	►	unchanged		Depending on operation mode	Bit 11	►	unchanged	
Manufacturer specific	Bit 12	►	unchanged		Depending on operation mode	Bit 12	►	unchanged	
Manufacturer specific	Bit 13	►	unchanged		Depending on operation mode	Bit 13	►	unchanged	
Manufacturer specific	Bit 14	►	unchanged		Depending on operation mode	Bit 14	►	unchanged	
Manufacturer specific	Bit 15	►	unchanged		Write protection	Bit 15	►	unchanged	

CANopen object	Index Value range	▸	P. No. Scaling		Controller parameters	P. No. Value range	▸	Index) Rescaling	Comment
status word	6041_{hex} /ro				Status word	**P0301**	▸	6041_{hex}	In the operating mode = 7 (IP mode): If control word Bit 4 will be set, then Bit 12 is set
	x =0 .. $FFFF_{hex}$					x =0 .. FFFF-hex	▸	y = x	
Ready to switch on					Ready-to-start	Bit 0	▸	unchanged	
Switched on					Switched on	Bit 1	▸	unchanged	
Operation enabled					Operation enabled	Bit 2	▸	unchanged	
Fault					Error	Bit 3	▸	unchanged	
Voltage disabled					Voltage disabled	Bit 4	▸	unchanged	
Quickstop					Quickstop	Bit 5	▸	unchanged	
Switched on enabled					Inhibit start	Bit 6	▸	unchanged	
Warning					Warning	Bit 7	▸	unchanged	In ProDrive/WinBASS via drive manager adjustable
Man. specific					Depending on operation mode	Bit 8	▸	unchanged	
Remote					Remote	Bit 9	▸	unchanged	
Target reached					Setpoint reached	Bit 10	▸	unchanged	
Internal limit active					Depending on operation mode	Bit 11	▸	unchanged	In ProDrive/WinBASS via drive manager adjustable
Operation mode specific					Depending on operation mode	Bit 12	▸	unchanged	
Operation mode specific					Depending on operation mode	Bit 13	▸	unchanged	
Manufacturer specific					Conf. status bits	Bit 14	▸	unchanged	
Manufacturer specific					Conf. status bits	Bit 15	▸	unchanged	
vl_target_velocity	6042_{hex}	▸	**P1171**		RFG1Input	**P1171**	▸	6042_{hex}	The user-defined unit (speed units) is interpreted in the b maXX® controller as RPM. Scaling of gear ratio is saved in FBO $604C_{hex}$. Only at changes in the option module G/H configuration 1 bit 2 = 1: Specification of desired speed in 1/10 RPM e. g.: 200.0 RPM ⇒ input 2000.
	x = -32768 .. 32767	▸	y = x *4000_{hex}/ MotorMax-Speed			x = -32768 .. 32767	▸	y = x*Motor-MaxSpeed / 4000_{hex}	

CANopen object	Index Value range	▸	P. No. Scaling		Controller parameters	P. No. Value range	▸	Index) Rescaling	Comment
vl_ velocity_demand	6043$_{hex}$ /ro				RFG output	**P0351**	▸	6043$_{hex}$	The user-defined unit (speed units) is interpreted in the b maXX® controller as RPM. Scaling of gear ratio is saved in FBO 604C$_{hex}$. Only at changes in the option module G/H configuration 1 bit 2 = 1: Internal setpoint for speed in 1/10 RPM, same unit as object 6042$_{hex}$. e. g.: 200.0 RPM ⇒input 2000.
						x = 8000$_{hex}$ .. 7FFF$_{hex}$	▸	y = x*Motor-MaxSpeed / 4000$_{hex}$	
vl_control_effort	6044$_{hex}$ /ro				SpeedActValue	**P0353**	▸	6044$_{hex}$	The user-defined unit (speed units) is interpreted in the b maXX®controller as RPM. Scaling of gear ratio is saved in FBO 604C$_{hex}$. Only at changes in option module G/H configuration 1 bit 2 = 1: Internal setpoint for speed in 1/10 RPM, same unit as object 6042$_{hex}$. e. g.: 200.0 RPM ⇒input 2000.
						x = 8000$_{hex}$ .. 7FFF$_{hex}$	▸	y = x*Motor-MaxSpeed / 4000$_{hex}$	
vl_control_effort	6045$_{hex}$ /ro				SpeedActValue	**P0352**	▸	6045$_{hex}$	The user-defined unit (speed units) is interpreted in the b maXX®controller as RPM. Scaling of gear ratio is saved in FBO 604C$_{hex}$. Only at modifications in option module G/H configuration 1 bit2 = 1: Internal setpoint for speed in 1/10 RPM, same unit as object 6042$_{hex}$. e. g.: 200.0 RPM ⇒input 2000.
						x = 8000$_{hex}$ .. 7FFF$_{hex}$	▸	y = x*Motor-MaxSpeed / 4000$_{hex}$	
vl_velocity_min_max_ amount	6046$_{hex}$ /ro								
vl_velocity_min_amount	Sub. 01$_{hex}$		’none’		SpeedSet_Ulim	„none“	▸	Sub. 01$_{hex}$	Sub. 1 is always zero, the min. limit is specified zero.
						x = 0	▸	y = x	
vl_velocity_max_amount	Sub. 02$_{hex}$	▸	**P1042** / **P1041**		SpeedSet_Llim	**P1042** / **P1041**	▸	Sub. 02$_{hex}$	The maximum limit symmetrical affects both speed directions in the b maXX®. The user-defined unit (speed units) is interpreted in the b maXX® controller as RPM
	x =0 .. FFFFFFFF-hex	▸	y = x* 40000000$_{hex}$ / MotorMax-Speed		**P1041**: x =0 .. 40000000$_{hex}$ **P1042**: x = C0000000$_{hex}$ .. 0		▸	y = x*Motor-MaxSpeed / 40000000$_{hex}$	
vl_velocity_acceleration	6048$_{hex}$	a = dv/dt; Because there is no acceleration parameter in the controller, the desired acceleration is achieved by varying the ramp-up time **P1172**. It is scaled to the maximum speed of the controller. The calculation for the desired acceleration follows when the input of dv in SIX1 and then dt in SIX2. Then the ramp-up time is written to the time, which was accurately calculated. After a reboot the reconstrution of the set acceleration is not possible anymore!							

CANopen object	Index Value range	P. No. Scaling	Controller parameters	P. No. Value range	Index) Rescaling	Comment
vl_delta_speed	Sub. 01_{hex}		Application parameter 16		Sub. 01_{hex}	Scaling of gear ratio is saved in FBO $604C_{hex}$. Only at changes in option module G/H configuration 1 bit 2 = 1: Internal setpoint for speed in 1/10 RPM, same unit as object 6042_{hex}. e. g.: 200.0 RPM ⇒input 2000.
	x =0 .. $FFFFFFFF_{hex}$	y = Δv		x =8000 .. $7FFF_{hex}$	y = Δv	
vl_delta_time	Sub. 02_{hex}	**P1172**	RFG1RampUpTime	**P1172**	Sub. 02_{hex}	delta_time is specified in seconds; corresponds to ramp-function generator-ramp-up time
	x =0 .. $FFFFFFFF_{hex}$	y = Δt*MotorMaxSpeed / Δv*100		x =0 .. 65000	y = Δt	
vl_velocity_deceleration	6049_{hex}	a = dv/dt; Because there is no deceleration parameter in the controller, the desired delay is achieved by varying the ramp-down time **P1172**. It is scaled to the maximum speed of the controller. The calculation for the desired delay follows when the input of dv in SIX1 and then dt in SIX2 was made. Then the correctly calculated ramp-up time is written to the delay. After a reboot the reconstrution of the set acceleration is not possible anymore!				
vl_delta_speed	Sub. 01_{hex}		Application parameter 17		Sub. 01_{hex}	Scaling of gear ratio is saved in FBO $604C_{hex}$. Only at changes in option module G/H configuration 1 bit 2 = 1: Internal setpoint for speed in 1/10 RPM, same unit as object 6042_{hex}. e. g.: 200.0 RPM ⇒input 2000.
	x =0 .. $FFFFFFFF_{hex}$	y = Δv		x =8000 .. $7FFF_{hex}$	y = Δv	
vl_delta_time	Sub. 02_{hex}	**P1173**	RFG1RampDownTime	**P1173**	Sub. 02_{hex}	delta_time is specified in seconds; corresponds to ramp-function generator ramp-down time
	x =0 .. $FFFFFFFF_{hex}$	y = Δt*MotorMaxSpeed / - Δv*100		x =0 .. 65000	y = Δt	
vl_dimension_factor	$604C_{hex}$				$604C_{hex}$	Calculation in the controller is e. g. as follows: Speed setpoint motor in the b maXX®: For vl_dimension_factor_numerator = 10 and vl_dimension_factor_denominator = 5 Speed setpoint motor = FBO [U/min]*vl_dimension_factor = 100*10 / 5 [RPM] = 200 [RPM]
vl_dimension_factor_ numerator	Sub. 01_{hex}				Sub. 01_{hex}	
	$X=-2^{31}$ .. $2^{31}-1$	y=X		x=-33000 .. 33000	y=X	
vl_dimension_factor_ denominator	Sub. 02_{hex}				Sub. 02_{hex}	
	$X=-2^{31}$ .. $2^{31}-1$	y=X		x=-33000 .. 33000	y=X	
vl_pole_number	$604D_{hex}$	**P0065**	Number of pole pairs	**P0065**	$604D_{hex}$	
	x =0 .. 255	y = x / 2		x = 1..120	y = x*2	
vl_ramp_function_time	$604F_{hex}$	**P1172**	RFG1RampUpTime	**P1172**	$604F_{hex}$	Ramp-function generator acceleration time (1 = 1/1000 s -> 1s = 1000). Resolution is 10 ms
	x = 0 .. $FFFFFFFF_{hex}$	y = x		x = 0 .. 65000	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
vl_slow_down_time	6050_{hex}	►	**P1173**		RFG1RampDownTime	**P1173**	►	6050_{hex}	Ramp-function generator ramp-up time (1 = 1/1000s, 1s = 1000). The resolution is 10 ms.
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x			x = 0..65000	►	y = x	
vl_quick_stop_time	6051_{hex}	►	**P1174**		RFG1StopTime	**P1174**	►	6051_{hex}	Ramp-function generator ramp-up time (1 = 1/1000 s, 1 s = 1000). The resolution is 10 ms.
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x			x = 0..65000	►	y = x	
quick_stop_option_code	$605A_{hex}$	►	**P1004**		QuickstopCode (quickstop)	**P1004**	►	$605A_{hex}$	
Conversion formalism	x = -32768 .. 32767	►	y = x			x = 0 .. 3	►	y = x	
Manufacturer specific	x = -32768 .. -1	►	y = x		not used	x = -32768 .. -1			
Disable drive	x = 0	►	y = x		Drive inhibited	x = 0	►	y = x	
Slow down on slow down ramp	x = 1	►	y = x		Ramp-down at deceleration ramp	x = 1	►	y = x	
Slow down on quickstop ramp	x = 2	►	y = x		Ramp down on quickstop ramp	x = 2	►	y = x	
Slow down on current ramp	x = 3	►	y = x		Ramp down at current limit	x = 3	►	y = x	
Slow down on voltage limit	x = 4	►	y = x		Ramp-down at voltage limit			y = 4	
Slow down on slow down ramp and remain in quick-stop	x = 5	►	y = x		Ramp-down ramp and remain in quickstop			y = 5	
Slow down on quickstop ramp and remain in quick-stop	x = 6	►	y = x		Ramp-down on quickstop ramp and remain in quickstop			y = 6	
Slow down on current and remain in quick-stop	x = 7	►	y = x		Ramp down at current limit and remain in quickstop.			y = 7	
Slow down on voltage limit and remain in quick-stop	x = 8	►	y = x		Ramp down at voltage limit and remain in quickstop			y = 8	
reserved	x = 9 .. 32767				not used			y = 9 .. 32767	
shutdown_option_code	$605B_{hex}$	►	**P1005**		ShutDownCode (shut down)	**P1005**	►	$605B_{hex}$	
Manufacturer specific	x = -32768 .. -3	►	y = x		not used	x = -32768..-3			
Manufacturer specific	x = -2	►	y = 3		Ramp down at current limit	x = 3	►	y = -2	
Manufacturer specific	x = -1	►	y = 2		Ramp-down at quickstop ramp	x = 2	►	y = -1	
Disable drive	x = 0	►	y = x		Drive inhibited	x = 0	►	y = x	
Slow down on slow down ramp	x = 1	►	y = x		Ramp-down at deceleration ramp	x = 1	►	y = x	of the selected RFG adjustable via **P1174** RFG stop time or in 6051 $_{hex}$

CANopen object	Index Value range	►	P. No. Scaling	Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
reserved	x = 2 .. 32767			not used			y = .. 32767	
disable_operation_ option_code	605C$_{hex}$	►	**P1006**	DisableOpCode (disable)	**P1006**	►	605C$_{hex}$	
Manufacturer specific	x = -32768 .. -3	►	y = x	not used	x = -32768..-3			
Manufacturer specific	x = -2	►	y = 3	not used			y = -2	
Manufacturer specific	x = -1	►	y = 2	not used			y = -1	
Disable drive	x = 0	►	y = x	Drive inhibited		►	y = 0	
Slow down ...	x = 1	►	y = x	Ramp-down at deceleration ramp	x = 1	►	y = x	
reserved	x = 2			Ramp down on quickstop ramp	x = 2	►	y = -1	
reserved	x = 3			Ramp down at current limit	x = 3	►	y = -2	
reserved	x = 4 .. 32767			not used			y = 4 .. 32767	
stop_option_code	605D$_{hex}$	►	**P1003**	StopOptionCode (Stop)	**P1003**	►	605D$_{hex}$	
Conversion formalism	x = -32768 .. 32767	►	y = x		x = 0 .. 3	►	y = x	
Manufacturer specific	x = -32768 .. -1	►	y = x	not used			y = -32768 .. -1	
Disable drive	x = 0	►	y = x	Drive inhibited	x = 0	►	y = x	
Slow down on slow down ramp	x = 1	►	y = x	Ramp-down at deceleration ramp	x = 1	►	y = x	of the selected RFG adjustable via **P1174** RFG stop time or in 6051$_{hex}$
Slow down on quickstop ramp	x = 2	►	y = x	Ramp down on quickstop ramp	x = 2	►	y = x	
Slow down on current ramp	x = 3	►	y = x	Ramp down at current limit	x = 3	►	y = x	
Slow down on voltage limit	x = 4	►	y = x	not used			y = 4	
reserved	x = 5 .. 32767			not used			y = 5 .. 32767	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
fault_reaction_option_ code	$605E_{hex}$	►	**P1007**		ErrorReactionCode	**P1007**	►	$605E_{hex}$	Presently for statical and dynamical position deviations and for the reactions for the FBO 6007_{hex} 'Mode 1 malfunction' adjustable
Conversion formalism	x = -32768 .. 32767	►	y = x			x = 0 .. 3	►	y = x	
Manufacturer specific	x = -32768 .. -1	►	y = x		not used	x = -32768 .. -1			
Disable Drive, motor is free to rotate	x = 0	►	y = x		Drive inhibited	x = 0	►	y = x	
Slow down on slow down ramp	x = 1	►	y = x		Ramp-down at deceleration ramp	x = 1	►	y = x	
Slow down on quickstop ramp	x = 2	►	y = x		Ramp down on quickstop ramp	x = 2	►	y = x	
Slow down on current ramp	x = 3	►	y = x		Ramp down at current limit	x = 3	►	y = x	
Slow down on voltage limit	x = 4	►	y = x		Ramp-down at voltage limit			y = 4	
reserved	x = 5 .. 32767				not used			y = 5 .. 32767	

CANopen object	Index Value range	P. No. Scaling	Controller parameters	P. No. Value range	Index) Rescaling	Comment
modes_of_operation	6060_{hex} /wo	**P1000**	OperationModeSet	**P1000**	6060	
Conversion formalism	x = -128 .. 127	y = x		x = -128 .. 127	y = x	
Manufacturer specific	x = -106	y' = -6				
Manufacturer specific	x = -7	y = x	Autotuning	x = -7	y = x	y', x' ≙ only internally in controller
Manufacturer specific	x = -6	y = 5	Spindle positioning	x = -6	y = -106	
Manufacturer specific	x = -5	y = x	Synchronous operation with electronic gearing	x = -5	y = x	According to the set operating mode: Operating mode = -4 Position control Operating mode = 7 IP mode (controller switches internally to operating mode -4)
Manufacturer specific	x = -4	y = x	Position control or Interpolated Position Mode	x = -4	y = -4 or y = 7	
Manufacturer specific	x = -3	y = x	Speed control	x = -3	y = 3	
Manufacturer specific	x = -2	y = x	Current control	x = -2	y = x	
Manufacturer specific	x = -1	y = x	Find notch position	x = -1	y = x	
reserved	x = 0		not used	x = 0		
Profile position mode	x = 1	y = x	Target position set value	x = 1	y = x	
Velocity mode	x = 2	y = x	Speed setting 1	x = 2	y = x	
Profile velocity mode	x = 3	y = -3	not used	x = 3	y = -3	
Torque profile mode	x = 4	y = x	not used	x = 4		
reserved	x = 5		Jog operation	x = 5	y = -6	
Homing mode	x = 6	y = x	Homing operation	x = 6	y = x	
Interpolated position mode	x = 7	y = x	not used	x = 7		
reserved	x = 8 .. 127		not used	x = 8 .. 127		

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
modes_of_operation_ display	6061_{hex} /ro				OperationModeAct (actual operating mode)	**P0304**	►	6061_{hex}	The CANopen standard name, see 6060_{hex}
Conversion formalism						x = -128 .. 127	►	y = x	
					Autotuning	x = -7	►	y = x	- Jog operation has value 5 in controller in CANopen jog operation has v alue -6 - Spindle positioning has value -6 in controller in CANopen spindle positioning has value -106
					Spindle positioning	x = -6	►	y = -106	
					Synchronous operation el. gear	x = -5	►	y = x	
					Position control	x = -4	►	y = x	
					Speed control	x = -3	►	y = 3	
					Current control	x = -2	►	y = x	
					Notch position	x = -1	►	y = x	
					not used	x = 0			
					Position set mode	x = 1	►	y = x	
					Speed setting 1	x = 2	►	y = x	
					not used	x = 3			
					not used	x = 4			
					Jog operation	x = 5	►	y = -6	
					Homing mode	x = 6	►	y = x	
					not used	x = 7			
					not used	x = 8 .. 127			
position_demand_value	6062_{hex} /ro				PPosSetValue (actual position value)	**P0463**	►	6062_{hex}	OUSIGN32 is provided on the CANopen option card with an offset of 2^{31}. USIGN32 -> INT32. (offset of 2^{31} is subtracted. UU - ratio added Only at changes in option module G/H con- figuration 1 bit 2 = 1, Specification of current position of the drive user units e. g. 1/100° for rotating move- ments starting from home position.
						x = 80000000 .. $7FFFFFFF_{hex}$	►	$y = x-2^{31}$	
position_actual_value*	6063_{hex} /ro				PPosActValue (actual position value)	**P0362**	►	6063_{hex}	UU - ratio added Only at changes in option module G/H con- figuration 1 bit 2 = 1: Specification of current position of the drive user units e. g. user units e. g. 1/100° for rotating movements starting from home position.
						x = 80000000 .. $7FFFFFFF_{hex}$	►	y = x	

CANopen object	Index Value range	►	P. No. Scaling	Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
position_actual_value	6064_{hex} /ro			PPosActValue (actual position value)	**P0462**	►	6064_{hex}	USIGN32 is provided on the CANopen option card with an offset of 2^{31}. USIGN32 ⇒INT32. (offset of 2^{31} is subtracted. UU - ratio added Only at changes in option module G/H configuration 1 bit 2 = 1: Specification of current position of the drive user units e. g. 1/100° for rotating movements starting from the home position.
					x = 80000000 .. $7FFFFFFF_{hex}$	►	$y = x-2^{31}$	
following_error_time_out	6066_{hex} /ro			PosDevTime	**P1056**	►	6066_{hex}	The unit in the CANopen object and in the b maXX®controller parameter is in ms.
					x = 0 .. 65000	►	y = x	
position_window	6067_{hex}	►	**P1194**	PPosWindow (pos. window)	**P1194**	►	6067_{hex}	
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x		x = 0 .. $FFFFFFFF_{hex}$	►	y = x	
position_window_time	6068_{hex}	►	**P1195**	PPosWindow Time (pos. window time)	**P1195**	►	6068_{hex}	
	x = 0 .. 65535	►	y = x		x = 1 .. $FFFF_{hex}$	►	y = x	
velocity_sensor_actual_ value	6069_{hex}	►	**P0362**	ENC1ActAngle	**P0391**	►	6069_{hex}	
	$x = -2^{15} .. 2^{15}-1$	►	y = x		x = 0 .. $FFFFFFFF_{hex}$	►	y = x	
sensor_selection_code	$606A_{hex}$ /ro			„none“				The b maXX®controller only supports the position encoder, therefore only display.
velocity_actual_value_from _position_encoder					x = 0	►	y = x	
velocity_actual_value_from _velocity_encoder				not supported				
velocity_demand_value	$606B_{hex}$ /ro			SetValueTotal	**P0352**	►	$606B_{hex}$	The user-defined unit (speed units) is interpreted in the controller as RPM. Only at changes in option module G/H configuration 1 bit 2 = 1: Specification of speed in 1/10 RPM. e. g.: 200.0 revolutions => input 2000.
					$x = 8000_{hex}$ .. $7FFF_{hex}$	►	y = x*MotorMaxSpeed / 4000_{hex}	
velocity_actual_value	$606C_{hex}$ /ro			SpeedActValue	**P0353**	►	$606C_{hex}$	Scaling of gear ratio is in FBO $604C_{hex}$ Only at changes in option module G/H configuration 1 bit 2 = 1: specification of current speed in 1/10 RPM e. g. 200.0 revolutions ⇒ input 2000.
					$x = -2^{31} .. 2^{31}-1$	►	y = x*MotorMaxSpeed / 40000000_{hex}	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
velocity_threshold	606F$_{hex}$	►	**P1073**		ENC1Mon_Llim	**P1073**	►	606F$_{hex}$	The threshold in the b maXX®controller is increased to 25% of maximum speed of the controller. The input then occurs in RPM e. g. max: 1000 RPM input for 25 % = 250 RPM
	x = -2^{31} .. 2^{31}-1	►	y = x*4000$_{hex}$ / 10000			x = -0 .. 1000$_{hex}$	►	y = x*10000 / 4000$_{hex}$	
max_torque	6072$_{hex}$	►	**P0357**		TrqSynDirect	**P0357**	►	6072$_{hex}$	
	x =0 ... FFFF$_{hex}$	►	y = x*4000$_{hex}$ / 1000			x = 0000 .. FFFF$_{hex}$	►	y = x*1000 / 4000$_{hex}$	
Torque actual value	6077$_{hex}$ /ro	►	**P0344**						1000 is equivalent to 100,0 % in relation to the nominal torque P1036
	x = -2^{16} ... 2^{16} - 1	►	y = x * 10000 / 1638						
target_position	607A$_{hex}$	►	**P0607** (**P0600**)		PPosTarget1	**P0607** (**P0600**)	►	607A$_{hex}$	UU - ratio added Only at changes in the option module G/H configuration 1 bit 2 = 1: User units e. g. 1/100° for rotating move-ments starting from home position.
	x = 80000000$_{hex}$ ..7FFFFFFF$_{hex}$	►	y = x			x = 800000000 .. 7FFFFFFF$_{hex}$	►	y = x	
home_offset	607C$_{hex}$	►	**P1200**		PPosEncoderOffset	**P1200**	►	607C$_{hex}$	Deviation of home position of homing- or limit switch UU - ratio and an offset of 2^{31} added (numerical scale conversion). Only at changes in the option module G/H configuration 1 bit 2 = 1: specified in UU e. g. 1/100 ° for rotating movements. No limit value monitoring.
	x = -2^{31} .. 2^{31}-1	►	y = x+2^{31}			x = 0 .. 2^{32}	►	y = x-2^{31}	
software_position_limit	607D$_{hex}$				SW limit switch			607D$_{hex}$	USIGN32 is provided with an offset of 2^{31} on the CANopen option card. USIGN32 ⇒ INT32. (offset of 2^{31} is subtracted. UU - ratio added.
	Sub. 01$_{hex}$	►	**P1196**		PPosSWLimitSwitch1	**P1196**	►	Sub. 01$_{hex}$	
	x = -2^{31} .. 2^{31}-1	►	y = x			x = 0 .. FFFFFFFF$_{hex}$	►	y = x	
	Sub. 02$_{hex}$	►	**P1197**		PPosSWLimitSwitch2	**P1197**	►	Sub. 02$_{hex}$	
	x = -2^{31} .. 2^{31}-1	►	y = x			x = 0 .. FFFFFFFF$_{hex}$	►	y = x	
max_profile_velocity	607F$_{hex}$	►	**P0057**		MotorNomSpeed	**P0057**	►	607F$_{hex}$	The user-defined unit (speed units) is inter-preted in the controller as RPM.
	x = 0 .. 2^{32}-1	►	y = x			x = 1 .. 24000	►	y = x	
max_motor_speed	6080$_{hex}$	►	**P1031**		SpeedMax	**P1031**	►	6080	The user-defined unit (speed units) is inter-preted in the controller as RPM.
	x = 0 .. FFFF$_{hex}$	►	y = x			x = 20 .. 24000	►	y = x	

CANopen object	Index Value range	P. No. Scaling	Controller parameters	P. No. Value range	Index) Rescaling	Comment
profile velocity	6081$_{hex}$	**P0602**	PPosSetSpeed1	**P0602**	6081$_{hex}$	Only at modifications in option module G/H configuration 1 bit2 = 1: Specification of desired moving speed of positioning in UU e. g. 1/100 °/s for rotating movements. s⇒ ms [1/1000]. UU added. At exceeding of limit value of parameter in b maXX® the maximum or minimum values are set without generating an error.
	x = 0 .. 2^{32}	y = x		x = 1 .. 13200	y = x	
profile acceleration	6083$_{hex}$	**P0603**	PPosAcceleraton1	**P0603**	6083$_{hex}$	Only at modifications in option module G/H configuration 1 bit2 = 1: Starting/braking acceleration of positioning is specified in UU e. g. 10°/s^2 for rotating movements. s^2 ⇒ ms^2 [1/10000] At exceeding of limit value of parameter in b maXX® the maximum or minimum values are set without generating an error.
	x = 0 .. 2^{32}	y = x		x = 25 .. 45000	y = x	
profile deceleration	6084$_{hex}$	**P0604**	PPosDeceleraton1	**P0604**	6084$_{hex}$	
	x = 0 .. 2^{32}	y = x		x = 25 .. 45000	y = x	
quick_stop_deceleration	6085$_{hex}$	**P1213**	PPosStopDeceleraton	**P1213**	6085$_{hex}$	Only at modifications in option module G/H configuration 1 bit2 = 1: Starting/braking acceleration of positioning is specified in UU e. g. 10°/s^2 for rotating movements. s^2 ⇒ ms^2 [1/10000] At exceeding of limit value of parameter in b maXX® the maximum or minimum values are set without generating an error.
	x = 0 .. 2^{32}	y = x		x = 25 .. 45000	y = x	
motion profile type	6086$_{hex}$	**P1190**	PPosMode	**P1190**	6086$_{hex}$	In controller
	x = -2^{16}... 2^{16}-1			x = 0.. FFFF$_{hex}$		
Manufacturer specific	x = -32768..-1		not used		y = x	
Linear ramp (trapezoidal profile)	x = 0	Bit 3 and bit 4:	Trapezium	Bit 3 and bit 4:	0	Speed profile: Bit 4 bit 3: 0 0: Trapezium 0 1: Sin² 1 0: S-curve 1 1: Reserved
Sin^2 ramp	x = 1	Bit 3 and bit 4:	Sin^2	Bit 3 and bit 4:	1	
Jerk-free ramp	x = 2	Bit 3 and bit 4:	S-curve	Bit 3 and bit 4:	2	
Jerk-limited ramp	x = 3		not used			
Reserved for future profile types	x = 4..	y = x	not used			

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
feed_constant	6092_{hex} /ro							6092_{hex}	
feed	Sub. 01_{hex}	►	**P3050**		PosScalingUserUnit	**P3050**	►	Sub. 01_{hex}	
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x			x = 2^{24} .. 1	►	y = x	
shaft_revolutions	Sub. 02_{hex}	►	**P3051**		PosScalingRevolution	**P3051**	►	Sub. 02_{hex}	
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x			x = 1 .. 2^{24}-1	►	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
homing_method	6098_{hex}	►	**P1205**		BM_i_Ds0_PPosHomingMod e (ref. homing mode)	**P1205**	►	6098_{hex}	
Manufacturer specific	x = -128 .. -12				not used			y = -128 .. -7	
Manufacturer specific	x = -10	►	y = -10		Reaching of mechanical stop with zero pulse, counter-clock-wise	x = -10	►	y = -10	
Manufacturer specific	x = -9	►	y = -9		Reaching of mechanical stop with zero pulse, clockwise rotation	x = -9	►	y = -9	
Manufacturer specific	x = -8	►	y = -8		Reaching of mechanical stop, counter-clockwise	x = -8	►	y = -8	
Manufacturer specific	x = -7	►	y = -7		Reaching of mechanical stop, clockwise rotation	x = -7	►	y = -7	
Manufacturer specific	x = -6	►	y = -6		Reaching of the next encoder zero angle	x = -6	►	y = -6	
Manufacturer specific	x = -5	►	y = -5		Reaching of the pos. limit switch	x = -5	►	y = -5	
Manufacturer specific	x = -4	►	y = -4		Reaching of the neg. limit switch	x = -4	►	y = -4	
Manufacturer specific	x = -3	►	y = -3		Setting of home position	x = -3	►	y = -3	
Manufacturer specific	x = -2	►	y = -2		Reaching the encoder zero angle or zero pulse with counter-clockwise rotation	x = -2	►	y = -2	
Manufacturer specific	x = -1	►	y = -1		Reaching the encoder zero angle or zero pulse with clock-wise rotation	x = -1	►	y = -1	
No homing operation	x = 0				not used			y = 0	
Homing on the neg. limit switch	x = 1	►	y = 1		Reaching of the neg. limit switch with encoder zero angle or zero pulse homing	x = 1	►	y = 1	
Homing on the pos. limit switch	x = 2	►	y = 2		Reaching of the pos. limit switch with encoder zero angle or zero pulse homing	x = 2	►	y = 2	
Homing on the positive home switch & index pulse	x = 3	►	y = 3		Reaching of the zero pulse reference switch with encoder zero angle or zero pulse hom-ing	x = 3	►	y = 3	
Homing on the positive home switch & index pulse	x = 4	►	y = 4		Reaching of the pos. zero pulse reference switch with encoder angle or zero pulse homing	x = 4	►	y = 4	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
Homing on the negative home switch & index pulse	x = 6	►	y = 6		Reaching of the neg. zero pulse reference switch with encoder zero angle or zero pulse homing	x = 6	►	y = 6	
Zero reference cam switch, left to pos. edge with Zero pulse; CW move	x = 7		y = 7		Zero point switch, to the left of pos. edge with zero pulse; clockwise rotation	x = 7		y = 7	
Zero reference cam switch, right fo pos. edge with zero pulse; CW move	x = 8		y = 8		Zero point switch, to the right of pos. Edge with zero pulse; clockwise rotation	x = 8		y = 8	
Zero reference cam switch, left to neg. edge with zero pulse; CW move	x = 9		y = 9		Zero point switch, to the left of neg. Edge with zero pulse; clockwise rotation	x = 9		y = 9	
Zero reference cam switch, right to neg. edge with zero pulse; CW move	x = 10		y = 10		Zero point switch, to the right of neg. edge with zero pulse; clockwise rotation	x = 10		y = 10	
Zero reference cam switch, right to neg. edge with zero pulse; CCW move	x = 11		y = 11		Zero point switch, to the right of neg. edge with zero pulse; counter-clockwise rotation	x = 11		y = 11	
Zero reference cam switch, right fo pos. edge with zero pulse; CCW move	x = 12		y = 12		Zero point switch, to the right of pos. edge with zero pulse; counter-clockwise rotation	x = 12		y = 12	
Zero reference cam switch, left to neg. edge with zero pulse; CCW move	x = 13		y = 13		Zero point switch, on the left of neg. angle with zero pulse; counter-clockwise rotation	x = 13		y = 13	
Zero reference cam switch, right to neg. edge with zero pulse; CCW move	x = 14		y = 14		Zero point switch, to the right of neg. edge with zero pulse; counter-clockwise rotation	x = 14		y = 14	
CANopen spec.	x = 15, 16				not used				
Negative limit switch	x = 17	►	y = 17		negative limit switch	x = 17	►	y = 17	
Positive limit switch	x = 18	►	y = 18		positive limit switch	x = 18	►	y = 18	
Positive zero reference switch, CCW move	x = 19	►	y = 19		positive zero point switch counter-clockwise rotation	x = 19	►	y = 19	
Positive zero reference switch, CW move	x = 20	►	y = 20		positive zero point switch clockwise rotation	x = 20	►	y = 20	
Negative zero reference switch, CW move	x = 21	►	y = 21		negative zero point switch clockwise rotation	x = 21	►	y = 21	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
Negative zero reference switch, CCW move	x = 22	►	y = 22		negative zero point switch counter-clockwise rotation	x = 22	►	y = 22	
Zero reference cam switch, left to pos. edge; CW move	x = 23	►	y = 23		Zero point switch, to the left of pos.; clockwise rotation	x = 23	►	y = 23	
Zero reference cam switch, right to pos. edge; CW move	x = 24	►	y = 24		Zero point switch, to the right of pos. edge; clockwise rotation	x = 24	►	y = 24	
Zero reference cam switch, left to neg. edge; CW move	x = 25	►	y = 25		Zero point switch, to the left of neg. edge; clockwise rotation	x = 25	►	y = 25	
Zero reference cam switch, right to neg. edge; CW move	x = 26	►	y = 26		Zero point switch, to the right of neg. edge; clockwise rotation	x = 26	►	y = 26	
Zero reference cam switch, right to neg. edge; CCW move	x = 27	►	y = 27		Zero point switch, to the right of neg. edge; counter-clockwise rotation	x = 27	►	y = 27	
Zero reference cam switch, left to neg. edge; CCW move	x = 28	►	y = 28		Zero point switch, to the left of neg. edge; counter-clockwise rotation	x = 28	►	y = 28	
Zero reference cam switch, right to pos. edge; CCW move	x = 29	►	y = 29		Zero point switch, to the right of pos. edge; counter-clockwise rotation	x = 29	►	y = 29	
Zero reference limit switch, left to pos. edge; CCW move	x = 30	►	y = 30		Zero point switch, to the left of pos. edge; counter-clockwise rotation	x = 30	►	y = 30	
CANopen spec.	31..32				not used	31..32			
Nearest zero pulse; CCW move	x = 33	►	y = 33		Next zero pulse; counter-clockwise rotation	x = 33	►	y = 33	
Nearest zero pulse; CW move	x = 34	►	y = 34		Next zero pulse with clockwise rotation	x = 34	►	y = 34	
Homing on the current position	x = 35	►	y = 35		Setting of home position	x = 35	►	y = 35	
reserved	x = 36 .. 127				not used				

CANopen object	Index Value range	▸	P. No. Scaling		Controller parameters	P. No. Value range	▸	Index) Rescaling	Comment
homing_speeds	6099_{hex}				(Ref. speed)			6099_{hex}	Only at changes in option module G/H configuration 1 bit 2 = 1: Specification of desired moving speed of positioning in UU e. g. 1/100 °/s for rotating movements. s⇒ ms [1/1000]. At exceeding of limit value of parameter in b maXX® the maximum or minimum values are set without generating an error.
speed_during_search_for_switch	Sub. 01_{hex}	▸	**P1201**		PPosHomingSpeed	**P1201**	▸	Sub. 01_{hex}	
	$x = 0 .. 2^{32}$	▸	y = x			x = 1..13200	▸	y = x	
speed_during_search_for_zero	Sub. 02_{hex}	▸	**P1202**		PPosHomingFinalSpeed	**P1202**	▸	Sub. 02_{hex}	
	$x = 0 .. 2^{32}$	▸	y = x			x = 1..50	▸	y = x	
homing_acceleration	$609A_{hex}$	▸	**P1203**		PPosHomingAcceler (homing acceleration)	**P1203**	▸	$609A_{hex}$	Only at changes in option module G/H configuration 1 bit 2 = 1: The user-defined unit (acceleration units) of homing_acceleration in UU e. g. 10 °/s for rotational movements. S^2⇒ms^2 [1/1000]. At exceeding of limit switch of the parameter in the b maXX® the max. or min. values are set without an error message.
	$x = 0 .. 2^{32}$	▸	y = x			x = 25 .. 45000	▸	y = x	
Interpolation_submode_select	$60C0_{hex}$ /ro		Sub. 0 = 1						Object $60C0_{hex}$ contains the information to the IP mode. The linear interpolation is supported only.
	$x = -2^{16} ... 2^{16} - 1$		-32768 ... -1,		It will be written to no parameter in the controller				
	x = 0	▸	linear Interpolation						
	x = 1. . 32767	▸	reserved						
Interpolation_data_record	$60C1_{hex}$ /ro								In Sub. 0 the size of the buffer is readable. Sub. 0 is supported only. The 16 higher bits relate to the number of revolutions and the 16 lower bits relate to the angle in inc. Synchronization occurs via the Sync telegram from the master.
	Sub. 0								
	x = 0 ... 255	▸	1						
	Sub. 1	▸	**P369**		**Position setpoint**	**P369**	▸	**60C1**	
	$x = 0 ... 2^{32} - 1$		y = x			$x = 0 ... 2^{32} - 1$		y = x	
Interpolation_time_period	$60C2_{hex}$		**Sub. 0 = 2**						Thera are only entries allowed which relate to 2000, 4000 and 8000 µs. **If no cycle time is recorded in P532 an error message is sent.**
Ip_time_units	**Sub. 1**	▸	**kein**		**Sync-Slot**	**kein**	▸	**Sub. 1**	**10^ip_time_index * seconds**
	x = 1 ... 255		$x = -2^{16} ... 2^{16} - 1$			$x = -2^{16} ... 2^{16} - 1$		x = 1 ... 255	Example: If 4000 µs shold set, then it must be written value 4 to Sub1 and value -3 to Sub2. Only when SIX2 is written, the cycle time is assumed in the controller.
Ip_time_index	**Sub. 2**	▸	**P532**		**Sync-Slot**	**P532**	▸	**Sub. 2**	
	x = -128 ... 63		$x = -2^{16} ... 2^{16} - 1$			$x = -2^{16} ... 2^{16} - 1$		x = 1 ... 255	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
Interpolation_sync_ definition	60C3$_{hex}$		**Sub. 0 = 2**						**Sub.1:** the synchronization mode is defined. **Confirmed -> value = 0:** Synchronization refers to the SYNC telegram. **Sub.2** is only readable and displays the number from which SYNC on the Phi values are incremented and are assumed in the buffer. At every received SYNC telegram **only** the assumption is supported, i.e. standard = „0“. A dummy parameter is used (application parameter 3331).
syncronize on group	**Sub. 1**	►	**P3331 (Application parameter)**		General Sync is used (Standard)	**P3331 (Application parameter)**	►	60C3$_{hex}$ Sub 1	
Ip_sync every n event	**Sub. 2** ro x = 0 ... 255	►	**P3331 (Application parameter)**						
Interpolation_data_configuration	60C4$_{hex}$	►	Sub. 0 = 6						
max_buffer_size	**Sub. 1 ro**	►	1						
actual_size	**Sub. 2**	►	1		Only one buffer is supported				
buffer_organisation	**Sub. 3**	►	0..1		0 = FIFO buffer; 1= Ring buffer				
buffer_position	**Sub. 4 ro**	►	1		Always = 1, because only one buffer available.				
size_ofdata_record	**Sub. 5 ro**	►	1		Always = 1, because only one buffer available.				
buffer_clear	**Sub. 6**	►	0..1		(no effect)				
position_control_ parameter_set	60FB$_{hex}$								Manufacturer-specific CANopen object
	Sub. 01$_{hex}$ /ro				PosCtrlStatus	**P0360**	►	Sub. 01$_{hex}$	Default = 0
						x = 0 .. 2^{16}-1	►	y = x	
	Sub. 02$_{hex}$	►	**P1050**		PosCtrlMode	**P1050**	►	Sub. 02$_{hex}$	Default = 0
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	y = x	
	Sub. 03$_{hex}$	►	**P1051**		PosCtrl_Kv-Factor	**P1051**	►	Sub. 03$_{hex}$	Default = 10.0
	x = 0 .. 32767	►	y = 0 .. 32767			x = 0 .. 32767	►	y = x	
	Sub. 04$_{hex}$	►	**P0364**		PosSetRev	**P0364**	►	Sub. 04$_{hex}$	Default = 0
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
	Sub. 05_{hex}	►	**P0363**		PosSetAngle	**P0363**	►	Sub. 05_{hex}	Default = 0
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	y = x	
	Sub. 06_{hex}	►	**P1053**		SpeedFeedForFactor	**P1053**	►	Sub. 06_{hex}	Default = 4000_{hex}
	x = 0 .. 2^{16}-1	►	y = 0 .. 5000_{hex}			x = 0 .. 5000_{hex}	►	y = 0 .. 2^{16}-1	
	Sub. 07_{hex} /ro				PosCtrlDev	**P0367**	►	Sub. 07_{hex}	Default = 0
						x = -2^{31} .. 2^{31}-1	►	y = x	
	Sub. 08_{hex}				PosActValue	**P0362**	►	Sub. 08_{hex}	Default = 0
						x = 0 .. 2^{32}-1	►	y = x	
	Sub. 09_{hex} /ro				Enc1ActRev	**P0392**	►	Sub. 09_{hex}	Default = 0
						x = 0 .. 2^{32}-1	►	y = x	
	Sub. $0A_{hex}$ /ro				Enc1ActAngle	**P0391**	►	Sub. $0A_{hex}$	
						x = 0 .. 2^{32}	►	y = x	
	Sub. $0B_{hex}$ /ro				SpeedFeedFor	**P0365**	►	Sub. $0B_{hex}$	
						x = -2^{31} .. 2^{31}-1	►	y = x	
	Sub. $0C_{hex}$ /ro				PPosStatus	**P0460**	►	Sub. $0C_{hex}$	
						x = 0 .. $FFFF_{hex}$	►	y = x	
	Sub. $0D_{hex}$	►	**P1191**		PPosActRecordNumber	**P1191**	►	Sub. $0D_{hex}$	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	y = x	
	Sub. $0E_{hex}$	►	**P1190**		PPosMode	**P1190**	►	Sub. $0E_{hex}$	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. $FFFF_{hex}$	►	y = x	
	Sub. $0F_{hex}$	►	**P1200**		PPosHomePosition	**P1200**	►	Sub. $0F_{hex}$	
	x = 0 .. 2^{32}-1	►	y = x			x = 0 .. $FFFFFFFF_{hex}$	►	y = x	
	Sub. 10_{hex}	►	**P1208**		PPosSwitchMode	**P1208**	►	Sub. 10_{hex}	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. $FFFF_{hex}$	►	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
	Sub. 11_{hex} /ro				PPosSpeedSetValue	**P0464**	►	Sub. 11_{hex}	
						x = -3276831 .. 32767	►	y = x	
	Sub. 12_{hex}	►	**P0605**		PPosBend0	**P0605**	►	Sub. 12_{hex}	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 8191	►	y = x	
	Sub. 13_{hex}	►	**P1198**		PPosClipEnvironment1	**P1198**	►	Sub. 13_{hex}	
	x = 0 .. 2^{32}-1	►	y = x			x = 0.. $FFFFFFFF_{hex}$	►	y = x	
	Sub. 14_{hex}	►	**P1199**		PPosClipEnvironment2	**P1199**	►	Sub. 14_{hex}	
	x = 0 .. 2^{32}-1	►	y = x			x = 0 .. $FFFFFFFF_{hex}$	►	y = x	
	Sub. 15_{hex}	►	**P0601**		PPosTargetInput0	**P0601**	►	Sub. 15_{hex}	
	x = -2^{15} .. 2^{15}-1	►	y = x			x = -2^{15} .. 2^{15}-1	►	y = x	
	Sub. 16_{hex}	►	**P0608**		PposTargetInput1	**P0608**	►	Sub. 16_{hex}	
	x = -2^{15} .. 2^{15}-1	►	y = x			-2^{15} .. 2^{15}-1	►	y = x	
	Sub. 17_{hex}	►	**P0370**		PosIpSetAngle	**P0370**	►	Sub. 17_{hex}	
	x = 0 .. 2^{32}	►	y = x			x = 0 .. 2^{32}	►	y = x	
	Sub. 18_{hex}	►	**P1209**		PPosEncoderOffset	**P1209**	►	Sub. 18_{hex}	
	x = 0 .. 2^{16}	►	y = x			x = 0 .. 2^{16}	►	y = x	
	Sub. 19_{hex}	►	**P1209**		PPosHomingDeceler	**P1204**	►	Sub. 19_{hex}	
	x = 0 .. 2^{16}	►	y = x			x = 0 .. 2^{16}	►	y = x	
	Sub. $1A_{hex}$ ro				SpeedActValue	**P0353**	►	Sub. $1A_{hex}$	The actual speed value **P0353** is rescaled by a 32 bit value to 16384. 100% of the maximum speed (in **P1031**) accords to 16384 units. The amount is issued.
						x = -2^{32} .. 2^{32}-1	►	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
	Sub. $1B_{hex}$ ro				AmpWarning/MotorWarning	**P0262**, **P0263**	►	Sub. $1B_{hex}$	Bit 0 **P0263** bit 1 Motor temperature threshold 2 exceeded Bit 1 **P0263** bit 1 Motor temperature has exceeded threshold 2 Bit 2 **P0262** bit 1 Power unit temperature > 80°C Bit 3 not assigned Bit 4 **P0263** bit 0 Motor temperature has exceeded threshold 1 Bit 5 **P0263** bit 0 Motor temperature has Threshold 1 exceeded
						x=0.. 2^{16}	►		
digital_inputs	$60FD_{hex}$ /ro	►	**P461**		DigInOutStatus	**P461**	►	$60FD_{hex}$	
	x=0.. 2^{16}					x=0.. $FFFF_{hex}$	►	y = 0 .. 2^{32}	
Negative limit switch					Status neg. limit switch	Bit 0	►	Bit 0	
Positive limit switch					Status pos. limit switch	Bit 1	►	Bit 1	
Home switch					Status zero point switch	Bit 2	►	Bit 2	
Interlock					reserved	Bit 4			
reserved					Reserved	Bit 3..15			
Man. specific					not used			Bit 16..31	
target_velocity	$60FF_{hex}$	►	**P1171**		RFG1Input	**P1171**	►	$60FF_{hex}$	The user-defined unit (velocity units) is interpreted in the b maXX® controller as RPM. Only at changes in option module G/H configuration 1 bit 2 = 1: Specification of current speed in 1/10 RPM. e. g.: 200.0 revolutions => input 2000.
	x = -2^{31} .. 2^{31}-1	►	y = x * 40000000_{hex}/ MotorMax-Speed			x = 8000_{hex} .. $7FFF_{hex}$	►	y = x * 40000000_{hex}/ MotorMax-Speed	
drive_data	6510_{hex}							6510_{hex}	
Manufacturer specific	Sub. 01_{hex} / ro					**P0001**	►	Sub. 01_{hex}	
					Controller type		►	y = x	
Manufacturer specific	Sub. 02_{hex} / ro					**P0002**	►	Sub. 02_{hex}	
					Software type		►	y = x	
Manufacturer specific	Sub. 03_{hex} / ro					**P0003**	►	Sub. 03_{hex}	
					SoftwareID		►	y = x	
Manufacturer specific	Sub. 04_{hex} / ro					**P0004**	►	Sub. 04_{hex}	
					Software version		►	y = x	

CANopen object	Index Value range	►	P. No. Scaling		Controller parameters	P. No. Value range	►	Index) Rescaling	Comment
Manufacturer specific	Sub. 05_{hex} / ro					**P0005**	►	Sub. 05_{hex}	
					ParamTableVersion		►	y = x	
Manufacturer specific	Sub. 06_{hex} / ro					**P0009**	►	Sub. 06_{hex}	
					AmpSW_Version		►	y = x	
Manufacturer specific	Sub. 07_{hex} / ro					**P0555**	►	Sub. 07_{hex}	
					FbgaVersion		►	y = x	
Manufacturer specific	Sub. 08_{hex} / ro					**P0556**		Sub. 08	
					Bootloader version			y = x	

APPENDIX D - TECHNICAL DATA

In this appendix you will find a survey of the technical data of the CANopen option card.

D.1 CANopen option card: technical features

CPU	SAB 80C167CR
FPGA	XCS2S15-5TQ144C of the SpartanII series (XILINX)
CAN controller	Integrated in the processor
Memory	4 kByte DP-RAM, 256 kByte RAM, 1 1 MByte Flash-Eprom
Baudrate	max. 1MBit/s, 500kBits, 250kBits, 125kBits, 20kBits
Physical layer	ISO 11898
Operating voltage	+5 V internal
Electrical isolation	Optocoupler, DC/DC converter
Plug-in connector	2 RJ45 sockets, 8-pin [1)]

[1)] The RJ45 terminals are internally connected 1:1 and isolated from the controller

D.2 CANopen option card: Data channels to the b maXX® controller

Three channels are available for data transfer from b maXX® controller to the option module CANopen slave:

- Two process data channels (4 PDOs per communication direction)
- One service data channel (server SDO)

With PDOs, objects can be transferred in cyclic data exchange. Not all objects are available for PDO transfer.

With the SDO transfer all b maXX®4400 parameters can be accessed via the object index.

D.3 CANopen option card: CAN buffer

The Baumüller CANopen interface corresponds to the full CAN concept, i. e. there is a CAN buffer for every type of message.

CAN buffer description	Transfer direction
NMT message	Receive
TX PDO1	Send
TX PDO2	Send
TX PDO3	Send
TX PDO4	Send
RX-PDO1	Receive
RX-PDO2	Receive
RX-PDO3	Receive
RX-PDO4	Receive
SDO_TX	Send
SDO_RX	Receive
Synchronization object	Receive
Emergency object	Send
Node guard object	Receive / transmit

List of Illustrations

Subject Index

Revision survey

Version	Version	Changes
5.02065.03	Feb. 15, 2006	Configuration CANopen option card, error telegrams, conversion tables
5.02065.04	Feb. 06, 2012	Extension for position control and synchronous operation Revision of error messages (Chap. 5.6.2) Revision of 6000' object numbers (App. B.2) Revision of Conversion tables (App. C)

Notes:

be in motion

Baumüller Nürnberg GmbH Ostendstraße 80-90 90482 Nürnberg T: +49(0)911-5432-0 F: +49(0)911-5432-130 **www.baumueller.de**