Programming manual

Language	**English**
	Translation
Document No.	5.07017.03
Part No.	432414
Status	14.08.2014

be in motion be in motion

BAUMÜLLER

CoE

CANopen over EtherCAT
b maXX® controller

E	5.07017.03

Read the Operating Manual before starting any work!

Preliminary information

Warning Insofar as this document is identified as being preliminary information, the following applies:
this version is regarded as providing advance technical information to users of the described devices and their functions at an early enough time in order to adapt to any possible changes or expanded functionality.
This information must be regarded as being preliminary, as it has not yet passed through Baumüller's internal review process. In particular, this information is still subject to changes, thus no legal liability can be derived from this preliminary information. Baumüller assumes no liability for damages that might arise from this possibly faulty or incomplete version.
If you detect or suspect any content errors and/or major form errors in this preliminary information, we request that you notify the Baumüller support specialist responsible for you. Please provide us, via this employee, with your insights and comments so that we can take them into account and include them when transitioning from the preliminary information to the final information (as reviewed by Baumüller).
The conditions stipulated in the following section under "Obligatory" are invalid in case of preliminary information.

Obligatory

These Programming manual are a part of the equipment/machine. These Programming manual must be available to the operator at all times and must be in legible condition. If the equipment/machine is sold or moved another location, these Programming manual must be passed on by the owner together with the equipment/machine.
After any sale of the equipment/machine, this original and all copies must be handed over to the buyer. After disposal or any other end use, this original and all copies must be destroyed.

When the present Programming manual are handed over, corresponding sets of operating instructions of a previous version are automatically invalidated.
Please note that the specifications/data/information **are current values according to the printing date**. These statements are **not legally binding** with regard to measurements, computation or calculations.
Baumüller Nürnberg GmbH reserves the right, in developing its products further, to change the technical specifications and handling of it products concerned without prior notice.

No liability can be accepted concerning the correctness of these Programming manual unless otherwise specified in the General Conditions of Sale and Delivery.

© **Baumüller Nürnberg GmbH**

Ostendstr. 80 - 90
90482 Nuremberg
Germany

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

Email : mail@baumueller.de
Internet: www.baumueller.de

Table of contents

1

INTRODUCTION

The program manual is an important part of the b maXX® 4400 device. Therefore this manual must be read completely, before starting any operation, last but not least on behalf of the own security. This manual describes how the company Baumüller Nürnberg GmbH implemented the option module **CANopen via EtherCAT** (**CoE** - interface on the CoE slave for the series b maXX®4400.

The introduction contains general information regarding the option module CoE slave.

1.1 General information

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

Information according option and function modules for the device series b maXX®4400 is found in the manual 5.01040.

Information according the programming of the b maXX®4400 controller is found in the parameter manual 5.03039.

1.2 Mounting and installation

The mounting of the option module CoE-slave is described in the manual 5.02014.

1.3 EDS file

The EDS file is an ASCII file and is for the description of the function range of a CANopen device. It is an electronic data sheet of the CoE device. The EDS file is used by CANopen masters or bus configurators. The EDS file contains information about all supported objects and further features.

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

1.4 XML-file

There is information in the XML file, which needs a master in order to e. g. configure the FMMU and the
SyncManager on the CoE slave.

2

FUNDAMENTAL SAFETY INSTRUCTIONS

In this chapter the dangers are prescribed, which can arise during parameterization of the Baumüller b maXX® 4400 controller unit and the meaning of the information sign is explained.

2.1 Safety notes and mandatories

WARNING!

Danger from modification of the parameter settings!

The change of parameters affects the behavior of the Baumüller-unit and consequently the behavior of the construction and its components. If you change the adjustments of the parameters, you may cause a dangerous behavior of the construction and/or of its components.

Therefore:

- After each modification of the parameter settings, a commissioning with consideration to all safety instructions and safety regulations must be executed.

2.2 Information sign

NOTE!

This note is a very important information.

2.2 Information sign

3

BASICS ETHERCAT

3.1 Literature concerning EtherCAT and CANopen

On behalf of basic information with reference to EtherCAT and CANopen the following literature is recommended:

- [1]
  „EtherCAT Technology Group“
  http://www.ethercat.org/

- [2]
  EtherCAT communication specification,
  Version 1.0 provided by
  the EtherCAT Technology Group

- [3]
  EtherCAT slave controller ESC10/20 hardware data sheet,
  Version 1.1 provided by
  the EtherCAT Technology Group

- [4]
  Online Encyclopedia Wikipedia http://www.wikipedia.org/

- [5]
  Zeltwanger
  CANopen
  Holger Zeltwanger VDE-Verlag

- [6]
  www.can-cia.de
  CAN in Automation e.V.
  Kontumazgarten 3
  D-90429 Nuremberg

3.2 Basic principles EtherCAT

The Real Time Ethernet Control Automation Technology (EtherCAT) was developed by the company Beckhoff as a new field bus standard. The Ethernet Technology Group ETG was founded in order to distribute EtherCAT as an open standard. The ETG is an association of interested parties, manufacturers and users. This association had 421 members from 31 countries in December 2006. These members join forces to support and promote the further technology development. .

3.2.1 Topology data

Several bus topologies can be used, e. g. line-, tree- or star-topologies (▷Figure 1◁ on page 10).

Up to 65535 users can be reached, thus the network size is nearly unlimited (>500 km).

For the transmission a standard Ethernet patch cable (CAT5) is sufficient. The full duplex features of 100 BASE-TX are used to full capacity, so that effective data rates of >100 MBit/s
(>90 % of 2 x 100 MBit/s) can be reached. The cable length between two users is indicated with up to 100 m.

Fiber-optic cables variants from 50 m to 2000 m can also be used.

It is also advantageous, that during the operation devices can be connected or disconnected „hot connect / disconnect of bus segments".

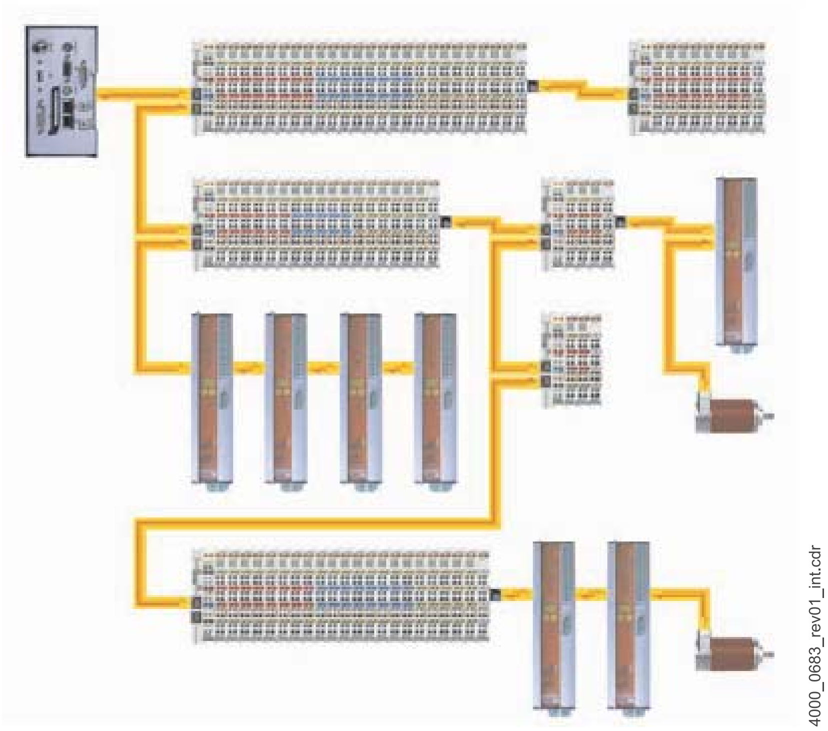

Figure 1: Flexible topology: line, tree or star [1]

3.2.2 Frame structure

The EtherCAT protocol was particularly optimized for the process data. This is possible because of a special Ether type (88A4h), which is directly transported in an Ethernet frame. It can consist of several sub-telegrams, which accordingly access a memory range of the great logic process image, which can be up to 4 Gigabyte. There is a random access to the data addressing, thereby the sequence of the physical sequence is independent of the data-technical sequence of the users in the network.

Sending is executed with a minimum displacement of few bit times.

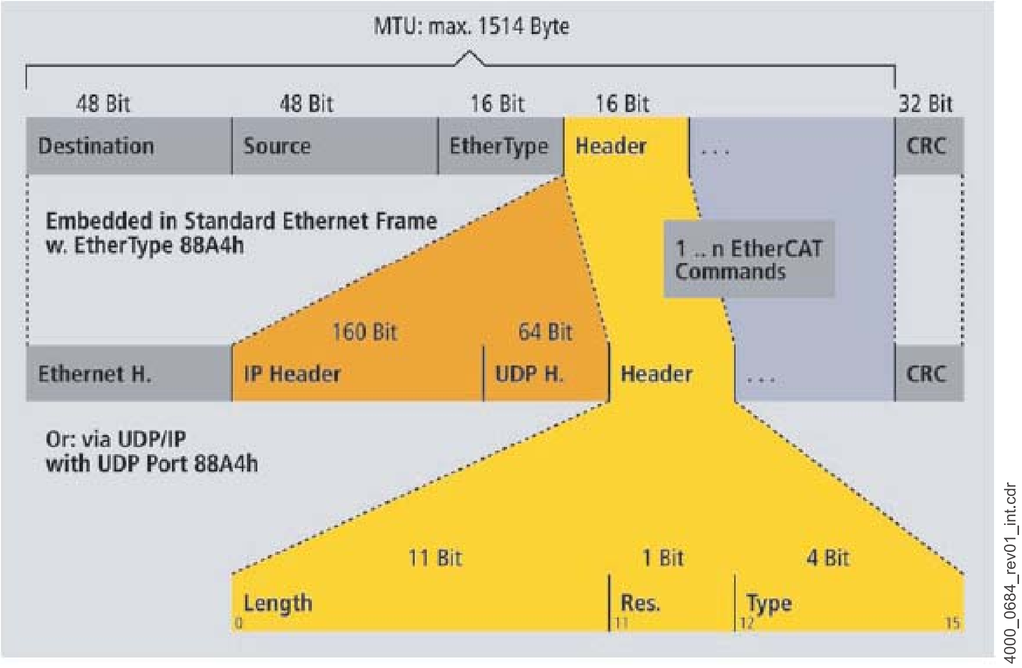

Figure 2: EtherCAT: standard -IEEE 802.3-frames [1]

3.2.3 Device profiles

For the different tasks in the automation there are special field bus systems e. g. CANopen. The field bus systems are often classified in standards. At the EtherCAT there are no own profiles for already existing standards developed, rather the already existing are improved.

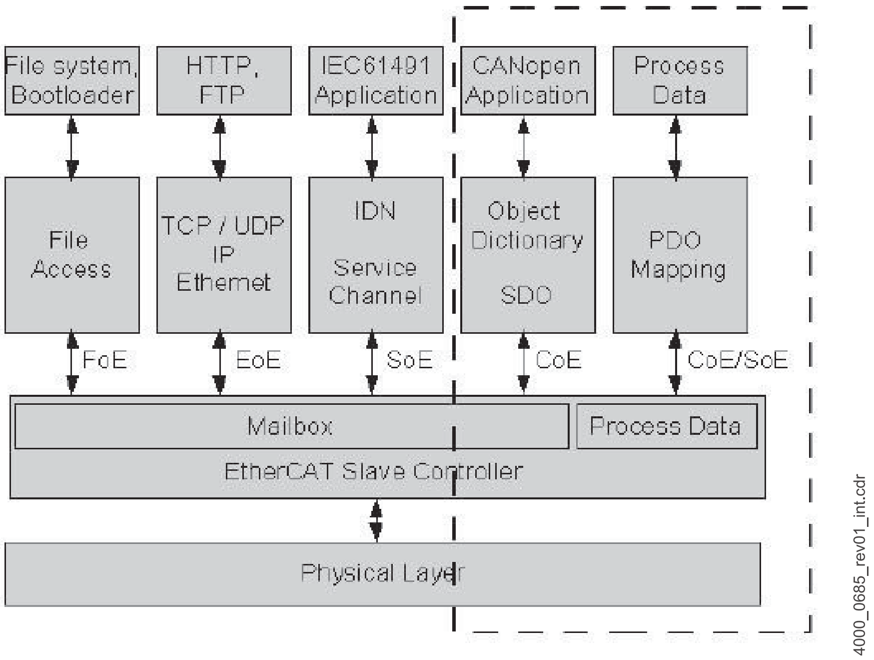

Figure 3: Device profile at EtherCAT[1]

3.2.4 EtherCAT frame structure

The EtherCAT telegrams, embedded into an Ethernet telegram, are send. The telegram contains an Ethernet header (a), an EtherCAT header (b) and in the following then n EtherCAT telegrams.

The EtherCAT telegram (c) is divided up in an EtherCAT header, data range and a counter range.

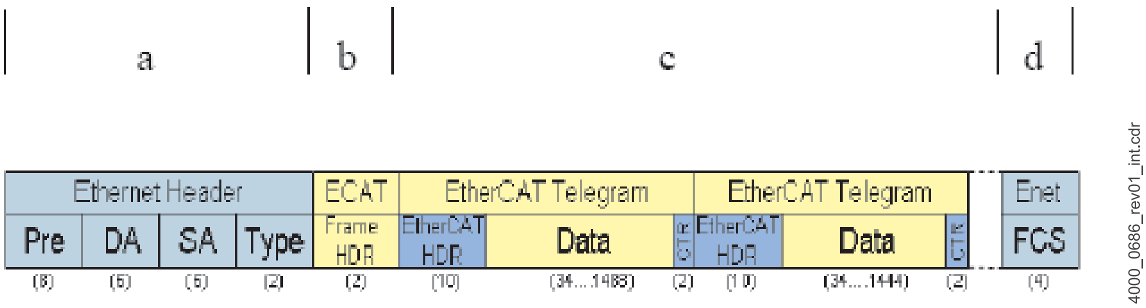

Figure 4: EtherCAT - frame [1]

a) Ethernet header:

Pre The preamble is used for the synchronization and the localization by the receiver, it consists of a sequence of '10101010' per byte.
The preamble contains the SFD byte: SFD: „Start of frame delimiter“ signifies the frame beginning; bit pattern 10101011.

DA Destination MAC address.
SA Source MAC address.
Target-/source address: specify the receiving (possibly several) and the Ethernet telegram, which needs to be send; within one LAN only one length permitted (16 or 48 bit)

Type Defines the EtherType. The EtherType shows, which protocol of the next higher layer* within the user data is used. $88A4_{hex}$ defines the EtherCAT type.

* ISO-OSI-layer model

b) EtherCAT frame header :

The EtherCAT frame header has a length of 2 byte. Here the information about the data length and the data type of the following telegrams is contained.

c) EtherCAT telegram :

The EtherCAT telegram is divided into the telegram header, into the data to be transmitted and the working counter. The working counter is incremented by each operating slave.

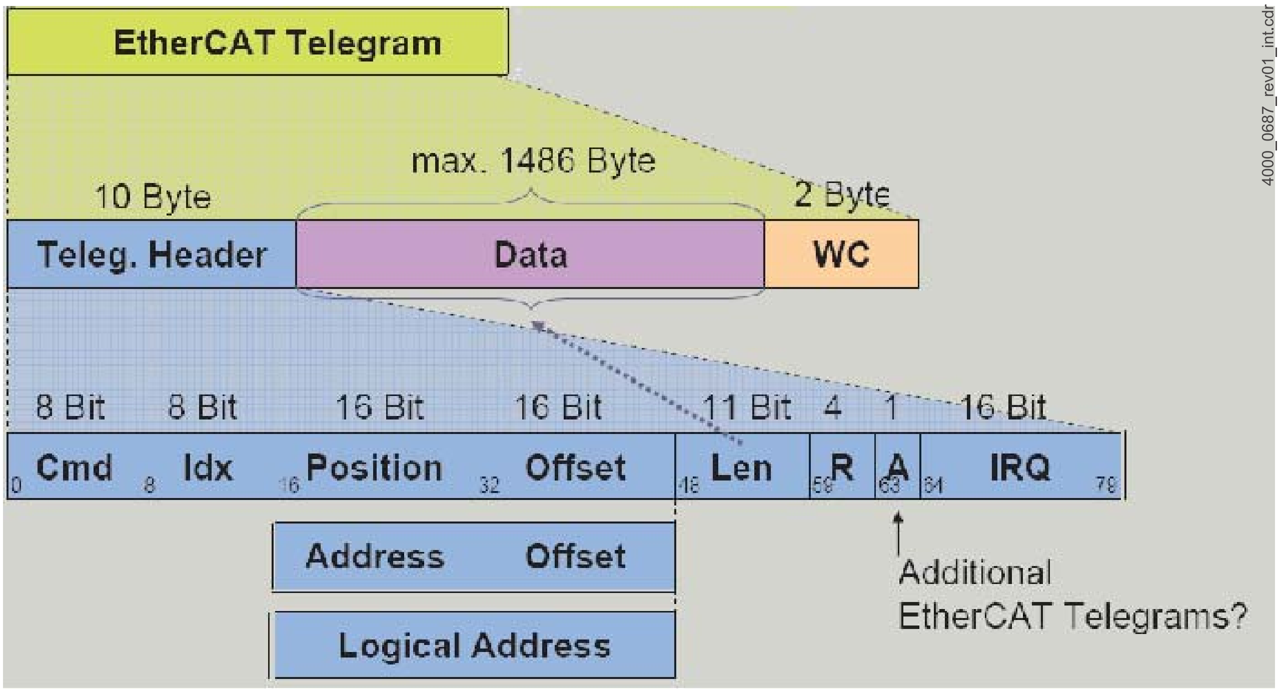

Figure 5: EtherCAT telegram [1]

Der „EtherCAT telegram header“ has a length of 10 byte. It contains information on the following data.

- **CMD**, 1 byte. Codes the EtherCAT command, which was transmitted by the master, which can be marked either written or read.
- **IDX**, 1 byte. Index of the frames. Is transmitted unchanged from the slave, with this the master can assign the telegram at reception more simple again.
- The **position**, shows the address or the physical position of the slave. Additionally an offset is indicated.
  Divided in: ADP (2 bytes) Address page dependent on the used command
  ADO (2 bytes) Address offset dependent on the used command
  INT Interrupt field
- **LEN**, 2 Bytes.
  In the bits 0 to 10 the length of the following data block is saved.
  The bits 11 to 15 are used as flags for different purposes.
  Bit 63(A) displays if an extra EtherCAT telegram is send subsequently.

The data range at maximum is 1486 bytes. Within the data range of an Ethernet frame there can be several EtherCAT frames and therewith several commands at different slaves contained. The physical sequence of the slave in the line generally must not be regarded. Due to the feature that several EtherCAT commands fit in an Ethernet frame and due to a memory mapping in the slaves, which allows the access to the memory range of several slaves with one EtherCAT command, the user data’s rate is considerably increased. Therewith the problem of the high overhead of Ethernet at low but repeating data volume is solved.

The EtherCAT telegram ends with a 2 byte great working counter. Each slave, which successfully received a telegram increments the counter. Therewith the master can recognize errors.

d) Frame check sequence (FCS):

The FCS field displays a 32 bit CRC checksum. If the checksum of the FCS is unequal zero, the transmission was incorrect.

3.2.5 EtherCAT communication statuses

The AL management in EtherCAT describes the handling of the EtherCAT state machine (ESM). The state and the state change of the according slave is described in one application. The actual state of the ECT slave is displayed in the state register and state changes are displayed in the control register, which is initiated by the master.

EtherCAT defines four communication states. The communication states (state) and its transitions (transitions) see ▷Figure 6◁.

State changes are inquired for by the master. The slave answers correctly if the change is completed or there is an error message if the change could not be done.

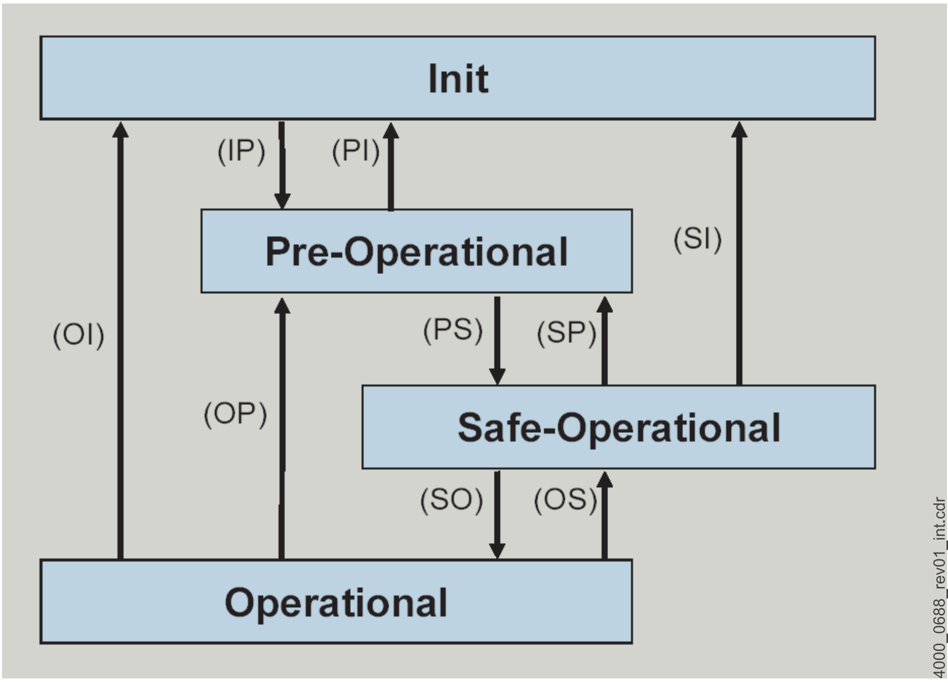

Figure 6: EtherCAT communication transitions [1]

States:

- **Init**:
  Initialization of the slaves. In the Init phase no direct communication is possible on the application level.
- **Pre-operational**:
  In this state a mailbox for a service data communication can be configured (if the slave supports it). Service data communication then is possible but not process data communication.
- **Safe-operational:**
  In this state the service data communication is possible further on. Only outgoing data from the slave, TX data, are send. RX data from the master are ignored. Mailbox is possible further on.
- **Operational**:
  Mailbox and cyclic communication in both directions (TxPDO and RxPDO) are now possible. Mailbox is possible further on.

The transitions are shown in the following table.

State transition	Local management service
IP	Start mailbox communication
PI	Stop mailbox communication
PS	Start input update
SP	Stop input update
SO	Start output update
OS	Stop output update
OP	Stop output update, stop input update
Si	Stop input update, stop mailbox communication
OI	Stop output update, stop input update, stop mailbox communication

Transitions:

If the demand of the master for a state change cannot be made by the slave, because e. g. of an incorrect mapping, the slave has the possibility to send an error message to the master. This message is similar to the subdivision of the device control.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
$A0_{hex}$	00_{hex}	08_{hex}	04_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

Byte 0 and **Byte 1** contain the emergency error code.

Two inputs of the CoE standard are defined.

$A000_{hex}$: Transition from PRE-OPERATIONAL to SAFE-OPERATIONAL was not successful

$A001_{hex}$: Transition from SAFE-OPERATIONAL to OPERATIONAL was not successful

Byte 2:

In the following table the messages are shown, which are displayed if an incorrect parameterization of the SyncManager was made.

SyncManager2 (process data out RxPDO)	08_{hex}	SyncManager incorrectly parameterized
	09_{hex}	PDO length is not in accordance with the mapping length of the objects
	$0A_{hex}$	SyncManager settings at an invalid address
SyncManager3 (Process data in TxPDO)	$0C_{hex}$	SyncManager incorrectly parameterized
	$0D_{hex}$	PDO length is not in accordance with the mapping length of the objects
	$0E_{hex}$	SyncManager settings at an invalid address

For the SyncManager0 and the SyncManager1 no message can be transmitted because therewith the mailboxes are written to. If the mailboxes are configured incorrect, the slave remains in status INIT. In this case the change to PRE-OPERATIONAL, which did not take place is transmitted only via the AL-status to the master.

When there are incorrect Syncmanager setting first the EMCY for the SyncManager2 is transmitted and it does not matter if SyncManager3 also was incorrectly configured. When the first error then was removed, the next emergency is send.

Byte 3:

Defines the number of the following bytes, either 4 byte (or 2 byte EMCY codes at error of device).

Byte 4-7:

	Byte 4	Byte 5	Byte 6	Byte 7
SM2 address error	0	0	0	0
SM2 incorrect length	High byte	Low byte	High byte	Low byte
	Minimum length of the Syncmanager		Maximum length of the Syncmanager	
SM2 incorrect parameterized	High byte	Low byte	High byte	Low byte
	Smallest permissible address		Greatest permissible address	
SM3 address error	0	0	0	0
SM3 incorrect length	High byte	Low byte	High byte	Low byte
	Minimum length of the Syncmanager		Maximum length of the Syncmanager	
SM3 incorrect parameterized	High byte	Low byte	High byte	Low byte
	Smallest permissible address		Greatest permissible address	

Manufacturer-specific error code:

Byte 0 and byte 1:

$A0A0_{hex}$:
the error code appears, if the drive shall operate synchronous, but after a defined time does still not run synchronously (dependent of b maXX®-device and from the and of device state, from 100 to 30 s).

Byte 2 contains FF_{hex} and byte 4-7 got the value zero.

Synchronization

The exact synchronization of users at the EtherCAT is made according on the principle of distributed clocks, as described in the latest standard IEEE 1588. Each slave has an independent operating clock implemented. Therewith the time of the master clock is transmitted via EtherCAT to the slave. In order to take into account the synchronization telegram an operating time measurement is made. For this the master sends a broadcast telegram, in which all slaves record the receiving point of time of this broadcast telegram according to their clock. Therewith the operating times are defined and can be accordingly regarded considered by the master. At EtherCAT the master clocks configured into a slave device, so that also for this no special hardware in the master is necessary. The accuracy of the synchronization therewith definite is under one µs, at 300 users and 120 m cable length deviations of +/- 20 ns were achieved [1].

The necessary settings of the slaves through the master or the setting in the data set are described in ▷Synchronization (SYNC)◁ on page 47.

3.2.6 Ethernet over EtherCAT (EoE) - TCP/IP- tunneling over EtherCAT

For the Ethernet communication to the EtherCAT slaves (e.g. to the b maXX ®-controller with EtherCAT slave, here particularly for the service console ProDrive) the TCP-packages are transmitted within the EtherCAT packages (tunneling). In this case for each EtherCAT slave an own IP address must be set. The EtherCAT slave is activated as Ethernet user via this IP address.

The possibilities for IP address adjustments are:

1 Setting via DIP switches

The value of the DIP switches is added to the base address 192.168.1.1.

192.168.1.XXX

192.168.1 is fixed.

XXX means setting of the DIP switches (SW13100 or S12100/S12200 to the hardware) + 1.

e. g. if address 192.168.1.5 is wanted, the DIP switch 4 must be set.

2 Setting via ProDrive

An user-defined IP address can be configurated at the ProDrive page *Option module G/H configuration*. For this bit 13 in configuration parameter 1 (P0830/P0840) must be set. The IP address can be set in configuration parameters 4 and 5 (P0833/P0843 and P0834/P0844). Configuration parameter 5 contains the first two numerals of the IP address and configuration parameter 4 contains the last two numerals. The configuration should be saved in the data set of the controller.

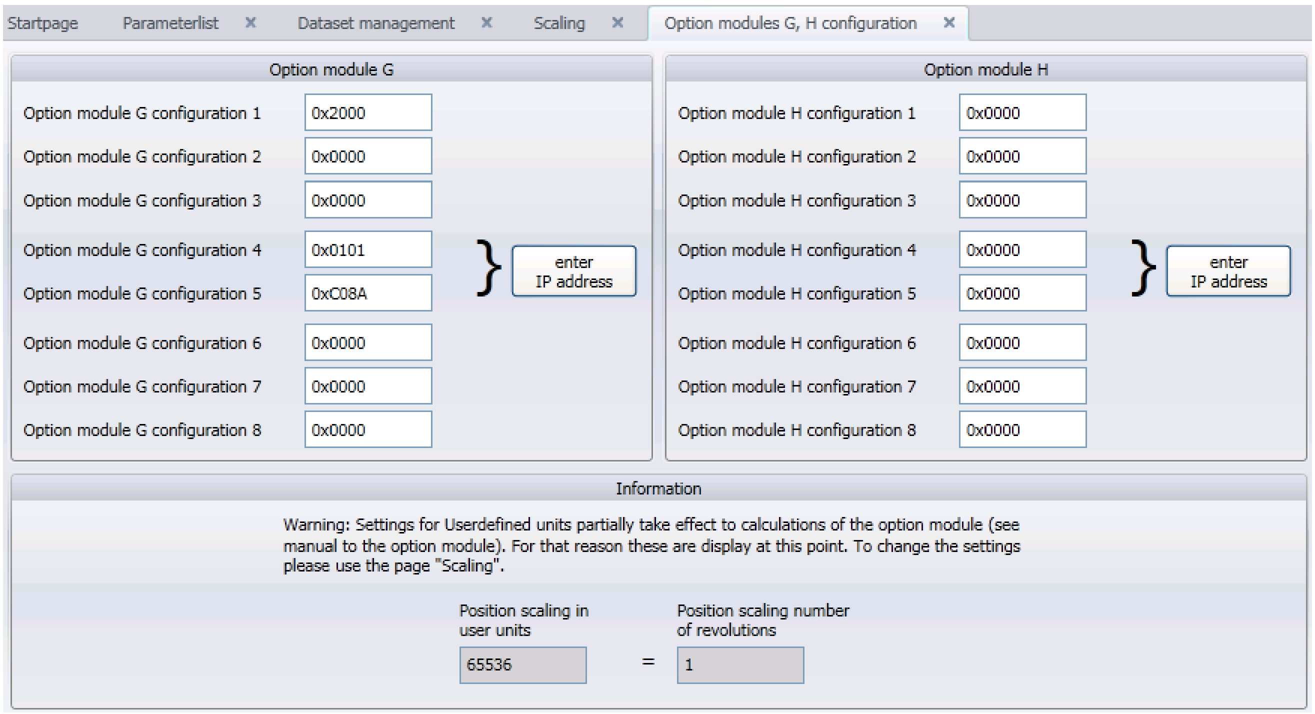

Figure 7: ProDrive page: Option module G/H configuration

3 Setting via EtherCAT master

It is also possible to define the IP address via the EtherCAT master (if this is supported by the master).

Thereby the IP address can be selected user-defined.

The port number for the EoE communication is 5043_{hex} (= 20547_{dez}).

As the EoE communication is made via the mailboxes of the EtherCAT, the mailbox shall be requested with less than 10 ms.

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 kind of the data exchange and general determinations, which are valid for all devices.

Device profile In the device profiles the user- and device-specific determinations, the meaning of data content and the device functionality are described. There are device profiles for drives, I/O-modules, encoder or programmable devices a.s.o. The option module CANopen slave for the b maXX® 4400 controller is implemented according to the device profile DSP402 (drives and motion control).

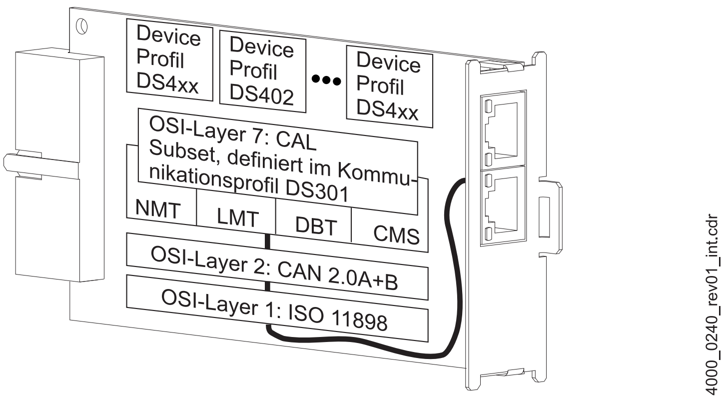

Figure 8: CANopen profile structure

3.3.1 Object directory

The central element of every CANopen device is the object directory of the 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

The function of SDOs is to transmit greater volumes of data of 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 byte maximum are transmitted with one SDO (SDO expedited transfer). Typically, SDOs are used for device configuration. SDOs are transmitted asynchronously and confirmed 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 transfer depends on the synchronization object. The content of the 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.

3.3.2 State machine CANopen

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.

In comparison to the CoE standard (see ▹EtherCAT communication statuses◃ on page 15) one state less is defined in the CANopen (safe-operational). Additionally at the CoE the state transitions are confirmed after the change was made from the slave or with one error message if there was no change.

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 available 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 available)	optional objects are partly existent

The following operating modes are not supported, i. e. at least one mandatory object is not existent, also optional objects can be existent.	
Interpolated position mode	no objects
Profile torque mode	an object

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	status word
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

Objects of position mode profiles **all mandatory objects, partially optional objects are supported** **(positioning)**			
$607A_{hex}$	mandatory		target_position
$607D_{hex}$	optional	SIX0 = 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
$60F8_{hex}$	optional	max_slippage

Objects of the 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
$604F_{hex}$	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 supported** **(general inputs in object directory)**			
$60FD_{hex}$	optional		digits_inputs
6510_{hex}	optional	SIX 0 = 08	drive_date

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

Torque mode profile objects **one optional object is supported** **(torque)**			
6072_{hex}	optional	SIX 0 = 2	max_torque

4

COMMUNICATION TO THE b maXX® CONTROLLER

In this chapter the data communication between the b maXX® 4400 device and the CoE slave option module is described.

4.1 Communication flow

The CoE option module exchanges via a FPGA data with the b maXX® 4400 controller. This data exchange is made with a defined time pattern via the BACI interface (Baumüller bus).

Therewith the CoE option module activates the 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 of a cycle, service data is transferred. The transmission of the process data is made in a settable time pattern.

NOTE!

The cyclic communication: (RxPDO) is active only in the CoE communication state OPERATIONAL. TxPDO can also be send in SAFE-OPERATIONAL.

4.2 Parameterizing the BACI communication times

Between the option module CoE slave and the b maXX® controller 8 set values and 8 actual values can be exchanged as process data in a communication cycle. Which set values and actual values are exchanged is specified in the mapping objects on the option module CoE-slave (setting via SDO by the master or the default setting, see ▷Process data◁ from page 44). The parameterization of the communication is specified in this chapter.

The setting of communication times between option module CoE-slave and b maXX® controller is automatically set by the CoE-slave and can not be changed by the user. Thereby the BACI-times are adapted to the cycle time. The setting of the cycle time is explained in ▷Data exchange and parameterization◁ from page 35.

On the WinBASS II / ProDrive side „BACI" (option module 1) it is possible to read the cycle communication time (rates set values, actual values), the cycle-offset of the set values and the cycle offset of the actual values.

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

The communication cycle time is a multiple of the call of the communication time slice of the controller (every 125 µs). In the EditBox „Rate set values, actual values" only the factor is determined, e. g. the value in the EditBox „Rate set values, actual values" is calculated as follows:

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

Example:

Communication cycle time = 500 µs $\Rightarrow$ set values, actual values = 4

NOTE!

When establishing BM_u_Baci1MPeriod note the following:
BACI can only be accessed every 250 µs.

Also on the WinBASS II / ProDrive page „BACI" (option module 1) parameter numbers of the set values and of the actual values are found. These are only for the display, because the setting of the parameter numbers for the process data exchange are specified in the mapping objects on the option module CoE slave. The mapping is entered at the transition from PRE-OPERATIONAL to SAFE-OPERATIONAL.

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.

4.3 Configuration possibilities of the CoE option card in WinBASS II / ProDrive.

WinBASS II / ProDrive „option module G/H - configuration 1"

NOTE!

Settings result in a modified behavior!

4.3.1 Settings from firmware version FW 03.00 b maXX®-controller

(Changing of several standardization functions

e. g. units 1/10 RPM or 1/100° degree resolution)

Bit 2 ⇒ 0: original behavior as e. g. FW 02.08 (LC1): e. g. 1 RPM

Bit 2 ⇒ 1: new functions are shown in the further sequence (e. g. 1/10 RPM)

4.3.2 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 (brake does not open), is then

displayed 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: original behavior e. g.: $FF00_{hex}$

4.4 General notes according CoE option card

Important: Changes, which are executed via WinBASS II / ProDrive, are not automatically updated on the CoE option card or are noticed. The access on the controller should, if existing in the DSP402, be made with FBO via CoE.

Changes via WinBASS II / ProDrive at the switching over between relative and absolute positioning modes are not noticed on the CoE option card during the positioning operation. This also includes e. g.changes of the operation mode via the Win BASS II. The switchover/change must be made via the CoE.

If the changes however must be made via WinBASS II / ProDrive, there is the possibility to update the parameters at transition of the CoE state machine OPERATIONAL to PRE-OPERATIONAL or to INIT. Furthermore an update is made after saving in the data set and rebooting the controller.

An access can be made to the following parameters/FBOs, both via WinBASS II / ProDrive as well as via the option card:

P0830	no FBO according to DSP 402, access only via the manufacturer-specific object possible
P0304	(FBO 6060_{hex}),
P1031	(FBO 6080_{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),
P1190	(FBO 6086_{hex}).

Following parameters can only be entered via the field bus:

P1172	(FBO 6048_{hex} SIX1, SIX2) for the determination of acceleration
P1173	(FBO 6049_{hex} SIX1, SIX2) for the determination of delay
P3314	(FBO $604C_{hex}$ SIX1),
P3315	(FBO $604C_{hex}$ SIX2)

Furthermore, also the FBOs, which are not available on a parameter in the controller. See ▷Appendix C - Conversion tables◁ from page 73, e. g. Mapping parameters.

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

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

4.4.1 Application parameters

Application parameters are **not** used (other than on the CANopen option card, which also is offered by Baumüller).

4.4.2 Speed profile at the positioning (FBO 6086_{hex}))

The speed profile can be set via the FBO 6086 $_{hex}$), also during the positioning. Thereby the current movement is completed and then the new movement 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 WinBASS II / ProDrive. It can be saved in the data set.

If new target outside ⇒ no movement;
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) is made. The drive behavior is settable via $605A_{hex}$. The error must be acknowledged and then a new movement can be executed.

If the actual position already is outside and the new target also is outside there is ⇒ no movement;
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) is made. Behavior of the drive via $605A_{hex}$ settable. The error must be acknowledged and a new movement then can be executed.

4.4.4 Error tripping at moving in hardware limit switch

The HW limit switch monitoring is settable WinBASS II / ProDrive 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 now be entered via WinBASS II / ProDrive ⇒ under „rescaling“. Then the data set must be saved (restarting of controller necessary).

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

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

In order to be able to enter the required input of e. g. position setpoint in user units UU, the FBO 6092_{hex} accordingly was adjusted.

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

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

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

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

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

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

The number of revolutions is limited to 255.

e.g.
The input via the field bus 360.00 degrees is converted on the option card to the required unit for the controller from 65536 increments for one revolution.

Example: position set value in UU = 36000; accords to 360.00 degrees.

The conversion on the CoE option card looks as follows:

Position setpoint [INC] in b maXX®

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

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

= 65536 [INC]

NOTE!

If the UUs are changed in WinBASS II / ProDrive, the CoE calculates after a reeboot with the new values.

NOTE!

The calculation of the UU is very time-consuming and if possible should not be used at cycle times under 0.5 ms. If the UU is set to 1:1 calculation is not necessary.

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

In addition to the user units there is a gear factor, which is set with the field bus object $604C_{hex}$. With the gear factor it now is possible to, e. g. consider the gear ratio or other scalings, where the necessary speed of the drive now can be calculated.

$604C_{hex}$:

vl_dimension_factor =
vl_dimension_factor_numerator / vl_dimension_factor_denominator

SIX1 = vl_dimension_factor_numerator

INT32 (-33000 ... 33000)

SIX2 = vl_dimension_factor_denominator

INT32 (-33000 ... 33000)

The conversion in the controller e. g. looks like the following:

Speed setpoint motor in the 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 factor is very time-consuming and if possible should not be used at cycle times under 0.5 ms. If the gear factor is set to 1:1 calculation is not necessary.

The gear factor has an affect on the following FBOs:

6042_{hex}, 6043_{hex}, 6048_{hex} Sub01/02, 6049_{hex} Sub/, $606B_{hex}$, C_{hex}, $60FF_{hex}$, 6044_{hex}, 6045_{hex}

4.4.7 CANopen offset

Mapping of numerical scale USIGN32 to INT32 (CANopen). At writing/reading of several FBOs an offset of 2^{31} is internally added or subtracted on the CANopen option card accordant to direction, this is mainly considered at the absolute positioning.

If the position actual values and the target position shall also be shown in WinBASS II / ProDrive 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}, $607A_{hex}$, C_{hex}, D_{hex} Sub ½) - 2^{31}

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

4.4.8 Homing for positioning is necessary

So that during the positioning the image of the numerical scale USIGN32 is made correctly, homing should be made before the positioning.

In WinBASS II / ProDrive on the page „**Homing**“ with the checkbox assigned for this, the drive can be activated to permit a positioning, if no first homing was made.

Deactivated:
In order to operate the operation mode positioning, an error message without reference is provided. It must be considered that in the positioning mode „CANopen“, due to the numerical scale conversion an incorrect homing is made.

Important:

Homing should, if possible, only be made via the field bus. If this is not possible or intended, the checkbox for the offset in WinBASS II / ProDrive must be activated (see ▹CANopen offset◃ on page 32).

Activated:
If the If the drive is enabled in operation mode positioning, without homing taking place, an error message (EMYC-telegram 8600_{hex} ⇒ controller error no. 200) is displayed and the drive remains position-controlled on its current position. Positioning requests are not executed. The positioning requests are executed not until homing was made (once after switching it on). After homing a positioning can be started.

4.4.9 Versions of positioning, target position dependent of the positioning mode (P0601)

NOTE!

It must be considered, that in WinBASS II / ProDrive under positioning 0 also the positioning data set 0 is set, otherwise positioning via the CoE will not be executed correctly. The switching over between the positioning modes „relative (negative/positive)“ and „absolute“ is executed only via the control word. Homing always should be made before positioning.

The CANopen mode (value 9 in the parameter P0601) default on the CoE option card. Thereby the target position in INT32 is calculated. The switch-over between absolute and relative only takes place via the control word. The CANopen mode should be preferred at the CoE option card.

At use of another positioning mode it must be considered, that the number range at absolute positioning is displaced (also see ▹CANopen offset◃ on page 32).

Positioning modes	Description
Absolute/relative CoE: Default (value 9)	• Target is in P0607 (INT32) • Switchover ’absolute/relative’ only occurs via the control word
Relative/positive/negative (value 4)	• Target is in P0607 (INT32) • No switchover ’absolute/relative’ via the control word. • Data type INT32
Absolute/relative (value 10)	• Target is in P0600 • Absolute = USIGN32 • Relative = INT32 • Switchover ’absolute/relative’ only occurs via the control word
Modulo positioning	• Towards the shorter distance (distance-optimized) • Target is in P0607 (INT32) • 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. • Data type INT32

Switchover ’absolute/relative’“. Via the control word bit 6

Control word bit 6 = 0 $\Rightarrow$ absolute

Control word bit 6 = 1 $\Rightarrow$ relative

Conversion of INT32 to UINT means, that an offset of 2^{31} is either added or subtracted, according to the direction ▹CANopen offset◃ on page 32.

In **P1190** with bit 9 it is possible to deactivate the automatic setting of the mode „absolute/ relative CoE“ during the INIT phase of the CoE option card.

P1190 bit 9 = 0 ⇒ activated

P1190 bit 9 = 1 ⇒ deactivated.

Thereby must be considered, that in order to start positioning bit 11 also is necessary, also see parameter manual b maXX® 4400, parameter **P1190**.

5

DATA EXCHANGE AND PARAMETER-IZATION

The access to data or parameter is made at CoE via the field bus 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 XXX and the XXX objects are to be found in ▷Appendix B - Quick reference◁ from page 67.

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 **P0300**, the control word will be transposed. These objects only have subindex 00 $_{hex}$.

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:
1	0	b maXX® 4400

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 present 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 present software version of the option module e. g. the character string „SV01.00“.

Name	Index	Subindex	Data type	Default value
Identity object	1018_{hex}	00_{hex}	U8	03_{hex}
Vendor ID		01_{hex}	U32	15_{hex}
Product code		02_{hex}	U32	0
Revision number		03_{hex}	U32	See below
Serial number		04_{hex}	U32	0

In this object there is information about the device.
The revision number contains the current version of firmware e. g. 00030002 for FW 03.02.

Name	Index	Subindex	Data type	Default value
EtherCAT address	1100_{hex}	00_{hex}	U16	-

In this object the EtherCAT address, that was specified by the master is displayed.

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}. The control word is entered according to default in subindex 01_{hex} (object 6040_{hex} subindex 00_{hex} length 10_{hex}). There are 8 inputs per 4 bytes possible at maximum (at a cycle time of 250 µs), also see ▷PDO mapping◁ from page 45.

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 are in subindex 00_{hex}. The status word is entered according to default in subindex 01_{hex} (object 6041_{hex} subindex 00_{hex} length 10_{hex}). There are 8 inputs at 4 bytes possible at maximum (at a cycle time of 250 µs), also see ▷PDO mapping◁ from page 45.

Name	Index	Subindex	Data type	Default value
SYNC MANAGER Communication Type	$1C00_{hex}$	00_{hex}	U8	04_{hex}
Communication type manager 0 Mailbox receive (master to slave)		01_{hex}	U8	1 (SM0)
Communication type manager 1 Mailbox transmit; (slave to master)		02_{hex}	U8	2 (SM1)
Communication type manager 2 Rx-PDO (master to slave)		03_{hex}	U8	3 (SM2)
Communication type manager 3 Tx-PDO (slave to master)		04_{hex}	U8	4 (SM3)

In this object there is information about the Sync manager settings. The object is read only.

Name	Index	Subindex	Data type	Default value
Sync manager communication type channel 0	$1C10_{hex}$	00_{hex}	U8	0

The object contains information about the communication type of the Sync manager channel 0 (mailbox received), especially the number of PDOs, which were configured for this. The CoE option card has zero PDOs on this channel. The object is read only.

Name	Index	Subindex	Data type	Default value
Sync manager communication type channel 1	$1C11_{hex}$	00_{hex}	U8	0

The object contains information about the communication type of the Sync manager channel 1 (mailbox transit), especially the number of PDOs, which were configured for this. The CoE option card has zero PDOs on this channel. The object is read only.

Name	Index	Subindex	Data type	Default value
Number of assigned RxPDOs	$1C12_{hex}$	00_{hex}	U8	0-255
PDO mapping object index of the associated RxPDO Sync manager communication type channel 2		01_{hex}	U16	1600_{hex}: RX-PDO1

The object contains information about the communication type of the Sync manager channel 2 (process data output). It is displayed how many and which RxPDOs are supported by the slave. At the CoE option card this is a RxPDO.

Name	Index	Subindex	Data type	Default value
Number of assigned RxPDOs	$1C13_{hex}$	00_{hex}	U8	0-255
PDO mapping object index of the associated RxPDO Sync manager communication type channel 3		01_{hex}	U16	$1A00_{hex}$: TX PDO1

The object contains information about the communication type of the Sync manager channel 3(process data input). It is displayed how many and which RxPDOs are supported by the slave. At the CoE option card this is a TxPDO.

Name	Index	Subindex	Data type	Default value
Sync manager synchronization	$1C32_{hex}$	00_{hex}	U8	02_{hex}
Synchronization type		01_{hex}	U16	see below
Cycle time		02_{hex}	U32	Here the cycle time is entered. Specification in ns.

Synchronization type

0	Controller is not synchronized
02_{hex} (2_{dez})	DC Sync0 Synchronization on AL Event Sync0
22_{hex} (34_{dez})	SyncSM2 Synchronization on AL Event Sync Manager2 (RxPDO from master)

This object contains information about the synchronization types of the Sync manager.

The cycle time is specified in ns, e. g. 1 ms ≙ 1 000 000 ns.

5.2 Service data (SDO)

Service data objects (SDO) serve as an exchange of messages without real-time requests. 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 acknowledged data, e. g. the transmitter receives an acknowledge from the receiver. For the transmission of the SDOs the mailbox services are used in ECT.

The mailbox is divided into a telegram header and the mailbox data bytes. In ▷Figure 9◁ on page 40 the mailbox structure is displayed schematically.

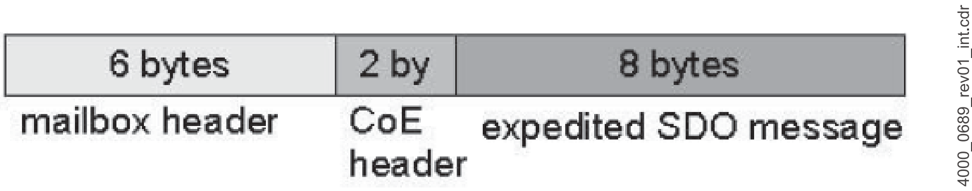

Figure 9: Structure of mailbox [1]

Furthermore the structure of the mailbox headers are divided in:

Figure 10: Mailbox header [1]

Length	Number of mailbox bytes, which follow the header
Address	ECT address of the according slaves
Type	Type of the used mailbox protocols e. g. 3rd CoE (CANopen over EtherCAT)

The CoE header is divided as follows:

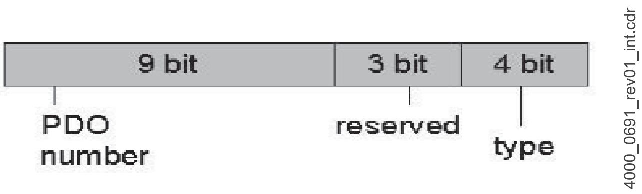

Figure 11: CoE-Header [1]

PDO number	With the mailbox it is also possible to transmit PDOs. Here is specified if the mailbox was configured for the PDO transmission.
Type	0: Reserved 1: Emergency message 2: SDO request 3: SDO response 4: TxPDO 5: RxPDO 6: Remote transmission of TxPDO 7: Remote transmission of RxPDO 8: SDO information 9 - 15: Reserved

5.2.1 Telegram structure according to CANopen

The telegram structure at ECT is defined in the data bytes according to CANopen standard. However the limit of 8 bytes is exceeded depending on whether or not the slave supports this.

The data field of the data telegram (8 bytes) for a SDO is divided in three parts, a command specifier CS (1 byte), a multiplexor M (3 bytes) and the actual service data range D0 - D3 (4 bytes).

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CS	M	M	M	D0	D1	D2	D3

The multiplexor M exist of the 16 bit index of an object and of the associated eight bit wide subindex.

The command specifier CS for a write request in the expedited transfer for the different lengths is:

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

The CS for a write request response is CS = 60_{hex} or in the error case CS = 80_{hex}.

The command specifier CS for a read request in the expedited transfer is CS = 40_{hex}.

The response for the different lengths then is:

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

5.2.2 Types of SDO transfers

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

Expedited Transfer Objects can be written or read, whereat its data includes 4 bytes at maximum. There are only two telegrams required, a request and a response. All objects with the indices $1XXX_{hex}$, $4XXX:_{hex}$, $6XXX_{hex}$ are activated via the expedited SDOs ansprechbar with 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 8-byte limit for the service data is exceeded. This is only possible at reading of the objects 1008_{hex}, 1009_{hex} and $100A_{hex}$.

5.2.3 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 5.2.1◁ on page 41. The data field contains an abort code with 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} 00_{hex}	SDO protocol time out
05_{hex} 04_{hex} 00_{hex} 01_{hex}	Client/server command specific CS not valid or unknown.
05_{hex} 04_{hex} 00_{hex} 05_{hex}	Memory range exceeded
06_{hex} 01_{hex} 00_{hex} 00_{hex}	Error in data format
06_{hex} 01_{hex} 00_{hex} 01_{hex}	Reading on a write-only object
06_{hex} 01_{hex} 00_{hex} 02_{hex}	Writing to a read-only object
06_{hex} 02_{hex} 00_{hex} 00_{hex}	Object does not exist im object directory
06_{hex} 04_{hex} 00_{hex} 41_{hex}	Data cannot be mapped (e. g. incorrect length indication)
06_{hex} 04_{hex} 00_{hex} 42_{hex}	The object number and the length of the objects which are to be mapped are outside the PDO length
06_{hex} 04_{hex} 00_{hex} 43_{hex}	General parameter compatibility
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)
06_{hex} 09_{hex} 00_{hex} 32_{hex}	Value too small (during write accesses)
08_{hex} 00_{hex} 00_{hex} 00_{hex}	General error
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)

5.3 Process data

Process data objects (PDO) are optimized to the exchange of data with real time requests. In the PDOs on the CoE option card at maximum there can be used 32 bytes per communication direction for the service data transmission/cyclic communication. For the data exchange via the PDOs the exact position of the objects in the EtherCAT frame must be defined before beginning the communication between transmitter and receiver. The „field bus memory management unit FMMU“ assigns the logic memory space of the EtherCAT buses to the physical memory space of the slaves. The configuration normally is made in the INIT phase by the master.

The process data of the EtherCAT slaves is described by the SyncManager channels. Every SyncManager describes a related memory range of the cyclic data. With the Sync Manager also a mailbox is described. The EtherCAT option card supports 4 SyncManagers, 2 for the mailbox one in each direction and 2 SyncManager as RxPDO or TxPDO. As at the FMMUs the configuration of the SyncManager is made by the master. Bitwise addressing is provided for in the CoE-standard, but is not possible at the CoE option card (only bytewise addressing is supported).

For the transmission of the cyclic data and the synchronization of the controller there are three synchronization methods at the CoE option card are possible. Synchronization deactivated (the operation is only for the status PRE-OPERATIONAL, from FW version V01.01 to the status OPERATIONAL possible), synchronization to SynchManager2 (RxPD01) and synchronization to distributed clocks DC.

NOTE!

All objects, which were configured in the PDOs are transmitted between the CoE option card and the b maXX® controller as cyclic data (also see ▷Communication flow◁ from page 25). As the cyclic data transmission (especially the RxPDOs) is only made in the state of OPERATIONAL, the communication monitoring in WinBASS II / ProDrive BACI should be only in this status be activated, because in other states (e.g. PRE-OPERATIONAL) otherwise an error message is generated (timeout for cyclic communication **P0836** BACI). This must then be acknowledged after the transition to OPERATIONAL.

5.3.1 PDO mapping

Mapping is a method of assigning variables/objects to PDOS. With these PDOs these variables/objects are transmitted via the bus. Due to mapping the cyclic data exchange is configured. SDOs are used for the parameterization. The mapping is set via addressable objects in the object library. There is such an object for each PDO.

Process data object	Object for content
TX PDO1	$1A00_{hex}$
RX-PDO1	1600_{hex}

NOTE!

In the status OPERATIONAL/SAFE-OP the mapping cannot be changed. At the transition to SAFE-OPERATIONAL/OPERATIONAL a new mapping is made.

Due to the mapping the logical content of the PDOs is determined. For this specification certain information on the object which is to be mapped is necessary: object index, sub index and length of date. From the object library the according objects are entered in the mapping object. The sequence of this entry, determined by the subindex of the mapping object, determines the sequence of the data in the EtherCAT telegram. In the mapping objects (1600_{hex}, $1A00_{hex}$) the objects, which are to be mapped are written to the according subindices (beginning with 01_{hex}), e.g. to the object 1600_{hex} subindex 01_{hex} the value 60400010_{hex}. That means that the first two bytes of the received data in RX-PDO1 are written to the control word (object 6040_{hex} subindex 00_{hex}). The object 6040_{hex} is implemented in the b maXX®-parameter **P0300** control word (also see ▷Appendix C - Conversion tables◁ from page 73). Therewith the first word of the received telegram, which was received in RX-PD01 is written to the control word of the b maXX®. In the subindex 00_{hex} the number of the objects, which are to be mapped (number of the subindices, which are assigned to valid objects) must be entered. An example for the mapping is described in ▷Example for PDO mapping◁ from page 49.

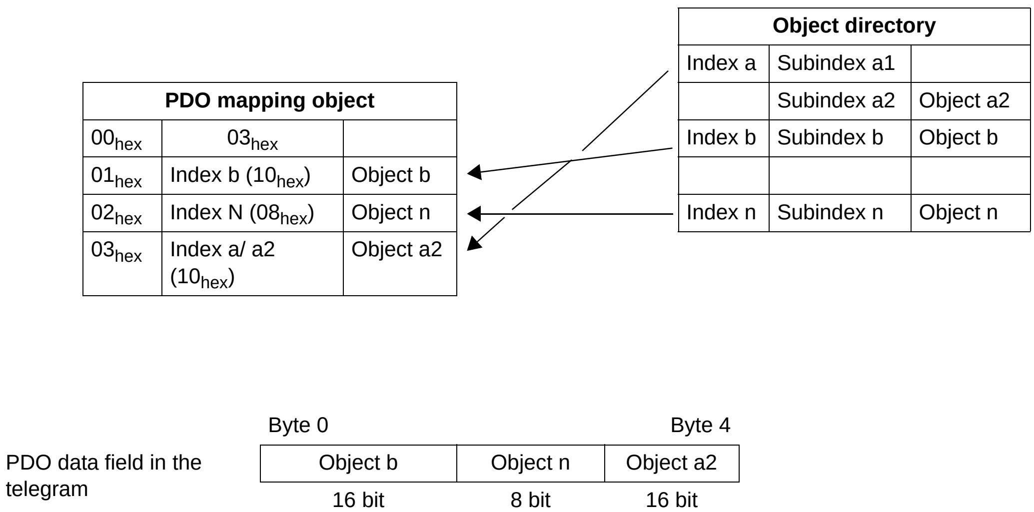

Figure 12: Mapping

Default mapping is described in ▷Directory of objects for communication control◁ from page 35.

In order to delete an existing mapping, the values in the subindices can be overwritten with new objects or can be set to zero. With the writing of „0“ to the subindex 00_{hex} of the according PDO (1600_{hex}, $1A00_{hex}$) the PDO is deactivated.

NOTE!

When setting the mapping in the (1600_{hex}, $1A00_{hex}$) the according subindex 00_{hex} is to be written with the correct number of mapped objects in the end.

Set values: 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 permitted cyclic actual values are marked in a table in column „PDO mapping“ as „TX“. The table is in appendix B.2 (for the 6000-object numbers). At the manufacturer-specific parameter (4000-objects) must be looked up in the parameter manual b maXX® 4400 basic unit (5.02017) for the b maXX ® 4400. A detailed description of the b maXX®-parameters is found in the parameter manual b maXX®.

Incorrect mapping configurations (invalid objects in 1600_{hex}, $1A00_{hex}$) are signalled with abort codes via SDO.

The cyclic set-/actual values are continuously initialized into the BACI, i. e. the first setpoint of PDO1 is on first position in the BACI, the second setpoint of PDO1 on second position s.s.o. Then the setpoints of the PDO2 follow. Analog for the actual value initialization the first actual value of PDO 1 is on first position in the BACI, the second actual value of PDO1 on second position a.s.o.

Dummy mapping The option module CoE slave provides 2 dummy objects: one 1 byte dummy object and one 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 dummy for the usage of certain objects within a telegram only (also see ▷Example for PDO mapping◁ from page 49).

NOTE!

The presently mapping, which was set drops away after a switchoff. Default mapping is set up on, if no new mapping is set.

5.3.2 Synchronization (SYNC)

For synchronization of the controller two synchronization mechanisms can be used. Firstly the synchronization to SM2 (RxPDO) and secondly with the distributed clocks (DC). The DCs were briefly introduced in the ▷Basics EtherCAT◁ from page 9.

Both kinds release an interrupt on the option module CoE, which is transmitted to the b maXX® controller. So this signal can be used for synchronization of the b maXX® controller.

Setting of cycle time

The setting of cycle time should preferably be made via the FBO 1C32 SIX2. Thereby the cycle time is specified in ns. Input 1 000 000 then e. g. corresponds to a cycle time of 1 ms. The input via the FBO is preferred by the option card and can change the cycle time, which was saved in the data set. If a cycle time is written, which is unequal the permitted cycle times of 8 ms, 4 ms, 2 ms, 1 ms, 500 µs and 250 µs synchronization is switched off (is to be identified in WinBASS II / ProDrive under „synchronization“).

Setting of cycle time also can be made via the visualization tool WinBASS II / ProDrive of the controller. On the page „synchronization“ the permitted (see above) cycle time is set. Additionally „Sync 1 signal of the BACI“ should be set as a sync source. After the setting the data set must be saved and the controller must be booted again. Default is booted to SM2, if the DC (distributed clocks) is not activated. At synchronizing to SM2 there is function activated in the FPGA of the option card , which compensates the jitter of the RxPDO from the master (PLL). With the use of the DC this function is deactivated.

If these possibilities are not used the synchronization of the controller is deactivated. **Thereby it must be considered that the option card cannot be switched after SAFE-OPERATIONAL.** A state change after OPERATIONAL is only possible from FW version V01.01.

Settings for the BACI are made by the option module.

NOTE!

If the synchronization is changed during the running operation the controller must be booted again.

Master setting for the use of distributed clocks (DC)

So that the DC can be activated in the ECT-FPGA (ASIC), the register address ECT ASIC 981 $_{hex}$ (for this see [3]) must be described by the master as follows:

Bit 0 ⇒ 1 „Activate cyclic operation“

Bit 1 ⇒ 1 „Activate Sync0“

The checking in the slave is made at transition from PRE-OPERATIONAL to SAFE-OPERATIONAL of ECT state machine.

Via FBO 1C32: SIX1 is set, which synchronization mode is wanted:

Value 0_{hex} ⇒ freerun, not synchronized

Value 2_{hex} ⇒ DC Sync0, synchronized with DC IRQ Sync0

Value 22_{hex} ⇒ SyncSM2, synchronized with SyncManager IRQ of the SM2 (SyncManager2 RxPDO)

All the other synchronization kinds are not supported. If, however it is tried to write an error message on it (06010000_{hex} = error in data format) is generated.

The cycle time must be set via FBO $1C32_{hex}$:SIX2 or via register $9A0_{hex}$ in the ECT(DWORD in ns). The master must provide that the setpoint telegrams at set CD 200 µs to 50 µs are not send before the SYNC event. With help of the sync offsets in the WinBASS II / ProDrive page „synchronize“ it is possible to shift this „prohibited range“, if the master has no chance to shift the setpoint telegrams out of the „prohibited range“.

If in both cases no cycle time was set, the cycle time is accepted accordingly to the page synchronization in WinBASS II / ProDrive.

NOTE!

It can be that there is another cycle time set in the controller as wanted.

If the DCs are activated the „PLL“ is deactivated due to the option card (in the FPGA) and the sync0 signal is directly transmitted from the DC to the controller. Additionally is monitored if setpoint failures occur. If there is no setpoint received in a cycle an EMCY is settled (8100_{hex} XX 02_{hex} AA_{hex} 00_{hex} 00_{hex}).

At transition to SAFE-OPERATIONAL it is checked if in register 981_{hex} the DCs were activated. If not FBO $1C32_{hex}$: SIX1 is reset to the value 22_{hex} (synchronization to SM2).

Necessary for the DC is a FPGA version greater B20, in the ESD file ETC03_121.eds the FBO $1C321_{hex}$ is contained.

5.3.3 Example for PDO mapping

The option module CoE slave with the node address 1 receives from the master a speed setpoint in RX-PD01. This speed setpoint must be written to the ramp function generator input. Furthermore node 1 receives the control word from the master its RX-PDO1. Node 1 sends its actual speed value as actual value and the status word. The cycle time must be 1 ms and must be synchronized to DC or to the SM2. With help of the master the slave is brought out of the init phase into the condition PRE-OPERATIONAL. This takes place with the defined application layer (AL) event-driven mechanism.

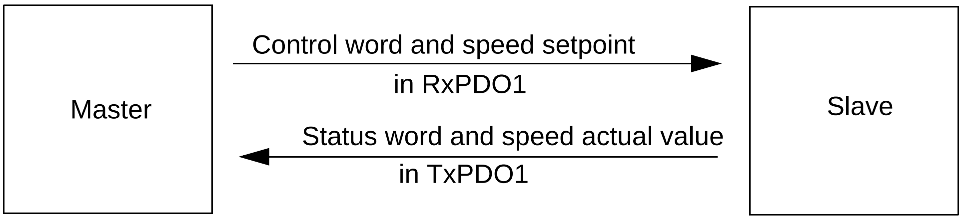

Figure 13: Example mapping with a b maXX®

1st step: determining the necessary objects

Ascertain the relevant object directory objects from the object list (see ▹Appendix C - Conversion tables◃ from page 73 and ▹Directory of objects for communication control◃ from page 35).

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

Parameter number, e.g. at the b maXX® 4400		CANopen field bus object
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

2nd step: configure mapping

Writing of the first object to be mapped with index (6041_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 01_{hex} (TxPDO 1).

Writing of the second object to be mapped with index (6044_{hex}), subindex (00_{hex}) and length (10_{hex}) to $1A00_{hex}$ subindex 02_{hex} (TxPDO 1).

Writing of amount of the mapped objects (02_{hex}) to $1A00_{hex}$ subindex 00_{hex} (TxPDO 1).

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

1A00$_{hex}$	**00**$_{hex}$	02_{hex}
	01$_{hex}$	60410010_{hex}
	02$_{hex}$	60440010_{hex}

Writing of the first object to be mapped with index (6040_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 01_{hex} (RxPDO 1).

Writing of the second object to be mapped with index (6042_{hex}), subindex (00_{hex}) and length (10_{hex}) to 1600_{hex} subindex 02_{hex} (RxPDO 1).

Writing of the number of mapped objects (02_{hex}) auf 1600_{hex} subindex 00_{hex} (RXPDO 1).

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

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

Now the cycle time of 1 ms still must be set as well as the synchronization mode (SM2 or sync0). This is made via SDOs with the FBO $1C32_{hex}$ SIX1/2. Additionally at synchronization to DC there still settings in the EtherCAT ASIC must be made. Here see ▷Synchronization (SYNC)◁ from page 47.

3rd step: synchronization

- Synchronization to DC

FBO $1C32_{hex}$ SIX1 type of synchronization:

Request

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
$2B_{hex}$	32_{hex}	$1C_{hex}$	01_{hex}	02_{hex}			

CS= $2B_{hex}$ for 1 byte;
$32_{hex}1C_{hex}$ is crossed and compounds of $1C32_{hex}$;
SIX1=02_{hex} for the sync type DC.

The response to this is:

Response

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
60_{hex}	32_{hex}	$1C_{hex}$	01_{hex}	02_{hex}			

- Synchronizing to SM2

FBO $1C32_{hex}$ SIX1 type of synchronization:

Request

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
$2B_{hex}$	32_{hex}	$1C_{hex}$	01_{hex}	22_{hex}			

CS= $2B_{hex}$ for 1 byte;
$32_{hex}1C_{hex}$ is crossed and compounds of $1C32_{hex}$;
SIX1=22_{hex} for the sync type SyncManager2.

The response to this is:

Response

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
60_{hex}	32_{hex}	$1C_{hex}$	01_{hex}	22_{hex}			

Alternatively the cycle time can also be saved in the data set. By default the synchronizing then is set to SM2, the FBO $1C32_{hex}$ SIX1/2 is not necessary.

5.3.4 Entry in BACI

Eight cyclic set values and eight cyclic actual values can be replaced between the option module CoE slave and the b maXX® controller at the same time. All values are updated in one cycle. The set-/actual values at CoE can be spread to one PDO each. 8 times 4 bytes can be transmitted cyclical per direction.

The updating time for the processing of the PDOs in the controller is dependent of the communication time, which was set in the b maXX® controller (see communication to the b maXX® controller). The inputs are made continuously starting with the 1st object of PDO1 the contents are checked for validity for the BACI configuration (no dummy) in turns. If the object is valid then this is entered at the next spare position of the BACI configuration. If the PDO mapping is faulty (incorrect parameter number or the like), there is no cyclic communication started between option card and b maXX® controller.

NOTE!

If, in the existing PDO of the same direction repeatedly several identical object numbers are mapped, then the object only appears once in the BACI configuration.

Thereby must be considered that the objects can possibly interact.

NOTE!

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

5.4 Error telegram (EMCY)

Emergency telegrams at cyclic data serve as a display of b maXX® errors. As soon as the b maXX® controller has recognized an internal error the EMCY telegram is transmitted. At each error which is added new an EMCY telegram is send, if the error number is smaller than the value shown before. A telegram repetition is not made. The EMCY telegram is send in CoE via the mailbox mechanism. The master thereby e.g. asks every 20 ms if an EMCY was entered from the slave into the sending mailbox.

5.4.1 Telegram structure

The user data area of the emergency message frame is organized into 3 sections:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Emergency error code		Error register	Manufacturer-specific error field				

The emergency error code (byte 0,1) is defined in CANopen DSP 402. The conversion to b maXX® error numbers is shown in ▹Conversion of error messages to DSP 402 V1.1◃ from page 54.

The error register is defined as follows.

Bit	Meaning
0	Error occurred, general error
1	Current error
2	Voltage error
3	Temperature error
4	Communication error
5	Device-specific error
6	Not used
7	Manufacturer-specific error

Byte 3: contains the number of bytes of the following emergency codes which is the length 2 bytes at device-specific errors.

Byte 4 and 5 of the manufacturer-specific error field contain the b maXX® 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:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00_{hex}	73_{hex}	81_{hex}	2_{hex}	73_{hex}	00_{hex}	00_{hex}	00_{hex}

If there are several errors and if an error is deleted the option module CoE slave transmits the EMCY telegram with the next error number. If all errors are acknowledged the telegram „error reset/no error“ is transmitted. Thereby all the bytes of the telegram are assigned to 0.

5.4.2 Conversion of error messages to DSP 402 V1.1

The description of the controller error messages and the notes concerning trouble shooting is found in the manual b maXX®. 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	7000_{hex}
0002_{hex}	Incorrect or unexpected interrupt has occurred	7000_{hex}
0003_{hex}	NMI interrupt has occurred - incorrect bus access	7000_{hex}
0010_{hex}	System boot error	5000_{hex}
0011_{hex}	Software error (e. g. switch)	5000_{hex}
0012_{hex}	Configuring error of the time-slice operating system	7000_{hex}
0013_{hex}	time slot - time error	7000_{hex}
0014_{hex}	No more free memory	7000_{hex}
0015_{hex}	Software error: invalid error code	7000_{hex}
0016_{hex}	Software error: invalid warning code	7000_{hex}
0017_{hex}	FPGA version is t not compatible with firmware	7000_{hex}
0020_{hex}	Timeout ProProg-protocol	8100_{hex}
0021_{hex}	Protocol error	7000_{hex}
0022_{hex}	Wrong module type	7000_{hex}
0023_{hex}	Too much data in list or telegram	7000_{hex}
0024_{hex}	Too little data in list or telegram	7000_{hex}
0025_{hex}	Invalid operand	7000_{hex}
0026_{hex}	Device supports only VARSTAT_MEMORY	7400_{hex}
0027_{hex}	Invalid operand address (logical address)	7000_{hex}
0028_{hex}	Value less than the minimum value	7000_{hex}
0029_{hex}	Value greater than the maximum value	7000_{hex}
$002A_{hex}$	Parameter is read-only	7000_{hex}
$002B_{hex}$	Parameter cannot be changed because of operational status	7000_{hex}
$002C_{hex}$	Invalid parameter value	7000_{hex}
$002D_{hex}$	WinBASS II / ProDrive is not connected anymore or does not react.	7000_{hex}
0030_{hex}	Error in the SmallModule_A (in order to find out the exact error number the parameter **P0240** must be read in the b maXX®, the error description of the **P0240** is described from ▹page 60◃).	$FF00_{hex}$

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0031_{hex}	Error in the SmallModule_B (in order to find out the exact error number the parameter **P0240** must be read in the b maXX®, the error description of the **P0241** is described from ▷page 60◁)	$FF00_{hex}$
0032_{hex}	Error in the SmallModule_C (in order to find out the exact error number the parameter **P0242** must be read in the b maXX ® , the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0033_{hex}	Error in the SmallModule_D (in order to find out the exact error number the parameter **P0243** must be read in the b maXX ®, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0034_{hex}	Error in the SmallModule_F (in order to find out the exact error number the parameter **P0244** must be read in the b maXX ®, the error description of the **P0240** is described from ▷page 60◁).	$FF00_{hex}$
0035_{hex}	Error in the BigModule_G (in order to find out the exact error number the parameter **P0245** must be read in the b maXX®, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0036_{hex}	Error in the BigModule_H (in order to find out the exact error number the parameter **P0246** must be read in the b maXX®, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0037_{hex}	Error in the BigModule_J (in order to find out the exact error number the parameter **P0247** must be read in the b maXX®, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0038_{hex}	Error in the BigModule_K (to determine the correct error number, the parameter **P0248** in the b maXX® must be read; the error designation of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
0039_{hex}	Error in the BigModule_L (in order to find out the exact error number the parameter **P0249** in the b maXX® must be read, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
$003A_{hex}$	Error in the BigModule_M (in order to find out the exact error number the parameter **P0250** in the b maXX® is read, the error description of the **P0240** is described from ▷page 60◁)	$FF00_{hex}$
$003B_{hex}$	Timeout at system initialization procedure	7000_{hex}
$003C_{hex}$	CRC error in SPI transmission module ⇒ controller	7000_{hex}
$003D_{hex}$	CRC error in SPI transmission controller ⇒ module	7000_{hex}
0040_{hex}	Mains failure	3100_{hex}

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0041_{hex}	Phase failure	3100_{hex}
0042_{hex}	Mains undervoltage	3100_{hex}
0043_{hex}	Mains overvoltage	3100_{hex}
0044_{hex}	Undervoltage 24V	3100_{hex}
0045_{hex}	Phase sequence identification error	3100_{hex}
0046_{hex}	Error frequency change	3100_{hex}
0047_{hex}	Frequency range error	5000_{hex}
0048_{hex}	Error/defective at the contactor (checkback although not controlled)	5000_{hex}
0049_{hex}	No checkback from the contactor	3100_{hex}
$004A_{hex}$	Error U DC link structure	3100_{hex}
$004B_{hex}$	Undervoltage U DC link	3100_{hex}
$004C_{hex}$	Mains connection error	3100_{hex}
$004D_{hex}$	Current limit reached	3100_{hex}
$004E_{hex}$	Synchronization errors	3100_{hex}
0050_{hex}	Communication error according to HIPERFACE specification In order to find out the accordant error number the parameter **P0233** in the b maXX® must be read, the error identification is described in the following from ▹page 60◃)	$FF00_{hex}$
0051_{hex}	Temperature threshold of heatsink exceeded	4200_{hex}
0052_{hex}	U DC link overvoltage	3200_{hex}
0053_{hex}	Overcurrent power unit	2300_{hex}
0054_{hex}	Ground current	2200_{hex}
0055_{hex}	Temperature threshold of inside air exceeded	4200_{hex}
0056_{hex}	Ixt) [%] limit value	$FF00_{hex}$
0057_{hex}	Safety relay off (or defect)	5000_{hex}
0058_{hex}	Safety relay off (safety relay okay, but voltage nonexistent)	5000_{hex}
0059_{hex}	Power unit not ready-to-operate	5000_{hex}
$005A_{hex}$	Phase failure	3100_{hex}
$005B_{hex}$	Mains failure	3100_{hex}
$005C_{hex}$	Mains undervoltage	3100_{hex}
$005D_{hex}$	Mains overvoltage	3100_{hex}
$005E_{hex}$	Undervoltage U DC link	3100_{hex}

These errors are supported **only** from b maXX®4100.

Controller Error code	Description (of b maXX®controller)	CANopen Error code
0060_{hex}	Temperature sensor of the motor short-circuited ($T_M \leq$ -30 °C)	4300_{hex}
0061_{hex}	Temperature sensor of the motor not connected (T_M > +300 °C)	4300_{hex}
0062_{hex}	Error motor temperature - Shutdown threshold exceeded	4300_{hex}
0063_{hex}	Error I^2t > 100% in the motor	7000_{hex}
0070_{hex}	Communication error according to HIPERFACE specification (in order to find out the accordant error number the parameter **P0234**/**P0235** in the b maXX® must be read, the error identification is described in the following from ▹page 60◃)	$FF00_{hex}$
0071_{hex}	Invalid module code	7000_{hex}
0072_{hex}	Error at writing of encoder position	7000_{hex}
0073_{hex}	Cable break encoder 1	7000_{hex}
0074_{hex}	Overspeed encoder 1	7000_{hex}
0075_{hex}	Amplitude limit exceeded	7000_{hex}
0076_{hex}	Encoder type unknown	7000_{hex}
0077_{hex}	Invalid data field for motor data	7000_{hex}
0078_{hex}	Incorrect motor data	7000_{hex}
0079_{hex}	Saving error of motor data	7000_{hex}
$007A_{hex}$	Motor data write-protected (not BM motors)	7000_{hex}
$007B_{hex}$	Error field angle	7000_{hex}
$007C_{hex}$	Encoder without temperature measuring	7000_{hex}
0080_{hex}	Communication error according to HIPERFACE specification (in order to find out the accordant error no. the parameter **P0235** in the b maXX® must be read, the error identification is described in the following from ▹page 60◃)	$FF00_{hex}$
0081_{hex}	Invalid module code	7000_{hex}
0082_{hex}	Error at writing of encoder position	7000_{hex}
0083_{hex}	Cable break encoder 2	7000_{hex}
0084_{hex}	Encoder 2 overspeed	7000_{hex}
0085_{hex}	Amplitude limit exceeded	7000_{hex}
0086_{hex}	Encoder type unknown	7000_{hex}
0087_{hex}	Invalid data field for motor data	7000_{hex}
0088_{hex}	Incorrect motor data	7000_{hex}
0089_{hex}	Saving error of motor data	7000_{hex}

Controller Error code	Description (of b maXX®controller)	CANopen Error code
008A$_{hex}$	Motor data write-protected (not BM motors)	7000$_{hex}$
008B$_{hex}$	Error field angle	7000$_{hex}$
008C$_{hex}$	Encoder without temperature measuring	7000$_{hex}$
0090$_{hex}$	Absolute position of the encoder unknown	7000$_{hex}$
0091$_{hex}$	Absolute position of the encoder unknown	7000$_{hex}$
0092$_{hex}$	Encoder module 1 is missing	7000$_{hex}$
0093$_{hex}$	Encoder module 2 is missing	7000$_{hex}$
0094$_{hex}$	Measurement storage for encoder module is missing	7000$_{hex}$
0095$_{hex}$	At resolver no measured value storage possible	7000$_{hex}$
0096$_{hex}$	Triggering on zero pulse and encoder is no incremental encoder	7000$_{hex}$
0097$_{hex}$	Digital I/O module required and missing	7000$_{hex}$
0098$_{hex}$	Incremental encoder emulation module required and missing	7000$_{hex}$
0099$_{hex}$	Encoder module 1 required for incremental encoder emulation and missing	7000$_{hex}$
009A$_{hex}$	Encoder module 2 required for incremental encoder emulation and missing	7000$_{hex}$
009B$_{hex}$	Initialization error of the incremental encoder emulation module	7000$_{hex}$
009C$_{hex}$	Incremental encoder emulation module signals error	7000$_{hex}$
009D$_{hex}$	Incremental encoder emulation: Selecting the option 'start after first zero pulse' for non-incremental encoder	7000$_{hex}$
009E$_{hex}$	SSI encoder emulation module is missing	7000$_{hex}$
009F$_{hex}$	Error in setpoint source encoder 1 or encoder 2	7000$_{hex}$
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$_{hex}$
00B0$_{hex}$	EEPROM copy error	5000$_{hex}$
00B1$_{hex}$	Timeout while writing to EEPROM	5000$_{hex}$

Controller Error code	Description (of b maXX®controller)	CANopen Error code
$00B2_{hex}$	Checksum error in EEPROM	5000_{hex}
$00B3_{hex}$	No boot record	5000_{hex}
$00B4_{hex}$	Incompatible SW	5000_{hex}
$00B5_{hex}$	Data record switching: DS not present	5530_{hex}
$00B6_{hex}$	Checksum error in the PSI	5000_{hex}
$00B7_{hex}$	PSI is reset	5000_{hex}
$00B8_{hex}$	PSI data invalid	5000_{hex}
$00B9_{hex}$	Self-optimization tables are invalid. Execute self-optimization again	5000_{hex}
$00BA_{hex}$	A/D correction table invalid	5000_{hex}
$00C0_{hex}$	Position deviation dynamic	8000_{hex}
$00C1_{hex}$	Position deviation static	8000_{hex}
$00C2_{hex}$	Encoder 1 for position control used, but inactive	7300_{hex}
$00C3_{hex}$	Encoder 2 for position control used, but inactive	7300_{hex}
$00C4_{hex}$	Software-limit switch monitoring 1 active	8600_{hex}
$00C5_{hex}$	Software-limit switch monitoring 2 active	8600_{hex}
$00C6_{hex}$	Hardware-limit switch monitoring 1 active	8600_{hex}
$00C7_{hex}$	Hardware-limit switch monitoring 2 active	8600_{hex}
$00C8_{hex}$	Positioning started without homing	8600_{hex}
$00C9_{hex}$	Setpoint in the mode Set-of-setpoints didn’t arrive in time	8600_{hex}
$00CA_{hex}$	Monitoring of modulo position active: Target position > modulo position	8600_{hex}
$00CB_{hex}$	Spindle positioning: error at initialization of the trigger	8600_{hex}
$00CC_{hex}$	Spindle positioning: timeout at trigger signal (zero pulse/switch input)	8600_{hex}
$00D0_{hex}$	Drive blocked	7000_{hex}
$00D1_{hex}$	Maximum speed reached	7000_{hex}
$00D2_{hex}$	Encoder 2 is used for motor control but inactive.	7000_{hex}
$00D3_{hex}$	Overspeed Open Loop	7000_{hex}
The manufacturer-specific error codes 0030_{hex} to $003B_{hex}$, 0050_{hex}, 0070_{hex} and 0080_{hex} are displayed summarized via an EMY telegram with the CANopen error code $FF00_{hex}$. The exact identification can be read out in the following parameters **P0233**, **P0234**, **P0235** and **P0240** to **P0250** and the description to the read-out error numbers is shown in the following.		

Controller Error code	Description (of b maXX® controller)	CANopen Error code
The following b maXX® error codes are n o t displayed via an EMY telegram. Error code (0050_{hex}) ⇒ **P0233** Communication error after 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 altered	
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 answer	
Error code (0070_{hex}) ⇒ **P0234** Communication error after HIPERFACE specification (Enc1HiperfaceError) Error code (0080_{hex}) ⇒ **P0235** Communication error after 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	
$0B_{hex}$	Unknown command code	
$0C_{hex}$	Number of the transmitted data is wrong	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
$0D_{hex}$	Invalid argument	
$0E_{hex}$	Data field is write protected	
$0F_{hex}$	Invalid access code	
10_{hex}	Data field size cannot be altered	
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 - position determination impossible	
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 and accordingly MRS code in response telegram is incorrect	
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	
61_{hex}	Error signal amplitude	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
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 in the 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 the 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	
1013_{hex}	Wrong parameter no. at actual value parameter 4	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
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	
1035_{hex}	BACI timeout at service data	

Controller Error code	Description (of b maXX® controller)	CANopen Error code
1036_{hex}	Checksum error during test	
1037_{hex}	Ramp-up: timeout during waiting for slave type or during waiting for reset of config-pending-flag	
1038_{hex}	Invalid data transfer structure type	
1039_{hex}	Internal error: invalid 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: wrong channel number (0 or > 6)	
$103E_{hex}$	Cmd-interface: stated channel does not exist	
$103F_{hex}$	Cmd-interface: internal error - invalid pointer	
1040_{hex}	Cmd-interface: internal error - invalid status	
1041_{hex}	Cmd-interface: wrong package number	
1042_{hex}	Cmd-interface: wrong command number	
1043_{hex}	Cmd-interface: wrong condition at package handling	
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 ID: PCI-error at reading	
$104A_{hex}$	Module ID: PCI-error at writing	
$104B_{hex}$	Module identification: general error when reading	
$104C_{hex}$	Module identification: general error when 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)	

APPENDIX A - ABBREVIATIONS

BACI	Baumüller drives serial interface
DC	Distributed Clocks
EMCY	Error telegram
FMMU	Fieldbus Memory Management Unit
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	Subindex
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 $number_{hex}$.
The subindex for all 4000-parameters always is 00_{hex}.

Example

Parameter **P0053** ⇒ object index 4035_{hex} subindex 00_{hex}

The information in italic type is a note that the parameter also can be influenced by a 6000-parameter or several 6000-parameters. Further notes are found in (▹B.2 6000 object numbers (device profile DSP 402)◃ ab Seite 68).

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

It is possible to access some parameters of the controller as well as via one or several 6000s **as well as** via 4000-objects (see *cursive* text in ▷B.1 4000 object numbers (manufacturer-specific objects)◁ ab Seite 67).

Access to some parameters only possible with a 6000-parameter ($606A_{hex}$, 6048_{hex} SIX1, 6049_{hex} SIX1 and $604C_{hex}$ SIX1/2).

It must be regarded that the standardizations between the manufacturer-specific objects and the device profile objects normally are different. If both field bus objects (via manufacturers-specific objects and via device profile object) for one and the same parameter in the b maXX® controller are mapped both field bus objects will interact. This must be avoided, therefore an access to the objects, which are not in the device profile must not be made via the manufacturer-specific objects.

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

CANopen object number		Parameter no.	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	**Subindex**				
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}		TX / RX	rw	Velocity mode
6048_{hex}	02_{hex}	**P1172**	TX / RX	rw	Velocity mode
6049_{hex}	01_{hex}		TX / RX	rw	Velocity mode
6049_{hex}	02_{hex}	**P1173**	TX / RX	rw	Velocity mode
$604C_{hex}$	01_{hex}		TX / RX	rw	Velocity mode
$604C_{hex}$	02_{hex}		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
6051_{hex}	00_{hex}	**P1174**	TX / RX	rw	Velocity mode
$605A_{hex}$	00_{hex}	**P1004**	TX	rw	Device control

CANopen object number		Parameter no.	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	Subindex				
$605B_{hex}$	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	wo	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
$607A_{hex}$	00_{hex}	**P0600**	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
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

CANopen object number		Parameter no.	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	Subindex				
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
$60F8_{hex}$	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
$60FB_{hex}$	$0C_{hex}$	**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

CANopen object number		Parameter no.	PDO mapping	Access type	Operating mode acc. to DSP 402
Index	Subindex				
$60FB_{hex}$	$1B_{hex}$	**P0262** **P0263**	TX	ro	Position control function
$60FD_{hex}$	00_{hex}	**P0410**	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 giving the value ranges ($x = x_{min}$ .. $x_{max.}$) and the representation 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.:	Representation of the CANopen object indices on b maXX® controller parameter
Controller parameters:	Identification of the controller parameters
P. no. ▸ index:	Conversion of the b maXX® controller parameters to CANopen object indices

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res- caling	Comment
Control word	6040_{hex}	►	**P0300**		Control word	**P0300**	►	6040_{hex}	Bit 6 in the control word now is supported; Bit 6 = 0: 6 = 0 : Positioning mode „abso-lute“ Bit 6 = 1: 6 = 1 : Positioning mode „relative, negative positive“ Via the CoE and the control word no other positioning mode is supported.
	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	
Status word	6041_{hex} /ro				Status word	**P0301**	►	6041_{hex}	
	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 WinBASS via drive manager adjustable

CANopen object	Index Value range	►	P. no. Scaling	Controller parameters	P. no. Value range	►	Index res-caling	Comment
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 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 $604C_{hex}$.
	x = -32768 .. 32767	►	y = x *4000_{hex}/ MotorMax-Speed		x = -32768 .. 32767	►	y = x*Motor-MaxSpeed / 4000_{hex}	
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 $604C_{hex}$.
					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 $604C_{hex}$.
					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 $604C_{hex}$.
					x = 8000_{hex} .. $7FFF_{hex}$	►	y = x*Motor-MaxSpeed / 4000_{hex}	
vl_velocity_min_max_amount	6046_{hex}							
vl_velocity_min_amount	Sub. 01_{hex}		„none“	SpeedSet_Ulim	„none“	►	Sub. 01_{hex}	Sub. 1 always is zero, the min. limit is determined 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}	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index res- caling	Comment
vl_velocity_acceleration	6048_{hex}	$a = \left(\frac{dv}{dt}\right)$; Because there is no parameter for the acceleration, the acceleration is attained by varying the ramp-up time **P1172**. It is scaled to the maximum speed of the controller. The required acceleration is calculated with the input of dv in SIX1 and then dt in SIX2. Only then the correct calculated time is written to the ramp-up time. The reconstruction of the set acceleration $\left(\frac{dv}{dt}\right)$ is not possible after a rebooting.							
vl_delta_speed	Sub. 01_{hex}						▸	Sub. 01_{hex}	Scaling of gear ratio is saved in FBO $604C_{hex}$.
	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*Motor-MaxSpeed / Δv*100			x = 0 .. 65000	▸	y = Δt	
vl_velocity_deceleration	6049_{hex}	$a = \left(\frac{dv}{dt}\right)$; Because there is no parameter for the delay in the controller, the delay is attained by varying the ramp-down time **P1173**. It is scaled to the maximum speed of the controller. The required delay is calculated with the input of dv in SIX1 and then dt in SIX2. Only then the correct calculated time is written to the ramp-down time. Only then the correct calculated time is written to the ramp-down time. The reconstruction of the set delay $\left(\frac{dv}{dt}\right)$ is not possible after a rebooting.							
vl_delta_speed	Sub. 01_{hex}						▸	Sub. 01_{hex}	Scaling of gear ratio is saved in FBO $604C_{hex}$.
	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*Motor-MaxSpeed / - Δv*100			x = 0 .. 65000	▸	y = Δt	
vl_dimension_factor	$604C_{hex}$							$604C_{hex}$	The calculation in the controller for example appears 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	

CANopen object	Index Value range		P. no. Scaling		Controller parameters	P. no. Value range		Index res- caling	Comment
vl_ramp_function_time	$604F_{hex}$	►	**P1172**		RFG1RampUpTime	**P1172**	►	$604F_{hex}$	Ramp function generator ramp-up time (1 = 1/1000 s ⇒ 1s = 1000). Resolution is 10 ms
	x = 0 .. $FFFFFFFF_{hex}$	►	y = x			x = 0 .. 65000	►	y = x	
vl_slow_down_time	6050_{hex}	►	**P1173**		RFG1RampDownTime	**P1173**	►	6050_{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	
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	x = 4		y = x	
Slow down on slow down ramp and remain in quick-stop	x = 5	►	y = x		Ramp-down ramp and remain in quickstop	x = 5		y = x	
Slow down on quickstop ramp and remain in quick-stop	x = 6	►	y = x		Ramp-down on quickstop ramp and remain in quickstop	x = 6		y = x	
Slow down on current and remain in quick-stop	x = 7	►	y = x		Ramp down at current limit and remain in quickstop.	x = 7		y = x	
Slow down on voltage limit and remain in quick-stop	x = 8	►	y = x		Ramp down at voltage limit and remain in quickstop	x = 8		y = x	
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	

CANopen object	Index Value range	►	P. no. Scaling	Controller parameters	P. no. Value range	►	Index res- caling	Comment
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}
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 res- caling	Comment
fault_reaction_option_ code	605E$_{hex}$	►	**P1007**		ErrorReactionCode	**P1007**	►	605E$_{hex}$	For static and dynamic position deviation and for the reactions for the FBO 6007$_{hex}$ „Mode 1 malfunction“ settable only.
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	
modes_of_operation	6060$_{hex}$ /wo	►	**P1000**						
Conversion formalism	x = -128 .. 127	►	y = x						
Manufacturer specific	x = -128 .. -7	►	y = x						
Manufacturer specific	x = -6	►	y = 5						
Manufacturer specific	x = -5	►	y = x						
Manufacturer specific	x = -4	►	y = x						
Manufacturer specific	x = -3	►	y = x						
Manufacturer specific	x = -2	►	y = x						
Manufacturer specific	x = -1	►	y = x						
reserved	x = 0								
Profile position mode	x = 1	►	y = x						
Velocity mode	x = 2	►	y = x						
Profile velocity mode	x = 3	►	y = -3						
Torque profile mode	x = 4	►	y = x						
reserved	x = 5								
Homing mode	x = 6	►	y = x						
Interpolated position mode	x = 7	►	y = x						
reserved	x = 8 .. 127								

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res-caling	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	
					Self-optimization	x = 1 .. -7			
					not used	x = -6			
					Synchronous operation el. gear	x = -5	►	y = x	
					Position control	x = -4	►	y = x	
					Speed control	x = -3	►	y = 3, y = -3	If via (FBO 6060_{hex}) the mode 3 was selected the value 3 is returned otherwise the value -3. After the power down the information about the mode which was selected is lost.
					Current control	x = -2	►	y = x	
					Notch position	x = -1	►	y = x	
					not used	x = 0			
					Target position setpoint	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}	An offset of 2^{31} is added to USIGN32 on the CoE option card. USIGN32 -> INT32. (offset of 2^{31} is subtracted. UU - ratio added.
						x = 80000000 .. $7FFFFFFF_{hex}$	►	$y = x-2^{31}$	
position_actual_value*	6063_{hex} /ro				PPosActValue (actual posi-tion value)	**P0462**	►	6063_{hex}	UU - ratio added.
						x = 80000000 .. $7FFFFFFF_{hex}$	►	y = x	
position_actual_value	6064_{hex} /ro				PPosActValue (actual posi-tion value)	**P0462**	►	6064_{hex}	An offset of 2^{31} is added to USIGN32 on the CoE option card. USIGN32 ? INT32 (offset of 2^{31} is subtracted. UU - ratio added.
						x = 80000000 .. $7FFFFFFF_{hex}$	►	$y = x-2^{31}$	

CANopen object	Index Value range	►	P. no. Scaling	Controller parameters	P. no. Value range	►	Index res- caling	Comment
following_error_time_out	6066_{hex} /ro			PosDevTime	**P1056**	►	6066_{hex}	The Unit is ms in the CANopen object and in the b maXX®controller parameter.
					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 (pro. 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.
					x = 8000_{hex} .. $7FFF_{hex}$	►	y = x*Motor- MaxSpeed / 4000_{hex}	
velocity_actual_value	$606C_{hex}$ /ro			SpeedActValue	**P0353**	►	$606C_{hex}$	Scaling of gear ratio is saved in FBO $604C_{hex}$.
					x = -2^{31} .. 2^{31}-1	►	y = x*Motor- MaxSpeed / 40000000_{hex}	
velocity_threshold	$606F_{hex}$	►	**P1073**	ENC1Mon_Llim	**P1073**	►	$606F_{hex}$	The threshold can be increased in the b maXX® controller up to 25% of the maximum speed of the controller. The input then is made 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}	
target_position	$607A_{hex}$	►	**P0607** (**P0600**)	PPosTarget1	**P0607** (**P0600**)	►	$607A_{hex}$	UU - ratio added.
	x = 80000000_{hex} ..$7FFFFFFF_{hex}$	►	y = x		x = 800000000 .. $7FFFFFFF_{hex}$	►	y = x	

CANopen object	Index Value range	►	P. no. Scaling	Controller parameters	P. no. Value range	►	Index res- caling	Comment
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).
	x = -2^{31} .. 2^{31}-1	►	y = x+2^{31}		x = 0 .. 2^{32}-1	►	y = x-2^{31}	
software_position_limit	607D$_{hex}$			SW limit switch			607D$_{hex}$	An offset of 2^{31} is added to the USIGN32 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 interpreted 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 interpreted in the controller as RPM.
	x =0 .. FFFF$_{hex}$	►	y = x		x = 1 .. 24000	►	y = x	
profile velocity	6081$_{hex}$	►	**P0602**	BM_u_PPosSetSpeed1	**P0602**	►	6081$_{hex}$	UU - ratio added.
	x = 0 .. 2^{32}-1	►	y = x		x = 1 .. 13200	►	y = x	
profile acceleration	6083$_{hex}$	►	**P0603**	PPosAcceleraton1	**P0603**	►	6083$_{hex}$	UU - ratio added.
	x = 0 .. 2^{32}-1	►	y = x		x = 25 .. 45000	►	y = x	
profile deceleration	6084$_{hex}$	►	**P0604**	PPosDeceleraton1	**P0604**	►	6084$_{hex}$	
	x = 0 .. 2^{32}-1	►	y = x		x = 25 .. 45000	►	y = x	
quick_stop_deceleration	6085$_{hex}$	►	**P1213**	PPosStopDeceleraton	**P1213**	►	6085$_{hex}$	UU - ratio added.
	x = 0 .. 2^{32}-1	►	y = x		x = 25 .. 45000	►	y = x	
motion profile type	6086$_{hex}$	►	**P1190**	PPosMode	**P1190**	►	6086$_{hex}$	
	x = -2^{16}... 2^{16}-1	►			x = 0.. FFFF$_{hex}$			
Manufacturer specific	x = -32768..1			not used				
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 1 0: sin² 0 1: 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				
For future profile type	x = 4..32767			not used				

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res-caling	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 res- caling	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 ..-13	
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		Moving to pos. limit switch	x = -5	►	y = -5	
Manufacturer specific	x = -4	►	y = -4		Moving to 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		Moving to the neg. limit switch with encoder zero angle or zero pulse reference	x = 1	►	y = 1	
Homing on the pos. limit switch	x = 2	►	y = 2		Moving to the pos. limit switch with encoder zero angle or zero pulse reference	x = 2	►	y = 2	
Homing on the positive home switch & index pulse	x = 3	►	y = 3		Reaching of pos. zero point switch with encoder zero angle or zero pulse reference	x = 3	►	y = 3	
Homing on the positive home switch & index pulse	x = 4	►	y = 4		Reaching of pos. zero point switch with encoder zero angle or zero pulse reference	x = 4	►	y = 4	
Homing on the negative home switch & index pulse	x = 5	►	y = 5		Reaching of neg. zero point switch with zero encoder angle or zero pulse referenc-ing	x = 5	►	y = 5	

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res- caling	Comment
Homing on the negative home switch & index **pulse**	x = 6	►	y = 6		Reaching of neg. zero point switch with zero encoder angle or zero pulse referenc-ing	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 direction	x = 7		y = 7	
zero point switch, to the right of 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 point 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 point 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 point switch, right to neg. edge with zero pulse; CCW move	x = 11		y = 11		Zero point switch, on the right of neg. edge with zero pulse; counter-clockwise rotation	x = 11		y = 11	
zero point switch, right to pos. edge with zero pulse; CCW move	x = 12		y = 12		Zero point switch, on the right of pos. edge with zero pulse; counter-clockwise rotation	x = 12		y = 12	
zero point switch, left. to neg. edge with zero pulse; CCW move	x = 13		y = 13		Zero point switch, on the left of neg. edge with zero pulse; counter-clockwise rotation	x = 13		y = 13	
zero point switch, right to neg. edge with zero pulse; CCW move	x = 14		y = 14		Zero point switch, on 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 res-caling**	**Comment**
Negative zero reference switch, CCW move	x = 22	►	y = 22		negative zero point switch; counter-clockwise rotation	x = 22	►	y = 22	
zero point switch, left. to pos. edge; CW move	x = 23	►	y = 23		Zero point switch, on the left of pos. clockwise rotation	x = 23	►	y = 23	
zero point switch, right to pos. edge; CW move	x = 24	►	y = 24		Zero point switch, to the right of pos. edge; clockwise rota-tion	x = 24	►	y = 24	
zero point switch, left. to neg. edge; CW move	x = 25	►	y = 25		Zero point switch, counter-clockwise neg. edge; clock-wise rotation	x = 25	►	y = 25	
Zero reference cam switch, right to neg. edge; CW move	x = 26	►	y = 26		Zero point switch, clockwise 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, on the right of neg. edge; counter-clock-wise rotation	x = 27	►	y = 27	
Zero reference cam switch, left to neg. edge; CCW move	x = 28	►	y = 28		Zero point switch; on the left of neg. edge; counter-clock-wise rotation	x = 28	►	y = 28	
Zero reference cam switch, right to pos. edge; CCW move	x = 29	►	y = 29		Zero point switch, on the right of pos. edge;counter-clock-wise rotation	x = 29	►	y = 29	
Zero reference cam switch, left to pos. edge; CCW move	x = 30	►	y = 30		Zero point switch, on the left of pos. edge; counter-clock-wise 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 clock-wise 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 res- caling	Comment
homing_speeds	6099$_{hex}$				(ref. speed)			6099$_{hex}$	
Speed_during_search_for_switch	Sub. 01$_{hex}$	►	**P1201**		PPosHomingSpeed	**P1201**	►	Sub. 01$_{hex}$	
	x = 0 .. 2^{32}-1	►	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}-1	►	y = x			x = 1..50	►	y = x	
homing_acceleration	609A$_{hex}$	►	**P1203**		PPosHomingAcceler (homing acceleration)	**P1203**	►	609A$_{hex}$	
	x = 0 .. 2^{32}-1	►	y = x			x = 25 .. 45000	►	y = x	
position_control_parameter_set	60FB$_{hex}$								Manufacturer-specific CANopen object

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res- caling	Comment
	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	
	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}-1	►	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	

CANopen object	Index Value range	►	P. no. Scaling		Controller parameters	P. no. Value range	►	Index res-caling	Comment
	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	
	Sub. 11_{hex} /ro				PPosSpeedSetValue	**P0464**	►	Sub. 11_{hex}	
						x = -32768 .. 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}-1	►	y = x			x = 0 .. 2^{32}-1	►	y = x	
	Sub. 18_{hex}	►	**P1209**		PPosEncoderOffset	**P1209**	►	Sub. 18_{hex}	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	y = x	
	Sub. 19_{hex}	►	**P1209**		PPosHomingDeceler	**P1204**	►	Sub. 19_{hex}	
	x = 0 .. 2^{16}-1	►	y = x			x = 0 .. 2^{16}-1	►	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 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 res- caling	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 Motor temperature has exceeded threshold 2 Bit 2 **P0262** bit 1 Power unit tempera-ture > 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
						$x = 0 .. 2^{16}-1$	►		
digital_inputs	$60FD_{hex}$ /ro	►	**P1208**		DigInOutStatus	**P1208**	►	$60FD_{hex}$	
	$x = 0 .. 2^{16}-1$					$x = 0 .. FFFF_{hex}$	►	$y = 0 .. 2^{32}-1$	
Negative limit switch					Status limit switch pos.	Bit 0	►	Bit 1	
Positive limit switch					Status limit switch neg.	Bit 1	►	Bit 0	
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 con-figuration 1 bit 2 = 1: Specification of actual speed in 1/10 RPM. e. g.: 200.0 revolutions ⇒ input 2000.
	$x = -2^{31} .. 2^{31}-1$	►	y = x			$x = 8000_{hex}$ .. $7FFF_{hex}$	►	y = x	
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 res- caling	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	

D

APPENDIX D - TECHNICAL DATA

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

D.1 CoE-option card: technical features

CPU	SH3
FPGA	XC35400 of the SpartanII series (Fa. XILINX)
Memory	512 kByte DPRAM, 8 MByte SDRAM, 8 MByte Flash-Eprom
Baud rate	100 Mbit
Operating voltage	+5 V internal
Plug-in connector	2 RJ45 sockets, 8-pin

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

For the data transmission of b maXX ®-controller to the option module CoE slave there are three channels:

- Two process data channels (1 PDO 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. (exception string parameter).

Table of figures

Index

Revision survey

Version	Status	Changes
5.07017.01	08.11.2007	First edition
5.07017.02	14.04.2011	Section 3.2.6 Ethernet over EtherCAT (EoE) - TCP/IP- tunneling over EtherCAT completed.
5.07017.03	14.08.2014	Section 3.2.6: revised

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**