Application Manual

Language	**English**
	Translation
Document No.	5.14006.06
Part No.	00450924
Status	2019-07-26

be in motion be in motion

BAUMÜLLER

b maXX

CoE and POWERLINK for b maXX 2500 / 3300 / 5000 and CANopen for b maXX 3300 / 5000

E	5.14006.06

Read the Operating Handbook 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

This Application Manual is a part of the equipment/machine. This Application 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, this Application 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 Application Manual is handed over, corresponding sets of application manuals 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 this Application 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

GENERAL INFORMATION

1.1 Information about the application manual CANopen / CoE / POWERLINK for b maXX® 2500 / 3300 / 5000

The application manual provides important information regarding handling the device. A prerequisite for safe working is compliance with all specified safety information and handling instructions.

Furthermore, the local accident prevention regulations and general safety requirements applicable to the area of application of the device must be observed.

Before starting any work on the device, completely read through the Instruction Handbook, in particular the chapter on safety information. The Instruction Handbook is an integral part of the product and must be kept in the immediate vicinity of the device in order to be accessible to personnel at all times.

For commissioning of the device the parameter manual must be used. The parameter manual contains information to the parameters of the device.

The application manual CANopen / CoE / POWERLINK provides information about the configuration and commissioning in a CANopen, EtherCAT or POWERLINK network of b maXX 2500 / 3300 / 5000 devices for controller firmware from version 01.08.

1.2 Explanation of symbols

Warnings

Warnings are identified by symbols in this Parameter Manual. The notices are introduced by signal words which express the magnitude of the danger.

Observe the notices without exception and exercise caution to prevent accidents, personal injury and damage to property.

DANGER!

....warns of an imminently dangerous situation which will result in death or serious injury if not avoided.

WARNING!

....warns of a potentially dangerous situation which may result in death or serious injury if not avoided.

CAUTION!

....warns of a potentially dangerous situation which may result in minor or slight injury if not avoided.

NOTICE!

....warns of a potentially dangerous situation which may result in material damage if not avoided.

Recommendations

NOTE!

....points out useful tips and recommendations, as well as information for efficient, trouble-free operation.

1.3 Limitation of liability

All specifications and information have been compiled taking account of the applicable standards and regulations, the state of the art and also our many years of expertise and experience.

The manufacturer accepts no liability for damage resulting from:

- Non-compliance with the Operating Manual
- Non-intended use
- Use of untrained personnel

The product actually supplied may deviate from the versions and illustrations described here in the case of special versions, the use of additional ordering options or as a result of the latest technical changes.

The user is responsible for carrying out servicing and maintenance in accordance with the safety regulations in the applicable standards and all other relevant national or local regulations concerning conductor dimensioning and protection, grounding, isolation switches, overcurrent protection, etc.

The person who carried out the assembly or installation is liable for damage arising during assembly or upon connection.

1.4 Copyright

Treat the Parameter Manual confidentially. It is intended exclusively for persons involved with the device. It must not be made available to third parties without the written permission of the manufacturer.

NOTE!

The details, text, drawings, pictures and other illustrations contained within are copyright protected and are subject to industrial property rights. Any improper exploitation is liable to prosecution.

CiA® and **CANopen®**	is a registered trademark of CAN in Automation e.V. 90429 Nürnberg, Germany
EtherCAT®	is a registered trademark of Beckhoff Automation GmbH, 33415 Verl, Germany
POWERLINK®	is a registered trademark of ETHERNET POWERLINK STANDARDIZATION GROUP, 15370 Fredersdorf, Germany
b maXX®	is a registered trademark of Baumüller Nürnberg GmbH, 90482 Nürnberg, Germany

NOTE!

Please note, that BAUMÜLLER is not responsible to examine whether any (industrial property) rights of third parties are infringed by the application-specific use of the BAUMÜLLER products/components or the execution.

1.5 Other applicable documents

Manual basic unit b maXX 3300 (5.11018) or Manual b maXX 5000 (5.09021) and

Parameter manual b maXX 3300 (5.12001) or Parameter manual b maXX 5000 (5.09022) in the current version at each case.

1.6 Guarantee conditions

The guarantee conditions are located as a separate document in the sales documents.

Operation of the devices described here in accordance with the stated methods/ procedures / requirements is permissible. Anything else, e.g. even the operation of devices in installed positions that are not shown here, is not permissible and must be checked with the factory in each individual case. If the devices are operated differently than described here, any guarantee will be invalidated.

1.7 Customer service

Our customer service department is available for technical information.

Information concerning the responsible contact person can be obtained at any time by telephone, fax, e-mail or over the internet.

1.8 Terms used

For abbreviations used, see ▷Appendix A - Abbreviations◁ from page 133.

2

FUNDAMENTAL SAFETY INSTRUCTIONS

In this chapter the dangers are prescribed, which can arise during parameterization of the Baumüller b maXX 3300 or b maXX 5000 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

OBJECT DIRECTORY

The main element of a device profile is the object directory. The basis for this is the CANopen object directory:

Index (hex)	Object
0x0000	Not used
0x0001 - 0x001F	Static data types
0x0020 - 0x003F	Complex data types
0x0040 - 0x005F	Manufacturer specific complex data types
0x0060 - 0x007F	Device profile specific static data types
0x0080 - 0x009F	Device profile specific complex data types
0x00A0 - 0x03FF	Reserved
0x0400 - 0x041F	POWERLINK specific static data types
0x0420 - 0x04FF	POWERLINK specific complex data types
0x0500 - 0x0FFF	Reserved
0x1000 - 0x1FFF	Communication profile area
0x2000 - 0x5FFF	Manufacturer specific profile area
0x6000 - 0x9FFF	Standardized device profile area
0xA000 - 0xBFFF	Standardized interface profile area
0xC000 - 0xFFFF	Reserved

The objects are always addressed by means of an index (16 bit) and additionally a sub-index (8 bit).

3.1 Communication profile

Different communication objects are used in parts for the different bus systems. There are additional objects for devices with double axis.

Index	Description	CANopen	CoE	Powerlink
0x1000	Device type	X	X	X
0x1001	Error register	X	X	X
0x1002	Vendor status register	X		
0x1005	COB-ID SYNC	X		
0x1006	Communication cycle period	X		X
0x1007	Synchronous window length	X		
0x1008	Device name	X	X	X
0x1009	Hardware version	X	X	X
0x100A	Software version	X	X	X
0x100C	Guard time	X		
0x100D	Life time factor	X		
0x1017	Producer heartbeat time	X		
0x1018	Identity	X	X	X
0x1020	Verify configuration			X
0x1030	Interface group			X
0x1050	Relative Latency Difference			X
0x1300	Sequence layer timeout			X
0x1400	Rx communication parameter 0 axis 1	X		X
0x1401	Rx communication parameter 1 axis 1	X		
0x1402	Rx communication parameter 2 axis 1	X		
0x1403	Rx communication parameter 3 axis 1	X		
0x1440	Rx communication parameter 0 axis 2	X		
0x1441	Rx communication parameter 1 axis 2	X		
0x1442	Rx communication parameter 2 axis 2	X		
0x1443	Rx communication parameter 3 axis 2	X		
0x1600	Rx mapping parameter 0 axis 1	X	X	X
0x1601	Rx mapping parameter 1 axis 1	X		
0x1602	Rx mapping parameter 2 axis 1	X		
0x1603	Rx mapping parameter 3 axis 1	X		
0x1640	Rx mapping parameter 0 axis 2	X		
0x1641	Rx mapping parameter 1 axis 2	X		
0x1642	Rx mapping parameter 2 axis 2	X		

Index	Description	CANopen	CoE	Powerlink
0x1643	Rx mapping parameter 3 axis 2	X		
0x1700	Rx mapping parameter 1 axis 2		X	
0x1800	Tx communication parameter 0 axis 1	X		X
0x1801	Tx communication parameter 1 axis 1	X		
0x1802	Tx communication parameter 2 axis 1	X		
0x1803	Tx communication parameter 3 axis 1	X		
0x1840	Tx communication parameter 0 axis 2	X		
0x1841	Tx communication parameter 1 axis 2	X		
0x1842	Tx communication parameter 2 axis 2	X		
0x1843	Tx communication parameter 3 axis 2	X		
0x1A00	Rx mapping parameter 0 axis 1	X	X	X
0x1A01	Rx mapping parameter 1 axis 1	X		
0x1A02	Rx mapping parameter 2 axis 1	X		
0x1A03	Rx mapping parameter 3 axis 1	X		
0x1A40	Rx mapping parameter 0 axis 2	X		
0x1A41	Rx mapping parameter 1 axis 2	X		
0x1A42	Rx mapping parameter 2 axis 2	X		
0x1A43	Rx mapping parameter 3 axis 2	X		
0x1B00	Rx mapping parameter 0 axis 2		X	
0x1C00	Sync manager type		X	
0x1C02	Cycle diagnosis		X	
0x1C0B	CN loss of SoC			X
0x1C0D	CN loss of PReq			X
0x1C0F	CN CRC error			X
0x1C12	Number of assigned RxPDOs	X	X	
0x1C13	Number of assigned TxPDOs	X	X	
0x1C14	CN loss of SoC tolerance			X
0x1C32	Sync manager output parameter		X	
0x1C33	Sync manager input parameter		X	
0x1E40	IP Address Table			X
0x1E4A	IP Group			X
0x1F81	Node Assignment			X
0x1F82	Feature flags			X
0x1F83	EPL version			X
0x1F8C	Current NMT state			X

Index	Description	CANopen	CoE	Powerlink
0x1F8D	PRes Payload Limit List			X
0x1F93	EPL Node ID			X
0x1F98	Cycle timing			X
0x1F99	CN basic Ethernet timeout			X
0x1F9A	Host Name			
0x1F9E	Reset Command			X

3.2 CANopen device profile CiA® 402

CANopen, CoE (CAN application protocol over EtherCAT) and POWERLINK support the CANopen device profile CiA® 402 (Device Profile for Drives and Motion Control).

The following operation modes are supported.

3.2.1 Short summery

The following operation modes are supported, i.e. all prescribed objects can be found.	
Device Control	optional objects not available completely
Homing Mode	optional objects completely available
Profile Position Mode	optional objects not available completely
Position Control Function	optional objects not available completely
Profile Velocity Mode	optional objects not available completely
Common Entries in the Object Dictionary (no prescribed objects available)	optional objects not available completely
Velocity Mode	optional objects not available completely
Cyclic Synchronous Position Mode	recommended objects not available completely
Cyclic Synchronous Velocity Mode	recommended objects not available completely
Cyclic Synchronous Torque Mode	recommended objects not available completely
Factor Group	optional objects completely available

The following operation modes are not supported, i.e. at least one prescribed object is not available and optional objects may be available.	
Profile Torque Mode	three objects
Interpolated Position Mode	no objects

3.2.2 Operating modes and field bus objects

Operating mode		
Fieldbus object number	mandatory / optional	Fieldbus object name

Device Control all prescribed and partly all optional objects are supported		
0x6040	mandatory	Controlword
0x6041	mandatory	Statusword
0x605A	optional	Quick stopp option code
0x605B	optional	Shutdown option code
0x605C	optional	Disable operation option code
0x6060	mandatory	Modes of operation
0x6061	mandatory	Modes of operation display

Homing Mode all prescribed and all optional objects are supported		
0x607C	optional	Home offset
0x6098	mandatory	Homing method
0x6099 SIX 0 = 2	mandatory	Homing speed
0x609A	optional	Homing acceleration
0x60B8	optional	Touch probe function
0x60B9	optional	Touch probe status
0x60BA	optional	Touch probe pos 1 pos value
0x60BB	optional	Touch probe pos 1 neg value
0x60BC	optional	Touch probe pos 2 pos value
0x60BD	optional	Touch probe pos 2 neg value

Profile Position Mode all prescribed and partly all optional objects are supported		
0x607A	mandatory	Target position
0x607D SIX 0 = 2	optional	Software position limit
0x6080	optional	Max motor speed
0x6081	mandatory	Profile velocitiy
0x6083	mandatory	Profile acceleration
0x6084	mandatory	Profile deceleration
0x6085	optional	Quick stop deceleration
0x6086	mandatory	Motion profile type

Position Control Function all prescribed and partly all optional objects are supported		
0x6063	optional	Position actual internal value
0x6064	mandatory	Position actual value
0x6067	optional	Position window
0x6068	optional	Position window time
0x60FB SIX 0=25	optional	Position control parameter set

Profile Velocity Mode all prescribed and partly all optional objects are supported		
0x6069	mandatory	Velocity sensor actual value
0x606A	mandatory	Sensor selection code
0x606B	mandatory	Velocity demand value
0x606C	mandatory	Velocity actual value
0x60FF	mandatory	Target velocity

Common Entries in the Object Dictionary no prescribed objects available, partly all optional objects are supported		
0x6007	optional	Abort connection option code
0x603F	optional	Error code
0x6502	optional	Supported drive modes
0x6510 SIX 0= 13	optional	Drive date

Velocity Mode all prescribed and partly all optional objects are supported		
0x6042	mandatory	vl target velocity
0x6043	mandatory	vl velocity demand
0x6044	mandatory	vl control effort
0x6046 SIX 0 = 2	mandatory	vl velocity min max amount
0x604F	optional	vl ramp function time
0x6050	optional	vl slow down time

Profile Torque Mode one prescribed and two optional objects are supported		
0x6071	mandatory	Target torque
0x6072	optional	Max torque
0x6077	optional	Torque actual value

Cyclic Synchronous Position Mode all prescribed and partly all recommended objects are supported		
0x6064	mandatory	Position actual value
0x6077	recommended	Torque actual value
0x607A	mandatory	Target position
0x607D SIX 0 = 2	recommended	Software position limit
0x60B1	recommended	Velocity offset
0x60F4	recommended	Following error actual value

Cyclic Synchronous Velocity Mode all prescribed and partly all recommended objects are supported		
0x6064	mandatory	Position actual value
0x606C	recommended	Velocity actual value
0x6077	recommended	Torque actual value
0x60FF	mandatory	Target velocity

Cyclic Synchronous Torque Mode all prescribed and partly all recommended objects are supported		
0x6064	recommended	Position actual value
0x6071	mandatory	Target torque
0x6077	mandatory	Torque actual value
0x60B2	recommended	Torque offset

Factor Group (group of the user units) no prescribed objects available, all recommended objects are supported		
0x607E	optional	Polarity
0x608F SIX 0 = 2	optional	Position encoder resolution
0x6090 SIX 0 = 2	optional	Velocity encoder resolution
0x6091 SIX 0 = 2	optional	Gear ratio
0x6092 SIX 0 = 2	optional	Feed constant

3.3 Manufacturer-specific objects

The access to the manufacturer-specific objects of the b maXX 2500 / 3300 / 5000 is made by the 0x2000 - 0x3FFF range for the axis 1 as well as by the 0x4000 - 0x5FFF range for the axis 2 (b maXX 5000, only).

The assignment of the CANopen object number and the controller parameter number is described in appendix B.

3.3.1 Array parameter

The addressing by a CANopen index and subindex is inadequate for the array parameters of the b maXX controller. Therefore, additional manufacturer-specific objects were defined to make the access to these parameters possible.

0x2120
Error number

Subindex 0x01 … 0x14

Display of the error numbers from controller parameter ▹100.3◃

Index	Subindex	Data type	Access	Description
0x2120	0x00	UINT8	RO	Number of subindices
	0x01	UINT32	RO	Error number 0
	0x02	UINT32	RO	Error number 1
	0x03	UINT32	RO	Error number 2
	0x04	UINT32	RO	Error number 3
	0x05	UINT32	RO	Error number 4
	0x06	UINT32	RO	Error number 5
	0x07	UINT32	RO	Error number 6
	0x08	UINT32	RO	Error number 7
	0x09	UINT32	RO	Error number 8
	0x0A	UINT32	RO	Error number 9
	0x0B	UINT32	RO	Error number 10
	0x0C	UINT32	RO	Error number 11
	0x0D	UINT32	RO	Error number 12
	0x0E	UINT32	RO	Error number 13
	0x0F	UINT32	RO	Error number 14
	0x10	UINT32	RO	Error number 15
	0x11	UINT32	RO	Error number 16
	0x12	UINT32	RO	Error number 17
	0x13	UINT32	RO	Error number 18
	0x14	UINT32	RO	Error number 19

The subindices of object 0x2120 are only readable.

0x2121
Error reactions

Error reaction according to controller error numbers from controller parameter ▹100.4◃

Index	Subindex	Data type	Access	Description
0x2121	0x00	UINT8	RO	Number of subindices
	0x01	UINT32	RW	Error code
	0x02	UINT32	RO	Confirmation error code
	0x03	INT32	RW	Error reaction

In subindex 0x01 the error number is entered which is to be changed and in subindex 0x03 the required error reaction. After writing object 0x2121.03 the error reactions are immediately updated in the controller.

Object 0x2121.02 serves as the acknowledgment of a correct error input in object 0x2121.01 and can be read, only. If the error number in the controller is unknown then 0 is the value in object 0x2121.02.

0x2125
Positioning sets

Positioning sets according to controller parameter ▹118.19◃

Index	Subindex	Data type	Access	Description
0x2125	0x00	UINT8	RO	Number of subindices
	0x01	UINT32	RW	Positioning set
	0x02	UINT32	RW	Target position
	0x03	UINT16	RW	Target mode
	0x04	UINT32	RW	Speed
	0x05	UINT32	RW	Acceleration
	0x06	UINT32	RW	Deceleration
	0x07	UINT32	RW	Jerk
	0x08	UINT16	RW	Smoothing time
	0x09	INT32	RW	Relative target position

In subindex 0x01 the positioning set 1 to 16 can be selected. The positioning set 0 is the active positioning set and cannot be selected.

By the writing of subindex 0x02 to 0x09 the data of the selected positioning sets can be changed.

3.3.2 File Transfer over EtherCAT

The b maXX 2500 / 3300 / 5000 CoE slave support the file transfer via EtherCAT (FoE, the transmission of files via the fieldbus EtherCAT). This makes firmware updates or the installation of data sets possible. The precondition for this is that the used EtherCAT master supports this function (FoE) as well. The activation of the firmware update is made by object 0x2300.

0x2300
File transfer

Subindex	Name	Data type	Access	Description
1	FoE file name	STRING	RO	FoE file name
2	FoE file length	UINT32	RO	FoE file length
3	Command	UINT16	RW	0 – Abort of the transfer or initialization of the internal state machine 1 – Saving a file in Baumüller FW Format (incl. data sets) in Flash 2 – Saving a file in Baumüller FW Format (incl. data sets) in RAM
4	Status	UINT32	RO	RC_NO_ERROR = 0 (no error) RC_BUSY = 1 (waiting state) RC_ERROR = 2 (general error) RC_DONE = 3 (process / command finished)
5	RC error	UINT32	RO	Baumüller RC error message
6	Extended info	UINT32	RO	For commands 1 to 2 – Send state in %
7	File Options	UINT32		Reserved
8	File Type	UINT32		Reserved
9	Axis Unit	UINT32		For BM5800 only. 1-8: Axis module (AM); 0xFF: SE
10	File Source/ Destination	UINT32		Reserved

4

COMMUNICATION TO THE B MAXX CONTROLLER

In this chapter the data communication between the fieldbus slave and the b maXX 2500 / 3300 / 5000 controller is described.

4.1 Communication flow

The fieldbus slave exchanges via a Dual-Port-RAM data with the b maXX 2500 / 3300 / 5000 controller. This data exchange is made with a defined time pattern.

The fieldbus slave activates the communication with the b maXX 2500 / 3300 / 5000 controller. During communication, two different types of data are transferred:

- Process data
- Service data

Process data is always transferred cyclically. In the remaining time, service data is transferred. The transmission of the process data is made in a settable time pattern (fieldbus cycle time), transmitting the reference values and the actual values with an offset to the interval of the fieldbus. The cycle time of the SoC frame must be in accordance with the fieldbus cycle time (▷131.18◁).

4.2 Parameterizing the fieldbus communication times

Between the fieldbus slave and the b maXX controller 16 set values and 16 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 fieldbus master, see ▹Data Exchange and Parameterization◃ on page 112.

The setting of communication times between fieldbus slave and b maXX controller is automatically set. The fieldbus cycle time of the controller in parameter 131.18 is set by means of the cycle time in object 0x1006 (CANopen, POWERLINK) or in object 0x1C32 SIX 2 (CoE)

If distributed clock is activated at EtherCAT, the cycle time is transferred from the distributed clock time.

The b maXX controller initiates a communication time slot in each fieldbus cycle, in which process data set values or process data actual values are transferred.

The process date set values and the process data actual values are transmitted in the same communication time slots. The transmission occurs with a time offset to the synchronization. This Sync offset is set in parameter 156.4 for set values and actual values together. The Sync offset is set automatically to the half cycle time.

Cycle time (0x1006)	**Cycle time (0x1C32 SIX 2)**	**Fieldbus cycle time (131.18)**	**Sync Offset (156.4)**
500 µs	500000 ns	500000 ns	250 µs
1000 µs	1000000 ns	1000000 ns	500 µs
2000 µs	2000000 ns	2000000 ns	1000 µs
4000 µs	4000000 ns	4000000 ns	2000 µs
8000 µs	8000000 ns	8000000 ns	4000 µs

The synchronization in the controller must be activated additionally to synchronize the controller with an external signal, e.g. the synchronization signal of the fieldbus master. Therefore bit 0 must be set in parameter 156.1 Synchronization mode.

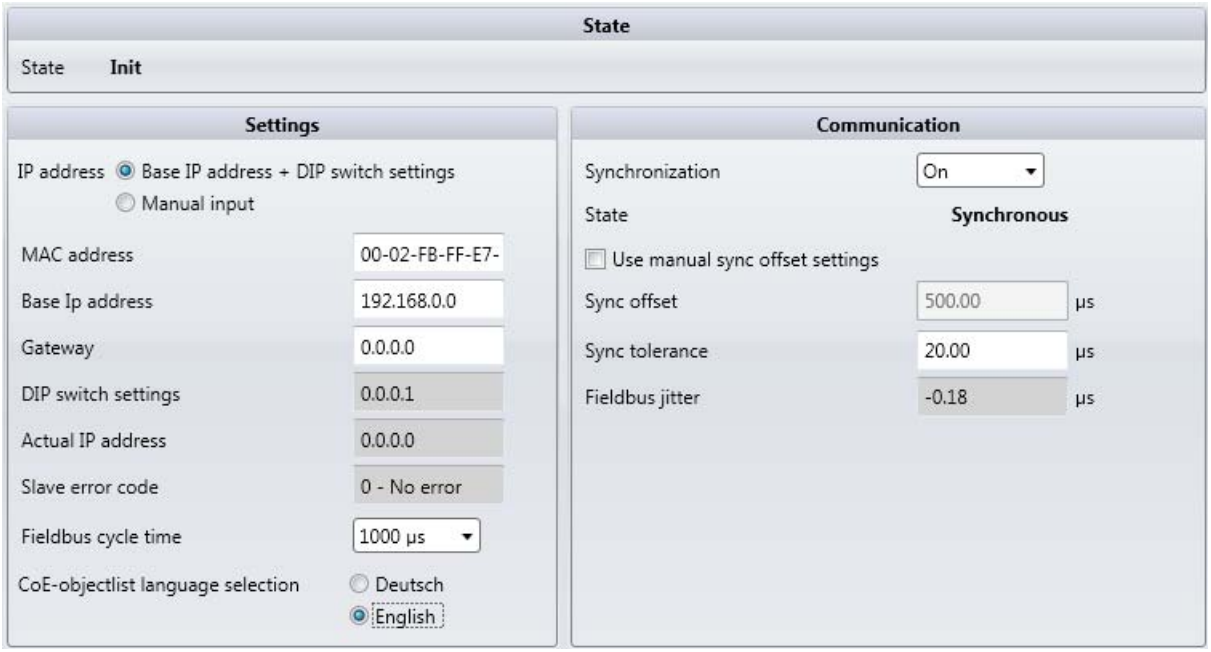


Figure 1: ProDrive fieldbus slave

If the transmission of the actual value between fieldbus and controller fails more than two fieldbus cycles, error number 1937 is set. If two set values are missing, error number 1938 is set.

The Sync offset also can be defined manually. Here bit 1 must be set in parameter 156.1.

The settings must be stored in the data set of the b maXX controller and the controller must be booted again.

NOTE!

The synchronization in parameter 156.1 must be switched on for the synchronization with an fieldbus signal.

NOTE!

If cyclic communication is interrupted, e. g. at restart of the bus the error 1937 or 1938 can occur.

5

CONFIGURATION POSSIBILITIES OF THE FIELDBUS SLAVE

The behavior of the fieldbus slave can be changed by changes in the slave settings in parameter 131.9.

NOTE!

Settings result in a modified behavior!

5.1 Network settings for EoE (Ethernet over EtherCAT)

The setting of the IP address for EtherCAT can occur via DIP switches at the device or via b maXX parameters.

Bit 0

0 IP address = base IP address + DIP switch (131.12 + 131.13),
Sub net mask = 255.255.0.0

1 Read out of the network settings for parameter 131.14, 131.16, 131.17.

5.2 Select a language online CoE object directory

At CoE the object directory can be recalled online by the master. The language can be switched between German and English in the slave settings.

Bit 1

0 German

1 English

5.3 Factor Group according to CiA® 402

(from Firmware version 01.09)

The calculation is done in the b maXX controller using the Factor Group (please refer as well to BM3000 Parameter Manual 5.12001 or BM5000 Parameter Manual 5.09022). The calculation must be enabled in the slave settings in parameter 131.9.

Bit 14

0 No calculation according to CiA® 402 Factor Group

1 Calculation according to CiA® 402 Factor Group

The Factor Group has effects on the following objects:

Object number	Standardization using Factor Group object
0x6064	0x608F, 0x6091, 0x6092
0x6067	0x608F, 0x6091, 0x6092
0x606C	0x607E, 0x6090, 0x6091, 0x6092
0x607A	0x607E, 0x608F, 0x6091, 0x6092
0x607C	0x608F, 0x6091, 0x6092
0x607D SIX 1	0x607E, 0x608F, 0x6091, 0x6092
0x607D SIX 2	0x607E, 0x608F, 0x6091, 0x6092
0x6081	0x6090, 0x6091, 0x6092
0x6083	0x6090, 0x6091, 0x6092
0x6084	0x6090, 0x6091, 0x6092
0x6085	0x6090, 0x6091, 0x6092
0x6099 SIX 1	0x6090, 0x6091, 0x6092
0x6099 SIX 2	0x6090, 0x6091, 0x6092
0x609A	0x6090, 0x6091, 0x6092
0x60B1	0x607E, 0x6090, 0x6091, 0x6092
0x60BA	0x608F, 0x6091, 0x6092
0x60BB	0x608F, 0x6091, 0x6092
0x60BC	0x608F, 0x6091, 0x6092
0x60BD	0x608F, 0x6091, 0x6092
0x60F4	0x608F, 0x6091, 0x6092
0x60FF	0x607E, 0x6090, 0x6091, 0x6092

$$\text{Position encoder resolution (0x608F)} = \frac{\text{Encoder increments (0x608F.01)}}{\text{Motor revolutions (0x608F.02)}}$$

$$\text{Velocity encoder resolution (0x6090)} = \frac{\text{Encoder increments per second (0x6090.01)}}{\text{Motor revolutions per second (0x6090.02)}}$$

$$\text{Gear ratio (0x6091)} = \frac{\text{Motor revolutions (0x6091.01)}}{\text{Shaft revolutions (0x6091.02)}}$$

$$\text{Feed constant (0x6092)} = \frac{\text{Feed (0x6092.01)}}{\text{Shaft revolutions (0x6092.02)}}$$

Polarity (0x607E): The position set value should be multiplied by -1, if the polarity bit is set. The polarity bit has no influence on the Homing mode.

Bit 7	Bit 6	Bit 0 - 5
Position polarity	Velocity polarity	reserved

The position polarity bit should be used only for Profile Position Mode (target position setting) and Cyclic Sync Position Mode (position control). The velocity polarity bit should be used only for Profile Velocity Mode (speed control) and Cyclic Sync Velocity Mode (speed control).

All values of the Factor Group are dimensionless.

The standardization occurs as follows:

Writing values from fieldbus master to slave:

$$\text{Position values internal} = \frac{\text{Position values} \cdot \text{Polarity (0x607E)} \cdot \text{Position encoder resolution (0x608F)} \cdot \text{Gear ratio (0x6091)}}{\text{Feed constant (0x6092)}}$$

$$\text{Speed values internal} = \frac{\text{Speed values} \cdot \text{Polarity (0x607E)} \cdot \text{Velocity encoder resolution (0x6090)} \cdot \text{Gear ratio (0x6091)}}{\text{Feed constant (0x6092)}}$$

Reading values from slave to fieldbus master:

$$\text{Position values} = \frac{\text{Position values internal} \cdot \text{Feed constant (0x6092)}}{\text{Position encoder resolution (0x608F)} \cdot \text{Gear ratio (0x6091)} \cdot \text{Polarity (0x607E)}}$$

$$\text{Speed values} = \frac{\text{Speed values internal} \cdot \text{Feed constant (0x6092)}}{\text{Velocity encoder resolution (0x6090)} \cdot \text{Gear ratio (0x6091)} \cdot \text{Polarity (0x607E)}}$$

5.4 CANopen offset

Mapping of the numerical scale from UINT32 to INT32 (CANopen mode). On writing/reading on some FB objects, an offset of 2^{31} is added respectively subtracted internally on the fieldbus slave depending on the direction.

Bit 16

0 Conversion of the numerical scale from UINT32 to INT32; depending on the direction, an offset of 2^{31} is added/subtracted to the appropriate fieldbus object during the positioning.

1 1no offset is added/subtracted
The following objects are concerned:
0x6062, 0x6064, 0x607A, 0x607C, 0x607D SIX1, 0x607D SIX2

If the position reference values and the target position shall also be displayed in ProDrive in the INT32 number scale, it is possible to activate a check box for the offset on the page „Scaling“.

5.5 Homing necessary for positioning

In ProDrive you can choose on page „Homing“ with the provided check box, whether the drive permits a positioning, if no first-time homing has been executed.

Deactivated: No homing is necessary for operating in operating mode positioning.

Activated: If the drive is activated in the operating mode ’positioning’ and no homing has been executed before, an error message (controller message no. 900) is displayed and the drive remains position-controlled in the current position. Positioning jobs are not executed. They will only be executed, after a homing has been completed (at least once after switching on). The error message can only be acknowledged, if a homing has been executed. After the homing, the positioning can be started.

NOTE!

A homing is necessary, in case the CANopen mode is defined as standard!

5.6 Types of positioning, depending on the positioning mode (parameter 118.10)

NOTE!

Make sure that also the positioning data set 0 is set in ProDrive under positioning 0, otherwise the positioning will not be effected correctly in the fieldbus. The switching between the positioning modes „relative“, „negative/positive“ and „absolute“ only takes place by means of the control word. A homing should always precede the positioning in the CANopen mode (standard).

There are the following positioning modes:

Positioning mode Parameter 118.10	**Description**
„Absolute/relative“ CANopen (standard value 9)	• Destination is displayed in parameter 118.16 (INT32) • Switching „absolute/relative“ via control word, only
„Relative, positive and negative“ (value 4)	• Destination is displayed in parameter 118.16 (INT32) • no switching „absolute/relative“ via control word
„Absolute relative“ (value 10)	• Destination is displayed in parameter 118.9 (UINT32) • Switching „absolute/relative“ via control word, only
„Absolute with shortest way / relative“ (value 12) (from V01.10)	• Destination is displayed in parameter 118.16 (INT32) • Switching „absolute/relative“ via control word, only • Positioning is made towards the shorter route at an absolute target
„Absolute without range offset / relative“ (value 16) (from V01.10)	• Destination is displayed in parameter 118.9 (UINT32) • Switching „absolute/relative“ via control word, only • no internal range offset about 2^{31} increments at absolute target
„Absolute with shortest way without range offset / relative“ (value 17) (from V01.10)	• Destination is displayed in parameter 118.9 (UINT32) • Switching „absolute/relative“ via control word, only • no internal range offset about 2^{31} increments at absolute target • Positioning is made towards the shorter route at an absolute target
All other modes	• Destination is displayed in parameter 118.9 (UINT32) • no switching „absolute/relative“ via control word • no conversion (data type = UINT32)

Switching „absolute/relative“, via control word Bit 6

Bit 6

0 Absolute

1 Relative

The conversion of data type INT32 ⇔ UINT32 means that an offset of 2^{31} is added or subtracted depending on the direction. This is necessary in order to achieve a standard presentation of the fieldbus object in data type INT, as some controllers parameters are realized for the positioning (see ▹CANopen offset◃ on page 32) as data type UINT. Thus, the existing fieldbus objects are displayed to the user in the positioning as data type INT.

NOTE!

The conversion of the positioning mode to parameter 118.16 „Absolute/relative CANopen“ is not being deactivated.

6

BASICS CAN / CANOPEN

6.1 Literature concerning CAN

On behalf of basic information with reference to CAN the following literature is recommended:

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

- CAN Controller Area Network. Grundlagen und Praxis
  Wolfhard Lawrenz
  Hüthig Verlag

- CANopen
  Holger Zeltwanger
  VDE-Verlag

- CANopen Device Profile for Drives and Motion Control
  CiA Draft Standard CiA® 402
  CAN in Automation (CiA)

- www.can-cia.de
  CAN in Automation (CiA)
  Kontumazgarten 3
  D-90429 Nürnberg

6.2 Basic principles of CAN

A CAN-based field bus system is executed in line structure. A three-wire cable provides the physical basis of data transfer with the connections CAN_High, CAN_Low and CAN_Ground. CAN employs transmissions balanced to ground for the suppression of common mode interference. For this reason, the difference signals are evaluated.

Network

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

Bus access

Bus access takes place using the CSMA/CA procedure (Carrier Sense Multiple Access / Collision Avoidance). Because after detecting the necessary bus quiescence every participant has the right to begin transmitting a message, collisions can occur. This is avoided by means of bit-by-bit arbitration of the messages to be sent. During this, two bus levels are differentiated, a dominant level, logical bit value 0 and a recessive level, logical bit value 1. In the worst case, all participants wishing to transmit simultaneously begin sending their messages onto the bus. If a recessive bit of a participant is overwritten by a dominant bit of another, then the "recessive" node withdraws from the bus and, after detecting bus quiescence, attempts once more to transmit its message. Consequently, it is guaranteed that the most important and highest priority message (with the lowest identifier) will be transmitted in a collision-free manner and without delay. For this reason it is of course necessary that every identifier is permitted to be placed onto the CAN bus only once.

Identifier

Different identifiers are available in accordance with specification CAN 2.0A 2032. Each participant can transmit in an unsolicited manner (multi-master capability). A transmitter sends its message to all CAN nodes (broadcast) and each then decides on the basis of the identifier whether they will continue processing the message or not.

Error

Up to 8 bytes of user data can be transmitted within a CAN data message frame. For error or overload signaling, a CAN node can send error or overload message frames. This occurs on Layer 2 of the OSI/ISO reference model, the Data Link layer, therefore independently of the application. Due to high quality error detection and handling on Layer 2, a Hamming distance (a measure of error detection) of HD=6 is achieved, i. e. a maximum of 5 simultaneously-occurring bit errors within a message frame will be reliably detected as an error.

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 (CiA 301) defines the method of data exchange and general definitions applicable for all devices.

Device profile

Device profiles contain application- and device-specific definitions describing the contents-related meaning of data and device functionality. Device profiles exist for drives, I/O modules, encoders or programmable devices. The CANopen slave for the b maXX® 2500 / 3300 / 5000 controller is implemented in accordance with the device profile CiA 402 (drives and motion control).

CANopen differentiates between 4 types of messages:

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

NMT

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

SDO

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

PDO

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

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

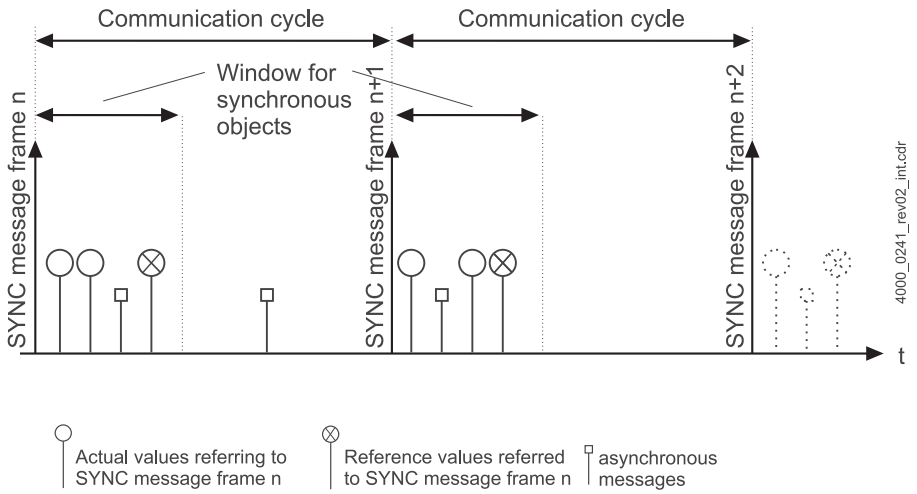


Figure 2: PDO transmission types

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

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

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

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

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

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

- INITIALIZATION
- PRE-OPERATIONAL
- STOPPED
- OPERATIONAL

The individual state transitions are triggered by NMT commands. After initializing, the CANopen slave switches automatically to the PRE-OPERATIONAL state. Additional data is available in ▹Network management (NMT)◃ from page 46.

7

CANopen at b maXX 2500 / 3300 / 5000

7.1 General information

The b maXX 2500 / 3300 / 5000 CANopen slave connects the b maXX® 2500 / 3300 / 5000 via the CAN bus with other CAN nodes (e. g. PC, PLC, further b maXX® devices, I/O modules).

Information according installation and handling with the device series b maXX® 3300 / 5000 is found in the manual 5.11018 / 5.09021.

Information according the programming of the **b maXX 3300 / 5000** controller is found in the parameter manual 5.12001 / 5.09022.

7.2 Address Setting

The node address setting of the b maXX 3300 / 5000 CANopen slave is described in the instruction manual b maXX 3300 / 5000 (5.11018 / 5.09021).

7.3 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 CANopen device. The EDS file is used by the CANopen masters or bus configurators. The EDS file contains information about the supported objects, baud rates and other features of the slave.

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

The file can be downloaded from the download area on Baumüller's home page www.baumueller.de.

7.4 Diagnosis

CANopen is a communication protocol based on CAN and follows the EN 50325-4 (CiA 301) standard. In this way standard tools and devices for analysis can be used for system diagnosis of CAN net works.

7.5 Data Exchange and Parameterization

The access to data or parameter is made always via CANopen objects.

Accordant to profile structure it is differed between objects for communication control (indices 0x1XXX) and user- or device-specific objects. The latter are divided into objects according to profile CiA 402 (indices 0x6XXX) and manufacturer-specific objects (indices 0x02XXX or 0x4XXX).

A listing of the 6XXX and the 2XXX / 4XXX objects are to be found in ▹Appendix B - Quick reference◃ from page 135.

NOTE!

The mapping of the manufacturer-specific parameter of the b maXX 5000 / 3300 / 2500 is specified in ▹B.1 2000 / 4000 object numbers (manufacturer-specific objects)◃ from page 135.

7.6 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 in accordance with CiA 301.

Name	Index	Subindex	Data type	Default value
Device type	0x1000	0x00	UINT32	0x00020192

This object is read-only and contains information on the related device (drive in accordance with CiA® 402).

Name	Index	Subindex	Data type	Default value
Error register	0x1001	0x00	UINT8	0x0

This object can only be read. The object 0x1001 contains an error bit string with the following meaning:

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

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x80 + address	0x08	Emergency error code		Error register	Manufacturer-specific error field				

EMCY telegram for error reset / No error

Name	Index	Subindex	Data type	Default value
Manufacturer status register	0x1002	0x00	UINT32	-

This object can only be read. The low byte contains the controller status word low byte from parameter **108.3**.

Name	Index	Subindex	Data type	Default value
COB ID SYNC message	0x1005	0x00	UINT32	0x80

This object contains information about the SYNC slave behavior. The slave is not a SYNC master, e. g. only SYNC telegrams can be received. The lower 11 bits in the low word specify the identifier of the SYNC telegram (0x80), only readable.

Name	Index	Subindex	Data type	Default value
Communication cycle period	0x1006	0x00	UINT32	0

In case the SYNC telegram is active, the SYNC interval must be set to the time of SYNC telegram (1000 µs, 2000 µs, 4000 µs or 8000 µs). The set time takes effects on the parameters **131.8** (Fieldbus cycle time) and **156.4** (SYNC offset) of the b maXX® controller.

Name	Index	Subindex	Data type	Default value
Synchronous window length	0x1007	0x00	UINT32	0

This object is not evaluated.

Name	Index	Subindex	Data type	Default value
Manufacturer device name	0x1008	0x00	VString	-

This object is read-only. It contains the device name.

Name	Index	Subindex	Data type	Default value
Manufacturer hardware version	0x1009	0x00	VString	-

This object is read-only. It contains the current hardware version of the option module from parameter **102.25**.

Name	Index	Subindex	Data type	Default value
Manufacturer software version	0x100A	0x00	VString	-

This object is read-only. It contains the current software version of the option module, e. g. the character string: „01.08.00 S (Build 109)“.

Name	Index	Subindex	Data type	Default value
Guard time	0x100C	0x00	UINT16	0

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

Name	Index	Subindex	Data type	Default value
Life time factor	0x100D	0x00	UINT8	0

The value of this object is multiplied by object 0x100C and from this the node guarding period results. By writing the value ’0’, node guarding will be deactivated.

Name	Index	Subindex	Data type	Default value
Producer heartbeat time	0x1017	0x00	UINT8	0x03

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

Name	Index	Subindex	Data type	Default value
Identity object	0x1018	0x00	UINT8	0x04
Vendor ID		0x01	UINT32	0x0000015H
Product code		0x02	UINT32	0x26483052
Revision number		0x03	UINT32	0x00000000
Serial Number		0x04	UINT32	0x00000000

This object contains some information on the device.

Name	Index	Subindex	Data type	Default value
RPDO0…3 Communcation Parameter Axis 1	0x1400 ... 0x1403	0x00	UINT8	0x05
COB-ID used by RPDO		0x01	UINT32	(0x200 ... 0x500) + address
Transmission type		0x02	UINT8	0
Inhibit Time		0x03	UINT16	0
Compatibility Entry		0x04	UINT8	0
Event Timer		0x05	UINT16	0

This object contains the information on receive-PDO 0 to 3 for axis 1. The identifier of the receive PDO 0 to 3 for axis 1 is entered in subindex 0x01. Subindex 0x02 contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 0x03. The input value is defined as a multiplier of 100 µs. Subindex 0x04 is not used. Subindex 0x05 is for setting the time of timer triggered transmit PDOs. The resolution is 1 ms.

Name	Index	Subindex	Data type	Default value
RPDO0…3 Communcation Parameter Axis 2	0x1440 ... 0x1443	0x00	UINT8	0x05
COB-ID used by RPDO		0x01	UINT32	0
Transmission type		0x02	UINT8	0
Inhibit Time		0x03	UINT16	0
Compatibility Entry		0x04	UINT8	0
Event Timer		0x05	UINT16	0

This object contains the information on receive-PDO 0 to 3 for axis 2. The identifier of the receive PDO 0 to 3 for axis 2 is entered in subindex 0x01. Subindex 0x02 contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 0x03. The input value is defined as a multiplier of 100 µs. Subindex 0x04 is not used. Subindex 0x05 is for setting the time of timer triggered transmit PDOs. The resolution is 1 ms.

Name	Index	Subindex	Data type	Default value
RPDO0…3 Mapping Axis 1	0x1600 ... 0x1603	0x00	UINT8	0x01
		0x01	UINT32	0x60400010
		:	:	
		n	UINT32	

This object contains the contents of receive PDO 0 to 3 for axis 1. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the set value limit of 16 objects max. (also ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
RPDO0…3 Mapping Axis 2	0x1640 ... 0x1643	0x00	UINT8	0x01
		0x01	UINT32	0x60400010
		:	:	
		n	UINT32	

This object contains the contents of receive PDO 0 to 3 for axis 2. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the set value limit of 16 objects max. (also ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
TPDO0…3 Communcation Parameter Axis 1	0x1800 ... 0x1803	0x00	UINT8	0x06
COB-ID used by TPDO		0x01	UINT32	(0x180 ... 0x480) + address
Transmission type		0x02	UINT8	0
Inhibit time		0x03	UINT16	0
Compatibility Entry		0x04	UINT8	0
Event timer		0x05	UINT16	0
SYNC Start Value		0x06	UINT8	0

This object contains information on transmit PDO 0 to 3 for axis 1. The transmit PDO 0 to 3 for axis 1 identifier is entered in subindex 0x01. Subindex 0x02 contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 0x03. The input value is defined as a multiplier of 100 µs. Subindex 0x04 is not used. Subindex 0x05 is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond. Subindex 0x06 contains the SYNC start time. It is defined with 0, because the counter of the SYNC message is not changed by the PDO.

Name	Index	Subindex	Data type	Default value
TPDO0…3 Communcation Parameter Axis 2	0x1840 ... 0x1843	0x00	UINT8	0x06
COB-ID used by TPDO		0x01	UINT32	0
Transmission type		0x02	UINT8	0
Inhibit time		0x03	UINT16	0
Compatibility Entry		0x04	UINT8	0
Event timer		0x05	UINT16	0
SYNC Start Value		0x06	UINT8	0

This object contains information on transmit PDO 0 to 3 for axis 2. The transmit PDO 0 to 3 for axis 2 identifier is entered in subindex 0x02. Subindex 0x02 contains the trigger type of this PDO. The inhibit time, which represents the minimum delay for a transmission interval, is set in subindex 0x03. The input value is defined as a multiplier of 100 µs. Subindex 0x04 is not used. Subindex 0x05 is for setting the time of timer triggered transmit PDOs. The resolution is 1 millisecond. Subindex 0x06 contains the SYNC start time. It is defined with 0, because the counter of the SYNC message is not changed by the PDO.

Name	Index	Subindex	Data type	Default value
TPDO0…3 Mapping Axis 1	0x1A00 ... 0x1A03	0x00	UINT8	0x01
		0x01	UINT32	0x60410010
		:	:	
		n	UINT32	

This object contains the contents of transmit PDO 0 to 3 for axis 1. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the actual value limit of 16 objects max. (also ▹PDO mapping◃ from page 61).

Name	Index	Subindex	Data type	Default value
TPDO0…3 Mapping Axis 2	0x1A40 ... 0x1A43	0x00	UINT8	0x01
		0x01	UINT32	0x60410010
		:	:	
		n	UINT32	

This object contains the contents of transmit PDO 0 to 3 for axis 2. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the actual value limit of 16 objects max. (also ▹PDO mapping◃ from page 61).

7.7 Network management (NMT)

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

7.7.1 Communication state machine

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

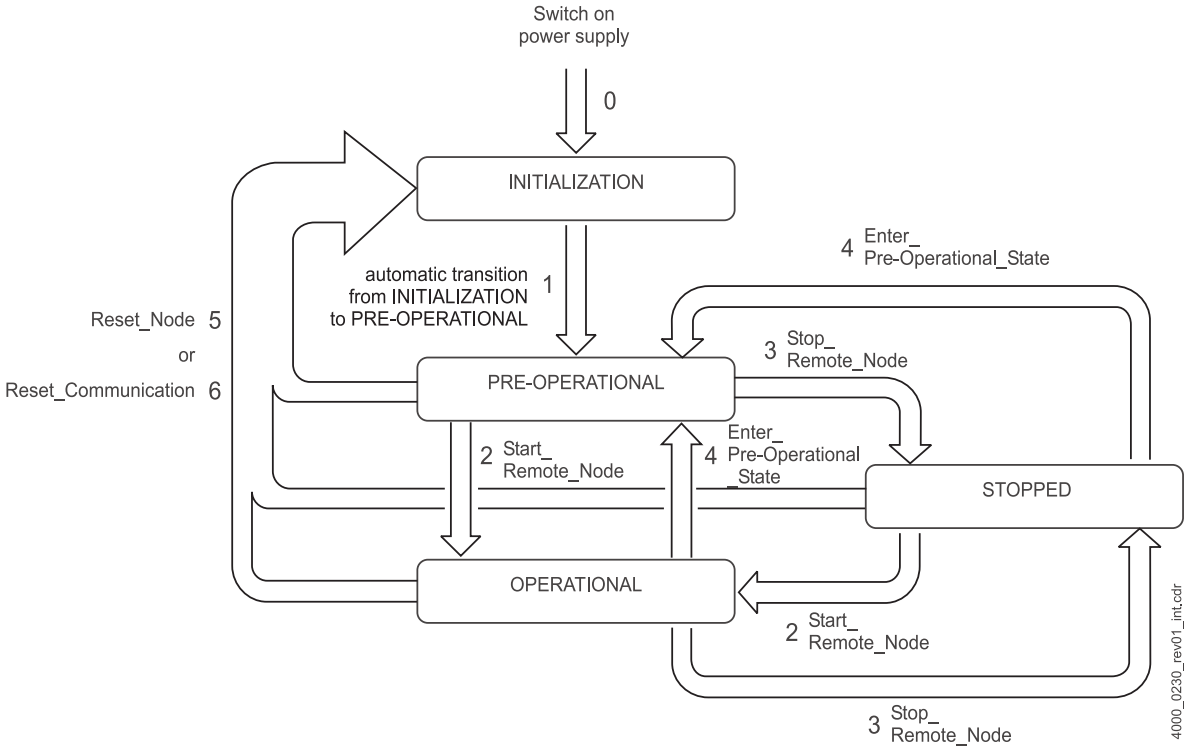


Figure 3: Communication state machine

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

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

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

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

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

2 Start_Remote_Node

3 Stop_Remote_Node

4 Enter_Pre-Operational_State

5 Reset_Node

6 Reset_Communication

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

7.7.2 Telegrams

NMT telegrams for communication control have the default identifier ’0’ in accordance with the predefined connection set (also see ▹Basics CAN / CANopen◃ from page 35).

7.7.2.1 State control

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

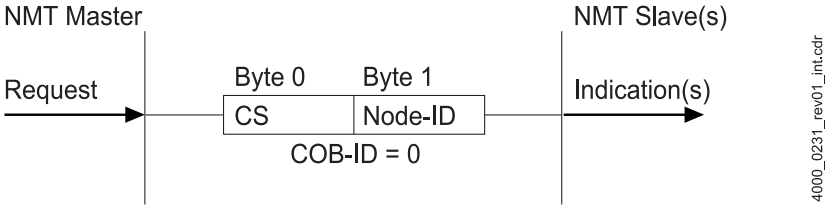


Figure 4: NMT telegram for controlling the communication states

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

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

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x00	0x02	0x80	0x10						

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

NOTE!

The b maXX® 5000 currently does not include device reset by software.

After turning on supply voltage and a reset the CANopen slave signals a boot up telegram (also see ▹Boot up◃ on page 49). The entire reset sequence lasts a few seconds, starting

with the receipt of the command reset_node to the acknowledgement using boot up telegram.

WARNING!

Danger from mechanical and electrical cause

If a reset is triggered during a running cyclic operation, this can lead to undesired states in the application, because the boot record will be loaded in the controller.

Therefore:

- Check the mapping after each reset.

7.7.2.2 Boot up

The boot up behavior according to CiA 301 V4 is supported by the device series b maXX 3300 / 5000.

Boot up according with CiA 301 V4,
Boot up telegram with ID = 0x700 + node ID, DLC = 1 byte 0 filled with the data = 0.

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x701	0x01	0x00							

CiA 301 V4

7.7.3 Node guarding

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

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

The COB ID results from 0x700 + address according to the predefined connection set. This COB ID can also be changed. The object required for this is 0x100E.

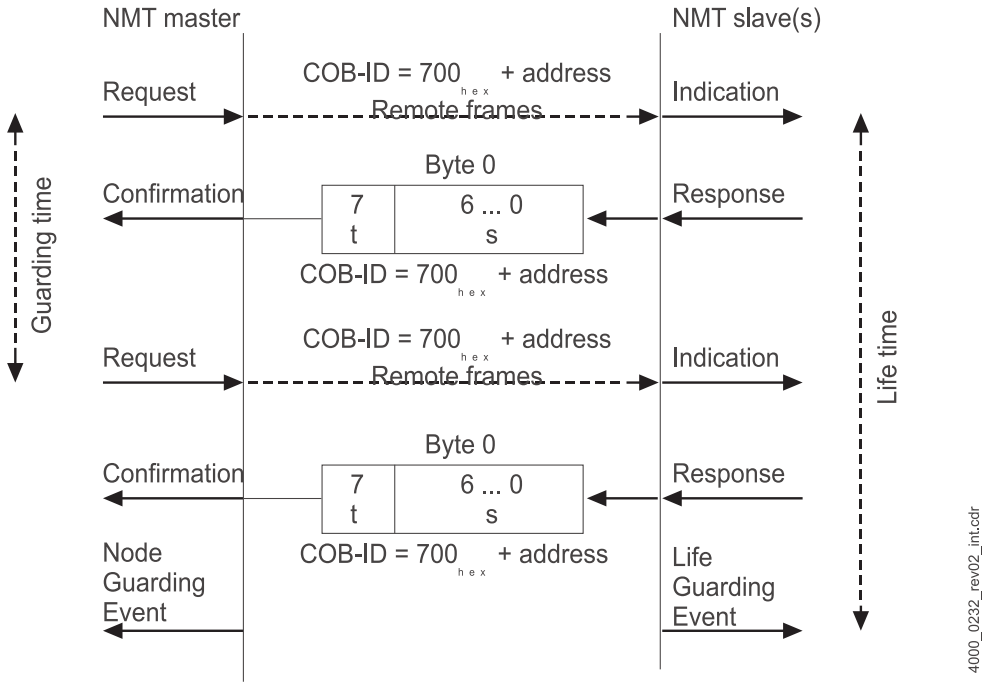


Figure 5: Node guarding protocol

The guarding time is set in objects 0x100C and 0x100D. Within this time, the slave must have received a guarding request (remote telegram) from the master. Should this not be the case, the life guarding event occurs in the slave. Through this, the slave switches to the PRE-OPERATIONAL state and the reaction specified in object 0x6007 is triggered in the controller.

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

The current communication state of the slave can be recognized from the response of the slave to a node guarding request from the master. The response telegram consists of one data byte (also see ▹Figure 5◃ on page 50). Field 's' differs according to the communication state. In addition, if there are two successive telegrams, toggle bit 't' will be changed.

Communication phase	Identifier s	Resulting data with	
		t = 0	t = 1
PRE-OPERATIONAL	0x7F (127)	0x7F (127)	0xFF (255)
OPERATIONAL	0x05 (5)	0x05	0x85 (133)
STOPPED	0x04 (4)	0x04	0x84 (132)

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

Node guarding is started in the slave after receipt of the first guarding request telegram. From the moment, the monitoring time parameterized in objects 0x100C and 0x100D runs in the slave.

NOTE!

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

7.8 Service data (SDO)

Service data objects (SDO) are used in the exchange of messages without real time requirements. Therefore low-priority COB IDs are provided for this in the predefined connection set (also see ▹Basics CAN / CANopen◃ from page 35). SDOs are used for parameterizing slaves and for setting the communication references for PDOs. Access on data occurs only via the object list. SDOs are always confirmed data, i. e. the transmitter receives an acknowledgement from the receiver. Data exchange via SDOs can only progress asynchronously (also see ▹Synchronization (SYNC)◃ from page 60).

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

7.8.1 Telegram structure

Die COB ID of the request SDO results from 600_{hex} + address, from the response SDOs from 0x580 + address. The data field of the CAN data telegram (8 bytes) for a SDO is divided into three parts, a command specifier CS (1 byte), a multiplexor M (3 bytes) and the actual user data D0 - D3 (4 bytes).

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600 + address	0x08	CS	M	M	M	D0	D1	D2	D3
0x580 + address	0x08	CS	M	M	M	D0	D1	D2	D3

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

7.8.2 Types of SDO transfers

The Baumüller CANopen interface supports the expedited transfer and segmented transfer, in that the latter is only used for objects 0x1008, 0x1009 and 0x100A manufacturer device name.

Expedited transfer

Objects can be written or read, with their data including a maximum of 4 bytes. Only two telegrams are required, a request and a response. All objects with the indices 0x1XXX, 0x4XXX, 0x6XXX can be addressed via expedited SDOs with the exception of objects 0x1008, 0x1009 and 0x100A.

Segmented transfer

The segmented transfer is necessary for objects with data greater than 4 bytes. Thereby the user data is divided to several telegrams. This is only necessary when reading the objects 0x1008, 0x1009 and 0x100A.

7.8.3 Writing object

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

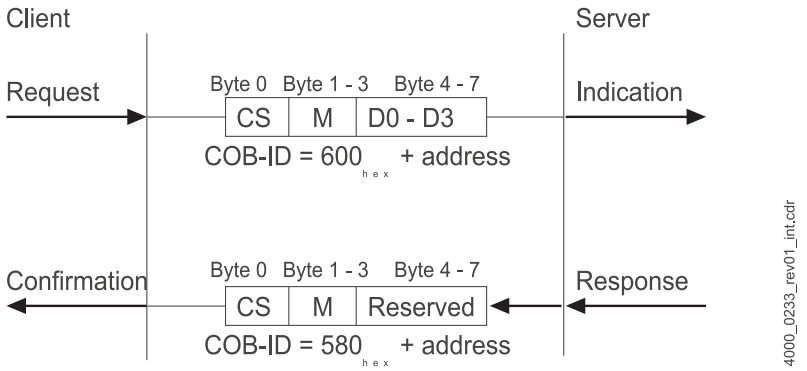


Figure 6: Initiate SDO download protocol

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

Data length in D0 - D3	Command specifier CS
1 byte	0x2F
2 byte	0x2B
4 byte	0x23

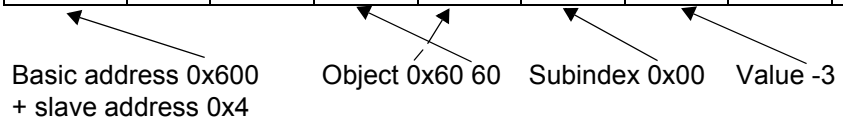
The command specifier CS for the response is 0x60, the multiplexor is identical to that of the request, the data field without meaning (reserved).

Example

The value ’-3’ (0xFD) to be written to object 0x6060, subindex 0x00, of the slave with the address 4. The data width of this object is 8 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x2F	0x60	0x60	0x00	0xFD	0x00	0x00	0x00



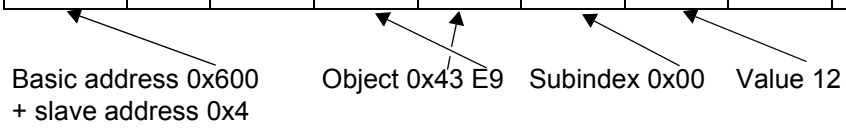
Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x60	0x60	0x60	0x00	0x00	0x00	0x00	0x00

The value ’12’ (0x0C) is to be written to object 0x43E9, subindex 0x00, of the slave with the address 4. The data width of this object is 16 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x2B	0xE9	0x43	0x00	0x0C	0x00	0x00	0x00



Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x60	0xE9	0x43	0x00	0x00	0x00	0x00	0x00

The value „0x60610008“ to be written to the object 0x1800, subindex 0x02, of the slave with the address 4. The data width of this object is 32 bits.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x23	0x00	0x18	0x02	0x08	0x00	0x61	0x60

Basic address 0x600 + slave address 0x4 — Object 0x18 00 — Subindex 0x02 — Value 0x60 61 00 08

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x60	0x00	0x18	0x02	0x00	0x00	0x00	0x00

7.8.4 Reading object

With the Baumüller CANopen interface, expedited transfer is used to read objects; with objects 0x1008, 0x1009 and 0x100A segmented transfer is used.

7.8.4.1 Expedited transfer

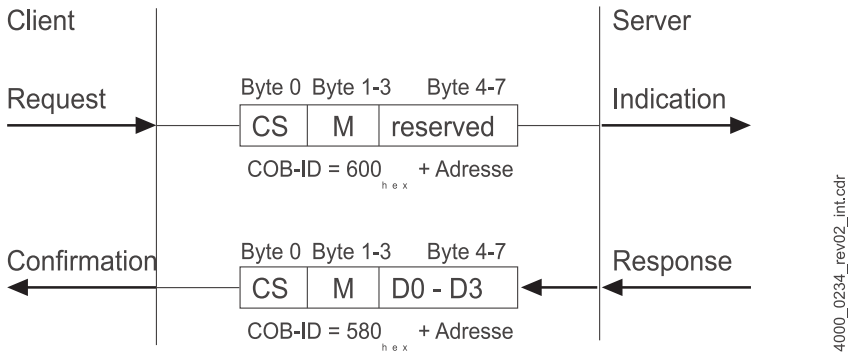


Figure 7: Initiate SDO upload expedited

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

The command specifier CS for the request is always 0x40. The command specifier CS for the response will depend on the length of the user data. D0 is the LSB and D3 the MSB.

Data length in D0 - D3	Command specifier CS
1 byte	0x4F
2 byte	0x4B
4 byte	0x43

The request and response multiplexors agree.

Example

Object 0x6061, subindex 0x00 of the slave with the address 4 is to be read. The data width of this object is 1 byte.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x40	0x61	0x60	0x00	0x00	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x4F	0x61	0x60	0x00	0xD0	0x00	0x00	0x00

Basic address 0x580 + slave address 0x4 — Object 0x60 61 — Subindex 0x00 — Value data

Object 0x6041, subindex 0x00 of the slave with the address 4 is to be read. The data width of this object is 2 byte.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x40	0x41	0x60	0x00	0x00	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x4B	0x41	0x60	0x00	D0	D1	0x00	0x00

Basic address 0x580 + slave address 0x4 — Object 0x60 41 — Subindex 0x00 — Value DB high DB low

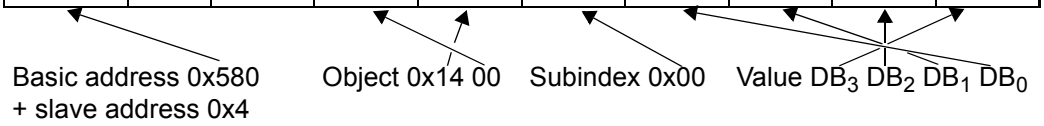
Object 0x1400, subindex 0x01 of the slave with the address 4 is to be read. The data width of this object is 4 byte.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x40	0x00	0x14	0x01	0x00	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x43	0x00	0x14	0x01	D0	D1	D2	D3



7.8.4.2 Segmented transfer

First of all a read request is sent to the slave with initiate SDO upload protocol. The slave responds with the command specifier CS 0x41. The total number of the user data bytes to be transferred is returned in the data field (request 1, response 1). This user data will be transferred in the following cycles (request 2, response 2, request 3 and response 3).

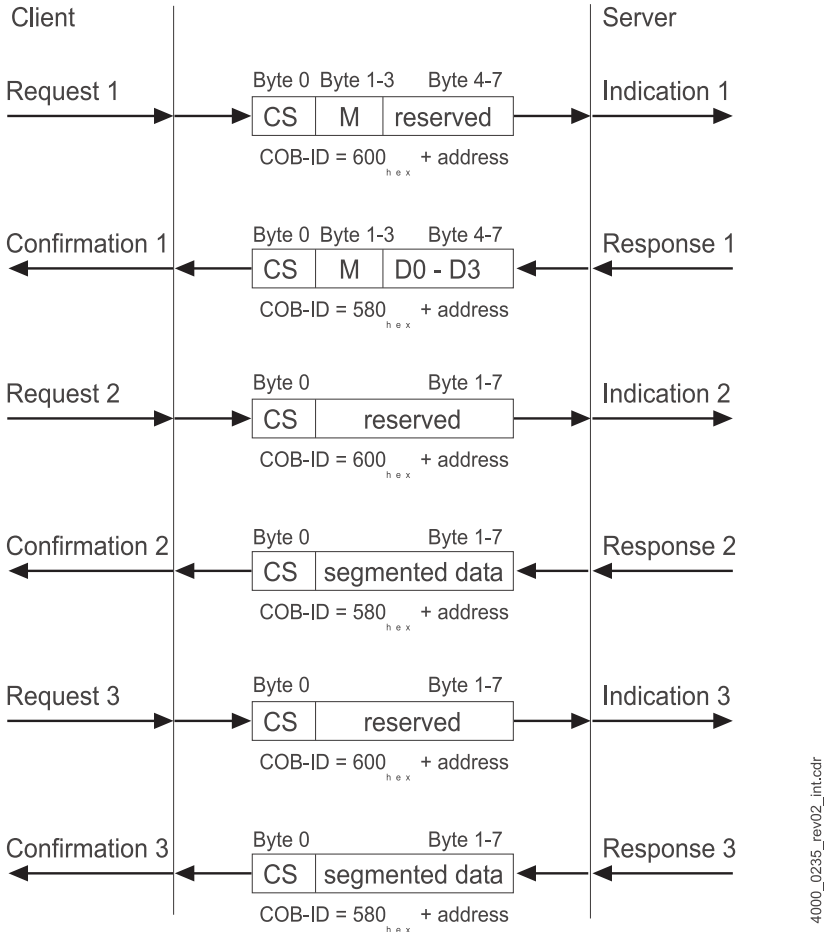


Figure 8: Upload SDO segmented protocol

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

for example to read object 0x1008 manufacturer device name of slave 4:

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x40	0x08	0x10	0x00	0x00	0x00	0x00	0x00

Response 1

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x41	0x08	0x10	0x00	0x06	0x00	0x00	0x00

Byte 0 in response 1 (command specifier 0x41) signifies that the user data field contains the number of the user data bytes to be transferred (6).

Request

		CS							
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x60	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Response

		CS	D0	D1	D2	D3	D4	D5	D6
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x14	0x62	0x20	0x6D	0x61	0x58	0x58	0x00

Byte 0 In request 2 (command specifier 0x60) means that the first segment (6 bytes) are to be transferred. Byte 0 in response 2 (command specifier, 0x14) signifies that the user data field (6 bytes) contains valid data and that this segment is at the same time the last.

The result of the transfer is: b maXX.

7.8.5 Error reactions

Invalid SDO accesses are refused with abort codes. The structure of these abort telegrams is identical to the SDO telegram illustrated in ▷page 51◁. The data field contains an abort code of 4 bytes.

With invalid accesses to communication-specific objects (0x1XXX) the following messages are differentiated:

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

Invalid accesses to all other objects (0x4XXX and 0x6XXX) are globally refused with the following codes:

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

Example

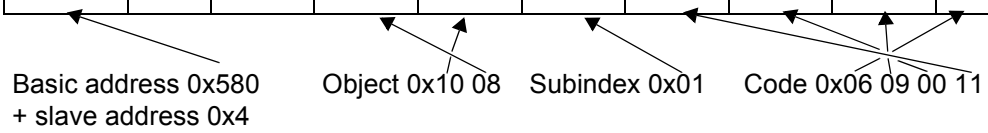
Slave 4 object 0x1008 subindex 0x01 is to be read. Object 0x*1008 manufacturer device name* has however only subindex 0x00.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x604	0x08	0x40	0x08	0x10	0x01	0x00	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x584	0x08	0x80	0x08	0x10	0x01	0x11	0x00	0x09	0x06



The command specifier CS (4 byte 0, 0x80) in the response telegram specifies, that it is an abort telegram. The request and response multiplexors agree.

7.9 Synchronization (SYNC)

In order to synchronize the slaves a SYNC telegram is used. This telegram is unconfirmed (broadcast). It contains no data. The COB ID is stipulated in object *0x1005 COB ID SYNC*. By default, 0x80 is specified. The CANopen slave option module can receive SYNC telegrams. It is not a SYNC master!

Receipt of a SYNC telegram with the identifier set in object 0x1005 generates an interrupt to the CANopen option module, which is passed to the b maXX®controller. Because of this, this signal can be used to synchronize the b maXX® controller. All relevant telegrams must be transmitted to all configured slaves within one SYNC interval (communication cycle). Transfer rate, cable length, number of nodes, size of telegrams as well as processing times on the CANopen option card are to be taken into consideration during this setting the cycle time for the SYNC telegram is undertaken in object 0x1006. For this, see ▹Directory of objects for communication control◃ from page 41.

The communication cycle time in the controller is accepted automatically from object 0x1006.

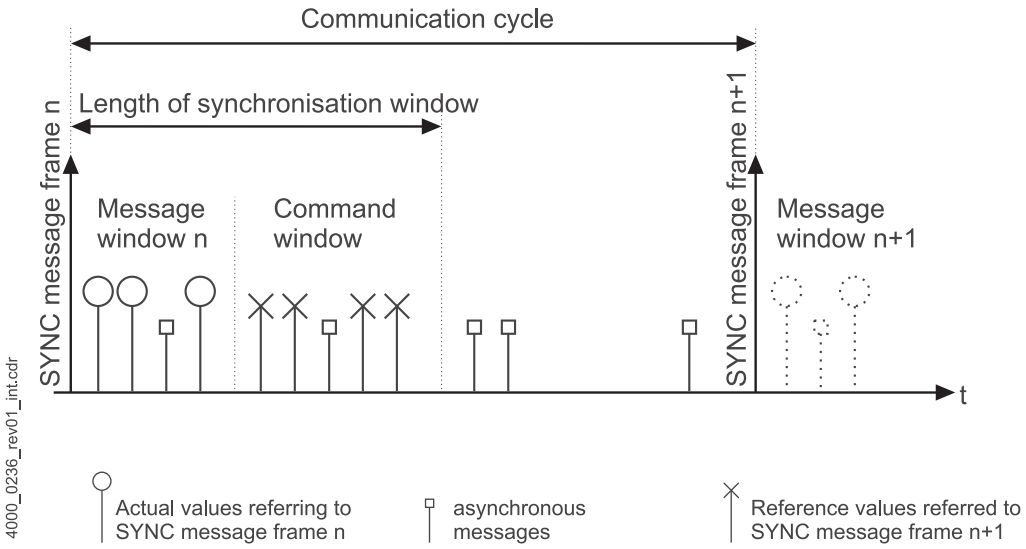


Figure 9: Communication cycle

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

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

7.10 Process data (PDO)

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

7.10.1 PDO mapping

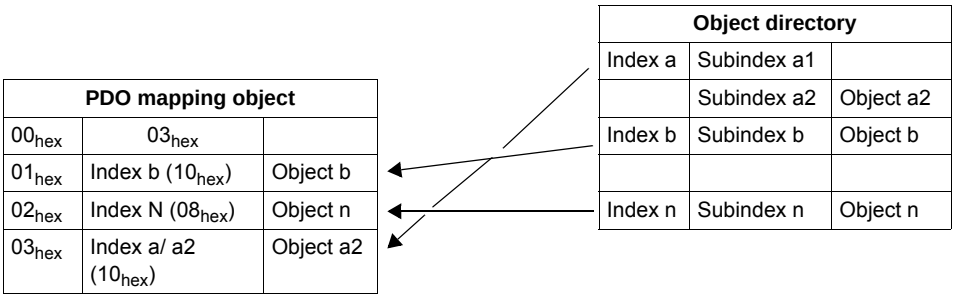
Mapping is a method of assigning variables/objects to PDOs. These variables/objects are moved across the CAN bus with the PDOs. Cyclic data exchange is configured by means of the mapping. SDOs are used for this parameterizing. Mapping is set in the object directory via addressable objects. There are four such objects for each PDO per axis (also see ▹Directory of objects for communication control◃ from page 41). One of the objects determines the content of the PDO, the second one the communication relationship or triggering.

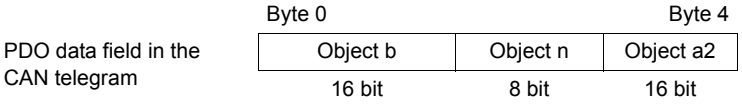
Process data object	Object for content	Object for the communication relationship
TX-PDO1 Axis 1	0x1A00	0x1800
TX-PDO2 Axis 1	0x1A01	0x1801
TX-PDO3 Axis 1	0x1A02	0x1802
TX-PDO4 Axis 1	0x1A03	0x1803
TX-PDO1 Axis 2	0x1A40	0x1840
TX-PDO2 Axis 2	0x1A41	0x1841
TX-PDO3 Axis 2	0x1A42	0x1842
TX-PDO4 Axis 2	0x1A43	0x1843
RX-PDO1 Axis 1	0x1600	0x1400
RX-PDO2 Axis 1	0x1601	0x1401
RX-PDO3 Axis 1	0x1602	0x1402
RX-PDO4 Axis 1	0x1603	0x1403
RX-PDO1 Axis 2	0x1640	0x1440
RX-PDO2 Axis 2	0x1641	0x1441
RX-PDO3 Axis 2	0x1642	0x1442
RX-PDO4 Axis 2	0x1643	0x1443

NOTE!

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

For the user data transmission a CAN data telegram provides a maximum of eight bytes. By mapping the logic content is determined by a maximum of eight bytes. For this determination certain information about the object, which is mapped is necessary: object index, subindex and length of datum. From the object index the according objects are entered in the mapping object. The sequence of this input, determined by the subindex of the mapping object, determines the data sequence in the CAN telegram. In the mapping objects (0x1600 ... 0x1603, 0x1640 ... 0x1643, 0x1A00 ... 0x1A03, 0x1A40 ... 0x1A43) the objects which are mapped, are written to the according subindices (start with 0x01), e. g. to object 0x1600 subindex 0x00 the value 0x60400010 is entered. This means, that the first two bytes of the data, which was received in RX-PD01 is written to the control word (object 0x6040 subindex 0x00). The object 0x6040 is implemented to the b maXX 3300/5000 parameter **108.1** control word (also see ▹Appendix C - Conversion tables◃ from page 141). Therewith the first word of the received telegram in the RX-PDO1 is written to the control word of the b maXX controller. In subindex 0x00 the number of the objects, which must be mapped (number of assigned subindices with the valid objects) must be entered. In ▹Example for PDO mapping◃ from page 67 is a detailed example for mapping.





Default mapping is described in ▹Directory of objects for communication control◃ from page 41.

In order to deactivate an existing mapping, the values in the subindices can be overwritten or the value '0' can be written to subindex 0x00 of the corresponding mapping object (0x1600 ... 0x1603, 0x1640 ... 0x1643, 0x1A00 ... 0x1A03, 0x1A40 ... 0x1A43). In this way, the entire mapping of the respective PDO will be deactivated, but the entry is preserved.

Furthermore deactivation of mapping objects via the associated communication objects 0x1400 to 0x1403, 0x1440 to 0x1443, 0x1800 to 0x1803, 0x1840 to 0x1843 in subindex 1 with bit 31 set to 1 can take place.
Note: therewith COBID must be written.

NOTE!

At the setting of mapping in the (0x1600 ... 0x1603, 0x1640 ... 0x1643, 0x1A00 ... 0x1A03, 0x1A40 ... 0x1A43) is accordingly the subindex 0x00 with the correct number of the mapped objects is to be written at the end.

Setpoints: The permissible cyclical setpoints are marked in a table with the column 'PDO mapping' as 'RX'. The table is found in appendix B.2 (for the six thousands object numbers). The manufacturer-specific parameters (two thousands and four thousands objects) must be checked up in the parameter manual b maXX®3300 (5.12001) or in parameter manual b maXX®5000 (5.09022).

Actual values: The permissible cyclic actual values are marked in a table with the column 'PDO mapping' as 'TX'. The table is to be found in appendix B.2 (for the six thousands object numbers). At manufacturer-specific parameters (two thousands and four thousands objects) it must be checked up in the parameter manual b maXX®3300 (5.12001) or in parameter manual b maXX®5000 (5.09022). A detailed description of the b maXX® parameters are found in these manuals.

Incorrect mapping configuration (invalid mapping configurations in 0x1600 ... 0x1603, 0x1640 ... 0x1643, 0x1A00 ... 0x1A03, 0x1A40 ... 0x1A43) are signaled by abort codes via SDO.

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

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

If a wrong parameter is mapped at the setpoints (e. g. an actual value parameter), the axis does not switch to OPERATIONAL mode.

Dummy mapping

The option module CANopen slave provides 2 dummy objects: a 1 byte dummy object and a 2 byte dummy object, which also can be mapped into a PDO. These objects have the indices 0x0005 (1 byte dummy) and 0x0006 (2 byte dummy). The dummy object serves as a dummy, in order to use only certain objects within a CAN telegram (also see ▹Example for PDO mapping◃ from page 67).

NOTE!

The present mapping, which was set gets lost after switching off or after a reset of the state machine.

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

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

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

NOTE!

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

7.10.2 Communication relationship via PDO

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

The object index has an offset of -0x200 for the corresponding mapping object.

Process data object mapping	Object for content	Object for communication relationship
TX-PDO1 Axis 1	0x1A00	0x1800
TX-PDO2 Axis 1	0x1A01	0x1801
TX-PDO3 Axis 1	0x1A02	0x1802
TX-PDO4 Axis 1	0x1A03	0x1803
TX-PDO1 Axis 2	0x1A40	0x1840
TX-PDO2 Axis 2	0x1A41	0x1841
TX-PDO3 Axis 2	0x1A42	0x1842
TX-PDO4 Axis 2	0x1A43	0x1843
RX-PDO1 Axis 1	0x1600	0x1400
RX-PDO2 Axis 1	0x1601	0x1401
RX-PDO3 Axis 1	0x1602	0x1402
RX-PDO4 Axis 1	0x1603	0x1403
RX-PDO1 Axis 2	0x1640	0x1440
RX-PDO2 Axis 2	0x1641	0x1441
RX-PDO3 Axis 2	0x1642	0x1442
RX-PDO4 Axis 2	0x1643	0x1443

The structure of these objects is described in ▹Directory of objects for communication control◃ from page 41.

The criterion for accepting a message transmitted onto the CAN bus into the CAN open slave option module is the matching COB ID. The COB ID is set in control objects 0x1400 - 0x1403, 0x1440 - 0x1443, 0x1800 - 0x1803, 0x1840 - 0x1843 under subindex 0x01. If the identifier parameterized here agrees with the message identifier transmitted via the CAN bus, the telegram will be accepted into its telegram buffer.

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

e. g. 0x1400 subindex 1

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	0x08	0x23	0x00	0x14	0x01	0x01	0x00	0x00	0x01

Object 0x14 00 Important Bit 31

Furthermore, triggering requirements for transmission and reception are defined for CANopen that group PDOs into synchronous and asynchronous. Triggering requirements are set in the objects 0x1400 - 0x1403, 0x1440 - 0x1443, 0x1800 - 0x1803, 0x1840 - 0x1843 accordingly in the subindex 0x02.

NOTE!

PDOs which are not needed should be deactivated, in order to exclude interferences. Deactivation occurs with the writing of the subindices 0 of the objects 0x1600 to 0x1603, 0x1640 to 0x1643, 0x1A00 to 0x1A03 and 0x1A40 to 0x1A43 with 0 or with bit 31 in SIX 2 of the objects 0x1401 to 0x1403, 0x1441 to 0x1443, 0x1801 to 0x1803 and 0x1841 to 0x1843.

Synchronous PDOs

Transmission and reception is linked with the SYNC telegram (also see ▹Synchronization (SYNC)◃ from page 60).

Asynchronous PDOs

Transmission and reception is linked with certain events.

Value in subindex 0x02	Type	Effect	
		TX-PDO 0x1800 to 0x1803, 0x1840 to 0x1843	**RX-PDO 0x1400 to 0x1403, 0x1440 to 0x1443**
0x00 (0)	asynchronous	Transmission takes place after each SYNC telegram received **and** an event has occurred.	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
0x01 (1)	synchronous	Transmission occurs after each received SYNC telegram.	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
0x02 - 0xF0 (2 - 240)	synchronous	Transmission occurs after receiving the set number of SYNC telegrams	PDOs with a matching COB ID received before the last SYNC telegram are accepted.
0xFC (252)	RTR synchronous	Transmission takes place after receipt of the RTR telegram with matching COB ID updating the PDO takes place after receipt of the last SYNC telegram.	none

Value in subindex 0x02	Type	Effect	
		TX-PDO 0x1800 to 0x1803, 0x1840 to 0x1843	RX-PDO 0x1400 to 0x1403, 0x1440 to 0x1443
0xFD (253)	RTR asynchro-nous	Transmission takes place after receipt of the RTR telegram with matching COB ID	none
0xFE (254)	asynchro-nous	Transmission takes place in a time-controlled man-ner	none
0xFF (255)	asynchro-nous	Transmission takes place in an event-controlled manner	Each PDO with a match-ing COB ID is accepted

Time-controlled transmission means that the transmission requirement is linked to a timer. This timer is set for the TX-PDO1 by means of subindex 0x05 in object 0x1800 (16 bit) is set in a similar manner. Analog the timer for TX-PDO2, TX-PDO3 and TX-PDO4 in the subindex 0x05 of object 0x1801, 0x1802, 0x1803 can be set. The resolution is 1 millisecond. The timer(s) is (are) started on change of state to OPERATIONAL. Transmission of the corresponding TX-PDO then takes place cyclically with the cycle time set in the timer. The timer will be cleared by writing the value ’0’ to subindices 0x05 of object 0x1800 - 0x1803.

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

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

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

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

7.10.3 Example for PDO mapping

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

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

Furthermore, node 2 receives the control word from the master in its RX-PDO1. Node 7 likewise receives a control word from the master in RX-PDO 2. The configuration is shown in ▹Figure 10◃ (see below). Object 0x6086 is used in connection with dummy mapping.

The b maXX®5000 with the address 2 transmits its speed actual value and the status word every 10 ms. Node 7 transmits its status word only after receiving a SYNC telegram (from the master) 3 times.

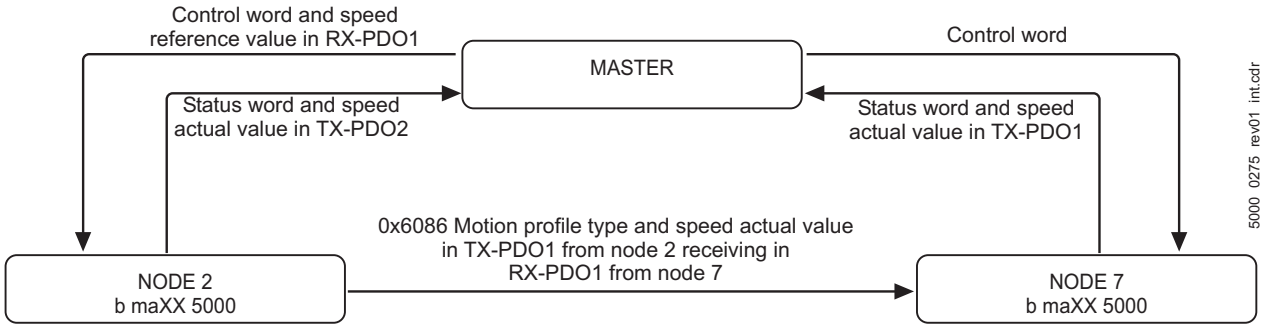


Figure 10: Example mapping with two b maXX®5000

1st step: determining the necessary objects

Ascertain the relevant object directory objects from the object list (see ▹Appendix C - Conversion tables◃ from page 141 and ▹Directory of objects for communication control◃ from page 41).

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

108.3 Status word	⇔	6041_{hex} Status word
108.1 Control word	⇔	6040_{hex} Control word
110.5 Setpoint selection HLG input	⇔	6042_{hex} Speed setpoint at the HLG
18.22 Speed actual value	⇔	6044_{hex} Control effort
118.2 Positioning mode	⇔	6086_{hex} Motion profile type

The following objects are necessary for setting mapping:

Node 2 0x1A00 (1. transmit PDO mapping), 0x1800 (1. transmit PDO parameters)
0x1A01 (2. transmit PDO mapping), 0x1801 (2. transmit PDO parameters)

Node 7 0x1600 (1. receive PDO mapping), 0x1400 (1. receive PDO parameters)
0x1601 (2. receive PDO mapping), 0x1401 (2. receive PDO parameters)

2nd step: configure mapping

In order to set mapping the SDOs of the expedited transfers (also see ▹Service data (SDO)◃ from page 51) are used. These can be initiated via a master, a bus configurator or similar.

Mapping for slave 2

Write the first object to be mapped with index (0x6086), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x01 (TX-PDO 1). The object shall not be evaluated by slave 7.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x23	0x00	0x1A	0x01	0x10	0x00	0x86	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x1A	0x01	0x00	0x00	0x00	0x00

Write the second object to be mapped with index (0x6044), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x02 (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x23	0x00	0x1A	0x02	0x10	0x00	0x44	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x1A	0x02	0x00	0x00	0x00	0x00

Writing the number of the mapped objects (0x02) to 0x1A00 subindex 0x00 (TXP-DO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x2F	0x00	0x1A	0x00	0x02	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x1A	0x00	0x00	0x00	0x00	0x00

The content of object 0x1A00 is as follows:

0x1A00	0x00	0x02
	0x01	0x60860010
	0x02	0x60440010

Write the first object to be mapped with index (0x6041), subindex (0x00) and length (0x10) to 0x1A01 subindex 0x01 (TX-PDO 2).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x602	0x08	0x23	0x01	0x1A	0x01	0x10	0x00	0x41	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x582	0x08	0x60	0x01	0x1A	0x01	0x00	0x00	0x00	0x00

Writing of the second object to be mapped with index (0x6044), subindex (0x00) and length (0x10) to 0x1A01 subindex 0x02.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x602	0x08	0x23	0x01	0x1A	0x02	0x10	0x00	0x44	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x582	0x08	0x60	0x10	0x1A	0x02	0x00	0x00	0x00	0x00

Write the number of the mapped objects (0x02) to 0x1A01 subindex 0x00 (TX-PDO 2).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x602	0x08	0x2F	0x01	0x1A	0x00	0x02	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x01	0x1A	0x00	0x00	0x00	0x00	0x00

The content of object 0x1A01 is as follows:

0x1A01	**0x00**	0x02
	0x01	0x60410010
	0x02	0x60440010

Write the first object to be mapped with index (0x6040), subindex (0x00) and length (0x10) to 0x1600 subindex 0x01 (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x23	0x00	0x16	0x01	0x10	0x00	0x40	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x16	0x01	0x00	0x00	0x00	0x00

Write the second object to be mapped with index (0x6042), subindex (0x00) and length (0x10) to 0x1600 subindex 0x02 (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x23	0x00	0x16	0x02	0x10	0x00	0x42	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x16	0x02	0x00	0x00	0x00	0x00

Write the number of the mapped objects (0x02) to 0x1600 subindex 0x00 (RX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x602	0x08	0x2F	0x00	0x16	0x00	0x02	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x582	0x08	0x60	0x00	0x16	0x00	0x00	0x00	0x00	0x00

The content of object 0x1600 is as follows:

0x1600	**0x00**	0x02
	0x01	0x60400010
	0x02	0x60420010

Mapping for slave 7

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

Write the first object to be mapped with index (0x6041), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x01 (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x23	0x00	0x1A	0x01	0x10	0x00	0x41	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x1A	0x01	0x00	0x00	0x00	0x00

Write the first object to be mapped with index (0x6044), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x02 (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x23	0x00	0x1A	0x02	0x10	0x00	0x44	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x1A	0x02	0x00	0x00	0x00	0x00

Write the number of the mapped objects (0x02) to 0x1A00 subindex 0x00 (TX-PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x2F	0x00	0x1A	0x00	0x02	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x1A	0x00	0x00	0x00	0x00	0x00

The content of object 0x1A00 is as follows:

0x1A00	**0x00**	0x02
	0x01	0x60410010
	0x02	0x60440010

Write the first object to be mapped (16 bit dummy object) with index (0x0006), subindex (0x00) and length (0x10) to 0x1600 subindex 0x01 (RX PDO 1).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x23	0x00	0x16	0x01	0x10	0x00	0x06	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x16	0x01	0x00	0x00	0x00	0x00

Write the second object to be mapped with index (0x6042), subindex (0x00) and length (0x10) to 0x1600 subindex 0x02.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x23	0x00	0x16	0x02	0x10	0x00	0x42	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x16	0x02	0x00	0x00	0x00	0x00

Write the number of the mapped objects (0x02) to 0x1600 subindex 0x00.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x2F	0x00	0x16	0x00	0x02	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x00	0x16	0x00	0x00	0x00	0x00	0x00

The content of object 0x1600 is as follows:

0x1600	**0x00**	0x02
	0x01	0x00060010
	0x02	0x60420010

Write the first object to be mapped with index (0x6040), subindex (0x00) and length (0x10) to 0x1601 subindex 0x01 (RX-PDO 2).

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x23	0x01	0x16	0x01	0x10	0x00	0x40	0x60

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x01	0x16	0x01	0x00	0x00	0x00	0x00

Write the number of the mapped objects (0x01) to 0x1601 subindex 0x00.

Request

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x607	0x08	0x2F	0x01	0x16	0x00	0x01	0x00	0x00	0x00

Response

		CS	Multiplexor			D0	D1	D2	D3
COB ID	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
0x587	0x08	0x60	0x01	0x16	0x00	0x00	0x00	0x00	0x00

The content of object 0x1601 is as follows:

0x1601	**0x00**	0x01
	0x01	0x60400010

Data exchange between the b maXX®5000 via the PDOs is shown in ▹Figure 11◃ on page 76. Example for a cross communication. The speed actual value of slave 2 becomes speed set value of slave 7.

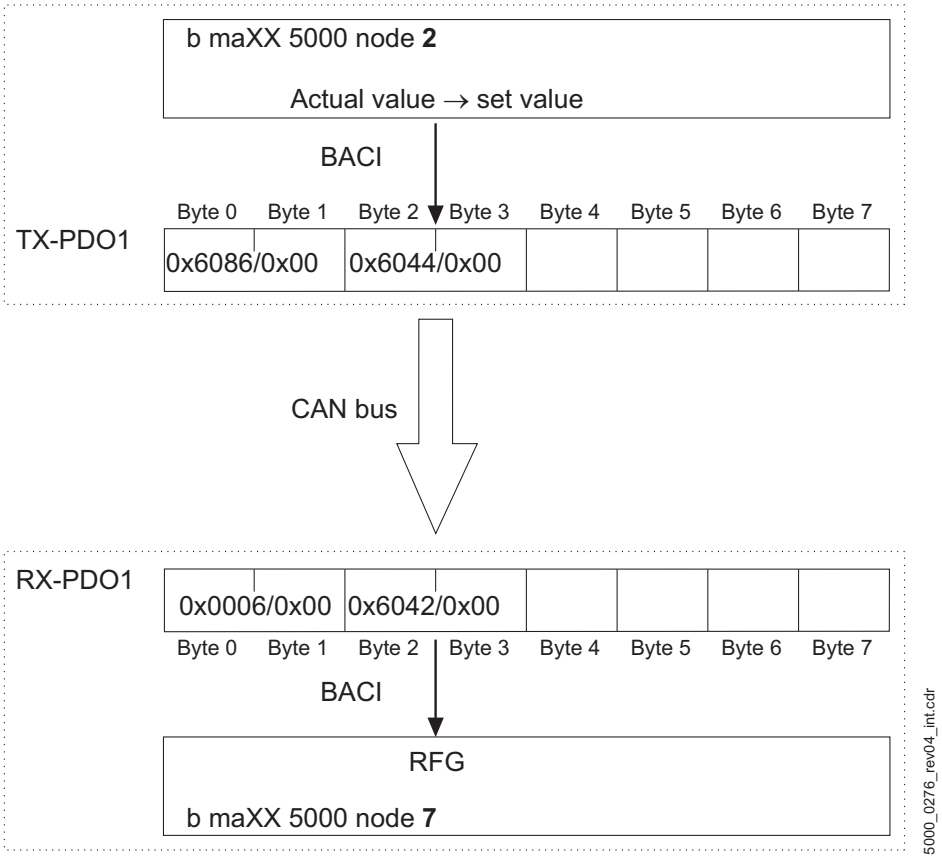


Figure 11: Telegram structure for example mapping

Only bytes 2 and 3 from TX-PDO1 of the b maXX® 5000 node 2 are evaluated in b maXX® 5000 node 7, because in RX-PDO1 only bytes 2 and 3 are validly linked with parameter numbers.

The communication parameters of both slaves are set via SDOs. So that a communication relationship can be built up, the COB-IDs of the transmitter and receiver must be in agreement. The COB-ID of the TX-PDO1 of slave 2 is by default set to 0x182. The COB-ID of the TX-PDO1 of slave 7 is by default 0x207. Which COB-ID must be used is the responsibility of the user. In the example, the COB-ID 0x207 is used.

Communication parameter for slave 2:

On object 0x1800 subindex 0x01 the value 0x207 is entered. On subindex 0x02 the value „0xFF“ (event-controlled) is written.

0x1800		
	0x01	0x207
	0x02	0xFF

The value 0xFE (timer triggered) is entered into object 0x1801 subindex 0x02. The value ’0x0A’ is written to subindex 0x05 (timer value = 10 ms).

0x1801		
	0x02	0xFE
	0x05	0x0A

The remaining subindices contain their default values.

The type of trigger for RX-PDO1 is set to event-triggered (
(0x1400 subindex 0x02 = 0xFF).

Communication parameter for Slave 7:

On object 0x1400 subindex 0x02 the value „0xFF“ (event controlled) is written. In subindex 0x01 is 0x207 by default.

0x1400		
	0x01	0x207
	0x02	0xFF

Both of the TX-PDOs of node 7 keep their default COB IDs. The trigger type of TX-PDO1 is set to SYNC triggered with the value 0x03. TX-PDO2 is parameterized as event triggered.

0x1800		
	0x01	0x187
	0x02	0x03

0x1801		
	0x01	0x287
	0x02	0xFF

7.10.4 Entry in fieldbus process data of the controller

A maximum of 16 cyclic setpoints and 16 cyclic actual values can be simultaneously exchanged between the CANopen slave and the b maXX® controller. All values are updated in one cycle. With CANopen, the setpoint and actual values can each be distributed across 4 PDOs.

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

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

The same method applies for RX PDOs.

The entries in the fieldbus process data list of the controller are made continuously, beginning with the first object of PDO1.

NOTE!

The dummy object is not taken into consideration in the initialization of process data.

Beginning with the first object of PDO1, the contents of the PDOs are alternately queried for their validity for the process data configuration (not a dummy). If the object is valid, then this is entered into the next free process data configuration position. If the PDO mapping is invalid (incorrect parameter numbers or the like), no cyclic communication between slave and b maXX® 5000 is started.

NOTE!

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

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

8

BASICS ETHERCAT

8.1 Literature concerning EtherCAT

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

- EtherCAT Communication Specification
  EtherCAT Technology Group (ETG)

- EtherCAT Slave Controller IP Core for Altera FPGAs Hardware Data Sheet
  EtherCAT Technology Group (ETG)

- www.ethercat.org
  EtherCAT Technology Group (ETG)
  Ostendstraße 196
  D-90482 Nürnberg

8.2 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.

8.2.1 Topology data

Several bus topologies can be used, e. g. line-, tree- or star-topologies (▹Figure 12◃).

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“.

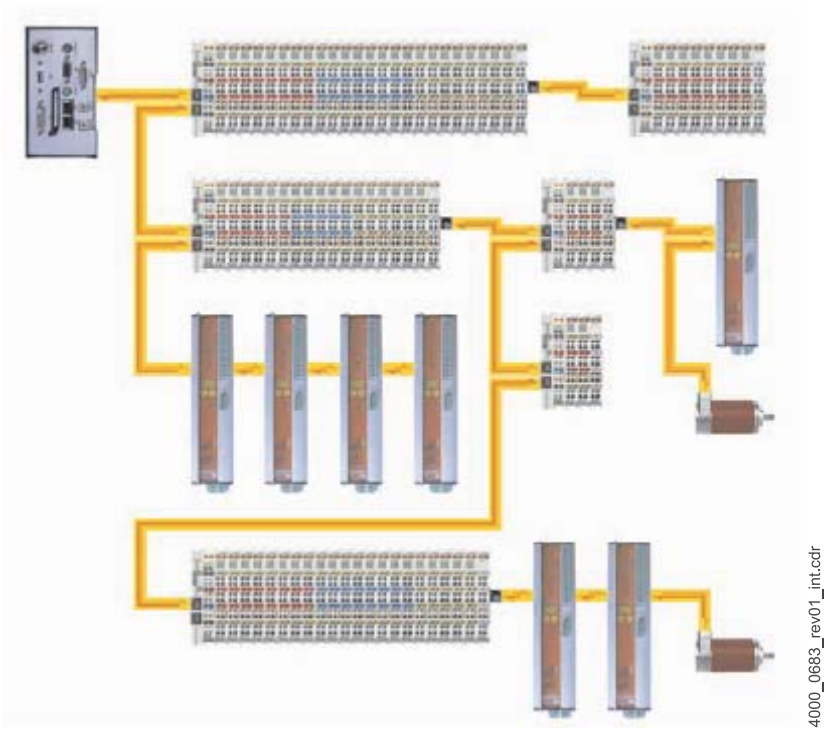


Figure 12: Flexible topology: line, tree or star [1]

8.2.2 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.

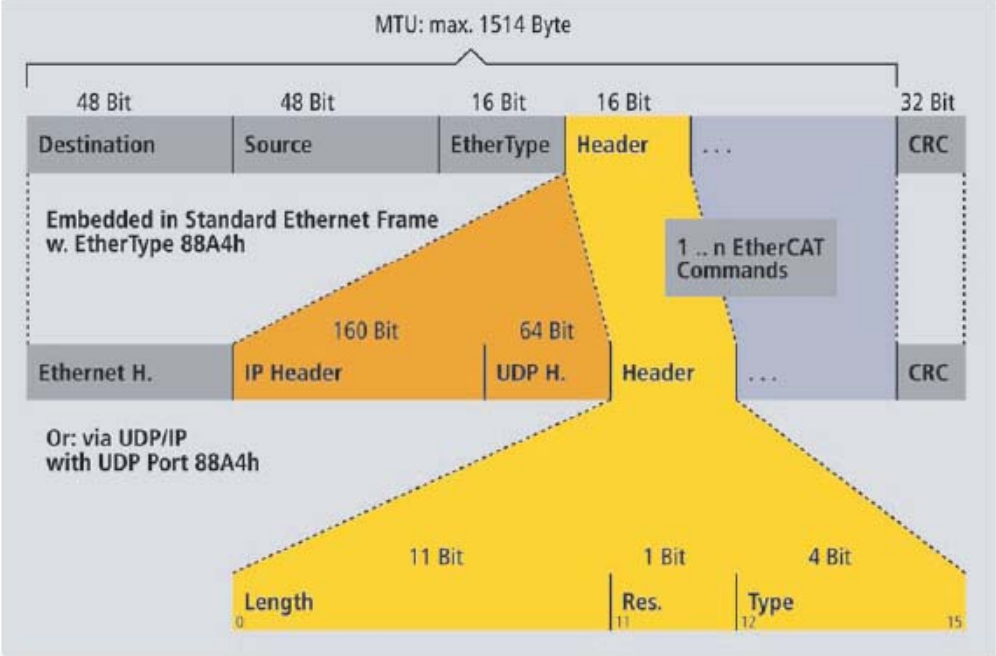


Figure 13: EtherCAT: standard -IEEE 802.3-frames [1]

8.2.3 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.

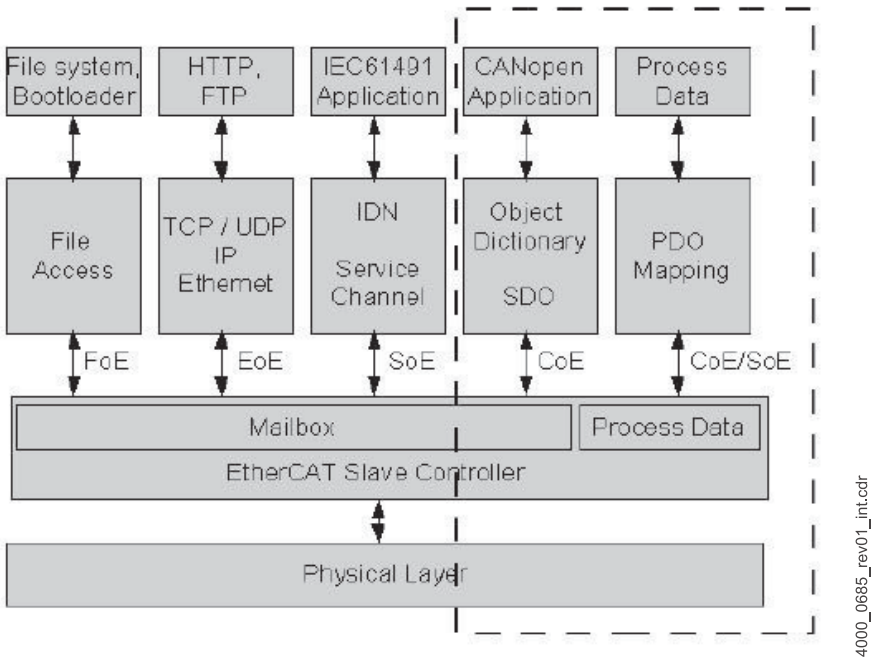


Figure 14: Device profile at EtherCAT[1]

8.2.4 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.

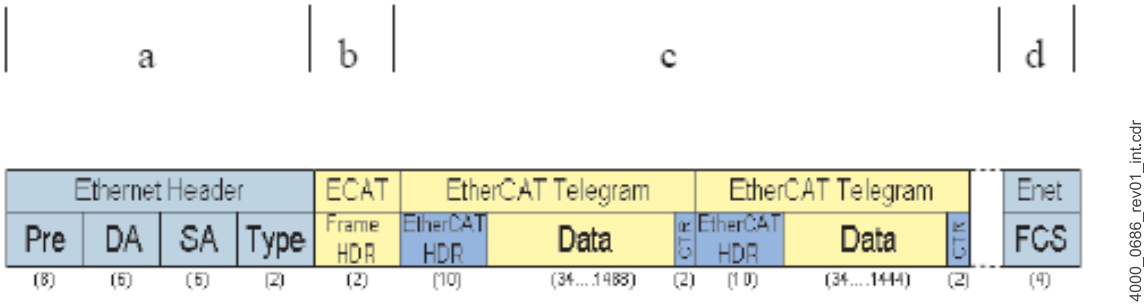


Figure 15: EtherCAT - frame [1]

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.

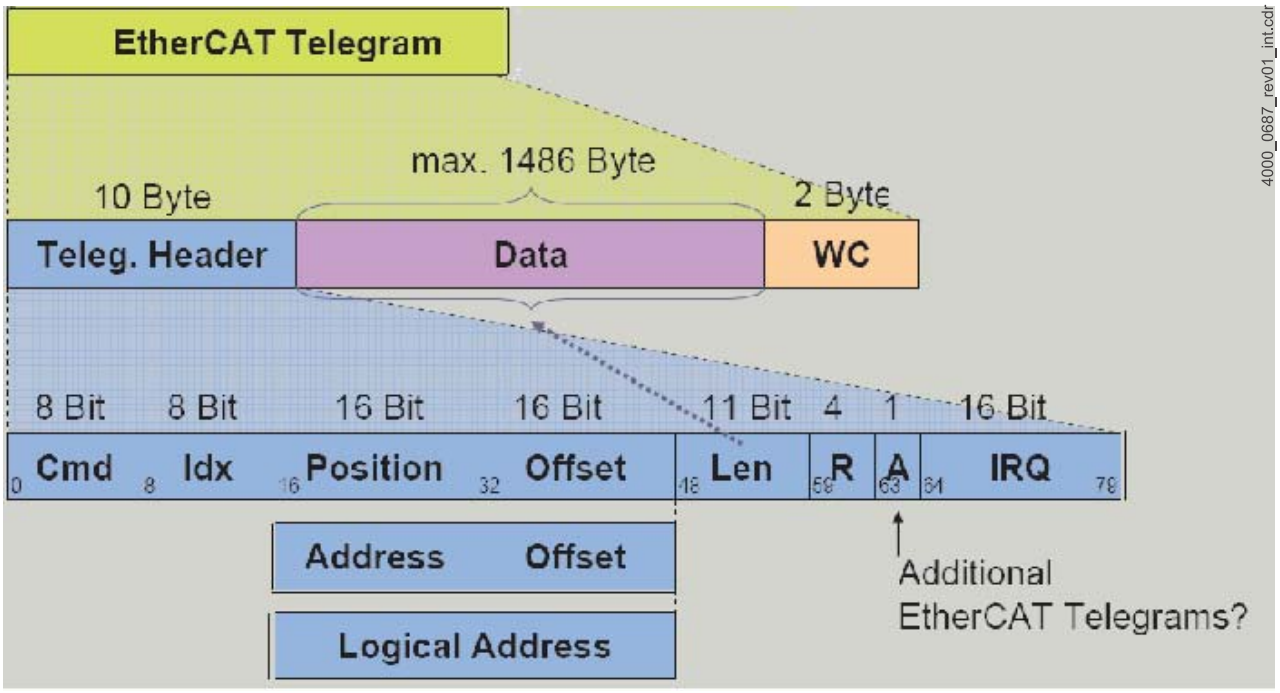


Figure 16: EtherCAT telegram [1]

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.

8.2.5 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 17◃.

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.

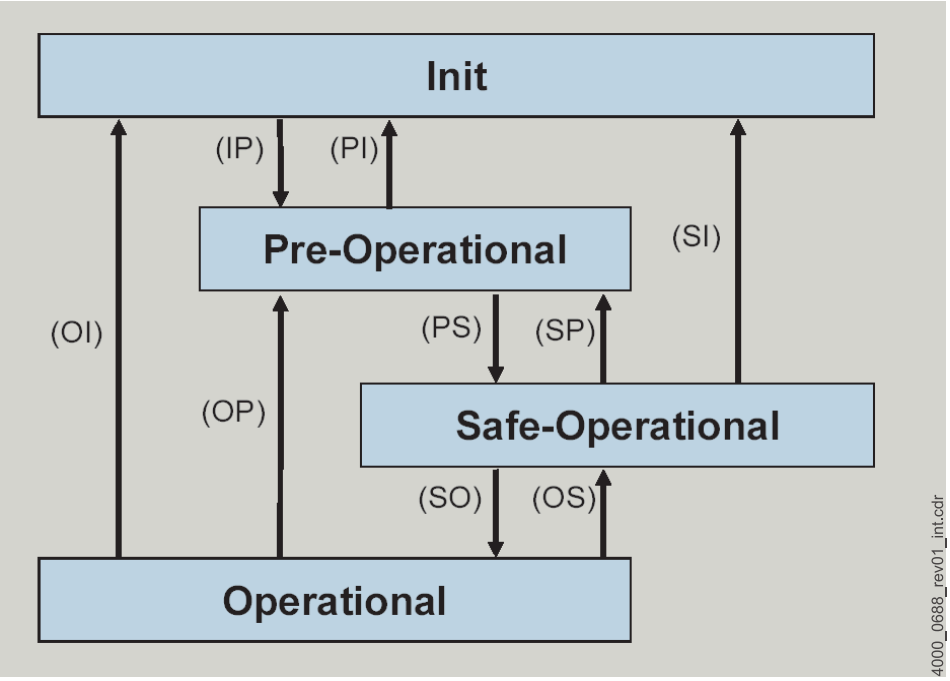


Figure 17: EtherCAT communication transitions [1]

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 addresses 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 121.

8.2.6 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 PROPROG-communication 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 setting of the IP address is:

	192.168 .XXX.XXX	
For DIP switch 192.168 is definitely assigned	XXX	means setting of DIP switch to the HW

An EtherCAT master also has the possibility to change the IP address (if this is supported by the master).

Thereby the IP address can be selected user-defined.

An IP address given by the master takes precedence over the DIP switch setting and over the controller parameters.

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

As the EoE communication is made via the mailboxes of the EtherCAT, the mailbox shall be checked several times (between 5 ms and 50 ms, at which 5 ms are perfect).

9

CoE at b maXX 2500 / 3300 / 5000

9.1 General information

The b maXX® 2500 / 3300 / 5000 CoE slave connects the b maXX® 2500 / 3300 / 5000 via the EtherCAT bus with other EtherCAT nodes (e. g. PC, PLC, further b maXX® devices, I/O modules).

Information according installation and handling with the device series b maXX® 3300 / 5000 is found in the manual 5.11018 / 5.09021.

Information according the programming of the **b maXX 3300 / 5000** controller is found in the parameter manual 5.12001 / 5.09022.

9.2 Address Setting

The node address setting of the b maXX 3300 / 5000 CoE slave is described in the instruction manual b maXX 3300 / 5000 (5.11018 / 5.09021).

9.3 XML file

The XML file contains information which a master needs, for example to configurate the FMMU (Fieldbus Memory Management Unit) and the SYNC manager on the slave.

The XML file can be downloaded from the download area on Baumüller's home page www.baumueller.de.

9.4 Diagnosis

CANopen follows the Ethernet standard IEEE 802.3. In this way standard tools (e. g. Wireshark (Freeware) or OmniPeek) and standard devices (hubs or switches and PC network interfaces) can be used for system diagnosis.

9.5 Data Exchange and Parameterization

The access to data or parameter is made always via CANopen objects. For a detailed description see ▹Data Exchange and Parameterization◃ from page 40.

9.6 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 CoE slave in accordance with CiA 301.

Name	Index	Subindex	Data type	Default value
Device type	0x1000	0x00	UINT32	0x00020192

This object is read-only and contains information on the related device (drive in accordance with CiA® 402).

Name	Index	Subindex	Data type	Default value
Error register	0x1001	0x00	UINT8	0x0

This object can only be read. The object 0x1001 contains an error bit string with the following meaning:

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

COB ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x80 + address	0x08	Emergency error code		Error register	Manufacturer-specific error field				

EMCY telegram for error reset / No error

Name	Index	Subindex	Data type	Default value
Manufacturer device name	0x1008	0x00	VString	-

This object is read-only. It contains the device name.

Name	Index	Subindex	Data type	Default value
Manufacturer hardware version	0x1009	0x00	VString	-

This object is read-only. It contains the current hardware version of the controller from parameter **102.25**.

Name	Index	Subindex	Data type	Default value
Manufacturer software version	0x100A	0x00	VString	-

This object is read-only. It contains the current software version of the controller, e. g. the character string: „01.08.00 S (Build 109)“.

Name	Index	Subindex	Data type	Default value
Identity object	0x1018	0x00	UINT8	0x04
Vendor ID		0x01	UINT32	0x0000015A
Product code		0x02	UINT32	0x26483052
Revision number		0x03	UINT32	0x00000000
Serial Number		0x04	UINT32	0x00000000

This object contains some information on the device.

Name	Index	Subindex	Data type	Default value
RPDO Mapping Axis 1	0x1600	0x00	UINT8	0x01
		0x01	UINT32	0x60400010
		:	:	
		n	UINT32	

This object contains the contents of receive PDO for axis 1. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the set value limit of 16 objects max. (also ▹PDO mapping◃ from page 99).

Name	Index	Subindex	Data type	Default value
RPDO Mapping Axis 2	0x1700	0x00	UINT8	0x01
		0x01	UINT32	0x60400010
		:	:	
		n	UINT32	

This object contains the contents of receive PDO for axis 2. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the set value limit of 16 objects max. (also ▹PDO mapping◃ from page 99).

Name	Index	Subindex	Data type	Default value
TPDO Mapping Axis 1	0x1A00	0x00	UINT8	0x01
		0x01	UINT32	0x60410010
		:	:	
		n	UINT32	

This object contains the contents of transmit PDO for axis 1. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the actual value limit of 16 objects max. (also ▹PDO mapping◃ from page 99).

Name	Index	Subindex	Data type	Default value
TPDO Mapping Axis 2	0x1B00	0x00	UINT8	0x01
		0x01	UINT32	0x60410010
		:	:	
		n	UINT32	

This object contains the contents of transmit PDO for axis 2. The total number of the following entries is in subindex 0x00. The total number of mapped objects may not exceed the actual value limit of 16 objects max. (also ▹PDO mapping◃ from page 99).

Name	Index	Subindex	Data type	Default value
Sync Manager Communication Type	0x1C00	0x00	UINT8	0x04
Communication Type Manager 0 Mailbox receive (Master to Master)		0x01	UINT8	1 (SM0)
Communication Type Manager 1 Mailbox transmit (Slave to Master)		0x02	UINT8	2 (SM1)
Communication Type Manager 2 RPDO (Master to Slave)		0x03	UINT8	3 (SM2)
Communication Type Manager 3 TPDO (Slave to Master)		0x04	UINT8	4 (SM3)

This object contains information on the SYNC manager settings. This object is read-only.

Name	Index	Subindex	Data type	Default value
Sync Manager Communication Type Channel 2 Number of assigned RPDOs	0x1C12	0x00	UINT8	0 ... 255
PDO Mapping Object Index of assigned RPDO		0x01	UINT16	0x1600 RPDO

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 slave this is a RxPDO.

Name	Index	Subindex	Data type	Default value
Sync Manager Communication Type Channel 3 Number of assigned TPDOs	0x1C13	0x00	UINT8	0 ... 255
PDO Mapping Object Index of assigned TPDO		0x01	UINT16	0x1A00 TPDO

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

Name	Index	Subindex	Data type	Default value
Sync manager output parameter	0x1C32	0x00	UINT8	0x20
Sync mode		0x01	UINT16	see below
Cycle time		0x02	UINT32	Cycle time in ns

SYNC mode

0	Controller is not synchronized
0x02	DC mode Sync0 Synchronization on AL Event Sync0
0x22	SM2 Synchronization on AL Event Sync manager 2 (RPDO 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.

Name	Index	Subindex	Data type	Default value
Sync manager input parameter	0x1C33	0x00	UINT8	0x20
Sync mode		0x01	UINT16	see below
Cycle time		0x02	UINT32	Cycle time in ns

SYNC mode

0	Controller is not synchronized
0x02	DC mode Sync0 Synchronization on AL Event Sync0
0x22	SM2 Synchronization on AL Event Sync manager 2 (RPDO 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.

9.7 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 18◃ the mailbox structure is displayed schematically.

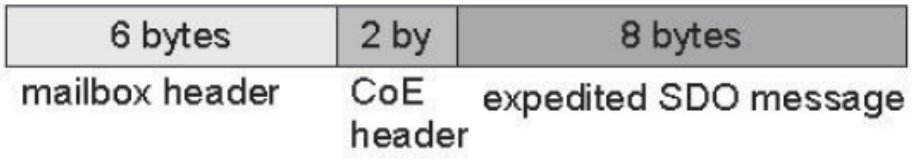


Figure 18: Structure of mailbox

Furthermore the structure of the mailbox headers are divided in:

Figure 19: Mailbox header

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:

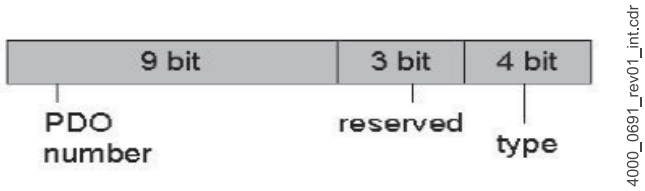


Figure 20: CoE-Header [1]

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

9.7.1 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	0x2F
2 byte	0x2B
4 byte	0x23

The CS for a write request response is CS = 0x60 or in the error case CS = 0x80.

The command specifier CS for a read request in the expedited transfer is CS = 0x40.

The response for the different lengths then is:

Data length in D0 - D3	Command specifier CS
1 byte	0x4F
2 byte	0x4B
4 byte	0x43

9.7.2 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 0x1008, 0x1009 and 0x100A 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 0x1XXX, 0x4XXX:, 0x6XXX are activated via the expedited SDOs accessible with exception of objects 0x1008, 0x1009 and 0x100A.

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 0x100, 0x1009 and 0x100A.

9.7.3 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 9.7.1◃ on page 96. The data field contains an abort code with 4 bytes.

With invalid accesses to communication-specific objects (0x1XXX) 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

Abort code	Meaning
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)

9.8 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 64 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 slave supports 4 SyncManagers, 2 for the mailbox one in each direction and 2 SyncManager as RPDO or TPDO. 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 are possible. Synchronization deactivated (the operation is only for the status SAFE-OPERATIONAL possible), synchronization to SyncManager2 (RPDO axis 1) and synchronization to distributed clocks DC.

NOTE!

All objects, which were configured in the PDOs are transmitted between the CoE slave and the b maXX® controller as cyclic data (also see ▹Communication flow◃ on page 25). As the cyclic data transmission (especially the RPDOs) is only made in the state of OPERATIONAL, the communication monitoring in ProDrive should be only in this status be activated, because in other states (e.g. PRE-OPERATIONAL) otherwise an error message is generated. This must then be acknowledged after the transition to OPERATIONAL.

9.8.1 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
TPDO axis 1	0x1A00
TPDO axis 2	0x1B00
RPDO axis 1	0x1600
RPDO axis 2	0x1700

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 (0x1600, 0x1A00) the objects, which are to be mapped are written to the according subindices (beginning with 0x01), e. g. to the object 0x1600 subindex 0x01 the value 0x60400010. That means that the first two bytes of the received data in RPDO axis 1 are written to the control word (object 0x6040 subindex 0x00). The object 0x6040 is implemented in the b maXX®-parameter **108.1** control word (also see ▹Appendix C - Conversion tables◃ from page 141). Therewith the first word of the received telegram, which was received in RPDO axis 1 is written to the control word of the b maXX®. In the subindex 0x00 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 103.

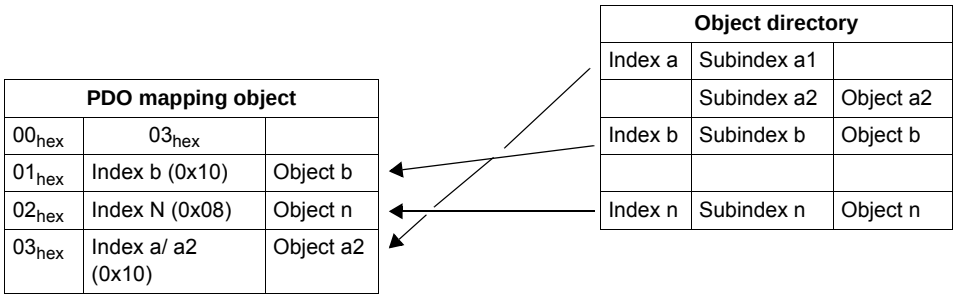


Figure 21: Mapping

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 0x00 of the according PDO (0x1600, 0x1A00) the PDO is deactivated.

NOTE!

When setting the mapping in the (0x1600, 0x1A00) the according subindex 0x00 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).

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). A detailed description of the b maXX®-parameters is found in the parameter manual b maXX® 5000 (5.09022) or in the parameter manual b maXX® 3300 (5.12001).

Incorrect mapping configurations (invalid objects in 0x1600, 0x1A00) are signalled with abort codes via SDO.

The cyclic set-/actual values are continuously initialized into the process data list, i. e. the first setpoint of PDO axis 1 is on first position, the second setpoint of PDO axis 1 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, 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 0x0005 (1 byte dummy) and 0x0006 (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 103).

NOTE!

The presently mapping, which was set drops away after a switchoff.

9.8.2 Synchronization (SYNC)

For synchronization of the controller two synchronization mechanisms can be used. Firstly the synchronization to SM2 (RPDO) and secondly with the distributed clocks (DC). The DCs were briefly introduced in the ▹Basics EtherCAT◃ from page 79.

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 slave 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 ProDrive under „Fieldbus slave“).

Setting of cycle time also can be made via the visualization tool ProDrive of the controller. On the page „Fieldbus slave“ the permitted (see above) cycle time is set. 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 RPDO 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 slave cannot be switched after OPERATIONAL.**

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 0x981 (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 0x1C32 subindex 0x01 is set, which synchronization mode is wanted:

Value 0x0 ⇒ freerun, not synchronized

Value 0x2 ⇒ DC Sync0, synchronized with DC IRQ Sync0

Value 0x22 ⇒ 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 (0x06010000 = error in data format) is generated.

The cycle time must be set via FBO 0x1C32 Subindex 0x02 or via register 0x9A0 in the ECT (DWORD in ns). The master must provide that the setpoint telegrams at set DC 200 µs to 50 µs are not send before the SYNC event. With help of the sync offsets in the ProDrive page „Fieldbus slave“ 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 Fieldbus slave in ProDrive.

If the DCs are activated the „PLL“ is deactivated due to the slave (in the FPGA) and the sync0 signal is directly transmitted from the DC to the controller.

At transition to SAFE-OPERATIONAL it is checked if in register 0x981 the DCs were activated. If not FBO 0x1C32 subindex 0x01 is reset to the value 0x22 (synchronization to SM2).

9.8.3 Example for PDO mapping

The option module CoE slave with the node address 1 receives from the master a speed setpoint in RPDO axis 1. This speed setpoint must be written to the ramp function generator input. Furthermore node 1 receives the control word from the master its RPDO axis 1. 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.

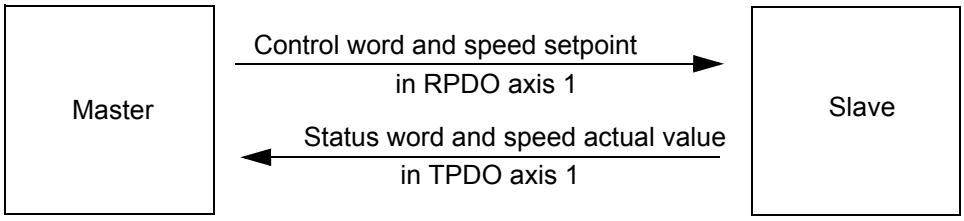


Figure 22: Example mapping with a b maXX®

1st step: determining the necessary objects

Ascertain the relevant object directory objects from the object list (see ▹Appendix C - Conversion tables◃ from page 141 and ▹Directory of objects for communication control◃ from page 90).

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

Parameter number, e.g. at the b maXX® 5000		CANopen field bus object
108.3 Status word	⇔	0x6041 Status word
108.1 Control word	⇔	0x6040 Control word
110.5 Set value selection HLG input	⇔	0x6042 Speed set value at the HLG
18.22 Speed actual value	⇔	0x6044 Control effort

2nd step: configure mapping

Writing of the first object to be mapped with index (0x6041), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x01 (TPDO axis 1).

Writing of the second object to be mapped with index (0x6044), subindex (0x00) and length (0x10) to 0x1A00 subindex 0x02 (TPDO axis 1).

Writing of amount of the mapped objects (0x02) to 0x1A00 subindex 0x00 (TPDO axis 1).

The content of object 0x1A00 is as follows:

0x1A00	0x00	0x02
	0x01	0x60410010
	0x02	0x60440010

Writing of the first object to be mapped with index (0x6040), subindex (0x00) and length (0x10) to 0x1600 subindex 0x01 (RPDO axis 1).

Writing of the second object to be mapped with index (0x6042), subindex (0x00) and length (0x10) to 0x1600 subindex 0x02 (RPDO axis 1).

Writing of the number of mapped objects (0x02) to 0x1600 subindex 0x00 (RPDO axis 1).

The content of Object 0x1600 is as follows:

0x1600	0x00	0x02
	0x01	0x60400010
	0x02	0x60420010

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 0x1C32 subindex 0x01 / 0x02. Additionally at synchronization to DC there still settings in the EtherCAT ASIC must be made. Here see ▹Synchronization (SYNC)◃ from page 101.

3rd step: synchronization

- Synchronization to DC

FBO 0x1C32 subindex 0x01 type of synchronization:

Request

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x2B	0x32	0x1C	0x01	0x02			

CS =0x 2B for 1 byte;
0x32 and 0x1C is crossed and compounds of 0x1C32;
SIX1 = 0x02 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
0x60	0x32	0x1C	0x01	0x02			

- Synchronizing to SM2

FBO 0x1C32 subindex 0x01 type of synchronization:

Request

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x2B	0x32	0x1C	0x01	0x22			

CS = 0x2B for 1 byte;
0x32 and 0x1C is crossed and compounds of 0x1C32;
SIX1 = 0x22 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
0x60	0x32	0x1C	0x01	0x22			

9.8.4 Entry in fieldbus process data of the controller

16 cyclic set values and 16 cyclic actual values can be replaced between the 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. 64 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 PDO axis 1 the contents are checked for validity (no dummy) in turns. If the object is valid then this is entered at the next spare position of the fieldbus process data list of the controller. If the PDO mapping is faulty (incorrect parameter number or the like), there is no cyclic communication started between slave and b maXX® controller.

NOTE!

If, in the existing PDO of the same direction repeatedly several identical object numbers are mapped, then the object appears several times in the process data configuration.

Thereby must be considered that the objects can possibly interact.

NOTE!

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

10

BASICS POWERLINK

10.1 Literature concerning POWERLINK

On behalf of basic information with reference to POWERLINK the following literature is recommended:

- [1]
  Ethernet **POWERLINK** Communication Profile Specification
  EPSG Draft Standard 301
  Ethernet POWERLINK Standardization Group (EPSG)

- [2]
  Ethernet **POWERLINK** XML Device Description
  EPSG Draft Standard 311
  Ethernet POWERLINK Standardization Group (EPSG)

- [3]
  www.ethernet-powerlink.org
  Ethernet **POWERLINK** Standardization Group (EPSG)
  Bonsaiweg 6
  D-15370 Fredersdorf

10.2 Basic principles POWERLINK

POWERLINK Version 2 (Ethernet type 0x88ab) is a published fieldbus system on the basis of real-time Ethernet, that integrates the mechanisms of CANopen completely.

Twisted pair cables (100Base-TX) serve as physical basis.

Net work

POWERLINK enables users to choose any topology. The network can be realized as line structure, tree structure, star structure or ring structure, whereas combinations are allowed, too. There is the possibility of adding and removing devices during run time (Hot-Plugging).

Bus access

The bus can be accessed via the CSMA/CD procedure (Carrier Sense Multiple Access / Collision Detection). Collisions may occur, as each participant is allowed to start sending his message, after recognizing the necessary idle bus. In this case, the collisions will be detected (Collision Detection) and the sending will be repeated after a random time interval. This ensures a transmission without data loss. For that reason, it is, of course, necessary that each participant can clearly be identified in the network by the respective MAC address.

The application of switches may lead to undefined conditions in the network.

MAC addressing

Each participant can send messages unrequested. Therefore, a clear sender and destination address is needed which is achieved by the MAC address.

IP addressing

Class Cv4 address 192.168.100.0 shall be used as net ID of a POWERLINK network. Each network supports 254 IP addresses whereby the last byte of the IP address (host ID) should correspond to the node number (node ID) of the participant.

192.168.100.POWERLINK Node ID

Net ID Host ID

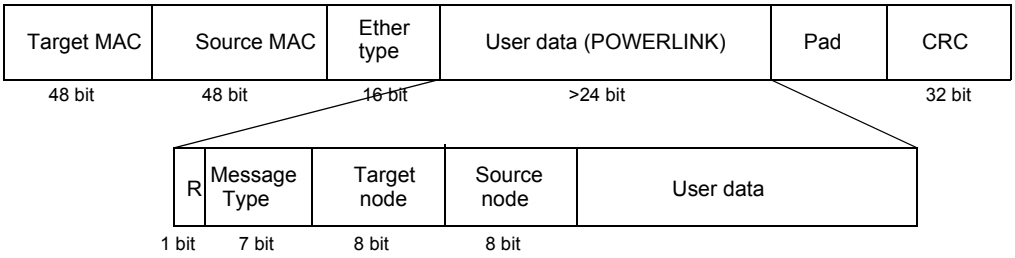
The following node IDs are reserved:

Address	Description
0x00	Invalid address
0xF0	POWERLINK default address of Managed Node
0xFB	Pseudo node address to be used by a node to address itself
0xFC	POWERLINK dummy node address
0xFD	POWERLINK default address of diagnostic device
0xFE	Default address of POWERLINK gateways/router
0xFF	POWERLINK broadcast address

As this is a class C network, the subnet mask of the POWERLINK node shall be 255.255.255.0.

Ethernet frame

The Ethernet frame consists of a header and the data payload. The header consists of the destination and the source MAC address as well as of the Ethertype field that contains some control information. The Ethernet payload field, including the POWERLINK frame, contains at least 46 and up to 1500 bytes. Concluding, the correctness of the frame is ensured by means of a checksum.



CANopen, CoE and POWERLINK frame

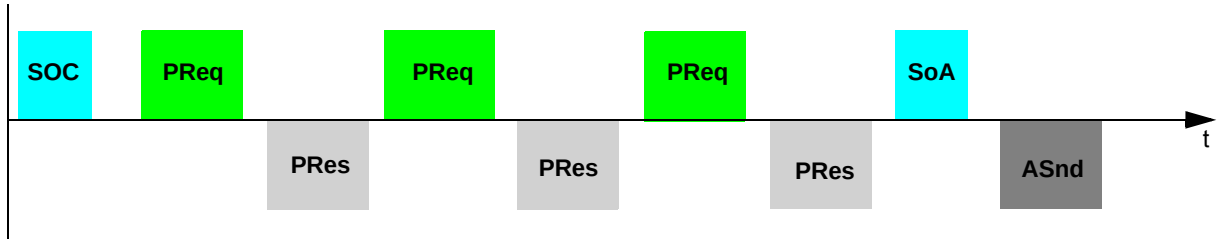
The POWERLINK frame consists of a header and the actual data payload as well. The header consists of a reserved-bit, the message type, the destination node and the source node. The following message types are defined.

Message type	ID	Designation	Use	Transfer type
SoC	0x01	Start of Cycle	Defines the start of a new cycle	Multicast
PReq	0x03	Poll Request	Ask for cyclic data of the CN	Unicast
PRes	0x04	Poll Response	Send current cyclic data of the CN	Multicast
SoA	0x05	Start of Asynchronous	Signalize the start of the asynchronous phase	Multicast
ASnd	0x06	Asynchronous Send	Sending of asynchronous data	Multicast

Determinism

The various participants in the network, the Controlled Nodes (CN), are controlled by a specific participant, the Managed Node (MN), and are only allowed to send, if they are asked to by the MN.

The POWERLINK cycle is divided into a synchronous and a asynchronous phase. At the beginning of the synchronous phase, the MN sends the SoC Frame. Subsequently, each single node is inquired by the MN with a PReq and responds with the PRes. After the cyclic phase, the MN starts the asynchronous phase with the sending of the SoA frame. A node determined by the MN can transmit acyclic data by means of a ASnd frame.



Device profile

POWERLINK supports the CANopen device profiles. These profiles describe application-specific and device-specific definitions, meaning of the data with regard to contents and device functionality. Amongst others, there are device profiles for drives, I/O modules, encoders or programmable devices. The option module POWERLINK for BM3300/5000 for the b maXX 2500 / 3300 / 5000 controller is implemented according to the device profile DSP402 (Drives and Motion Control).

11

POWERLINK AT B MAXX 2500 / 3300 / 5000

11.1 General information

The b maXX 2500 / 3300 / 5000 POWERLINK Controlled Node connects the b maXX 2500 / 3300 / 5000 via the POWERLINK bus with other POWERLINK nodes (e. g. PC, PLC, further b maXX devices, I/O modules).

Information according installation and handling with the device series b maXX 3300 is found in the manual 5.11018. Information according installation and handling with the device series b maXX 5000 is found in the manual 5.09021.

Information according the programming of the b maXX BM3300 controller is found in the parameter manual 5.12001. Information according the programming of the b maXX BM5000 controller is found in the parameter manual 5.09022.

11.2 Address Setting

The IP address setting of the b maXX 3300 / 5000 POWERLINK Controlled Node is described in the instruction manual b maXX 3300 / 5000 (5.11018 / 5.09021).

11.3 XDD file

The XDD file is a XML file and is for the description of the function range of a POWERLINK device. It is an electronic data sheet of the POWERLINK device. The XDD file is used by the POWERLINK Managed Node or the bus configurator. The XDD file contains information on all objects supported by the Controlled Node, the network management and further features.

The name extension of the XDD file is *.xdd.

The file can be downloaded from the download area on Baumüller's home page www.baumueller.de.

11.4 Diagnosis

Ethernet POWERLINK follows the Ethernet standard IEEE 802.3. Therefore standard tools e.g. Wireshark (Freeware) or OmniPeek and standard devices e.g. hubs or switches and PC network interfaces could be used.

11.5 Data Exchange and Parameterization

The access to data or parameter is made at POWERLINK via CANopen objects.

Accordant to profile structure it is differed between objects for communication control (indices 0x1XXX) and user- or device-specific objects. The latter are divided into objects according to profile CiA® 402 (indices 0x6XXX) and manufacturer-specific objects (indices 0x2XXX or 0x4XXX).

A listing of the 2XXX or 4XXX and the 6XXX objects are to be found in ▹Appendix B - Quick reference◃ from page 135.

NOTE!

The mapping of the manufacturer-specific parameter of the b maXX 5000 / 3300 / 2500 is specified in ▹B.1 2000 / 4000 object numbers (manufacturer-specific objects)◃ from page 135.

11.6 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 POWERLINK Controlled node in accordance with EPSG DS301.

Name	Index	Subindex	Data type	Default value
NMT_DeviceType_U32	0x1000	0x00	UINT32	0x00020192

This object is read-only and contains information on the related device (drive in accordance with CiA 402).

Name	Index	Subindex	Data type	Default value
NMT_CycleLen_U32	0x1006	0x00	UINT32	0x1000

If the sync frame is activated, the sync interval has to be set in accordance with the time of the sync frame (500 µs, 1000 µs, 2000 µs, 4000 µs or 8000 µs). The set time has an effect on the parameter 131.18 (Fieldbus cycle time) and 156.4 (Sync offset) of the b maXX controller.

Name	Index	Subindex	Data type	Default value
NMT_ManufactDevName_VS	0x1008	0x00	VAR	-

This object is read-only. It contains the following character strings: „b maXX 5000“.

Name	Index	Subindex	Data type	Default value
NMT_ManufactHwVers_VS	0x1009	0x00	VAR	-

This object is read-only. It contains the present hardware version of the option module, e.g. the character string: „01.00“.

Name	Index	Subindex	Data type	Default value
NMT_ManufactSwVers_VS	0x100A	0x00	VAR	-

This object is read-only. It contains the present POWERLINK Stack Version of the option module, e.g. the character string: „EPL V2 V1.8 r1“.

Name	Index	Subindex	Data type	Default value
NMT_IdentityObject_REC	0x1018	0x00	UINT8	0x04
VendorId_U32		0x01	UINT32	0x0000015A
ProductCode_U32		0x02	UINT32	0x00010000
RevisionNo_U32		0x03	UINT32	0x00010000
SerialNo_U32		0x04	UINT32	0x00000000

In this object there is information about the device.

RevisionNo_U32 contains the current version of firmware e.g. 00010002 for FW 01.02.

Name	Index	Subindex	Data type	Default value
CFM_VerifyConfiguration_REC	0x1020	0x00	UINT8	0x02
ConfDate_U32		0x01	UINT32	0x00000000
ConfTime_U32		0x02	UINT32	0x00000000

This object contains the information on the local configuration time of the device.

ConfDate_U32 contains the configuration time which including the days since 01.01.1984.

Name	Index	Subindex	Data type	Default value
NMT_InterfaceGroup_Xh_REC	0x1030	0x00	UINT8	0x09
InterfaceIndex_U16		0x01	UINT16	0x0001
InterfaceDescription_VSTR		0x02	VISIBLE_STRING	"Interface 1"
InterfaceType_U8		0x03	UINT8	0x06
InterfaceMtu_U16		0x04	UINT16	0x1500
InterfacePhysAddress_OSTR		0x05	OCTET_STRING6	0x06
InterfaceName_VSTR		0x06	VISIBLE_STRING	"Interface 1"
InterfaceOperStatus_U8		0x07	UINT8	0x01
InterfaceAdminState_U8		0x08	UINT8	0x01
Valid_BOOL		0x09	BOOL	0x1
HubPortEnableMask_U64		0x0A	UINT64	0x00000000 00000000

Parameters of the network interface are configured by means of this object via SDO.

Name	Index	Subindex	Data type	Default value
SDO_SequLayerTimeout_U32	0x1300	0x00	VAR	-

This object contains the timeout value in [ms] for the recognition of an SDO abort.

Name	Index	Subindex	Data type	Default value
PDO_RxCommParam_00h_REC	0x1400	0x00	UINT8	0x02
Node_ID_U8		0x01	UINT8	0x00
MappingVersion_U8		0x02	UINT8	0x00

This object contains information on the receive PDO.

Name	Index	Subindex	Data type	Default value
PDO_RxMappParam_00h_REC	0x1600	0x00	UINT8	0x01
		0x01	UINT64	0x0010000000006040
		:	:	
		n	UINT64	

This object contains the information of receive-PDO. The total number of the following entries is in subindex 0x00.

The total number of the mapped objects may not exceed the controller reference value limits of a maximum of 16 objects. (also see ▹PDO Mapping◃ from page 123).

Name	Index	Subindex	Data type	Default value
PDO_TxCommParam_00h_REC	0x1800	0x00	UINT8	0x02
Node_ID_U8		0x01	UINT8	0x00
MappingVersion_U8		0x02	UINT8	0x00

This object contains information on the transmit PDO.

Name	Index	Subindex	Data type	Default value
PDO_TxMappParam_00h_REC	0x1A00	0x00	UINT8	0x01
		0x01	UINT64	0x0010000000006041
		:	:	
		n	UINT64	

This object contains the contents of transmit PDO. The total number of the following entries are in subindex 0x00.

The total number of mapped objects may not exceed the controller reference value limits of a maximum of 16 objects. (also see ▹PDO Mapping◃ from page 123).

11.7 Net work management (NMT)

Commands of the network management serve mainly the control of the communication states in POWERLINK net.

The state diagram of the communication of the POWERLINK Controlled Node is illustrated here:

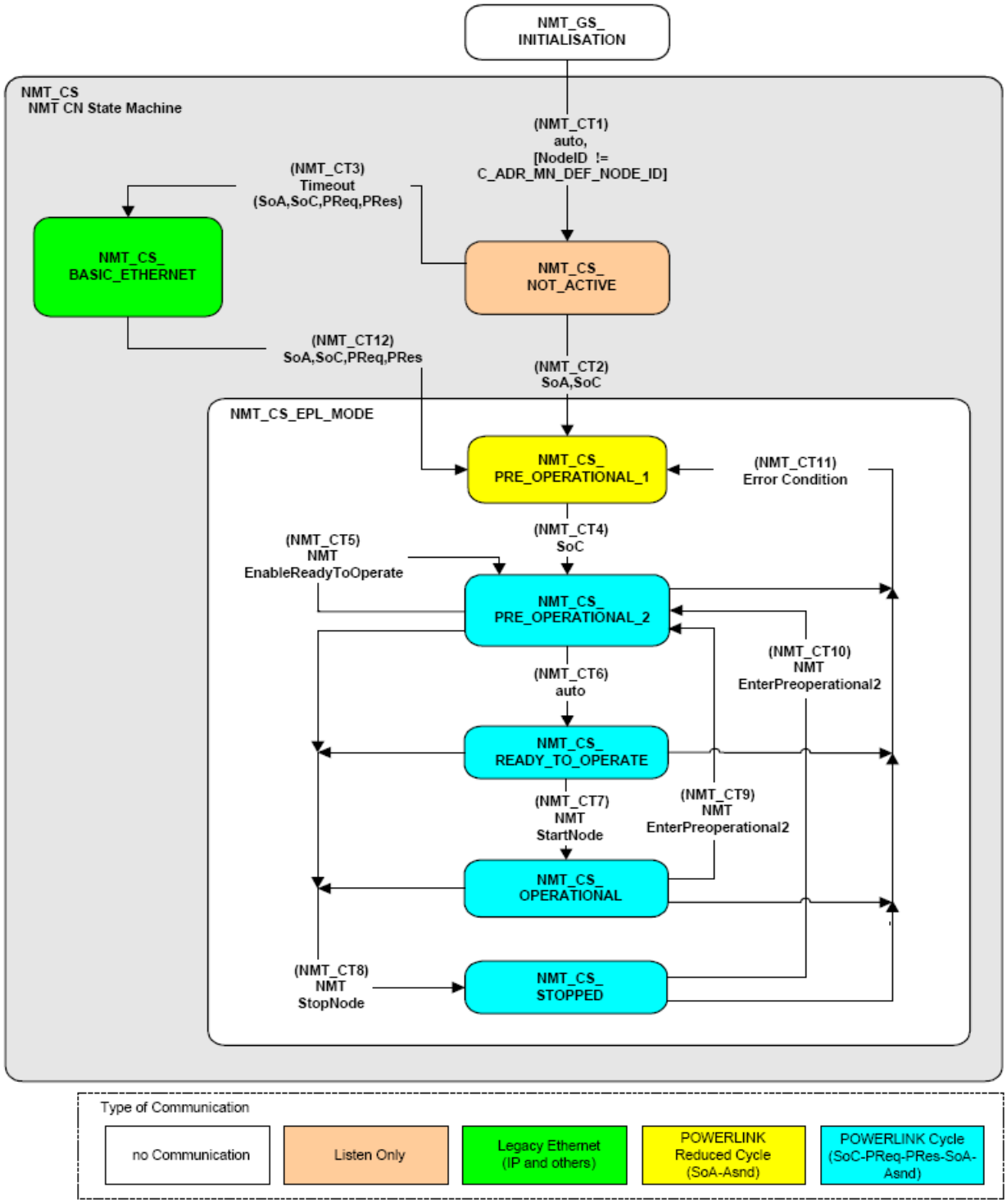


Figure 23: State diagram POWERLINK Controlled Node

State	Description
NMT_CS_NOT_ACIVE	Transient state for starting the CN and for identifying in the network
NMT_CS_PRE_OPERATIONAL_1	Identification of the node by the MN via IdentRequest
NMT_CS_PRE_OPERATIONAL_2	PDO configuration by the MN; configuration examination
NMT_CS_READY_TO_OPERATE	Signalizing the service readiness to the MN; Starting the PDO communication
NMT_CS_OPERATIONAL	Normal operating state of the CN; CN takes part at the cyclic data exchange
NMT_CS_STOPPED	CN is passive; controlled shut down of the CN; CN does not take part at the cyclic data exchange
NMT_CS_BASIC_ETHERNET	CN takes part at usual ethernet communication according to IEEE 802.3

11.8 Service data (SDO)

Service data objects (SDO) serve as an exchange of messages without real-time requests. SDOs are used for parameterizing CNs and for setting the communication references for PDOs. Access on data occurs only via the object list. SDOs are always acknowledged data, i. e. the transmitter receives an acknowledge from the receiver. The data exchange by means of SDOs can only be executed asynchronously (see also ▹Synchronization (SYNC)◃ from page 121).

SDOs follow the client-server-model. The client (MN) initiates the communication and the server (CN) responds to it. A server is not able to start a SDO communication. The Baumüller POWERLINK option module supports a server SDO but no client SDO.

The MN starts the asynchronous cycle by sending the Start of Asynchronous (SoA) frame. The SDO transfer is answered by the CN via ASnd.

11.8.1 Frame structure SoA

The structure of the Start of Asynchronous (SoA) frame is as follows:

	Bit offset							
Byte offset	7	6	5	4	3	2	1	0
0	Message Type = SoA (0x05)							
1	Destination = Broadcast address (0xFF)							
2	Source = Node ID of the MN (0xF0)							
3	NMTStatus							
4	res	res	res	res	res	EA/res	ER/res	res
5	reserved							
6	RequestedServiceID (see the following table)							
7	RequestedServiceTarget							
8	EPLVersion							
9 ... 45	reserved							

EA (Exception Acknowledge)
ER (Exception Reset)

RequestedService	**ID**	**Description**
NoService	0x00	Asynchronous slot is not assigned to any node
IdentRequest	0x01	Identification of inactive nodes
StatusRequest	0x02	Request the status and error information of single nodes
NMTRequestInvite	0x03	Requesting a node to send an indicated NMT command.
Manufacturer specific	0xA0 .. 0xFE	Manufacturer specific purposes
UnspecificInvite	0xFF	Requesting a node to send an indicated transmit request.

11.8.2 Frame structure ASnd

The structure of the asynchronous Send (ASnd) frame is as follows:

	Bit offset							
Byte offset	7	6	5	4	3	2	1	0
0	Message Type = ASnd (0x06)							
1	Destination (Node ID of the addressed nodes)							
2	Source (Node ID of the transmitting node)							
3	Service ID (see the following table)							
4 .. 7	Sequence Layer Protocol							
8 .. k-1	Command Layer Protocol							
k .. 1472	SDO Payload Data							

RequestedService	**ID**	**Description**
IdentResponse	0x01	Response of a node to an IdentRequest via SoA
StatusResponse	0x02	Response of a node to a StatusRequest via SoA
NMTRequest	0x03	Response of a CN to a NMTRequestInvite via SoA
NMTCommand	0x04	Response of the MN to an internal request or an external request via NMTRequest
SDO	0x05	Response of a CN to an UnspecificInvite via SoA
Manufacturer specific	0xA0 .. 0xFE	Manufacturer specific service

11.8.3 Error reactions

Faulty SDO accesses are rejected by means of abort codes. The structure of these abort frames is identical to the SDO frame described in ▹Frame structure SoA◃ on page 118.

The data field contains an abort code with 4 bytes.

The following different messages may occur in case of faulty SDO accesses:

Abort code	Description
0x05040000	SDO protocol timed out.
0x05040001	Unknown command ID
0x05040002	Invalid block size
0x05040003	Invalid sequence number
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write-only object
0x06010002	Attempt to write a read-only object
0x06020000	Object does not exist in the object dictionary
0x06040041	Object cannot be mapped to the PDO.
0x06040042	The number and length of the objects to be mapped would exceed PDO length
0x06040043	General parameter incompatibility
0x06040044	Invalid heartbeat declaration
0x06040047	General internal incompatibility in the device
0x06060000	Access failed due to an hardware error
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Sub-index does not exist
0x06090030	Value range of parameter exceeded (only for write access)
0x06090031	Value of parameter written too high
0x06090032	Value of parameter written too low
0x06090036	Maximum value is less than minimum value
0x08000000	General error
0x08000020	Data cannot be transferred or stored to the application.
0x08000021	Data cannot be transferred or stored to the application because of local control
0x08000022	Data cannot be transferred or stored to the application because of the present device state.
0x08000023	Object dictionary is not available
0x08000024	Configuration data set is empty

11.9 Synchronization (SYNC)

The Start of Cycle (SoC) frame is used for synchronizing the CNs. This frame is unconfirmed (multicast) and is sent by the MN. It does not contain any data. The POWERLINK Controlled Node is able to receive SoC frames.

The receipt of a SoC frame generates an interrupt on the POWERLINK for BM3300/5000; this interrupt is forwarded to the b maXX controller. Thus, this signal can be used for the synchronization of the b maXX controller. Within a communication cycle all corresponding frames have to be sent to all configured slaves and in doing so the number of nodes and the processing time have to be considered. The cycle time for the SoC frame is set in object 0x1006. For this see ▹Directory of objects for communication control◃ from page 112. Furthermore, the communication cycle time has to be stored in the data set of the controller.

The so-called multiplexing in POWERLINK allows not to request single nodes in each cycle. Thus, several nodes can share one time section in the transmission phase, so that the bandwidth of the synchronous phase can be used in an optimum way on the POWERLINK bus. The configuration of this assignment is effected by the Managed Node.

11.9.1 Frame structure SoC

The structure of the Start of Cycle (SoC) frame is as follows:

	Bit offset							
Byte offset	7	6	5	4	3	2	1	0
0	Message Type = SoC (0x01)							
1	Destination = Broadcast address (0xFF)							
2	Source = Node ID of the MN (0xF0)							
3	reserved							
4	MC	PS	res	res	res	res	res	res
5	reserved							
6 ... 13	NetTime (optional)							
14 ... 21	RelativeTime (optional)							
22 ... 45	reserved							

MC (Multiplexed Cycle Completed)
PS (Prescaled Slot)

11.10 Process data (PDO)

Process data objects are unconfirmed frames that are optimized for the exchange of data with real-time requirements. There are two kinds of PDOs, differing in the direction of the data transmission from the perspective of the device. The POWERLINK Controlled Node for b maXX controllers supports a transmit PDO (TPDO) as well as a receive PDO (RPDO). Up to 16 objects can be transmitted in each PDO.

The PDO communication in POWERLINK is executed by synchronous PReq respectively PRes frames. In the synchronous phase the MN sends the PollRequest (PReq) as unicast frame. The corresponding CN sends the PollResponse (PRes) as broadcast.

The format of the data exchange has to be defined before starting the communication between sender and receiver (mapping). The sending and receiving of PDOs can be initiated in different kind of ways (see ▹PDO Mapping◃ from page 123).

NOTE!

All objects, which were configured in the PDOs are transmitted between the POWERLINK Controlled Node and the b maXX controller as cyclic data (also see ▹Communication flow◃ on page 25). As the cyclic data transmission is only made in the state of NMT_CS_OPERATIONAL, the communication monitoring in ProDrive should be only in this status be activated.

11.10.1 Frame structure PReq and PRes

The structure of the PollRequest (PReq) frame is as follows:

	Bit offset							
Byte offset	7	6	5	4	3	2	1	0
0	Message Type = PReq (0x03)							
1	Destination = Node ID of the CN							
2	Source = Node ID of the MN (0xF0)							
3	reserved							
4	res	res	MS	res	res	EA	res	RD
5	reserved							
6	PDOVersion							
7	reserved							
8 ... 9	Size (Size of the process data in byte)							
10 ... n	Payload							

MS (Multiplexed Slot)
EA (Exception Acknowledge)
RD (Ready)

The structure of the PollResponse (PRes) frame is as follows:

	Bit offset							
Byte offset	7	6	5	4	3	2	1	0
0	Message Type = PRes (0x04)							
1	Destination = Broadcast Address (0xFF)							
2	Source = Node ID of the CN							
3	NMT status							
4	res	res	MS	EN	res	res	res	RD
5	res	res	PR			RS		
6	PDOVersion							
7	reserved							
8 ... 9	Size (Size of the process data in byte)							
10 ... n	Payload							

MS (Multiplexed Slot)
EN (Exception New)
RD (Ready)
PR (Priority)
RS (Request To Send)

11.10.2 PDO Mapping

Mapping is a method of assigning variables/objects to PDOS. With these PDOs these variables/objects are transmitted via POWERLINK. 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.

Two objects each exist for the TPDO and the RPDO (see also ▹Directory of objects for communication control◃ from page 112). One of these objects determines the contents of the PDO, the second one determines the communication relation respectively triggering.

Process data object	Object for content	Object for the communication relation
TPDO	0x1A00	0x1800
RPDO	0x1600	0x1400

NOTE!

The mapping **cannot** be changed in state NMT_CS_OPERATIONAL. A new mapping will be activated only after transition to NMT_CS_READY_TO_OPERATE.

A maximum of 1490 bytes are provided by the PReq respectively the PRes data frame for the payload data transmission. The POWERLINK Controlled Node is able to transmit the contents of up to 16 variables / objects in each direction. The logic content of the user data is determined by the mapping.

Specific information about the objects to be mapped are needed for this determination:

Object index, subindex and the length of the date as well as the sequence of the objects to be mapped. The objects to be mapped are written from the object directory to the subindices starting with 0x01 of the mapping object (0x1600, 0x1A00), e. g. the value 0x0010.0000.0000.6040 is entered in object 0x1600 subindex 0x01. This means that the first two bytes (length 0x0010, offset 0x0000) of the data received in RXPD are written on the control word (object 0x6040, Subindex 0x00). The object 0x6040 is transferred to the b maXX parameter **108.1** control word (see also ▹Appendix C - Conversion tables◃ from page 141). This means that the first word of the frame received in RPDO is written on the control word of the b maXX. The number of objects to be mapped (number of subindices occupied with valid objects) has to be entered in subindex 0x00.

The structure of the subindices for the mapping objects 0x1600 and 0x1A00 is as follows:

Byte offset	Name	Description
0 ... 1	Index	Index of the object to be mapped
2	Subindex	Subindex of the object to be mapped
3	Reserved	
4 ... 5	Offset	Offset related to start of PDO payload (Bit count)
6 ... 7	Length	Length of the object to be mapped (Bit count)

NOTE!

On setting the mapping in the mapping parameters (0x1600, 0x1A00), it is necessary to describe the respective subindex 0x00 with the correct number of mapped objects.

Set values: The permissible cyclical set values are marked in a table with the column 'PDO mapping' as 'RX', see table ▹6000 object numbers (device profile CiA® 402)◃ from page 137.

Actual values

The permitted cyclic actual values are marked in a table in column „PDO mapping“ as „TX“, see table ▹6000 object numbers (device profile CiA® 402)◃ from page 137. A detailed description of the b maXX parameters see parameter manual BM5000 (5.09022) or parameter manual BM3300 (5.12001).

Incorrect mapping configurations (invalid objects in 0x1600, 0x1A00) are signaled with abort codes via SDO.

The cyclic set-/actual values are continuously initialized into the fieldbus process data list of the controller, i. e. the first set value of RPDO is on first position in the process data list, the second set value on second position a.s.o. Analog for the actual value initialization the first actual value of TPDO is on first position in the process data list, the second actual value of TPDO on second position a.s.o.

Writing of similar field bus objects (FBO) via service data SD and process data PD.

Normally, PD write accesses overwrite SD write accesses cyclically on the same FBO. In single cases a write access via SD may have been successful, but this is not certain.

NOTE!

In this context you please avoid access to the same field bus object via SD and via PD.

11.11 Configuration Example with B&R X20 PLC

The following chapter describes the configuration of the POWERLINK Controlled Node for b maXX 2500 / 3300 / 5000 controller with a B&R X20 PLC by means of Automation Studio (V4.0.16.81).

To integrate the POWERLINK Controlled Node, the XDD-file must be imported into the Automation Studio Project.

In this connection the device description file must be chosen in the menu under *Extras* → *Import Fieldbus Device..* . The file can be downloaded from the download area on Baumuller's homepage for different device series.

As the device description is store in the Automation Studio project file, this process must be repeated on creating a new project.

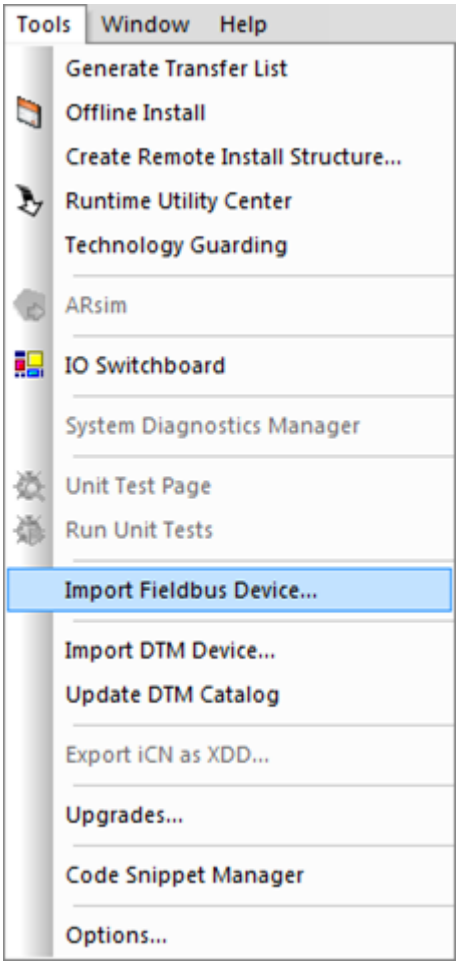


Figure 24: Configuration - Import fieldbus device

The imported device is shown under network type *POWERLINK* in the *Hardware Catalog of the Toolbox.*

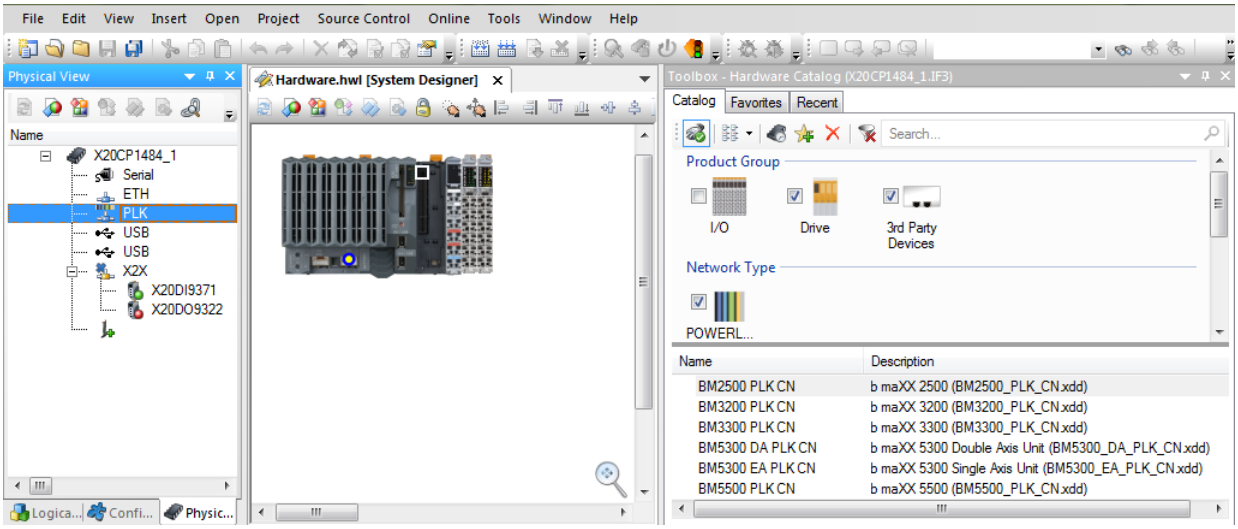


Figure 25: Configuration - Hardware Catalog

The corresponding Controlled Node is shown in the *Hardware Catalog* under the following designation:

BM2500_PLK_CN	b maXX 25xx
BM3300_PLK_CN	b maXX 32xx / 33xx
BM5000_NWR_PLK_CN	b maXX 51xx Active mains rectifier unit
BM5000_EA_PLK_CN	b maXX 52xx / 53xx Single axis unit
BM5000_DA_PLK_CN	b maXX 52xx / 53xx Double axis unit
BM5000_MA_PLK_CN	b maXX 54xx / 55xx Mono unit

The device can be added and connected with the *System Designer* using drag and drop or doubleclick.

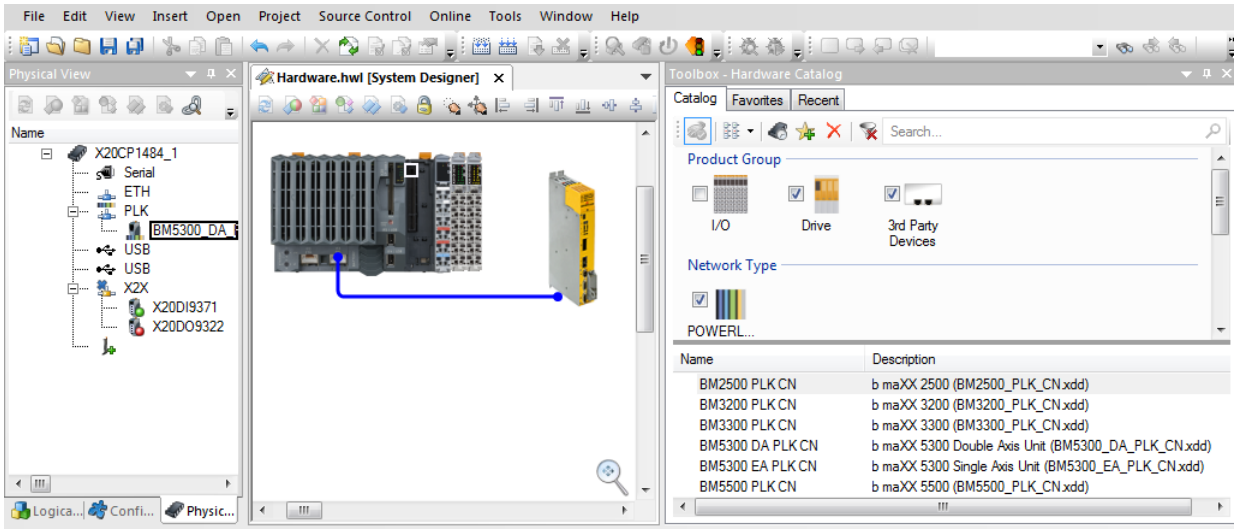


Figure 26: Configuration - System Designer

The controlled node is shown in Automation Studio *Physical View* after successful import.

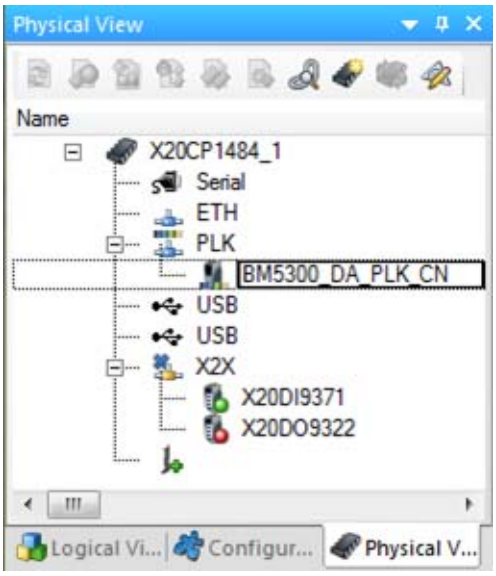


Figure 27: Configuration - Controlled Node in Physical View

11.12 PollResponse Chaining (from version 01.10)

PollResponse chaining is used to reduce the telegram number in the POWERLINK network and to increase the performance this way. Instead of the usual poll request/poll response telegram sequence the individual poll request telegrams are summarized to the controlled nodes (CN) in a poll response telegram of the managing node (PResMN). This is send by the broadcast to all nodes. This procedure mainly reduces the processing time.

Figure 28: Without PollResponse Chaining

Figure 29: With PollResponse Chaining

It is possible to combine conventional poll request/poll response nodes with poll response chaining nodes in one single network as well.

11.13 Ethernet communication (from version 01.10)

The user software ProDrive configures the POWERLINK controlled node for b maXX 2500 / 3300 / 5000. The access by ProDrive can be made via Ethernet. Thereby the controlled node (CN) can directly be approached from the PC or indirectly can be operated by an Ethernet gateway.

11.13.1 Direct Ethernet communication

When having a direct connection with a PC the IP address must be in the same subnet as the POWERLINK controlled nodes. At the line topology the connection to the PC can be made at the output port of the last slave as well as at the input port of the first slave.

11.13.1.1 Configuration of the IP settings with ProDrive

ProDrive reads and configures the IP setting of the POWERLINK controlled node for b maXX 2500 / 3300 / 5000.

In the menu under *Tools* → *Ethernet configuration* it is possible to browse POWERLINK controlled nodes for b maXX 2500 / 3300 / 5000.

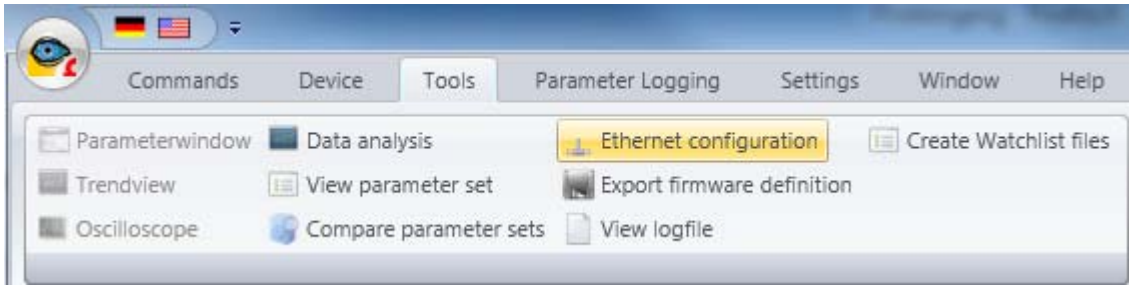


Figure 30: Ethernet configuration

The settings for the IP address, gateway address and subnet mask can be changed at the detected devices.

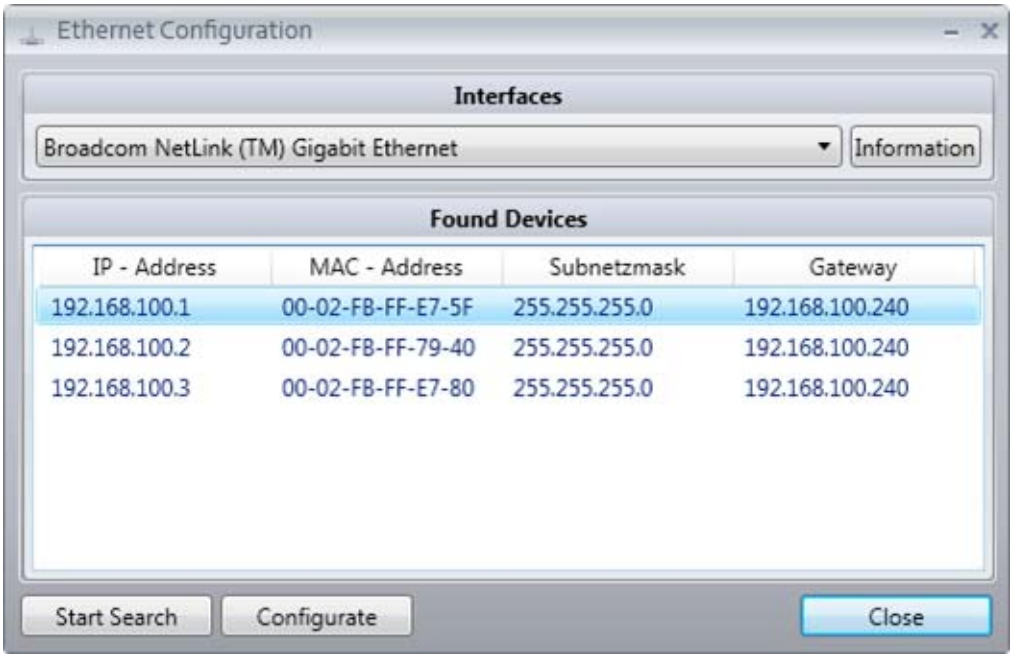


Figure 31: Detected devices

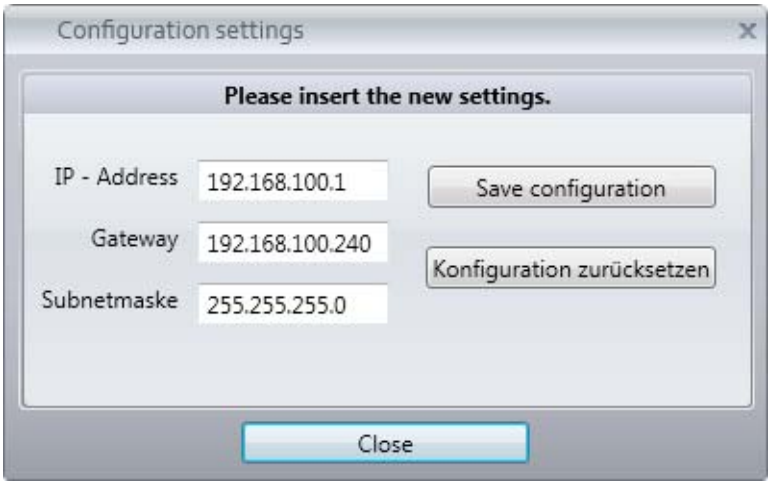


Figure 32: Configure device

11.13.2 Communication with ProDrive

The operating software ProDrive can be configured by a direct TCP/IP connection of the POWERLINK controlled node for b maXX 2500 / 3300 / 5000. The access of the Ethernet communication is executed by the port 20547.

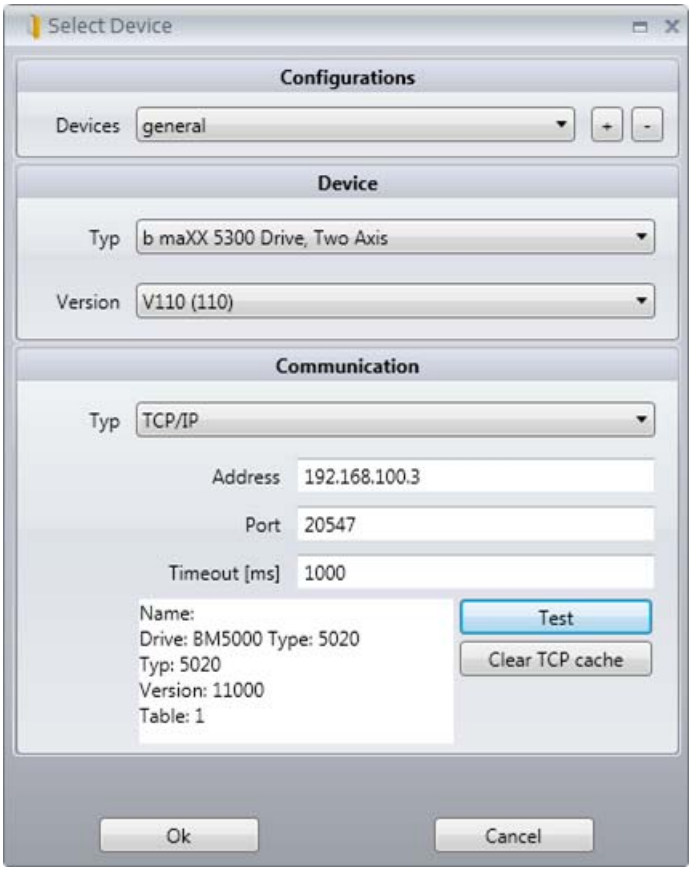


Figure 33: Select device

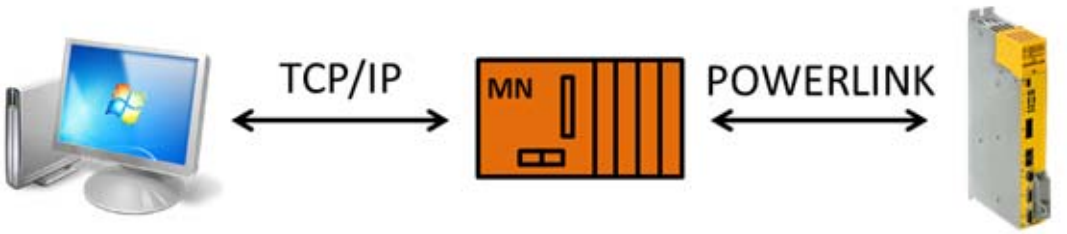
11.13.3 Indirect Ethernet communication

The Ethernet access to the POWERLINK controlled node can be made by an Ethernet gateway. This gateway could e.g. be a managing node (MN) if this function is supported. Thereby the Ethernet telegrams are tunneled in the provided asynchronous time intervals. The size of the user data range of the asynchronous POWERLINK telegrams (maximum transmission unit (MTU)) thereby should be 988 bytes. This corresponds to the ProDrive telegram sequence sizes.

An indirect Ethernet communication must consider the correct IP setting of the POWERLINK controlled nodes and of the gateway. For routing the Ethernet address of the gateway must be entered in the routing table of the PC, such as entering by the command window.

Example:		ROUTE ADD [host] MASK [subnet] [gateway]
	host	host address of the POWERLINK net work e.g. 192.168.100.0
	subnet	subnet mask for routing entry e.g. 255.255.255.0
	gateway	address of the gateway e.g. 192.168.1.2

Example of a MN as a gateway:



IP address: 192.168.1.1	IP address TCP/IP: 192.168.1.2	IP address: 192.168.100.1
Net mask: 255.255.255.0	Net mask TCP/IP: 255.255.255.0	Net mask: 255.255.255.0
Route: ROUTE ADD 192.168.100.0 MASK 255.255.255.0 192.168.1.2	POWERLINK NAT subnet: 192.168.100.0	Gateway: 192.168.100.240
	Net mask POWERLINK: 255.255.255.0	
	MTU size: 988	

The address of MN in the POWERLINK net work is 192.168.100.240.

11.14 Dynamic Node Allocation (from version 01.10 onward)

If the dynamic node allocation is supported by the MN the node number of the POWERLINK controlled nodes for b maXX 2500 / 3300 / 5000 can automatically be set by the MN.

The nodes of all controlled nodes which receive their number by dynamic node allocation must be set to node number 0. The node number accords to the last digit of the IP address. The setting of the IP address is described in the instruction handbook 3300 / 5500 (documentation 5.11018 / 5.09021).

The correct cabling of in- and output port during dynamic node allocation must be considered because the output port (port 2) of the CNs was temporarily deactivated.

The dynamic node allocation is supported only completely at the line topology. The node numbers thereby are incremented sequentially. The other topologies the initial nodes of a line as the main position must be allocated a fixed node number.

When making use of dynamic node allocation procedure the initial operation with the node numbers is delayed, which receive their number by the MN.

APPENDIX A - ABBREVIATIONS

CD	Collision Detection
CN	Controlled Node
CoE	CAN application protocol over EtherCAT
CSMA	Carrier Sense Multiple Access
DC	Distributed Clocks
DSP	Draft Standard Proposal
EMCY	Emergency Telegramm
FB	Function block
FBO	Field bus object
FMMU	Fieldbus Memory Management Unit
ID	Ident number
MAC	Media Access Control
MN	Managed Node
NMT	Net work management
PC	Personal Computer
PDO	Process Data Object
SDO	Service Data Object
SIX	Subindex
SPS	Programmable logic controller
SYNC	Synchronization
XDD	XML Device Description

B

APPENDIX B - QUICK REFERENCE

The following quick reference shows the connection between CANopen object numbers and the b maXX controller parameter numbers (see parameter manual b maXX 5000 (5.09022) or parameter manual b maXX 3300 (5.12001)).

B.1 2000 / 4000 object numbers (manufacturer-specific objects)

The following rule is true for the mapping of the manufacturer-specific objects:

Object ID = 0x2000 + AxisOffset + FbNumber + (0x200 * Instance)

with AxisOffset axis 1: 0x000
AxisOffset axis 2: 0x2000

object subindex = parameter number

CAN Index: bit's	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	x	B1	B2	Instance				Res	FB funktion blocks							
		0	1	Axis 1												
		1	0	Axis 2												

Special objects for array and struct parameters are defined in the reserved area.

Example **Error number axis 1** — Parameter number 100.5.0.0

FbNumber	= 100 = 0x64
Instance	= 0
Axis Offset	= 0
Object ID	= 0x2000 + 0 + 0x64 + 0 = 0x2064
Subindex	= 5
→	Object ID = 0x2064, Subindex 5

Isq actual value axis 2 — Parameter number 47.3.0.0

FbNumber	= 47 = 0x2F
Instance	= 0
Axis Offset	= 0x2000
Object ID	= 0x2000 + 0x2000 + 0x2F + 0 = 0x402F
Subindex	= 3
→	Object ID = 0x402F, Subindex 3

Position actual value angle 32 bit instance 1 — Parameter number 106.10.1.0

FbNumber	= 106 = 0x6A
Instance	= 1
Axis Offset	= 0
Object ID	= 0x2000 + 0 + 0x6A + (0x200*1) = 0x226A
Subindex	= 10
→	Object ID = 0x226A, Subindex 10

The error list of the drive is set as an independent object.

Object ID: 0x2120 for axis 1 or 0x4120 for axis 2.

The respective entry is addressed in the error list with the subindex (starting with 1)

B.2 6000 object numbers (device profile CiA® 402)

It is possible to access some parameters of the controller via 2000 / 4000-objects **as well as** via one or several 6000s.

NOTE!

There may be different standardizations for the 6000 objects and the 2000 / 4000 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 CiA® 402
Index	Subindex				
0x6007	0x00	-	TX / RX	rw	Common Entries
0x603F	0x00	**100.5**	TX / RX	rw	Common Entries
0x6040	0x00	**108.1**	TX / RX	rw	Device Control
0x6041	0x00	**108.3**	TX	ro	Device Control
0x6042	0x00	**110.5**	TX / RX	rw	Velocity Mode
0x6043	0x00	**18.21**	TX	ro	Velocity Mode
0x6044	0x00	**18.22**	TX	ro	Velocity Mode
0x6046	0x01	**110.16**	TX	ro	Velocity Mode
0x6046	0x02	**110.15**	TX / RX	rw	Velocity Mode
0x604F	0x00	**110.6**	TX / RX	rw	Velocity Mode
0x6050	0x00	**110.7**	TX / RX	rw	Velocity Mode
0x605A	0x00	**108.13**	TX	rw	Device Control
60x05B	0x00	**108.14**	TX	rw	Device Control
0x605C	0x00	**108.15**	TX	rw	Device Control
0x6060	0x00	**109.1**	TX / RX	rw	Device Control
0x6061	0x00	**109.2**	TX	ro	Device Control
0x6064	0x00	**121.9**	TX	ro	Position Control Function
0x6067	0x00	**121.5**	TX	rw	Position Control Function
0x6068	0x00	**121.6**	TX / RX	rw	Position Control Function
0x6069	0x00	**106.10**	TX	ro	Profile Velocity Mode
0x606A	0x00	-	-	ro	Profile Velocity Mode

CANopen object number		Parameter No.	PDO mapping	Access type	Operating mode acc. to CiA® 402
Index	Subindex				
0x606B	0x00	**18.21**	TX	ro	Profile Velocity Mode
0x606C	0x00	**18.22**	TX	ro	Profile Velocity Mode
0x6071	0x00	**18.50**	TX / RX	rw	Profile Torque Mode
0x6072	0x00	**138.14**	TX / RX	rw	Profile Torque Mode
0x6077	0x00	**47.3**	TX	ro	Profile Torque Mode
0x607A	0x00	**118.16 / 136.3**	TX / RX	rw	Profile Position Mode
0x607C	0x00	**120.3**	TX	rw	Homing Mode
0x607D	0x01	**121.3**	TX	rw	Profile Position Mode
0x607D	0x02	**121.4**	TX	rw	Profile Position Mode
0x607E	0x00	**179.1**	-	rw	Factor Group
0x6080	0x00	**110.13**	TX	rw	Profile Position Mode
0x6081	0x00	**118.11**	TX	rw	Profile Position Mode
0x6083	0x00	**118.12**	TX	rw	Profile Position Mode
0x6084	0x00	**118.13**	TX	rw	Profile Position Mode
0x6085	0x00	**110.8**	TX	rw	Profile Position Mode
0x6086	0x00	**118.2**	TX / RX	rw	Profile Position Mode
0x608F	0x01	**179.2**	-	rw	Factor Group
0x608F	0x02	**179.3**	-	rw	Factor Group
0x6090	0x01	**179.4**	-	rw	Factor Group
0x6090	0x02	**179.5**	-	rw	Factor Group
0x6091	0x01	**179.6**	-	rw	Factor Group
0x6091	0x02	**179.7**	-	rw	Factor Group
0x6092	0x01	**179.8**	-	rw	Factor Group
0x6092	0x02	**179.9**	-	rw	Factor Group
0x6098	0x00	**120.4**	TX	rw	Homing Mode
0x6099	0x01	**120.5**	TX	rw	Homing Mode
0x6099	0x02	**120.6**	TX	rw	Homing Mode
0x609A	0x00	**120.7**	TX	rw	Homing Mode
0x60B1	0x00	**18.68**	TX / RX	rw	Cyclic Synchronous Position Mode

CANopen object number		Parameter No.	PDO mapping	Access type	Operating mode acc. to CiA® 402
Index	Subindex				
0x60B8	0x00	**124.2, 124.1, 124.30, 124.35**	-	rw	Homing Mode
0x60B9	0x00	**124.3**	TX	ro	Homing Mode
0x60BA	0x00	**124.31**	TX	ro	Homing Mode
0x60BB	0x00	**124.32**	TX	ro	Homing Mode
0x60BC	0x00	**124.33**	TX	ro	Homing Mode
0x60BD	0x00	**124.34**	TX	ro	Homing Mode
0x60F4	0x00	**18.60**	TX	ro	Cyclic Synchronous Position Mode
0x60FB	0x0A	**18.76**	TX	ro	Position Control Function
0x60FB	0x0D	**118.6**	TX / RX	rw	Position Control Function
0x60FB	0x15	**118.10**	TX / RX	rw	Position Control Function
0x60FB	0x17	**136.5**	TX / RX	rw	Position Control Function
0x60FB	0x18	**120.10**	TX / RX	rw	Position Control Function
0x60FB	0x19	**120.8**	TX / RX	rw	Position Control Function
0x60FF	0x00	**110.4**	TX / RX	rw	Profile Velocity Mode
0x6502	0x00	**-**	TX	ro	Device Control
0x6510	0x03	**102.18**	TX	ro	Info
0x6510	0x04	**102.19**	TX	ro	Info
0x6510	0x05	**102.22**	TX	ro	Info
0x6510	0x06	**122.21**	TX	ro	Info
0x6510	0x07	**102.10**	TX	ro	Info
0x6510	0x08	**102.14**	TX	ro	Info
0x6510	0x09	**102.15**	TX	ro	Info
0x6510	0x0A	**102.1**	TX	ro	Info
0x6510	0x0B	**102.2**	TX	ro	Info
0x6510	0x0C	**102.4**	TX	ro	Info
0x6510	0x0D	**102.3**	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 CiA® 402
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 re-scaling	Comment
Abort connection option code	0x6007	▸					▸	0x6007	
	y = -2^{15} ... 2^{15}-1							y = -2^{15} ... 2^{15}-1	
Manufacturer-specific								y = -2^{15} ... -1	
No action								y = 0	
Fault signal								y = 1	
Disable voltage command								y = 2	
Quick stop command								y = 3	
Reserved								y = 4 ... 2^{15}-1	
Error Code	0x603F /ro				**First Error**	**100.5**		0x603F	Conversion of the internal controller error in emergency messages according to CiA DSP 402-3
	x = 0 ... 0xFFFF					x = 0 ... 0xFFFF		y = 0 ... 0xFFFF	
Control word	0x6040		**108.1**		**Control word**	**108.1**		0x6040	Bit 4 is ignored in operating mode = 9 (Cyclic sync velocity mode)
	x = 0 .. 0xFFFF	▸	y = x			x = 0 .. 0xFFFF	▸	y = x	
Switch on	Bit 0	▸	unchanged		Switch on	Bit 0	▸	unchanged	
Disable voltage	Bit 1	▸	unchanged		Inhibit voltage	Bit 1	▸	unchanged	
Quick stop	Bit 2	▸	unchanged		Quickstop	Bit 2	▸	unchanged	
Enable operation	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	
Fault reset	Bit 7	▸	unchanged		Reset error	Bit 7	▸	unchanged	
Halt	Bit 8	▸	unchanged		Depending on operation mode	Bit 8	▸	unchanged	
Operation mode specific	Bit 9	▸	unchanged		Depending on operation mode	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		reserved (always 0)	Bit 14	▸	unchanged	
Manufacturer specific	Bit 15	▸	unchanged		reserved (always 0)	Bit 15	▸	unchanged	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Status word	0x6041/ro				**Status word**	**108.3**	▸	0x6041	
	x = 0 .. 0xFFFF					x = 0 .. 0xFFFF	▸	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 enabled					Voltage enabled	Bit 4	▸	unchanged	
Quick stop					Quickstop	Bit 5	▸	unchanged	
Switched on disabled					Inhibit start	Bit 6	▸	unchanged	
Warning					Warning	Bit 7	▸	unchanged	
Manufacturer specific					Depending on operation mode	Bit 8	▸	unchanged	
Remote					Remote	Bit 9	▸	unchanged	
Targed reached					Set value reached	Bit 10	▸	unchanged	
Internal limit active					Internal limit active	Bit 11	▸	unchanged	
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	0x6042	▸	**110.5**		**Input 16 bit**	**110.5**	▸	0x6042	
	y = -2^{15} ... 2^{15}-1	▸	y = x * 0x4000/ MotorMaxSpeed			x = -2^{15}... 2^{15} -1	▸	y = x * Motor MaxSpeed/ 0x4000	
vl_velocity_demand	0x6043 /ro				**w2 speed set value**	**18.21**	▸	0x6043	
						x = -1000000 ... 1000000	▸	y = x / 6	
vl_control_effort	0x6044 /ro				**x2 speed actual value**	**18.22**	▸	0x6044	
						x = -1000000 ... 1000000	▸	y = x / 6	
vl_velocity_min_max_-amount	0x6046							0x6046	
vl_velocity_min_amount	SIX. 0x01		**110.16**		**Input min. amount**	**110.16**	▸	SIX 0x01	SIX. 1 is always zero, the minimum limit is set at zero.
	y = 0 ... 2^{32}-1		y = x			x = 0 ... 2^{32}-1	▸	y = x	
vl_velocity_max_amount	SIX. 0x02	▸	**110.15**		**Input max. amount**	**110.15**	▸	SIX 0x02	
	y = 0 ... 2^{32}-1	▸	y = x				▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
vl_ramp_function_time	0x604F	▸	**110.6**		**Ramp-up time**	**110.6**	▸	0x604F	Ramp function generator ramp function time (1 = 1/1000 s ⇒ 1s = 1000). The resolution is 10 ms
	y = 0 ... 2^{31}-1	▸	y = x			x = 0 ... 650000	▸	y = x	
vl_slow_down_time	0x6050	▸	**110.7**		**Ramp-down time**	**110.7**	▸	0x6050	Ramp function generator ramp function time (1 = 1/1000 s ⇒ 1s = 1000). The resolution is 10 ms
	y = 0 ... 2^{31}-1	▸	y = x			x = 0 ... 650000	▸	y = x	
quick_stop_option_code	0x605A	▸	**108.13**		**QUICK STOP reaction code**	**108.13**	▸	0x605A	
	y = -2^{15} ... 2^{15}-1	▸	y = x			x = 0 .. 8	▸	y = x	
Manufacturer specific	x = -2^{15} ... -1	▸	y = x		not used	x = -2^{15} .. -1			
Disable drive function	x = 0	▸	y = x		Drive inhibited	x = 0	▸	y = x	
Slow down on slow down ramp	x = 1	▸	y = x		Ramp down on 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 limit	x = 3	▸	y = x		Ramp down on current limit	x = 3	▸	y = x	
Slow down on voltage limit	x = 4	▸	y = x		Ramp down on voltage limit	x = 4		y = x	
Slow down on slow down ramp and stay in Quick Stop Active	x = 5	▸	y = x		Ramp down on deceleration ramp and stay in quickstop	x = 5		y = x	
Slow down on quickstop ramp and stay in Quick Stop Active	x = 6	▸	y = x		Ramp down on quickstop ramp and stay in quickstop	x = 6		y = x	
Slow down on current limit and stay in Quick Stop active	x = 7	▸	y = x		Ramp down on current limit and stay in quickstop.	x = 7		y = x	
Slow down on voltage limit and stay in Quick Stop Active	x = 8	▸	y = x		Ramp down on voltage limit and stay in quickstop	x = 8		y = x	
reserved	x = 9 .. 2^{15}-1				not used			y = 9 .. 2^{15}-1	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Shutdown_option_code	0x605B	▸	**108.14**		**SHUTDOWN reaction code**	**108.14**	▸	0x605B	
	x = -2^{15} ... $2^{15}-1$	▸	y = x			x = 0 ... 3		y = x	
Manufacturer specific	x = -2^{15} ... -1								
Disable Drive function	x = 0	▸	y = x		Drive inhibited	x = 0	▸	y = x	
Slow down on slow down ramp	x = 1	▸	y = x		Ramp down on deceleration ramp	x = 1	▸	y = x	
Manufacturer specific	x = 2	▸	y = x		Ramp down on quickstop ramp	x = 2	▸	y = x	
Manufacturer specific	x = 3	▸	y = x		Ramp down on current limit	x = 3	▸	y = x	
Reserved	x = 4 .. $2^{15}-1$								
Disable_operation_ option_code	0x605C	▸	**108.15**		**DISABLE OPERATION reaction code**	**108.15**	▸	0x605C	
	x = -2^{15} ... $2^{15}-1$	▸	y = x			x = 0 ... 3		y = x	
Manufacturer specific	x = -2^{15} ... -1								
Manufacturer specific	x = 0	▸	y = x		Drive inhibited	x = 0		y = x	
Manufacturer specific	x = 1	▸	y = x		Ramp down on deceleration ramp	x = 1	▸	y = x	
Disable drive function	x = 2	▸	y = x		Ramp down on quickstop ramp	x = 2	▸	y = x	
Slow down on slow down ramp	x = 3	▸	y = x		Ramp down on current limit	x = 3	▸	y = x	
Reserved	x = 4 ... $2^{15}-1$								

CANopen object	**Index** **Value range**	▸	**P. no.** **Scaling**		**Controller parameters**	**P. no.** **Value range**	▸	**Index** **re-scaling**	**Comment**
Modes_of_operation	0x6060	▸	**109.1**		**Operation Mode Set**	**109.1**	▸	0x6060	
	x = -128 .. 127	▸	y = x			x = -128 .. 127	▸	y = x	
Manufacturer specific	x = -128 ... -12		y = x						
Manufacturer specific	x = -11		y = x		reserved	x = -11		y = x	
Manufacturer specific	x = -10		y = x		U/f operation	x = -10		y = x	
Manufacturer specific	x = -9		y = x		Voltage setting	x = -9		y = x	
Manufacturer specific	x = -8		y = x		Current setting	x = -8		y = x	
Manufacturer specific	x = -7	▸	y = x		Autotuning	x = -7	▸	y = x	
Manufacturer specific	x = -6	▸	y = 5		Spindle positioning	x = -6	▸	x = -106	
Manufacturer specific	x = -5	▸	y = x		Synchronous operation	x = -5	▸	y = x	
Manufacturer specific	x = -4	▸	y = x		Position control	x = -4	▸	y = x , y = 8	
Manufacturer specific	x = -3	▸	y = x		Speed control	x = -3	▸	y = x; y = 3; y = 9	
Manufacturer specific	x = -2	▸	y = x		Current control	x = -2	▸	y = x; y = 4; y = 10	
Manufacturer specific	x = -1	▸	y = x		Notch position search	x = -1	▸	y = x	
No mode change / assigned	x = 0								
Profile position mode	x = 1	▸	y = x		Target position setting	x = 1	▸	y = x	
Velocity mode	x = 2	▸	y = x		Speed setting 1	x = 2	▸	y = x	
Profile velocity mode	x = 3	▸	y = -3						
Torque profile mode	x = 4	▸	y = -2						
Reserved	x = 5	▸	y = x		Manual drive operation	x = 5	▸	y = x	
Homing mode	x = 6	▸	y = x		Reference run operation	x = 6	▸	y = x	
Interpolated position mode	x = 7								
Cyclic sync position mode	x = 8	▸	y = -4						
Cyclic sync velocity mode	x = 9	▸	y = -3						
Cyclic sync torque mode	x = 10	▸	y = -2						
reserved	x = 11 ... 127								

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Modes_of_operation_display	0x6061/ro				Operation Mode Act	**109.2**	▸	0x6061	
						x = -11 ... 6	▸	y = x	
					reserved	x = -11	▸	y = x	
					U-f operation	x = -10	▸	y = x	
					Voltage setting	x = -9	▸	y = x	
					Current setting	x = -8	▸	y = x	
					Autotuning	x = -7	▸	y = x	
					Spindle positioning	x = -6	▸	y = x	
					Synchronous operation	x = -5	▸	y = x	
					Position control	x = -4	▸	y = x; y = 8	
					Speed control	x = -3	▸	y = x; y = 3; y = 9	
					Current control	x = -2	▸	y = x; y = 4; y = 10	
					Notch position search	x = -1	▸	y = x	
					Target position setting	x = 1	▸	y = x	
					Speed setting 1	x = 2	▸	y = x	
					Manual drive operation	x = 5	▸	y = x	
					Reference run operation	x = 6	▸	y = x	
Position_actual_internal_value	0x6063 /ro				**Position actual value rev+angle**	**18.54**	▸	0x6063	In the controller UINT32 is provided from the fieldbus with an offset of 2^{31} (UINT32 ⇒ INT32). Only in case of changes parameter 131.9 bit 16 = 1, no offset of -2^{31}.
						x = 0 .. 2^{32} - 1	▸	y = x - 2^{31}	
Position_actual_value	0x6064 /ro				**Positioning position actual value**	**121.9**	▸	0x6064	In the controller UINT32 is provided from the fieldbus with an offset of 2^{31} (UINT32 ⇒ INT32). Only in case of changes parameter 131.9 bit 16 = 1, no offset of -2^{31}.
						x = 0 .. 2^{32} - 1	▸	y = x - 2^{31}	
Position_window	0x6067	▸	**121.5**		**Positioning window**	**121.5**	▸	0x6067	
	x = 0 .. 2^{32} - 1	▸	y = x			x = 0 .. 2^{32} - 1	▸	y = x	
Position_window_time	0x6068	▸	**121.6**		**Positioning window time**	**121.6**	▸	0x6068	
	x = 0 .. 2^{32} - 1	▸	y = x			x = 1 ... 2^{32} - 1	▸	y = x	
Velocity_sensor_actual_value	0x6069 /ro				**Position actual angle 32 bit**	**106.10**	▸	0x6069	
						x = 0 ... 2^{32} - 1	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Sensor_selection_code	0x606A					-		0x606A	The controller only supports position encoders, thus display only.
velocity_actual_- value_from_position_en- coder					-	x = 0	▸	y = x	
velocity_actual_- value_from_velocity_en- coder					not supported				
Velocity_demand_value	0x606B /ro	▸	**18.21**		**w2 speed set value**	**18.21**	▸	0x606B	
	x = -2^{15} ... $2^{15}-1$	▸				x = -1000000 ... 1000000	▸	y = x / 6	
Velocity_actual_value	0x606C /ro				**x2 speed actual value**	**18.22**	▸	0x606C	
						x = -1000000 ... 1000000	▸	y = x / 6	
Target_torque	0x6071	▸	**18.50**		**isq set value for torque control**	**18.50**	▸	0x6071	1000 corresponds to 100,0% related to the max. available torque current parameter 19.8
	x = -2^{15} ... 2^{15} -1	▸	y = x*0x4000 / 1000			x = -2^{15} .. 2^{15} -1	▸	y = x*1000 / 0x4000	
Max_torque	0x6072	▸	**138.14**		**Iq cyclic bipolar limit**	**138.14**	▸	0x6072	1000 corresponds to 100,0% related to the max. available torque current parameter 19.8
	x = 0 ... 2^{16} - 1	▸	y = x*0x4000 / 1000			x = 0 ... 2^{16} - 1	▸	y = x*1000 / 0x4000	
Torque_actual_value	0x6077 /ro				**Isq actual value**	**47.3**	▸	0x6077	1000 corresponds to 100,0% related to the max. available torque current parameter 19.8
						x = -4,096e+3 ... 4,096e+3	▸	y = x * 1000 / P19.8	
Target_position	0x607A	▸	**118.16 / 136.3**		**Relative target position, Target position**	**118.16 / 136.3**	▸	0x607A	In the controller UINT32 is provided from the fieldbus with an offset of 2^{31} (UINT32 ⇒ INT32). Only in case of changes parameter 131.9 bit 16 = 1, no offset of -2^{31}. If the data are standardized from the controller in the fieldbus direction, only parameter 136.8 is used at operating mode 8 and parameter 118.16 is used else.
	x = -2^{31} ... $2^{31}-1$	▸	y = x, y = x + 2^{31}			x = -2^{31} .. $2^{31}-1$ x = 0 ... 2^{32} - 1	▸	y = x - 2^{31}	
Home_offset	0x607C	▸	**120.3**		**Home position**	**120.3**	▸	0x607C	In the controller UINT32 is provided from the fieldbus with an offset of 2^{31} (UINT32 ⇒ INT32). Only in case of changes parameter 131.9 bit 16 = 1, no offset of -2^{31}.
	x = -2^{31} ... $2^{31}-1$	▸	y = x + 2^{31}			x = 0 ... 2^{32} - 1	▸	y = x - 2^{31}	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Software_position_limit	0x607D							0x607D	In the controller UINT32 is provided from the fieldbus with an offset of 2^{31} (UINT32 ⇒ INT32). Only in case of changes parameter 131.9 bit 16 = 1, no offset of -2^{31}.
Min position limit	SIX 0x01	▸	**121.3**		**Negative software limit switch**	**121.3**	▸	SIX 0x01	
	$x = -2^{31} .. 2^{31}-1$	▸	$y = x + 2^{31}$			$x = 0 ... 2^{32} - 1$	▸	$y = x - 2^{31}$	
Max position limit	SIX. 0x02	▸	**121.4**		**Positive software limit switch**	**121.4**	▸	SIX 0x02	
	$x = -2^{31} .. 2^{31}-1$	▸	$y = x + 2^{31}$			$x = 0 ... 2^{32} - 1$	▸	$y = x - 2^{31}$	
Polarity	0x607E	▸	**179.1**		**Polarity**	**179.1**		0x607E	Effective only if P131.9 bit 14 = 1, acts on 0x607A, 0x607D SIX 1, 0x607D SIX 2, 0x606C, 0x60B1, 0x60FF
	x = 0 ... 0xFFFF	▸	y = x			x = 0 ... 0xFFFF	▸	y = x	
Reserved	Bit 0 ... 5								
Velocity polarity	Bit 6	▸	unchanged		Speed polarity	Bit 6	▸	unchanged	Effective only if operating mode = 3 (Profile velocity mode) or operating mode = 9 (Cyclic sync velocity mode)
Position polarity	Bit 7	▸	unchanged		Position polarity	Bit 7	▸	unchanged	Effective only if operating mode = 1 (Profile position mode) or operating mode = 8 (Cyclic sync position mode)
Max_motor_speed	0x6080	▸	**110.13**		**Maximum drive speed**	**110.13**	▸	0x6080	The user-defined unit (speed units) is interpreted in the controller as rpm
	$x = 0 .. 2^{16}-1$	▸	y = x			x = 1 .. 1.0e+6	▸	y = x	
Profile velocity	0x6081	▸	**118.11**		**Speed**	**118.11**	▸	0x6081	
	$x = 0 .. 2^{32}-1$	▸	y = x			x = 1 ... 65535	▸	y = x	
Profile acceleration	0x6083	▸	**118.12**		**Acceleration**	**118.12**	▸	0x6083	
	$x = 0 .. 2^{32}-1$	▸	y = x			x = 7 ... 65535	▸	y = x	
Profile deceleration	0x6084	▸	**118.13**		**Deceleration**	**118.13**	▸	0x6084	
	$x = 0 .. 2^{32}-1$	▸	y = x			x = 7 ... 65535	▸	y = x	
Quick_stop_deceleration	0x6085	▸	**110.8**		**Quick stop time**	**110.8**	▸	0x6085	
	$x = 0 .. 2^{32}-1$	▸	y = x			x = 0 .. 650000	▸	y = x	
Motion profile type	0x6086	▸	**118.2**		**Mode**	**118.2**	▸	0x6086	
	$x = -2^{15} ... 2^{15} - 1$								
Position encoder resolution	0x608F							0x608F	Effective only if P131.9 bit 14 = 1, acts on 0x6064, 0x6067, 0x607A, 0x607C, 0x607D SIX 1, 0x607D SIX 2, 0x60BA, 0x60BB, 0x60BC, 0x60BD, 0x60F4
Encoder increments	SIX 0x01	▸	**179.2**		**Position resolution encoder increments**	**179.2**	▸	SIX 0x01	
	$x = 0 .. 2^{32}-1$	▸	y = x			$x = 0 .. 2^{32}-1$	▸	y = x	
Motor revolutions	SIX 0x02	▸	**179.3**		**Position resolution motor revolutions**	**179.3**	▸	SIX 0x02	
	$x = 0 .. 2^{32}-1$	▸	y = x			$x = 0 .. 2^{32}-1$	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Velocity encoder resolution	0x6090							0x6090	Effective only if P131.9 bit 14 = 1, acts on 0x606C, 0x6081, 0x6083, 0x6084, 0x6085, 0x6099 SIX 1, 0x6099 SIX 2, 0x609A, 0x60B1, 0x60FF
Encoder increments per second	SIX 0x01	▸	**179.4**		**Speed resolution encoder increments/s**	**179.4**	▸	SIX 0x01	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Motor revolutions per second	SIX 0x02	▸	**179.5**		**Speed resolution motor revolutions/s**	**179.5**	▸	SIX 0x02	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Gear ratio	0x6091							0x6091	Effective only if P131.9 bit 14 = 1, acts on 0x6064, 0x6067, 0x606C, 0x607A, 0x607C, 0x607D SIX 1, 0x607D SIX 2, 0x6081, 0x6083, 0x6084, 0x6085, 0x6099 SIX 1, 0x6099 SIX 2, 0x609A, 0x60B1, 0x60BA, 0x60BB, 0x60BC, 0x60BD, 0x60F4, 0x60FF
Motor revolutions	SIX 0x01	▸	**179.6**		**Gear ratio motor shaft revolutions**	**179.6**	▸	SIX 0x01	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Shaft revolutions	SIX 0x02	▸	**179.7**		**Gear ratio drive shaft revolutions**	**179.7**	▸	SIX 0x02	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Feed constant	0x6092							0x6092	Effective only if P131.9 bit 14 = 1, acts on 0x6064, 0x6067, 0x606C, 0x607A, 0x607C, 0x607D SIX 1, 0x607D SIX 2, 0x6081, 0x6083, 0x6084, 0x6085, 0x6099 SIX 1, 0x6099 SIX 2, 0x609A, 0x60B1, 0x60BA, 0x60BB, 0x60BC, 0x60BD, 0x60F4, 0x60FF
Feed	SIX 0x01	▸	**179.8**		**Feed constant feed**	**179.8**	▸	SIX 0x01	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Shaft revolutions	SIX 0x02	▸	**179.9**		**Feed constant drive shaft revolutions**	**179.9**	▸	SIX 0x02	
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Homing_method	0x6098	▸	**120.4**		**Homing method**	**120.4**	▸	0x6098	
	x = -128 ... 127	▸	y = x			x = -10 ... 35	▸	y = x	
Manufacturer specific	x = -128 ... -11								
Manufacturer specific	x = -10	▸	y = x		Reaching of mechanical stop with zero pulse, counter-clockwise	x = -10	▸	y = x	
Manufacturer specific	x = -9	▸	y = x		Reaching of mechanical stop with zero pulse, clockwise rotation	x = -9	▸	y = x	
Manufacturer specific	x = -8	▸	y = x		Reaching of mechanical stop, counter-clockwise	x = -8	▸	y = x	
Manufacturer specific	x = -7	▸	y = x		Reaching of mechanical stop, clockwise rotation	x = -7	▸	y = x	
Manufacturer specific	x = -6	▸	y = x		Reaching of the next encoder zero angle	x = -6	▸	y = x	
Manufacturer specific	x = -5	▸	y = x		Moving to pos. limit switch	x = -5	▸	y = x	
Manufacturer specific	x = -4	▸	y = x		Moving to neg. limit switch	x = -4	▸	y = x	
Manufacturer specific	x = -3	▸	y = x		Setting of home position	x = -3	▸	y = x	
Manufacturer specific	x = -2	▸	y = x		Reaching the encoder zero angle or zero pulse with counter-clockwise rotation	x = -2	▸	y = x	
Manufacturer specific	x = -1	▸	y = x		Reaching the encoder zero angle or zero pulse with clockwise rotation	x = -1	▸	y = x	
No homing operation	x = 0				reserved			y = 0	
Homing on the neg. limit switch	x = 1	▸	y = x		Neg. limit switch with encoder zero angle or zero pulse reference	x = 1	▸	y = x	
Homing on the pos. limit switch	x = 2	▸	y = x		Pos. limit switch with encoder zero angle or zero pulse reference	x = 2	▸	y = x	
Homing on the positive home switch & index pulse	x = 3	▸	y = x		Pos. zero point switch with encoder zero angle or zero pulse reference, counter-clockwise rotation	x = 3	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Homing on the positive home switch & index pulse	x = 4	▸	y = x		Pos. zero point switch with encoder zero angle or zero pulse reference, clockwise rotation	x = 4	▸	y = x	
Homing on the negative Home Switch & Index Pulse	x = 5	▸	y = x		Neg. zero point switch with zero encoder angle or zero pulse reference, clockwise rotation	x = 5	▸	y = x	
Homing on the negative Home Switch & Index Pulse	x = 6	▸	y = x		Neg. zero point switch with encoder zero angle or zero pulse reference, counter-clockwise rotation	x = 6	▸	y = x	
Zero reference cam switch, left to pos. edge with Zero pulse; CW move	x = 7	▸	y = x		Zero point switch, to the left of pos. edge with zero pulse; clockwise direction	x = 7	▸	y = x	
Zero reference cam switch, right to pos. edge with Zero pulse; CW move	x = 8	▸	y = x		Zero point switch, to the right of pos. edge with zero pulse, clockwise rotation	x = 8	▸	y = x	
Zero reference cam switch, left to neg. edge with Zero pulse; CW move	x = 9	▸	y = x		Zero point switch, to the left of neg. edge with zero pulse, clockwise rotation	x = 9	▸	y = x	
Zero reference cam switch, right to neg. edge with Zero pulse; CW move	x = 10	▸	y = x		Zero point switch, to the right of neg. edge with zero pulse; clockwise rotation	x = 10	▸	y = x	
Zero reference cam switch, right to neg. edge with Zero pulse; CCW move	x = 11	▸	y = x		Zero point switch, on the right of neg. edge with zero pulse; counter-clockwise rotation	x = 11	▸	y = x	
Zero reference cam switch, right to pos. edge with Zero pulse; CCW move	x = 12	▸	y = x		Zero point switch, on the right of pos. edge with zero pulse; counter-clockwise rotation	x = 12	▸	y = x	
Zero reference cam switch, left to neg. edge with Zero pulse; CCW move	x = 13	▸	y = x		Zero point switch, on the right of pos. edge with zero pulse; counter-clockwise rotation	x = 13	▸	y = x	
Zero reference cam switch, right to neg. edge with Zero pulse; CCW move	x = 14	▸	y = x		Zero point switch, on the right of neg. edge with zero pulse; counter-clockwise rotation	x = 14	▸	y = x	
reserved	x = 15, 16				reserved				
Negative limit switch	x = 17	▸	y = x		Negative limit switch	x = 17	▸	y = x	
Positive limit switch	x = 18	▸	y = x		Positive limit switch	x = 18	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Positive Zero reference switch, CCW move	x = 19	▸	y = x		Positive zero point switch; counter-clockwise rotation	x = 19	▸	y = x	
Positive Zero reference switch, CW move	x = 20	▸	y = x		Positive zero point switch; clockwise rotation	x = 20	▸	y = x	
Negative Zero reference switch, CW move	x = 21	▸	y = x		Negative zero point switch; clockwise rotation	x = 21	▸	y = x	
Negative Zero reference switch, CCW move	x = 22	▸	y = x		Negative zero point switch; counter-clockwise rotation	x = 22	▸	y = x	
Zero reference cam switch, left to pos. edge; CW move	x = 23	▸	y = x		Zero point switch, to the left of pos. edge; clockwise rotation	x = 23	▸	y = x	
Zero reference cam switch, right to pos. edge; CW move	x = 24	▸	y = x		Zero point switch, to the right of pos. edge; clockwise rotation	x = 24	▸	y = x	
Zero reference cam switch, left to neg. edge; CW move	x = 25	▸	y = x		Zero point switch, to the left of neg. edge; clockwise rotation	x = 25	▸	y = x	
Zero reference cam switch, right to neg. edge; CW move	x = 26	▸	y = x		Zero point switch, to the right of neg. edge; clockwise rotation	x = 26	▸	y = x	
Zero reference cam switch, right to neg. edge; CCW move	x = 27	▸	y = x		Zero point switch, on the right of neg. edge; counter-clockwise rotation	x = 27	▸	y = x	
Zero reference cam switch, left to neg. edge; CCW move	x = 28	▸	y = x		Zero point switch; on the left of neg. edge; counter-clockwise rotation	x = 28	▸	y = x	
Zero reference cam switch, right to pos. edge; CCW move	x = 29	▸	y = x		Zero point switch, on the right of pos. edge; counter-clockwise rotation	x = 29	▸	y = x	
Zero reference cam switch, left to pos. edge; CCW move	x = 30	▸	y = x		Zero point switch, on the left of pos. edge; counter-clockwise rotation	x = 30	▸	y = x	
reserved	31 ... 32				reserved	31 ... 32			
Nearest zero pulse; CCW move	x = 33	▸	y = x		Next zero pulse; counter-clockwise rotation	x = 33	▸	y = x	
Nearest Zero pulse; CW move	x = 34	▸	y = x		Next zero pulse with clockwise rotation	x = 34	▸	y = x	
Homing on the Current Position	x = 35	▸	y = x		Setting of home position	x = 35	▸	y = x	
reserved	x = 36 .. 127				not used				

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Homing_speeds	0x6099							0x6099	
Speed_during_search_for_switch	SIX 0x01	▸	**120.5**		**Homing speed**	**120.5**	▸	SIX 0x01	
	x = 0 .. 2^{32}-1	▸	y = x			x = 1 ... 65535	▸	y = x	
Speed_during_search_-for_zero	SIX 0x02	▸	**120.6**		**Homing final speed**	**120.6**	▸	SIX 0x02	
	x = 0 .. 2^{32}-1	▸	y = x			x = 1 ... 65535	▸	y = x	
Homing_acceleration	0x609A	▸	**120.7**		**Homing acceleration**	**120.7**	▸	0x609A	
	x = 0 .. 2^{32}-1	▸	y = x			x = 7 ... 65535	▸	y = x	
Velocity offset	0x60B1	▸	**18.68**		**Speed additional value**	**18.68**	▸	0x60B1	
	x = -2^{31} ... 2^{31} -1	▸	y = x * 6			x = -150000 ... 150000	▸	y = x / 6	
Torque offset	0x60B2	▸	**19.17**		**Isq additive set value**	**19.17**	▸	0x60B2	
	x = -2^{15} ... 2^{15} -1	▸	y = x * MaxTor-queCurrent / 1000			x = -300 ... 300	▸	y = x * 1000 / MaxTorque-Current	
Touch probe pos 1 pos value	0x60BA /ro				**DS402 touch probe 1 pos. value**	**124.31**	▸	0x60BA	The UINT32 is added an offset of -2^{31} in the controller and on the fieldbus side (UINT32 → INT32). Only if parameter 131.9 bit 16 = 1 is changed, no offset is added.
						x = 0 ...2^{32}-1	▸	y = x - 2^{31}	
Touch probe pos 1 neg value	0x60BB /ro				**DS402 touch probe 1 neg. value**	**124.32**	▸	0x60BB	The UINT32 is added an offset of -2^{31} in the controller and on the fieldbus side (UINT32 → INT32). Only if parameter 131.9 bit 16 = 1 is changed, no offset is added.
						x = 0 ...2^{32}-1	▸	y = x - 2^{31}	
Touch probe pos 2 pos value	0x60BC /ro				**DS402 touch probe 2 pos. value**	**124.33**	▸	0x60BC	The UINT32 is added an offset of -2^{31} in the controller and on the fieldbus side (UINT32 → INT32). Only if parameter 131.9 bit 16 = 1 is changed, no offset is added.
						x = 0 ...2^{32}-1	▸	y = x - 2^{31}	
Touch probe pos 2 neg value	0x60BD /ro				**DS402 touch probe 2 neg. value**	**124.34**	▸	0x60BD	The UINT32 is added an offset of -2^{31} in the controller and on the fieldbus side (UINT32 → INT32). Only if parameter 131.9 bit 16 = 1 is changed, no offset is added.
						x = 0 ...2^{32}-1	▸	y = x - 2^{31}	
Following error actual value	0x60F4		**18.60**		**Position error rev+angle**	**18.60**		0x60F4	
	x = -2^{31} ... 2^{31} -1		y = x			x = -2^{31} ... 2^{31}-1		y = x	
position_control_parameter_set	0x60FB							0x60FB	Manufacturer-specific CANopen object

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Manufacturer specific	SIX 0x0A /ro				**Position act value angle**	**18.76**	▸	SIX 0x0A	Default = 0
						x = 0 .. 2^{32}-1	▸	y = x	
Manufacturer specific	SIX 0x0D	▸	**118.6**		**Record number actual**	**118.6**	▸	SIX 0x0A	Default = 0
	x = 0 ... 16	▸	y = x			x = 0 .. 16	▸	y = x	
Manufacturer specific	SIX 0x0F	▸	**120.3**		**Home position**	**120.3**	▸	SIX 0x0F	Default = 0x00020000
	x = 0 .. 2^{32}-1	▸	y = x			x = 0 .. 2^{32}-1	▸	y = x	
Manufacturer specific	SIX 0x15	▸	**118.10**		**Target mode**	**118.10**	▸	SIX 0x15	Default = 0
	x = -2 ... 13	▸	y = x			x = -2 ... 13	▸	y = x	
Manufacturer specific	SIX 0x17	▸	**136.5**		**Target angle**	**136.5**	▸	SIX 0x17	Default = 0
	x = 0 .. 2^{32} - 1	▸	y = x			x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x18	▸	**120.10**		**Homing encoder offset**	**120.10**	▸	SIX 0x18	Default = 0
	x = 0 .. 2^{16}-1	▸	y = x			x = 0 .. 2^{16}-1	▸	y = x	
Manufacturer specific	SIX 0x19		**120.8**		**Homing deceleration**	**120.8**	▸	SIX 0x19	Default = 200
	x = 7 ... 2^{16}-1		y = x			x = 7 ... 2^{16}-1	▸	y = x	
target_velocity	0x60FF	▸	**110.4**		**Input 32 bit**	**110.4**	▸	0x60FF	Default = 0
	x = -2^{31} .. 2^{31}-1	▸	y = x * 0x40000000 / MotorMaxSpeed			x = -2^{31} .. 2^{31} -1	▸	y = x * Motor-MaxSpeed / 0x4000000	
Supported drive modes	0x6502 / ro							0x6502	
						0x000003BF	▸	y = x	
Drive_data	0x6510							0x6510	
Manufacturer specific	SIX 0x01/ ro				**reserved**				
Manufacturer specific	SIX 0x02 / ro				**reserved**				
Manufacturer specific	SIX 0x03 / ro				**Fieldbus controller firmware number**	**102.18**	▸	SIX 0x03	Baumueller internal fieldbus firmware number
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x04 / ro				**Feldbus Controller Firm-ware Version**	**102.19**	▸	SIX 0x04	Display in the format: Major[2] . Minor[2] . Fix[2]
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x05 / ro				**Fieldbus controller firmware build number**	**102.22**	▸	SIX 0x05	Number for counting beta states, proto-types or even nightly builds.
						x = 0 .. 2^{32} - 1	▸	y = x	

CANopen object	Index Value range	▸	P. no. Scaling		Controller parameters	P. no. Value range	▸	Index re-scaling	Comment
Manufacturer specific	SIX 0x06 / ro				**Fieldbus controller firmware type**	**102.21**	▸	SIX 0x06	
						x = 0 ... 3	▸	y = x	
					Series	0	▸	y = x	
					Beta	1	▸	y = x	
					Prototype	2	▸	y = x	
					Nightly Build	3	▸	y = x	
Manufacturer specific	SIX 0x07 / ro				**Fpga Id**	**102.10**	▸	SIX 0x07	Baumueller internal FPGA identifier
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x08 / ro				**FPGA version**	**102.14**	▸	SIX 0x08	Display in the format: Major[2] . Minor[2] . Fix[2]
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x09 / ro				**FPGA Firmware number**	**102.15**	▸	SIX 0x09	Baumueller internal FPGA Firmware number
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x0A / ro				**Firmware number**	**102.1**	▸	SIX 0x0A	Baumueller internal controller Firmware number
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x0B / ro				**Firmware version**	**102.2**	▸	SIX 0x0B	Display in the format: Major[2] . Minor[2] . Fix[2]
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x0C / ro				**Firmware build number**	**102.4**	▸	SIX 0x0C	Number for counting beta states, prototypes or even nightly builds.
						x = 0 .. 2^{32} - 1	▸	y = x	
Manufacturer specific	SIX 0x0D / ro				**Firmware type**	**102.3**	▸	SIX 0x0D	
					Series	0	▸	y = x	
					Beta	1	▸	y = x	
					Prototyp	2	▸	y = x	
					Nightly Build	3	▸	y = x	
					Developer Build	4	▸	y = x	

D

APPENDIX D - TECHNICAL DATA: POWERLINK CONTROLLED NODE

In this appendix you will find a survey of the technical data of the **CANopen, CoE and POWERLINK** for BM3300/5000.

D.1 Technical features

CPU	Nios® II
FPGA	Cyclone IV (Fa. Altera)
Memory	8 kByte DPRAM, 64 MByte DDR2 SDRAM, 8 MByte Flash-Eprom
Baud rate	100 Mbit
Operating voltage	+5 V internal
Plug-in connector	2 RJ45 sockets, 8-pin

D.2 Data channels to the b maXX controller

For the data transmission of **b maXX** controller to the option module **CANopen, CoE and POWERLINK** for BM3300/5000 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** parameters can be accessed via the object index (exception string parameter).

Table of figures

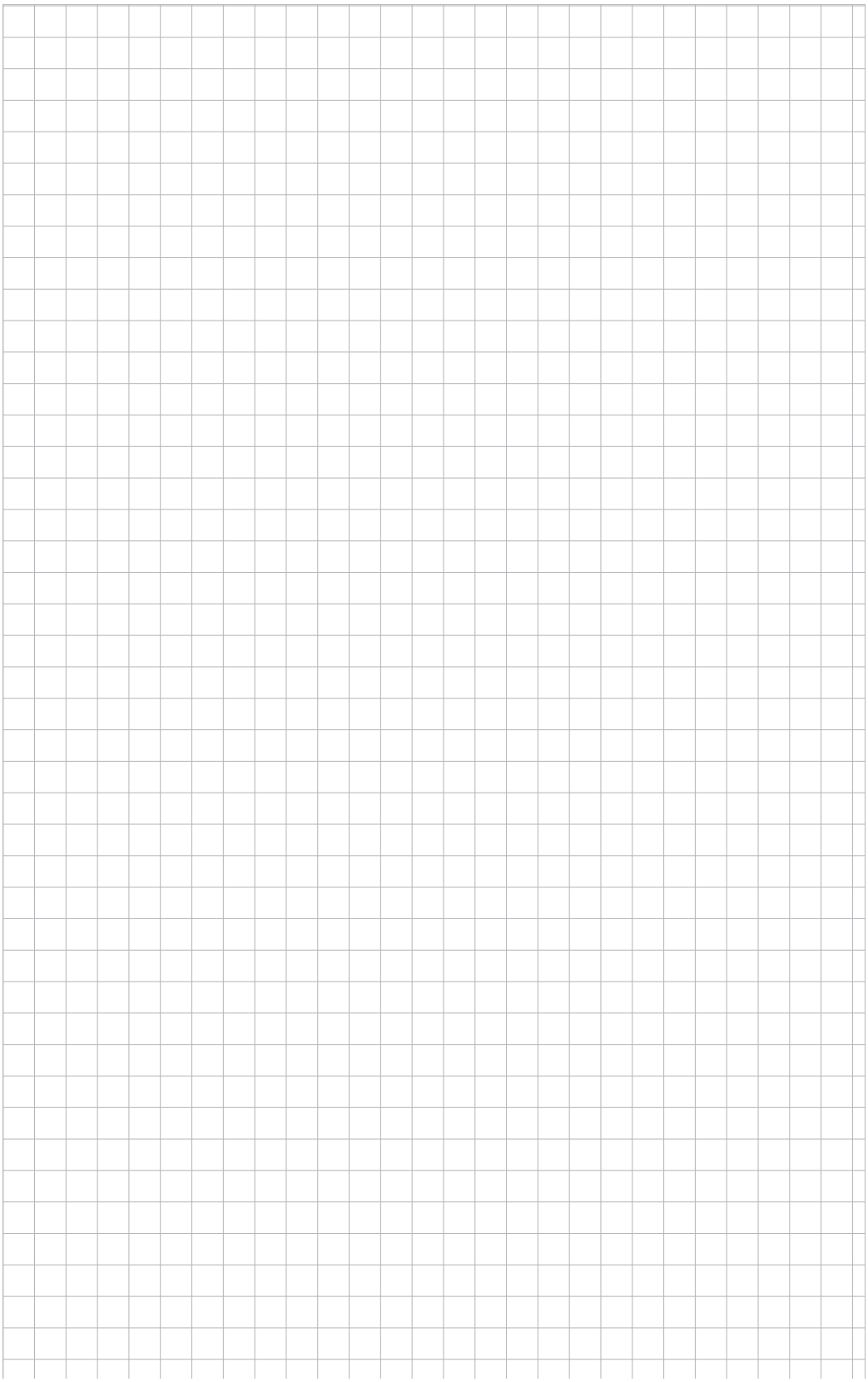
Table of figures

Index

Revision survey

Version	Status	Changes
5.14006.01	26.06.2014	First edition
5.14006.02	09.09.2014	Ch. 3.1 Communication profile Powerlink changed
5.14006.03	27.07.2015	Ch. 4.2 Fieldbus communication times: Fig. added App. B1: Example added
5.14006.04	07.09.2016	Factor Group added in Ch. 3.2.1, Ch. 3.2.2, Ch. 5.3 (new), App. B.2 and App. C
5.14006.06	26.07.2019	Ch. 3.3.2: No. 7 to 10 new App. B2: Line 0x60B8 changed App. B2 and App C: Line 0x60FB changed

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**