BAUMÜLLER

be in motion be in motion

BM4-O-ETH-01/2, ...-CAN-04

Ethernet with CANopen-Master for b maXX drive PLC

Application Manual

E	5.03002.02

Title	Application Manual
Product	Ethernet with CANopen-Master for b maXX drive PLC BM4-O-ETH-01/2, ...-CAN-04
Last Revision:	28 February 2007

Manufacturer

Baumüller Nürnberg GmbH
Ostendstr. 80 - 90
90482 Nuremberg
Germany
Tel. +49 9 11 54 32 - 0 Fax: +49 9 11 54 32 - 1 30
www.baumueller.de

Table of contents

TABLE OF CONTENTS

1

INTRODUCTION

This Application Manual is an important component of your b maXX 4400; this means that you must thoroughly read this document, not least to ensure your own safety.

In this chapter, we will describe the first steps.

1.1 First Steps

1 To program the Ethernet with CANopen option module, you need the following hardware:
a b maXX 4400 basic unit,
a b maXX PLC option module and
an ETH-01, ETH-02 or CAN-04 option module.
You must have installed the hardware in accordance with the Operating Instructions in each case and it must be ready for operation.

2 Apart from this, you need the following software:

- ProMaster for configuration of an CANopen network
- ProProg wt III for programming the b maXX drive PLC and the Ethernet with CANopen master for b maXX drive PLC option module

or

- PROPROG wt II for programming the b maXX drive PLC and the Ethernet with CANopen master for b maXX drive PLC option module.

NOTE

The ProProg wt III programming system is required to program the b maXX drive PLC BM4-O-PLC-0x in conjunction with ProMaster.

1.2 Terms Used

In this documentation, we will also refer to the following Baumüller products: "BM4-O-ETH-01" (Ethernet option module), "BM4-O-ETH-02" (Ethernet option module with CANopen-Master) or "„BM4-O-CAN-04" (CANopen-Master option module) as "Ethernet plug-in module“ for the BM4-O-ETH-01 or BM4-O-ETH-02 or "CANopen-Master plug-in module" for the BM4-O-ETH-02 or BM4-O-CAN-04.

We will use the term "option module" on its own when the text refers to the ETH-01, ETH-02 or CAN-04 products in general.

We will also use the term "b maXX" for the "Basic Unit b maXX 4400" product. The controller in the basic unit is also referred to as the "b maXX controller".

For a list of the abbreviations that are used, refer to ▹Appendix A - Abbreviations◃ from page 159 onward.

1.3 Conditions

This manual builds on the "b maXX PLC Application Manual" and assumes that you have knowledge of the PROPROG wt II programming tools and have read its manual or of the ProProg wt III programming tools and its Online Help System.

2

BASIC SAFETY INSTRUCTIONS

We have designed and manufactured each Baumüller plug-in module in accordance with the strictest safety regulations. Despite this, working with the plug-in module can be dangerous for you.

In this chapter, we will describe the risks that can occur when working with a Baumüller plug-in module. Risks are illustrated by icons. All the symbols that are used in this documentation are listed and explained.

In this chapter, we cannot explain how you can protect yourself from specific risks in individual cases. This chapter contains only general protective measures. We will go into concrete protective measures in subsequent chapters directly after information about the individual risk.

2.1 Hazard information and instructions

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

The hazard information is showing you the hazards which can lead to injury or even to death.

Always observe the hazard information given in this documentation.

Hazards are always divided into three danger classifications. Each danger classification is identified by one of the following words:

DANGER

- Considerable damage to property
- Serious personal injury
- Death **will** occur

WARNING

- Considerable damage to property
- Serious personal injury
- Death **can** occur

CAUTION

- Damage to property
- Slight to medium personal injury **can** occur

2.1.1 Structure of hazard information

The following two examples show how hazard information is structured in principle. A triangle is used to warn you about danger to living things. If there is no triangle, the hazard information refers exclusively to damage to property.

A triangle indicates that there is danger to living things.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is.

The icon in the rectangle represents the hazard.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is.

The icon in the circle represents an instruction. Users must follow this instruction.
(The circle is shown dashed, since an instruction is not available as an icon for each hazard advisory).

The circle shows that there is a risk of damage to property.

The icon in the rectangle represents the hazard.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is. (The rectangle is shown dashed, since the danger is not represented as an icon with every hazard advisory)

The text next to the icons is structured as follows:

THE SIGNAL WORD IS HERE THAT SHOWS THE DEGREE OF RISK

Here we indicate whether one or more of the results below occurs if you do not observe this warning.

- Here, we describe the possible results. The worst result is always at the extreme right.

Here, we describe the hazard.

Here, we describe what you can do to avoid the hazard.

2.1.2 Hazard advisories that are used

If a signal word is preceded by one of the following danger signs: ⚠ or ⚠ or ⚠, the safety information refers to injury to people.

If a signal word is preceded by a round danger sign: (!), the safety information refers to damage to property.

2.1.2.1 Hazard advisories about injuries to people

To be able to differentiate visually, we use a separate border for each class of hazard information with the triangular and rectangular pictograms.

For danger classification **DANGER,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

DANGER

The following **will occur**, if you do not observe this danger information:

- serious personal injury
- death

*Danger from: **electricity.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

DANGER

The following **will occur**, if you do not observe this danger information:

- serious personal injury
- death

*Danger from: **mechanical effects.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

For danger classification **WARNING,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

*Danger from: **electricity.**The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

For danger classification **CAUTION,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

CAUTION

The following **may occur**, if you do not observe this caution information:

- minor to medium personal injury.

Danger from: ***sharp edges.*** *The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

CAUTION

The following **may occur**, if you do not observe this danger information:

- environmental pollution.

Danger from: ***incorrect disposal.*** *The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

2.1.2.2 Hazard advisories about damage to property

If a signal word is preceded by a round danger sign: ⓘ, the safety information refers to damage to property.

CAUTION

The following **may occur**, if you do not observe this caution information:

- property damage.

Danger from: ***electrostatic discharge.*** *The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

2.1.2.3 Instruction signs that are used

wear safety gloves

wear safety shoes

2.2 Information signs

NOTE

This indicates particularly important information.

2.3 Legal information

This documentation is intended for technically qualified personnel that has been specially trained and is completely familiar with all warnings and maintenance measures.

The equipment is manufactured to the state of the art and is safe in operation. It can be put into operation and function without problems if you ensure that the information in the documentation is complied with.

Operators are responsible for carrying out servicing and commissioning in accordance with the safety regulations, applicable standards and any and all other relevant national or local regulations with regard to cable rating and protection, grounding, isolators, overcurrent protection, etc.

Operators are legally responsible for any damage that occurs during assembly or connection.

2.4 Appropriate Use

You must always use the plug-in module appropriately. Some important information is listed below. The information below should give you an idea of what is meant by appropriate use of the plug-in module. The information below has no claim to being complete; always observe all the information that is given in these operating instructions.

- You must only install the plug-in module in series b maXX 4400 units.
- Configure the application such that the plug-in module is always operating within its specifications.
- Ensure that only qualified personnel works with this plug-in module.
- Mount the plug-in module only in the specified slot/slots.
- Install the plug-in module as specified in this documentation.
- Ensure that connections always comply with the stipulated specifications.
- Operate the plug-in module only when it is in technically perfect condition.
- Always operate the plug-in module in an environment that is specified in the technical data.
- Always operate the plug-in module in a standard condition.
  For safety reasons, you must not make any changes to the plug-in module.
- Observe all the information on this topic if you intend to store the plug-in module.

You will be using the plug-in module in an appropriate way if you observe all the comments and information in these operating instructions.

2.5 Inappropriate Use

Below, we will list some examples of inappropriate use. The information below should give you an idea of what is meant by inappropriate use of the plug-in module. We cannon, however, list all possible cases of inappropriate use here. Any and all applications in which you ignore the information in this documentation are inappropriate; particularly, in the following cases:

- You installed the plug-in module in units that are not Series b maXX 4400.
- You ignored information in the operating instructions of this option module.
- You did not use the plug-in module as intended.
- You handled the plug-in module as follows
  - you mounted it incorrectly,
  - you connected it incorrectly,
  - you commissioned it incorrectly,
  - you operated it incorrectly,
  - you allowed non-qualified or insufficiently qualified personnel to mount the module, commission it and operate it,
  - you overloaded it,
  - You operated the module
    - with defective safety devices,
    - with incorrectly mounted guards or without guards at all,
    - with non-functional safety devices and guards
    - outside the specified environmental operating conditions
- You modified the plug-in module without written permission from Baumüller Nürnberg GmbH.
- You ignored the maintenance instructions in the component descriptions.
- You incorrectly combined the plug-in module with third-party products.
- You combined the drive system with faulty and/or incorrectly documented third-party products.
- Your self-written PLC software contains programming errors that lead to a malfunction.

Version 1.1 of Baumüller Nürnberg GmbH's General Conditions of Sale and Conditions of Delivery dated 2/15/02 or the respective latest version applies in all cases. These will have been available to you since the conclusion of the contract at the latest.

2.6 Protective equipment

In transit, the plug-in modules are protected by their packaging. Do not remove the plug-in module from its packaging until just before you intend to mount it.

The cover on the b maXX units' controller sections provides IP20 protection to the plug-in modules from dirt and damage due to static discharges from contact. This means that you must replace the cover after successfully mounting the plug-in module.

2.7 Personnel training

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

Only qualified personnel are allowed to mount, install, operate and maintain equipment made by Baumüller Nürnberg GmbH.

Qualified personnel (specialists) are defined as follows:

Qualified Personnel

Electrical engineers and electricians of the customer or of third parties who are authorized by Baumüller Nürnberg GmbH and who have been trained in installing and commissioning Baumüller drive systems and who are authorized to commission, ground and mark circuits and equipment in accordance with recognized safety standards.

Qualified personnel has been trained or instructed in accordance with recognized safety standards in the care and use of appropriate safety equipment.

Requirements of the operating staff

The drive system may only be operated by persons who have been trained and are authorized.

Only trained personnel are allowed to eliminate disturbances, carry out preventive maintenance, cleaning, maintenance and to replace parts. These persons must be familiar with the Operating Instructions and act in accordance with them.

Commissioning and instruction must only be carried out by qualified personnel.

2.8 Safety measures in normal operation

- At the unit's place of installation, observe the applicable safety regulations for the plant in which this unit is installed.
- Provide the unit with additional monitoring and protective equipment if the safety regulations demand this.
- Observe the safety measures for the unit in which the plug-in module is installed.

2.9 Responsibility and liability

To be able to work with these option modules in accordance with the safety requirements, you must be familiar with and observe the hazard information and safety instructions in this documentation.

2.9.1 Observing the hazard information and safety instructions

In the operating instructions of this option module, we use visually consistent safety instructions that are intended to prevent injury to people or damage to property.

2.9 Responsibility and liability

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

Any and all persons who work on and with Series b maXX units must always have available these Operating Instructions and must observe the instructions and information they contain – this applies in particular to the safety instructions.

Apart from this, any and all persons who work on this unit must be familiar with and observe all the rules and regulations that apply at the place of use.

2.9.2 Danger arising from using this module

The option module has been developed and manufactured to the state of the art and complies with applicable guidelines and standards. It is still possible that hazards can arise during use. For an overview of possible hazards, refer to the chapter entitled ▷Basic Safety Instructions ◁ from page 9 onward..
We will also warn you of acute hazards at the appropriate locations in this documentation.

2.9.3 Warranty and Liability

All the information in this documentation is non-binding customer information; it is subject to ongoing further development and is updated on a continuous basis by our permanent change management system.

Warranty and liability claims against Baumüller Nürnberg GmbH are excluded; this applies in particular if one or more of the causes listed in ▷Inappropriate Use ◁ from page 14 onward or below caused the fault:

- Disaster due to the influence of foreign bodies or force majeure.

ETHERNET

In this chapter, you will find information about connecting the option module to an Ethernet. Ethernet is only available with the BM4-O-ETH-01 and BM4-O-ETH-02 option modules.

3.1 General information on Ethernet and the Ethernet option module

Ethernet is an established and proven technology for data transfer in information technology and in office communication. In the Ethernet option module, a twisted pair cable is used as the transfer medium (for more information, see the option module Operating Instructions). This means that you can easily implement a link to an existing 10BaseT network or to a 100BaseTX network (IEEE 802.3 standard).

You can use the Ethernet option module on a network containing other components like PCs, hubs, switches and repeaters. Alternatively, you can make a direct connection to a PC's network adapter by means of a crosslink cable; in this case, you do not need any additional network components.

For communication, an Ethernet-based TCP/IP protocol stack and the

- PROPROG / OmegaOS communication

application protocol are implemented.

This protocol allows you to use the following functionality for PROPROG wt and OmegaOS:

- PLC control functions like Start and Stop
- Program management
- Online debugging, watch windows
- Logic Analyzer/oscilloscope
- OPC Server.

For further details and other functionality, refer to the Application Manual for the b maXX drive PLC and to the OPC server.

To use the functions stated above, you only need to set the communications port and the IP address in your PROPROG wt project. On the option module side, you must set the IP address, the subnet mask and the gateway or you must code them if you cannot use the default settings.

For more information on the individual steps that you have to carry out, refer to the Chapter entitled ▷Configuring Ethernet TCP/IP networks ◁ from page 34 onward.

The following chapters will give you an introduction to the structure of Ethernet-TCP/IP networks and the conditions for structuring communications connections between the Ethernet option module and other components on the network.

NOTE

The branching of Ethernet-TCP/IP networks can be very complex. There can also be a large number of components installed that are configured differently. The components influence the data traffic on the network. Apart from this, you must consider aspects of data security and unauthorized network access. This complexity means that the manual can only give you an overview in principle of using the Ethernet option module on Ethernet-TCP/IP networks. To get more details about your local area network (LAN), ask your network administrator and refer to the manuals of the network components.

3.2 Basics of Ethernet Networks and TCP/IP

3.2.1 Transfer standards

The Ethernet option module supports both of the most common physical models in the IEEE 802.3 standard:

10BaseT

Each network node is connected via its own twisted-pair cable to a (star) hub that forwards all the data packets equally to all the network nodes. Eight-pin RJ45 types are used as the plug-in connectors. The maximum length of a segment (the connection between the hub and the terminal equipment) is limited to 100 meters.

100BaseTX

The 100BaseTX transmission standard is similar to 10BaseT. Each network node is also connected via its own twisted-pair cable to a (star) hub that forwards all the data packets equally to all the network nodes. Eight-pin RJ45 types are also used as the plug-in connectors. Here too, the maximum length of a segment (the connection between the hub and the terminal equipment) is limited to 100 meters. However, all the components (cables, RJ45 outlets, etc.) must be rated for the higher transfer rate of 100 MHz.

The Ethernet option module automatically detects the transfer rate that is being used.

3.2.2 Structure of Ethernet networks with a hub and switch

There are various options for structuring a star topology network.

All the nodes are connected via one hub.

If a node sends a data packet, it is broadcast across the entire network and is available to each of the connected nodes. The message is further-processed in each case only by the node with the correct target address. Due to the large amounts of data, collisions can occur and the system must transfer messages again.

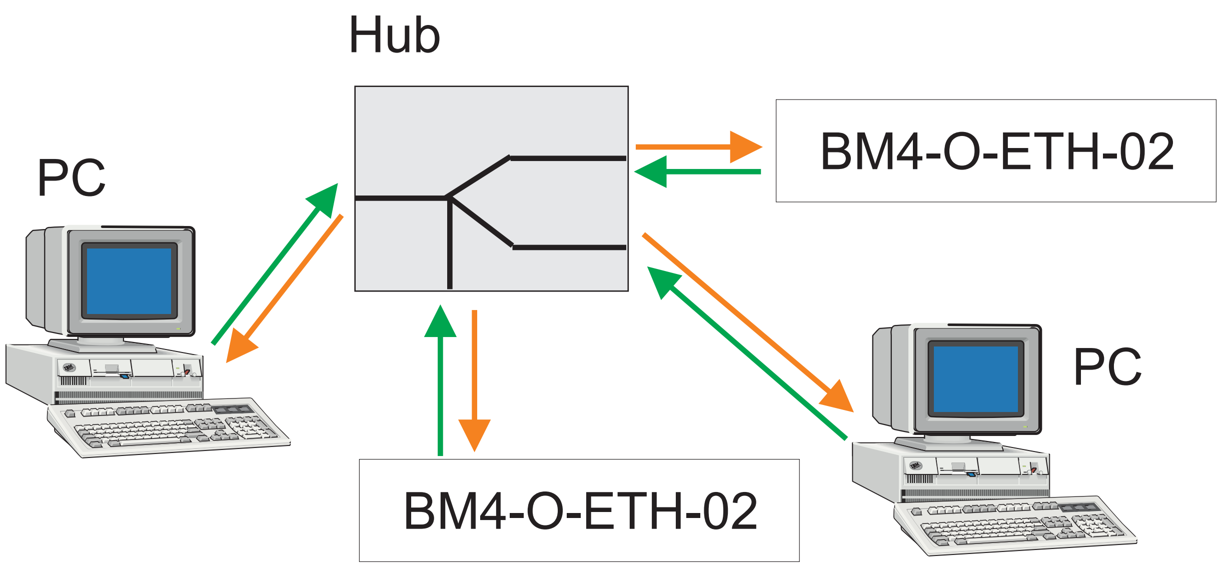


Figure 1: Principle of Shared Ethernet

All the nodes are connected via one switch.

In the case of a Switched Ethernet, a switch is used to connect several nodes. If data reaches the switch from a segment, the system checks the segment to which this data is to be transmitted. The system sends the data exclusively to the node with the correct target address. This reduces the amount of data and, with this, the risk of data collisions on the network.

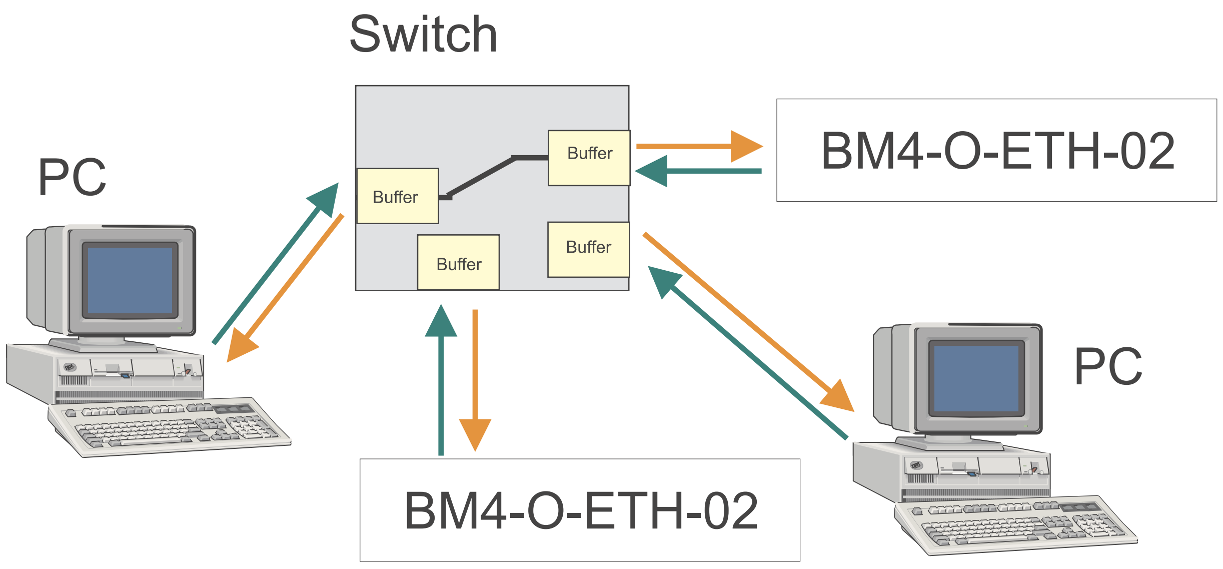


Figure 2: Principle of Switched Ethernet

By combining star structures, it is possible to form tree structures. Apart from this, it is possible to make the connection between the nodes at the TCP/IP level (see Chapter ▹TCP/IP ◃ from page 21 onward)

3.2.3 Ethernet address and MAC address

Each Ethernet option module has its own unique global Ethernet address. Ethernet addresses are also referred to as MAC addresses or MAC-IDs (Media Access Control Identity). The Ethernet address is permanently stored in the option module and you cannot change it. Note the difference between this address and the IP address, which has a different meaning and which you can change. The MAC address is used at the Ethernet level for addressing. It has a fixed length of 6 bytes (48 bits) and contains the manufacturer code and a manufacturer serial number.

3.2.4 Ethernet data packet

Data packets are transferred on a connection-free basis, i.e. the sender does not get a feedback message from the receiver. The user data is packed in a frame of address information. The structure of an Ethernet data packet of this type looks like this:

Preamble	Destination	Source	Type	Data	FCS

Figure 3: Ethernet data packet

Preamble:	A bit sequence for identifying the start of the packet
Destination:	The receiver's Ethernet address
Source:	The sender's Ethernet address
Type:	Indicates the higher level purpose (e.g. IP)
Data:	User data contains the higher level protocols, e.g. IP
FCS:	Checksum

The preamble is for synchronizing between the sending and receiving station.

The Ethernet header contains the MAC addresses of the sender and the receiver and a type field for identifying the subsequent protocol that is contained in the data area.

3.2.5 Bridges and Repeaters

Bridges also work at the level of the Ethernet (passing on data on the basis of the MAC-ID). Bridges connect subnets and decide on the basis of the MAC-ID the data packets that are to pass the bridge and the ones that are not to pass. By contrast with the switch, a bridge does not pass on broadcast messages (data packets to several nodes at the same time).

Repeaters are amplifiers for refreshing signals and they do not change the contents of the data packets. If a repeater detects an error on one of the connected segments, the system disconnects the connection to this segment. As soon as the error has been eliminated, the system establishes the connection again.

3.2.6 TCP/IP

Ethernet alone is not enough to connect several different networks for data transfer. Let us take a PC, for example, that is connected to an Internet Service Provider by means of an external modem and can use this connection to communicate with other servers on the Internet. In this case, there are other connections like the serial or USB connection to the modem and the phone line as transfer media in addition to the Ethernet. The TCP/IP protocol was developed to continue a connection outside the Ethernet too. TCP/IP is usually used as a common term, you must, however, differentiate between two different protocols.

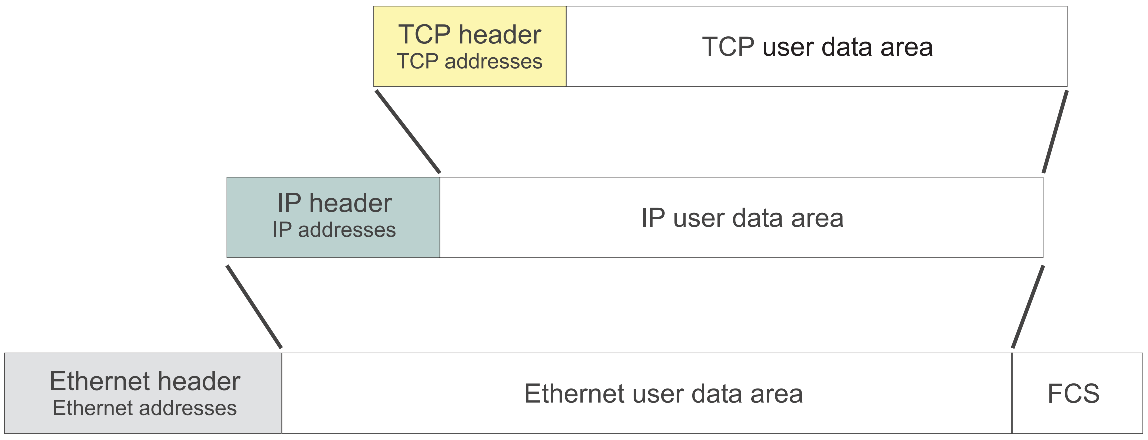


Figure 4: Structure of a TCP/IP-Ethernet data packet

3.2.6.1 IP protocol

Using **IP (Internet Protocol),** you go beyond the limits of a LAN (local area network). IP carries out correct addressing and delivery of data packet across a gateway to other networks. IP is in the Ethernet user data area. This means that IP packs the data that is received from higher levels into a separate frame. This packet is passed on to the Ethernet below it and it represents the user data of one or more Ethernet message frames.

Amongst other things, this IP frame contains a new form of addresses (Internet address, IP number). Under IP, each network node has its own unique Internet address (at least in a specific subnet). This is the basis of forwarding across the limits of an Ethernet segment and linking to non-Ethernet-based networks.

a) Router

You connect two or more different IP networks via routers. These routers decide on the basis of the IP address whether a data packet is to be forwarded to another network or not.

b) IP address

You must not confuse the IP address with the MAC-ID (or Ethernet address).

Users assign the IP address. As-delivered, the Ethernet option module has the IP address 192.168.1.1. For information on how to change this setting, refer to the chapter entitled ▹Configuring the Ethernet option module and checking settings ◃ from page 38 onward.

c) Structure of an IP address

The IP address consists of a total of 4 bytes that are normally shown in decimal form separated by periods, e.g. 192.168.1.1.

If you use the option module on a network whose nodes are only connected via a hub or switch and that does not have a connection to another network, you can assign virtually any IP address you like for each node. You must just ensure that you never set all the bits in any one byte equal to 0 or to 1 (byte = 0 or 255). These are reserved for special functions and you must not assign them. For example, you cannot use the address 194.11.0.13 due to the zero in the third byte.

If there is a connection to another network or to the Internet, you can, with the exceptions below, no longer freely assign the IP address:

For use on private networks, a corresponding IP standard reserves three address classes that may not be routed on the Internet, which means that they are not visible there. The addresses / address ranges are as follows:

- 10.xxx.xxx.xxx
- 172.16.xxx.xxx - 172.31.xxx.xxx
- 192.168.x.x

Routers on the LAN are generally set such that these address ranges are not routed. If it is not possible to communicate with an Ethernet option module, this may be due to the router on your LAN not routing the set IP address. The Ethernet option module's default IP address of 192.168.1.1 is an IP address of this type. Ask your network administrator which IP address range you may use if there is a connection to another network.

The address 127.0.0.1 is special, since it is always addressed as the local host. According to the standard, you are not allowed to use network address 127.x.x.x. This means that you can use address 127.0.0.1 to test the installation of your own equipment (e.g. a PC).

If you intend to connect network directly to the Internet, you can only use the unique IP addresses that are assigned by the IANA organization. Assignment depends on the country in which the network is operated.

NOTE

If the option module is in a network that has a connection to another network or if you want to make a direct Internet connection, you will need comprehensive knowledge of the entire LAN, of assigning IP addresses, of routing mechanisms and, in particular, of security requirements. This means that you should only make a connection to another network or establish an Internet connection together with an authorized network administrators.

d) Subnet mask and gateway

To make a connection to another network, you must address this network. The subnet mask is used for this. If a network node is to send a data packet, its own IP address is logically ANDed with the subnet mask and the destination IP address is logically ANDed with the subnet mask. If the network node gets the same result both times, it knows that the other node is on the same network. If the results are different, it is not possible to directly address the other network node. In the case, the node passes on the data packet to a gateway or router for transmission. A gateway differs from a router inasmuch as gateways can access non-TCP/IP networks. Since the receiver of the data packet must also carry out the logical operation with the subnet mask for the response, the subnet mask must also be set correctly on the receiver side.

NOTE

Since a router is only a special gateway, when configuring the IP address people often use the term "gateway" even though a router is physically present. This is particularly the case with the network configuration in Windows. With the Ethernet option module, too, we use the more general term "Gateway".

Users assign the subnet mask and the gateway or router IP address. For information on how to make these settings in the Ethernet option module, refer to the chapter entitled ▹Configuring the Ethernet option module and checking settings ◃ from page 38 onward. The default settings are as follows:

○ Subnet mask:	255.255.255.0
○ Gateway:	0.0.0.0

The gateway IP address 0.0.0.0 with the Ethernet option module means that no gateway is used. This means that the data packet is also sent even if the system detects that due to the subnet mask link the receiver is not on the same network.

e) Subnet mask and network class

The InterNIC (International Network Information Center) divided the IP address into a "network section" and a "host section" to create what are known as address classes. The table below shows the different address classes, the assigned values of the most significant bit of the IP address and the division into the "network section" and the "host section".

Address class	Description	Address range of the network section	Possible number of hosts
Class A	The first byte of the IP address is for addressing the network section; the last three bytes address the host section	1.xxx.xxx.xxx to 126.xxx.xxx.xxx	Approximately 16 million
Class B	The first two bytes of the IP address are for addressing the network section; the last two bytes address the host section	128.0.xxx.xxx - 191.255.xxx.xxx	Approximately 65 thousand
Class C	The first three bytes of the IP address are for addressing the network section; the last byte addresses the host section	192.0.0.xxx to 223.255.255.xxx	254

In addition, there are Class D and Class E networks that are of no great significance in practical applications.

This subdivision also affects the subnet mask. Depending on the address class to which an IP address belongs, the subnet mask has a minimum value that is dependent on the permitted range of the IP address's "network section".

Address class	Minimum subnet mask
Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

The Ethernet option module complies with these address class-dependent subnet mask, i.e. if the subnet mask that a user enters is less than the address class mask that belongs to the IP address, the system uses the address class mask.

Example:

A PC with PROPROG wt II has an IP address of 192.075.191.188; an Ethernet option module has an IP address of 192.168.1.1. Both network nodes are located physically on the same network.

Since only the first bytes of both IP addresses are identical, you should choose 255.0.0.0 as the subnet mask on both systems to get the same result of logically ANDing the "IP address AND the subnet mask". However, since the IP addresses belong to a Class C network, the option module uses subnet mask 255.255.255.0. Logically ANDing the "IP address AND the subnet mask" yields different networks: 192.075.191.0 and 192.168.1.0. This means that communication is not possible, since the data packet are transferred to the set gateway.

You can avoid this problem and establish communication by deactivating the gateway and assigning gateway IP address 0.0.0.0.

f) Examples of IP networks

Example 1:

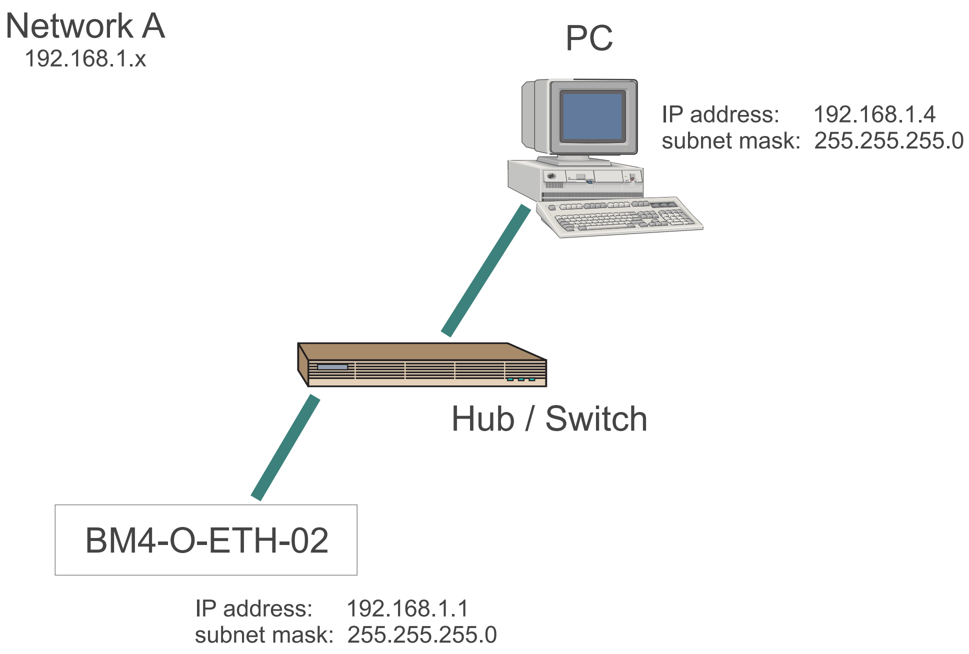


Figure 5: Example: Option module and PC on the same IP network

On network A, the components have the stated settings. Communication is possible between the PC and the option module, since logically ANDing both components "IP address AND subnet mask" yields the same network: 192.168.1.0

Example 2:

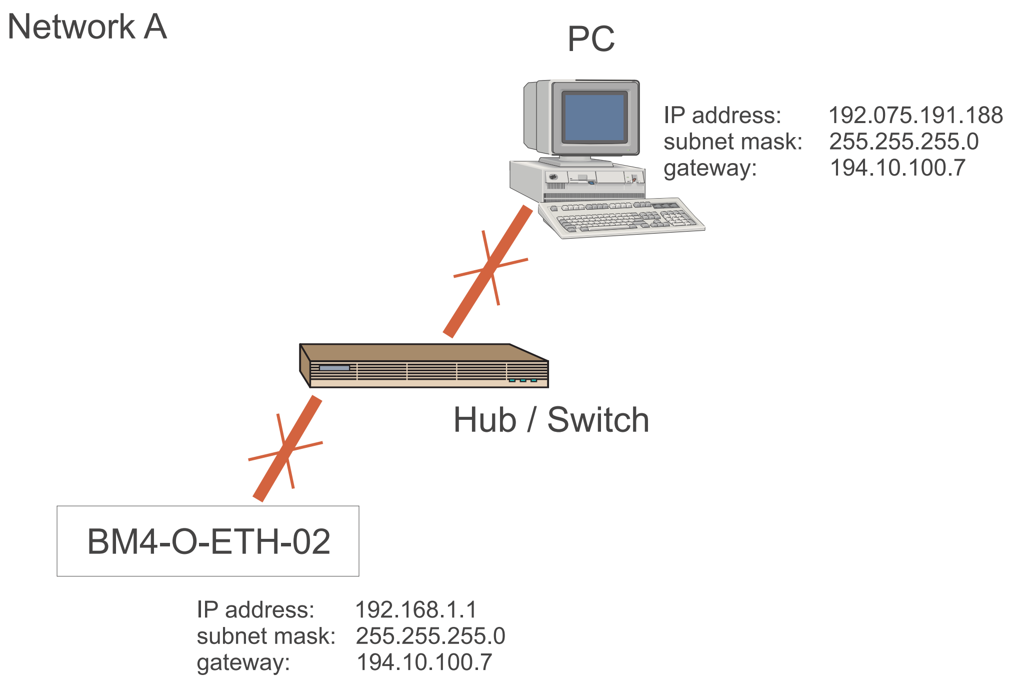


Figure 6: Example: Option module and PC on different IP networks

On network A, the components have the stated settings. Communication between the PC and the option module is not possible, since logically ANDing both components "IP address AND subnet mask" yields different networks. The PC and the option module would transfer the data packet to a gateway.

Example 3:

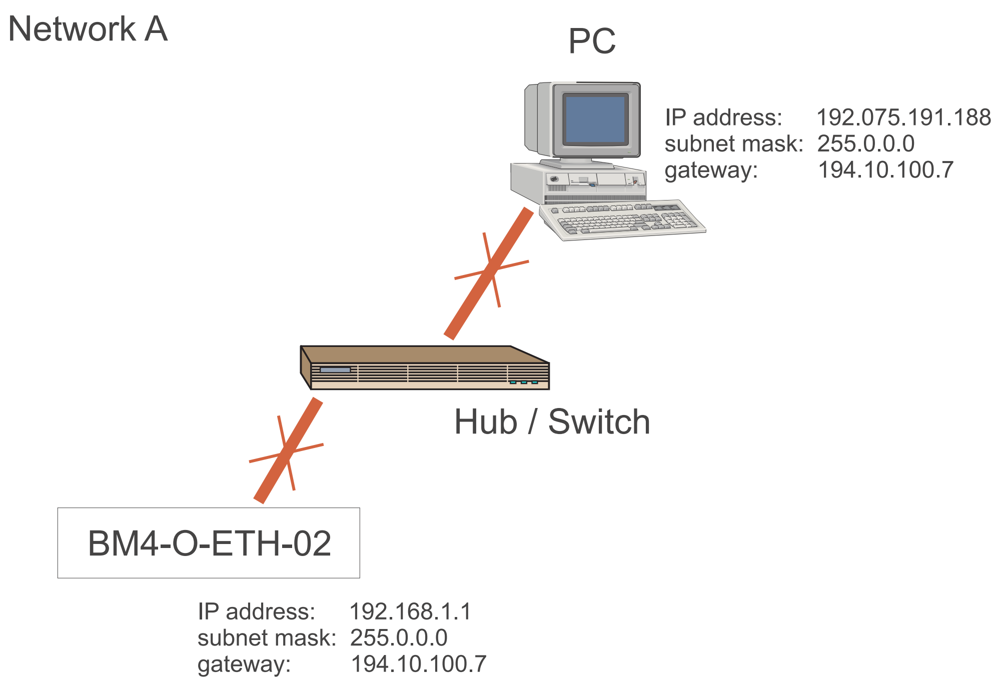


Figure 7: Example: Option module and PC with the wrong subnet mask

On network A, the components have the stated settings. Communication between the PC and the option module is not possible. Logically ANDing the "IP address AND the subnet mask" yields the same network 192.x.x.x. However, the option module detects a Class C network and uses subnet mask 255.255.255.0. and would transfer the data packets to a gateway. For a remedy, see example 4.

Example 4:

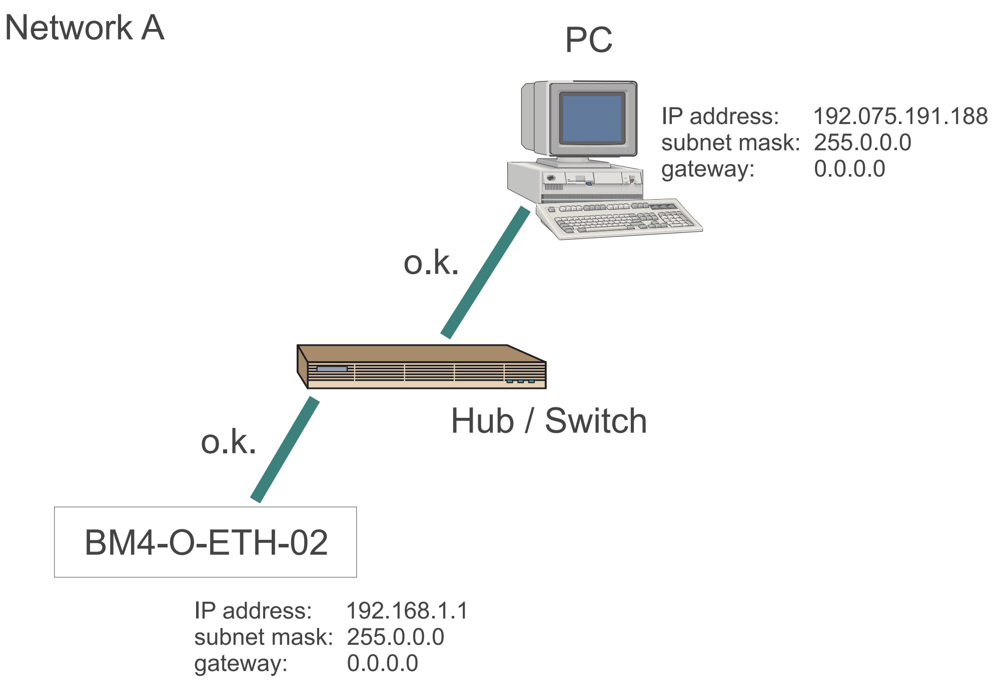


Figure 8: Example: Option module and PC without a gateway

On network A, the components have the stated settings. Communication between the PC and the option module is possible. Logically ANDing the "IP address AND the subnet mask" yields the network 192.x.x.x. However, the option module detects a Class C network and uses subnet mask 255.255.255.0. and would transfer the data packets to a gateway. However, since no gateway is set, the system transfers any data.

Example 5:

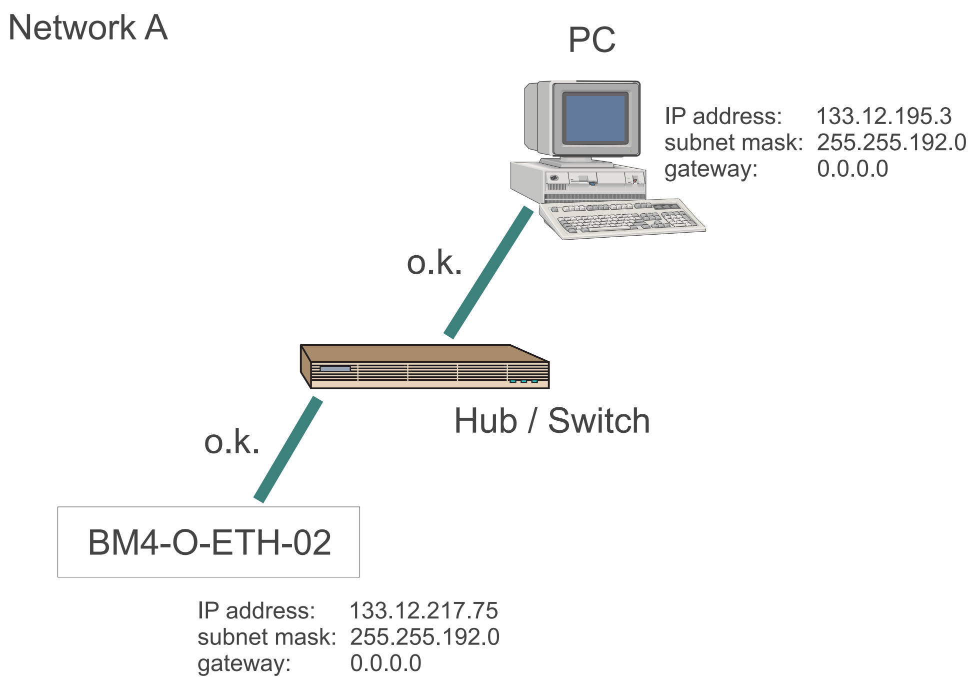


Figure 9: Example: Option module and PC on a Class-B IP network

On network A, the components have the stated settings. Communication is possible between the PC and the option module, since logically ANDing both components "IP address AND subnet mask" yields the same network: 133.12.192.0

Example 6:

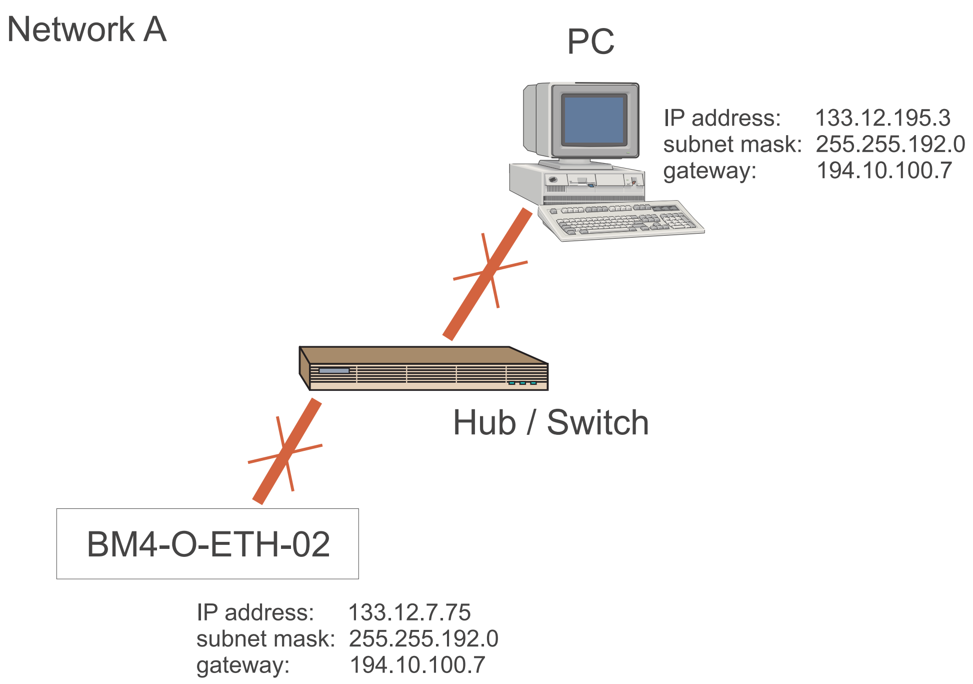


Figure 10: Example: Option module and PC on different Class-B IP networks

On network A, the components have the stated settings. Communication between the PC and the option module is not possible, since logically ANDing "the IP address AND the subnet mask" yields different networks: 133.12.192.0 and 133.12.0.0. The system would transfer the data packets to a gateway.

Example 7:

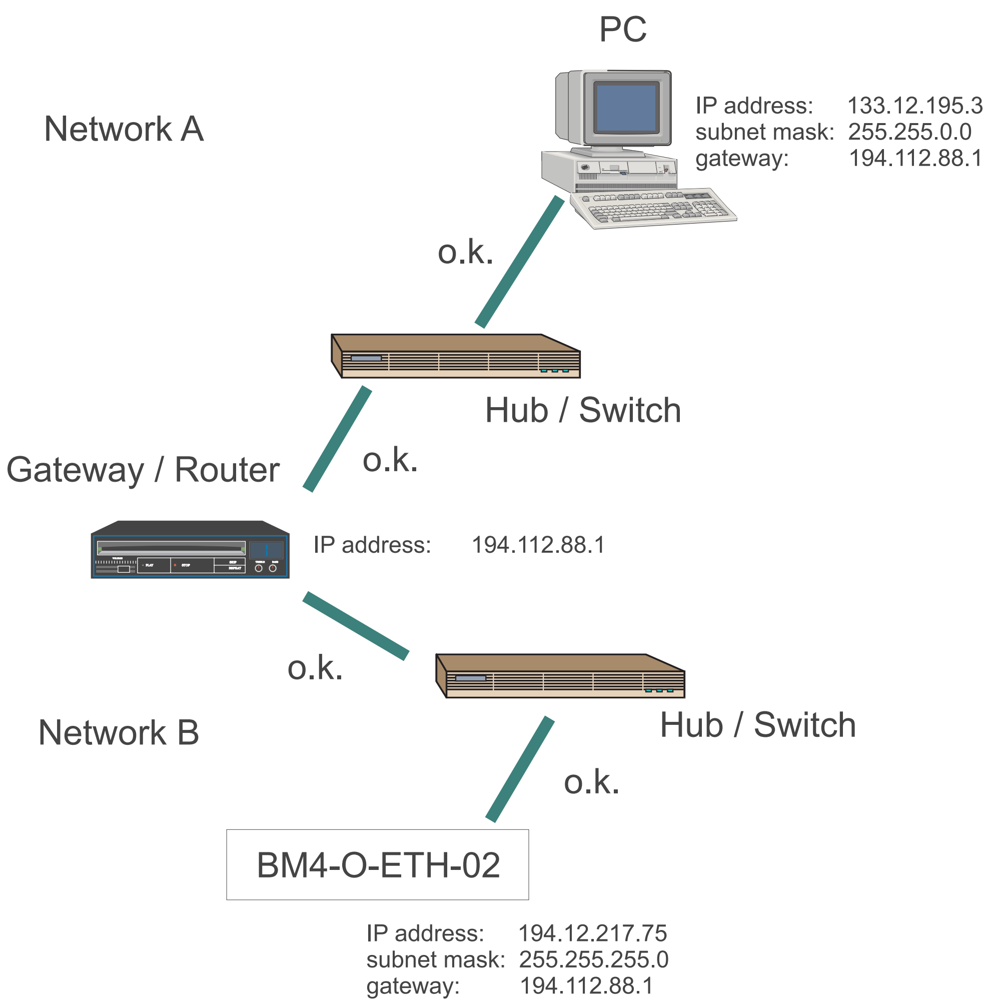


Figure 11: Example: Option module and PC with a gateway

On network A and network B, the components have the stated settings. The networks are connected via a gateway with IP address 194.112.88.1. Communication between the PC and the option module is possible, since logically ANDing both components with "IP address AND subnet mask" yields different networks – 133.12.0.0 and 194.12.217.0 – but both transfer data packets to the same gateway that carries out further- distribution.

3.2.6.2 TCP protocol

The **TCP (Transport Control Protocol)** is responsible for transporting and saving data and is located in the IP user data area. TCP works on the client server principle. For the duration of the data transfer, it makes a connection between two network nodes that is known as a session. Within any one session, the system divides up user data to individual numbered TCP packets and puts them back together again such that if necessary they can be transferred over different paths with different runtimes. Checksums make it possible to check and confirm correct reception. The system detects that individual TCP packets have been lost and it can request them again.

TCP packets are inserted in the user data area of an IP packet. Apart from this, TCP forwards the user data to the correct application program on the destination computer by assigning different port numbers to different applications (services).

The port number for die PROPROG communication is 0x5043 (= 20547 dec). The originating port is the port number of the sending application and at the same time the return sending port of the response.

As a user, you do not need to make any settings at the TCP level.

3.2.7 Interaction between Ethernet and TCP/IP, ARP

Up to now, addressing the nodes is only ensured by the IP addresses. At the Ethernet level, the network node only knows its own MAC-ID. To be able to deliver the IP packet in an Ethernet frame, however, the node must know the receiver's MAC-ID. The ARP protocol (Address Resolution Protocol) was developed to determine this. To find out the pair of Ethernet IP addresses of a network node, the ARP protocol sends an Ethernet broadcast containing the known IP address of the receiver. The receiver that has this IP address replies to the requesting system with a packet that contains the pair of Ethernet IP addresses. To avoid unnecessary ARP requests, the reply packet is stored on the requesting system in an ARP table. To keep this ARP table as small as possible, the system deletes the entries at specific, system-dependent time intervals. From the command line, you can read out the current ARP table for your PC:

```
arp -a
```

If communication has already taken place with the Ethernet option module, you will find the option module in the listing. To trigger communication, you can use the ping command from the command line.

```
ping 192.168.1.1
```

Assuming that your network is correctly configured (the IP addresses, subnet masks and the gateways), the option module with IP address 192.168.1.1 will reply in the following form:

```
Reply from 192.168.1.1: Bytes.....
```

If you do not get a reply, check your network settings.

If you do get a reply, the option module is now also listed in your PC's ARP table. It is not possible to read out the Ethernet option module's ARP table.

3.2.8 Proxy

A proxy is a computer that accepts requests from a network node and forwards them to the intended destination. The destination transfers the result of the request to the proxy, which, for its part, can return the result to the other network node. A proxy can work on different application levels and check the data to be transferred. If often has a large amount of memory (a proxy cache) to buffer requested data and, in the case of a possible subsequent request from a different network node or from the same one, to make available the data to the actual destination on a direct basis, i.e. without needing an additional request. A firewall proxy is generally used if an internal network, e.g. an Intranet/LAN is intended to make a protected connection to another network (e.g. the Internet). In this connection, the proxy acts as a kind of gateway with IP and packet filters. On the basis of the source, destination, port and packet type information that each packet contains, the proxy can inspect data transfer from one network to another one and if necessary it can limit it. Using appropriate software functions, the proxy can carry out further security tasks above the IP level.

3.2.9 Application layer PROPROG wt / OmegaOS

PROPROG wt / OmegaOS data packets are transferred in the user data area of TCP packets. Users do not need to make any further settings here. You must only set the communications port in PROPROG wt (see the chapter entitled ▷Setting a connection via Ethernet-TCP/IP in PROPROG wt II / ProProg wt III ◁ from page 47 onward). The Ethernet option module makes available two communications channels; i.e. using the option module you can operate in-parallel a maximum of

two ProMaster

or

one ProMaster and one ProProg wt III

or

two ProProg wt III

or

one ProMaster (or ProProg wt III) and one OPC application

or

two OPC applications

OR

two PROPROG wt II

or

one PROPROG wt II and one OPC application

or

two OPC applications

3.3 Configuring Ethernet TCP/IP networks

3.3.1 Overview

To be able to use the Ethernet option module via Ethernet with TCP/IP using ProMaster or PROPROG or via OPC, you must carry out the following steps:

- physically commission the network (see the Operating Instructions of option module BM4-O-ETH-01 / BM4-O-ETH-02 and the Operating Instructions of the network components)
- specify the network data: the IP addresses and subnet masks (see the previous chapter ▹TCP/IP ◃ from page 21 onward)
- set IP addresses, subnet masks and gateways in the network components (see the next chapter)
- ProMaster:
  - Set the communication port in ProMaster (see chapter entitled ▹Setting a connection via Ethernet-TCP/IP in ProMaster ◃ from page 46 onward).
  - Configuring the CANopen network with ProMaster (see chapter entitled ▹ProMaster, ProCANopen, ProProg wt III and Motion Control ◃ from page 57 onward).

Or

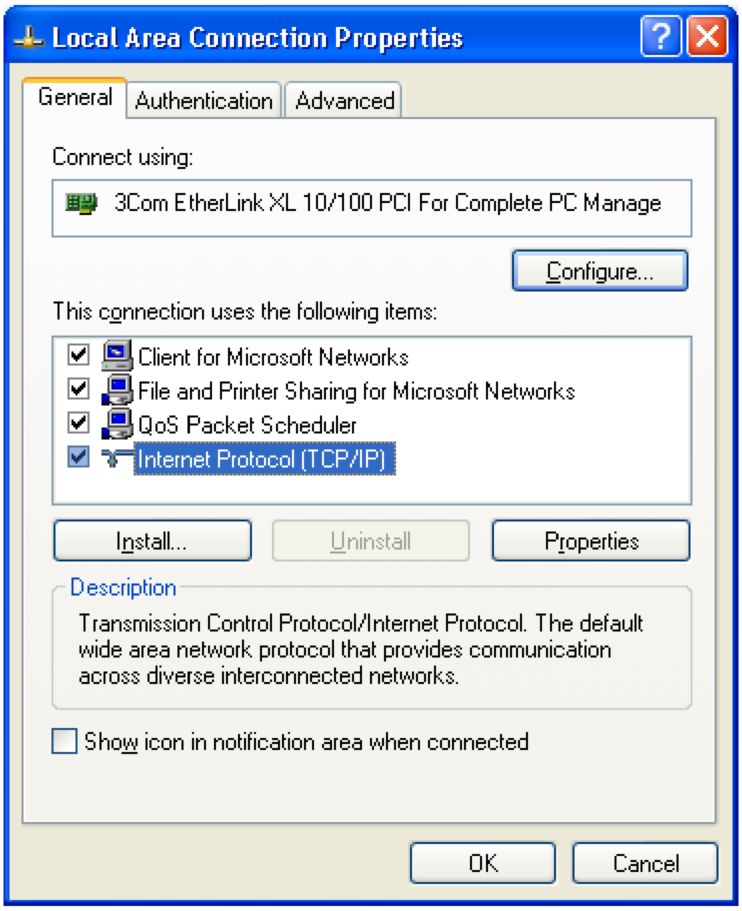
- PROPROG wt II / ProProg wt III:
  Set the communication port in PROPROG (see chapter entitled ▹Setting a connection via Ethernet-TCP/IP in PROPROG wt II / ProProg wt III ◃ from page 47 onward).

3.3.2 Configuring the Windows PC

Since PROPROG wt and the OPC server need a Windows operating system, we will only describe the settings that are necessary for Windows operating systems. The basic condition is that the TCP/IP protocol is installed on your Windows PC and that your computer has a configured network adapter. For details about this installation, refer to the Windows documentation.

3.3.2.1 Configuring TCP/IP under Windows XP

- Click on the *Start* pushbutton, and then choose My Network Places. Click on *Network Tasks* to View *Network Connections.*
- Double-click on the *Local Area Connection* icon and click on the *Properties* → pushbutton; the system displays a list of the components that use your network adapter.
- Check *Internet Protocol (TCP/IP)* and click on the *Properties* pushbutton.


Local Area Connection Properties

General | Authentication | Advanced

Connect using:

3Com EtherLink XL 10/100 PCI For Complete PC Manage

Configure...

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol (TCP/IP)

Install... | Uninstall | Properties

Description

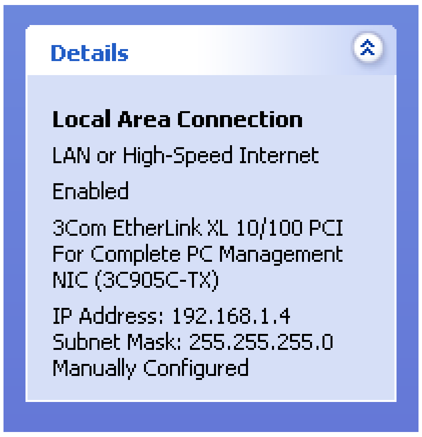
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

☐ Show icon in notification area when connected

OK | Cancel

Figure 12: Setting up Windows XP with TCP/IP – overview of LAN connections

- You can now choose between *Obtain an IP address automatically* or *Use the following IP address*. If you obtain the IP address automatically, you can query the current setting either in field *Details* of *LAN connection* in the *Network Connections* window, or


Details

Local Area Connection

LAN or High-Speed Internet

Enabled

3Com EtherLink XL 10/100 PCI For Complete PC Management NIC (3C905C-TX)

IP Address: 192.168.1.4
Subnet Mask: 255.255.255.0
Manually Configured

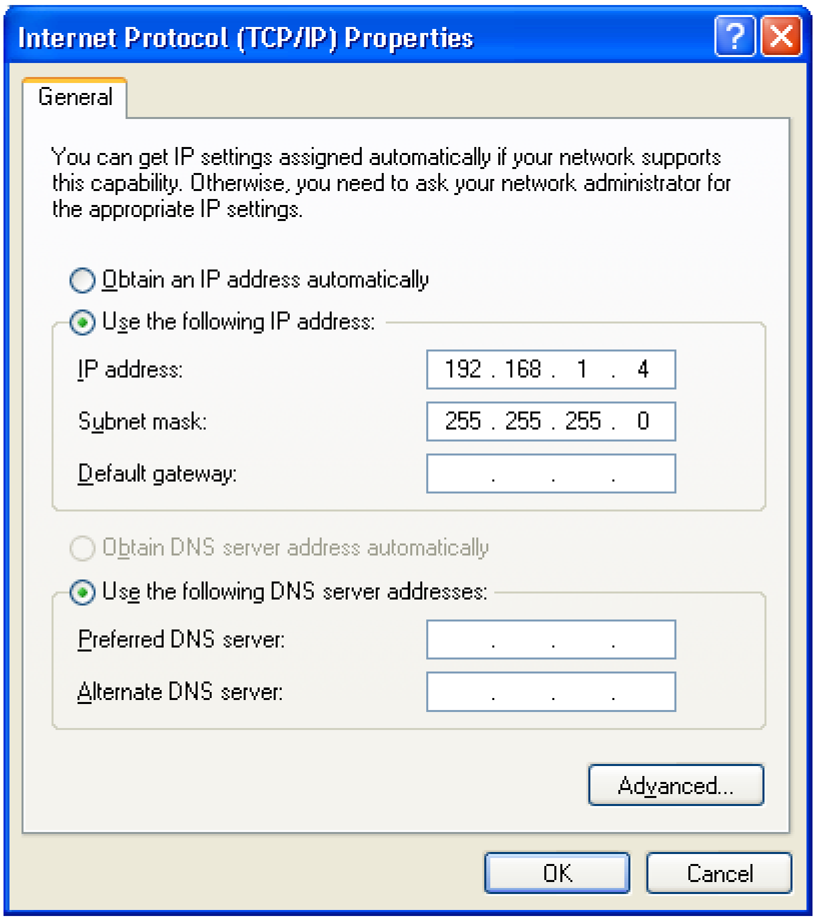
Figure 13: Setting up Windows XP with TCP/IP – details of LAN connections

on the command line by entering

```
ipconfig
```

If the settings match the ones you specified before, you can end the dialog and configuration. Note, however, that the system makes these settings dynamically, i.e. the next time you start your PC, this may result in different settings.

Check *Use the following IP address* if you want to enter special values for the IP address, the subnet mask and the gateway:


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address:	192 . 168 . 1 . 4
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

Preferred DNS server:	. . .
Alternate DNS server:	. . .

Figure 14: Setting up Windows XP with TCP/IP – TCP/IP properties

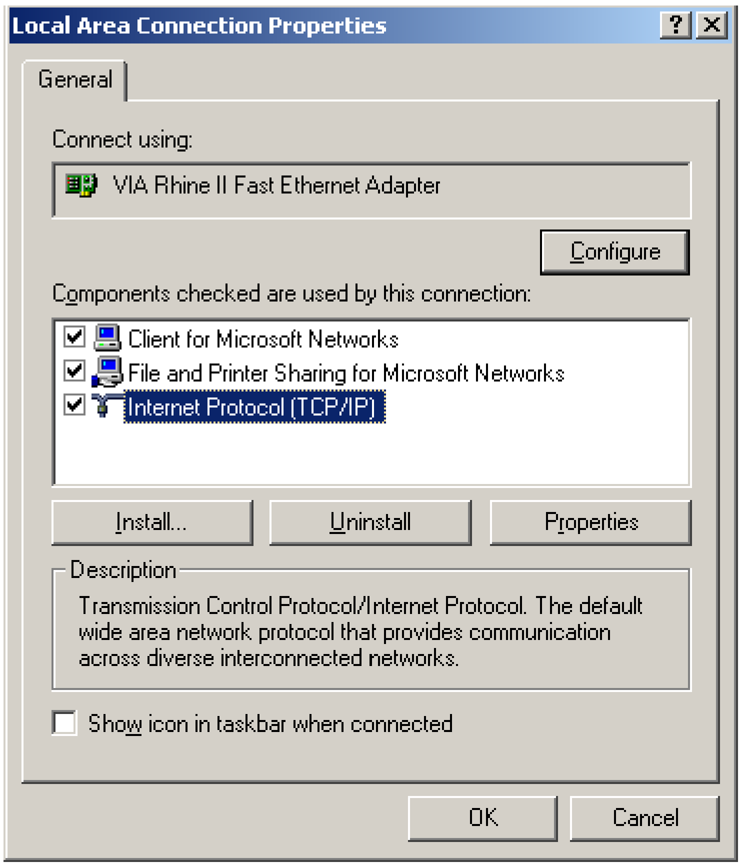
- Close the windows that you opened for configuration by clicking on the *OK* pushbutton.

This completes installation of TCP/IP support on your Windows XP system. Under Windows XP, you do not need to reboot your computer.

3.3.2.2 Configuring TCP/IP under Windows 2000

- Click on the *Start* pushbutton and under *Settings* open *Network and Dial-up Connections*

- Double-click on the *Local Area Connection* icon and click on the *Properties* → pushbutton; the system displays a list of the components that use your network adapter.
- Check *Internet Protocol (TCP/IP)* and click on the *Properties* pushbutton.


Local Area Connection Properties

General

Connect using:

VIA Rhine II Fast Ethernet Adapter

Components checked are used by this connection:

- ✓ Client for Microsoft Networks
- ✓ File and Printer Sharing for Microsoft Networks
- ✓ Internet Protocol (TCP/IP)

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

☐ Show icon in taskbar when connected

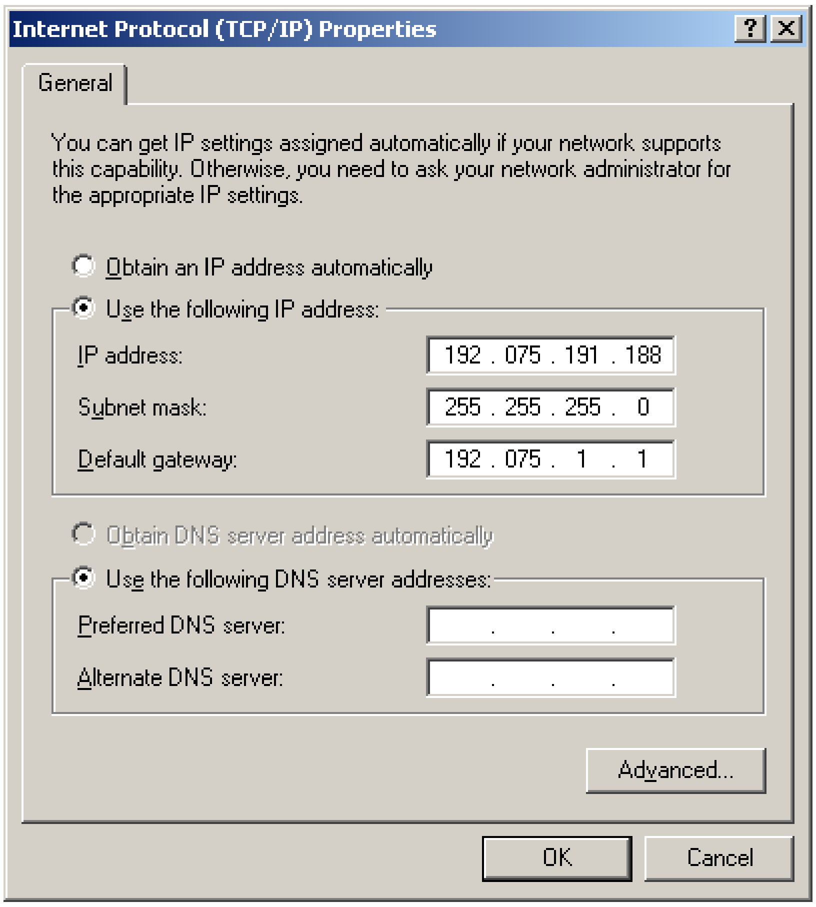
Figure 15: Setting up Windows 2000 with TCP/IP – overview of LAN connections

- You can now choose between *Obtain IP address automatically* or *Use the following IP address*. If you want to *Obtain IP address automatically,* you can now query the current setting on the command line by entering

```
ipconfig
```

If the settings match the ones you specified before, you can end the dialog and configuration. Note, however, that the system makes these settings dynamically, i.e. the next time you start your PC, this may result in different settings.

Check *Use the following IP address* if you want to enter special values for the IP address, the subnet mask and the gateway:


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:

IP address: 192 . 075 . 191 . 188
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 075 . 1 . 1

( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:

Preferred DNS server: . . .
Alternate DNS server: . . .

Figure 16: Setting up Windows 2000 with TCP/IP – TCP/IP properties

- Close the windows that you opened for configuration by clicking on the *OK* pushbutton.

This completes installation of TCP/IP support on your Windows 2000 system. Under Windows 2000, you do not need to reboot your computer.

3.3.3 Configuring the Ethernet option module and checking settings

You have two options for configuring the Ethernet option module for TCP/IP communication:

- by using the default settings
- by freely configuring using PROPROG wt II / ProProg wt III

3.3.3.1 Default settings for TCP/IP

The default settings for TCP/IP communication of the Ethernet option module are:

IP address:	192.168.1.1+ "DIP switch"
Subnet mask:	255.255.255.0
Gateway:	0.0.0.0

Using DIP switches 1 - 5 (S5000 on the option module), you can set IP addresses in the range 192.168.1.1 to 192.168.1.32. In order to set the DIP switches of the option module see ▷Operating Instructions Ethernet with CANopen-Master for b maXX drive PLC◁.

3.3.3.2 Freely configuring TCP/IP

a) Prepare for configuration using the PROPROG wt II / ProProg wt III

You can also freely set the values for the IP address, the subnet mask and the gateway. To do this, proceed as follows:

1 Create a PROPROG wt II / ProProg wt III project for the b maXX drive PLC if you have not already created a separate project for your application.

2 PROPROG wt II: In this project, link library BM_TYPES_20bd03 (or higher) if it is not already present.
ProProg wt III: In this project, link library BM_TYPES_30bd01 (or higher) if it is not already present.

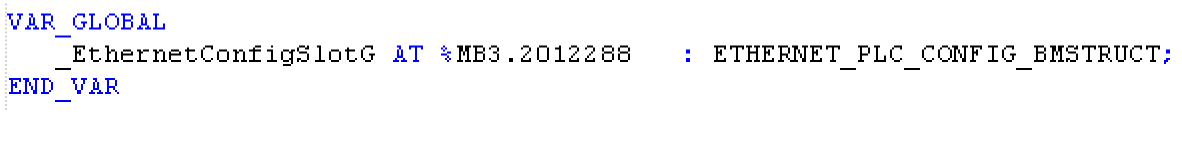
3 Create a POU that can later be called in a cold boot and warm restart task, assuming that one is not present.

4 Create a global variable of data type ETHERNET_PLC_CONFIG_BMSTRUCT. You must connect this global variable to the base address to the Ethernet configuration. The base address depends on the slot in which the Ethernet option module is fitted. The following addresses are possible:

Slot	Base address for Ethernet configuration
G	%MB3.2012288
H	%MB3.3012288
J	%MB3.4012288
K	%MB3.5012288
L	%MB3.6012288
M	%MB3.7012288

Example if the Ethernet option module is fitted in slot G:

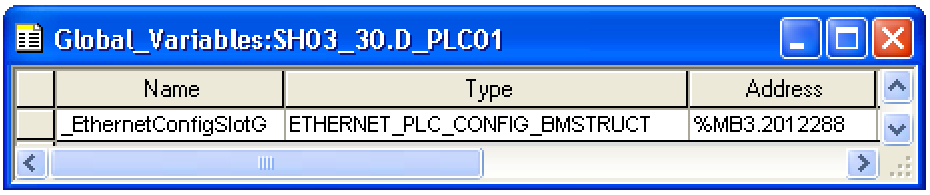
PROPROG wt II:



```
VAR_GLOBAL
    _EthernetConfigSlotG AT %MB3.2012288    : ETHERNET_PLC_CONFIG_BMSTRUCT;
END_VAR
```

Figure 17: Example: Global variable of type ETHERNET_PLC_CONFIG_BMSTRUCT (PROPROG wt II)

ProProg wt III:



Name	Type	Address
_EthernetConfigSlotG	ETHERNET_PLC_CONFIG_BMSTRUCT	%MB3.2012288

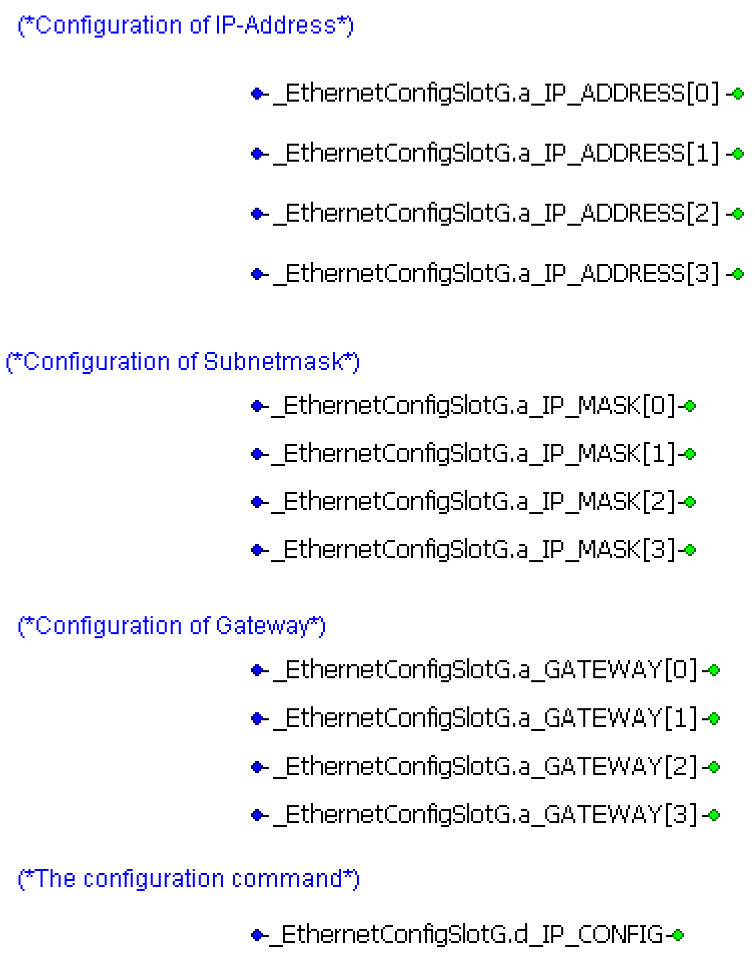
Figure 18: Example: Global variable of type ETHERNET_PLC_CONFIG_BMSTRUCT (ProProg wt III)

Where:

`_EthernetConfigSlotG`	is the variable name with data type short designation "_" for STRUCT
`%MB3.2012288`	is the address for Ethernet configuration of slot G
`ETHERNET_PLC_CONFIG_BMSTRUCT`	is the data type of the variable

5 To set the IP address, subnet mask and gateway, you need structure elements a_IP_ADDRESS, a_IP_MASK and a_GATEWAY of the created global variable. All three elements consist of one field (array) containing four entries each of data type USINT. Each address starts in field index 0 with the network section. To activate the configuration, you additionally need structure element d_IP_CONFIG of data type DWORD. Create these elements in the POU for cold boot/warm restart.

Example:



```
(*Configuration of IP-Address*)

        _EthernetConfigSlotG.a_IP_ADDRESS[0]
        _EthernetConfigSlotG.a_IP_ADDRESS[1]
        _EthernetConfigSlotG.a_IP_ADDRESS[2]
        _EthernetConfigSlotG.a_IP_ADDRESS[3]

(*Configuration of Subnetmask*)
        _EthernetConfigSlotG.a_IP_MASK[0]
        _EthernetConfigSlotG.a_IP_MASK[1]
        _EthernetConfigSlotG.a_IP_MASK[2]
        _EthernetConfigSlotG.a_IP_MASK[3]

(*Configuration of Gateway*)
        _EthernetConfigSlotG.a_GATEWAY[0]
        _EthernetConfigSlotG.a_GATEWAY[1]
        _EthernetConfigSlotG.a_GATEWAY[2]
        _EthernetConfigSlotG.a_GATEWAY[3]

(*The configuration command*)

        _EthernetConfigSlotG.d_IP_CONFIG
```

Figure 19: Example: Elements of a global variable of type ETHERNET_PLC_CONFIG_BMSTRUCT

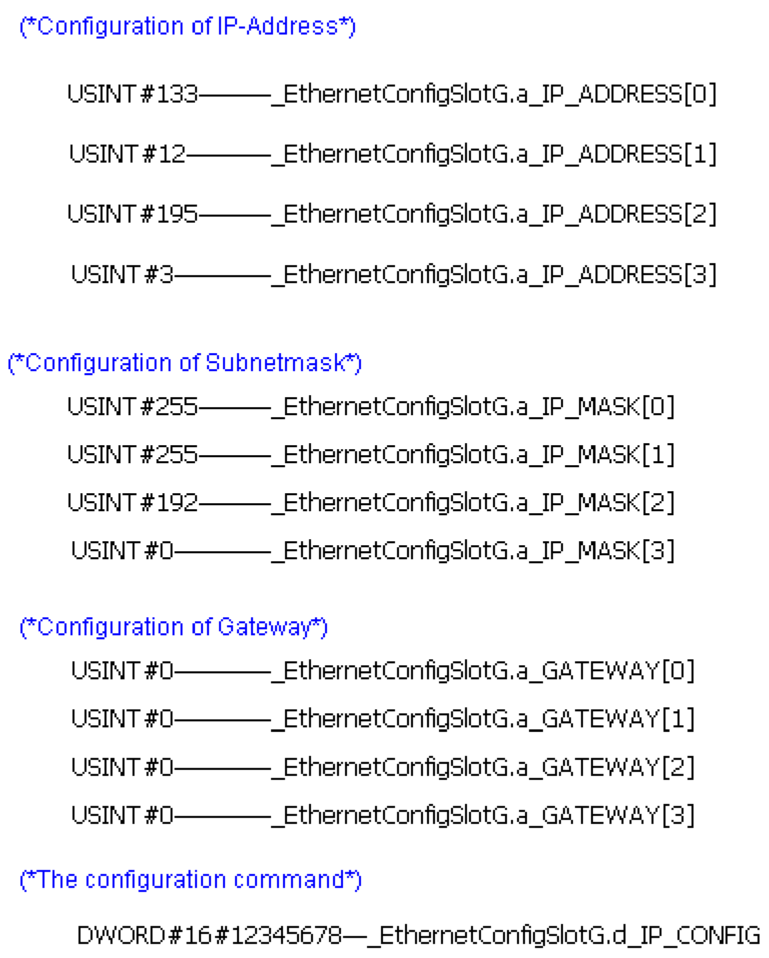
6 There are two options for configuring the IP address:
A fixed IP address or a DIP switch-dependent IP address. In the first case, you set an IP address that is independent of the DIP switches. In the second case, you add the value (1-5) of the DIP switch to the configured IP address. The value of element d_IP_CONFIG decides which procedure is to be used.

b) Assigning a fixed IP address

To assign a fixed IP address that is independent of the DIP switches, you assign the structure elements of the global variable for Ethernet configuration with the required values. Finally, you must write element d_IP_CONFIG with DWORD#16#12345678. When writing to the elements, you must absolutely comply with the following sequence

a_IP_ADDRESS → a_IP_MASK → a_GATEWAY → d_IP_CONFIG

Example:



```
(*Configuration of IP-Address*)

USINT#133———_EthernetConfigSlotG.a_IP_ADDRESS[0]

USINT#12————_EthernetConfigSlotG.a_IP_ADDRESS[1]

USINT#195———_EthernetConfigSlotG.a_IP_ADDRESS[2]

USINT#3—————_EthernetConfigSlotG.a_IP_ADDRESS[3]

(*Configuration of Subnetmask*)
USINT#255———_EthernetConfigSlotG.a_IP_MASK[0]
USINT#255———_EthernetConfigSlotG.a_IP_MASK[1]
USINT#192———_EthernetConfigSlotG.a_IP_MASK[2]
USINT#0—————_EthernetConfigSlotG.a_IP_MASK[3]

(*Configuration of Gateway*)
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[0]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[1]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[2]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[3]

(*The configuration command*)

DWORD#16#12345678—_EthernetConfigSlotG.d_IP_CONFIG
```

Figure 20: Example: Assigning a fixed IP address, gateway and subnet mask for the option module

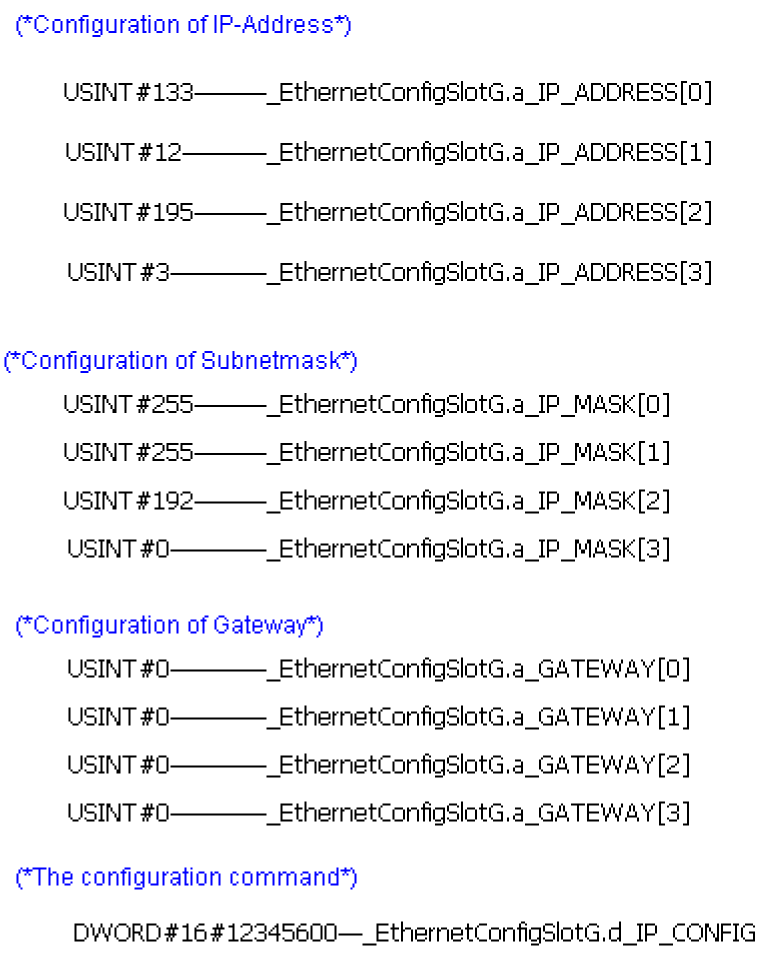
For the Ethernet option module, this yields the IP address 133.12.195.3 and the subnet mask 255.255.192.0. The gateway has the address 0.0.0.0, i.e. no gateway is used.

c) Assigning a variable, DIP switch-dependent IP address

To assign a variable IP address that is dependent on the DIP switches, you assign the structure elements of the global variable for Ethernet configuration with the required values. Finally, you must write element d_IP_CONFIG with DWORD#16#12345600. When writing to the elements, you must absolutely comply with the following sequence

a_IP_ADDRESS → a_IP_MASK → a_GATEWAY → d_IP_CONFIG

Example:



```
(*Configuration of IP-Address*)

USINT#133———_EthernetConfigSlotG.a_IP_ADDRESS[0]
USINT#12————_EthernetConfigSlotG.a_IP_ADDRESS[1]
USINT#195———_EthernetConfigSlotG.a_IP_ADDRESS[2]
USINT#3—————_EthernetConfigSlotG.a_IP_ADDRESS[3]

(*Configuration of Subnetmask*)
USINT#255———_EthernetConfigSlotG.a_IP_MASK[0]
USINT#255———_EthernetConfigSlotG.a_IP_MASK[1]
USINT#192———_EthernetConfigSlotG.a_IP_MASK[2]
USINT#0—————_EthernetConfigSlotG.a_IP_MASK[3]

(*Configuration of Gateway*)
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[0]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[1]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[2]
USINT#0—————_EthernetConfigSlotG.a_GATEWAY[3]

(*The configuration command*)

DWORD#16#12345600—_EthernetConfigSlotG.d_IP_CONFIG
```

Figure 21: Example: Assigning a variable IP address, gateway and subnet mask for the option module

For the Ethernet option module, this yields the IP address 133.12.195.3 + DIP switch and the subnet mask 255.255.192.0. The gateway has the address 0.0.0.0, i.e. no gateway is used.

d) Activating the TCP/IP configuration

The POU containing the elements for TCP/IP configuration has been created. After this, carry out the following steps:

1 Now link the POU in a cold boot and a warm restart task. After this, compile the program and load it as a boot project on the b maXX drive PLC.

2 Switch the b maXX unit off and back on again twice at an interval of approximately 10 seconds. It is absolutely crucial to switch off and back on again twice.

The Ethernet option module now has the TCP/IP configuration that you set. You can remove the program code from the created POU. The TCP/IP configuration is retained. In the same way, you can load another project (even a boot project) on the b maXX drive PLC without losing the TCP/IP configuration. You can only change the TCP/IP configuration of the Ethernet option module by implementing again the code lines that were described above.

e) Checking the TCP/IP configuration

You can use PROPROG wt II / ProProg wt III to check the settings of the Ethernet option module's TCP/IP configuration.

To do this, proceed as follows:

1 Create a PROPROG wt II / ProProg wt III project for the b maXX drive PLC if you have not already created a separate project for your application or by the configuration.

2 PROPROG wt II: In this project, link library BM_TYPES_20bd03 (or higher) if it is not already present.
ProProg wt III: In this project, link library BM_TYPES_30bd01 (or higher) if it is not already present.

3 Create a global variable of data type ETHERNET_PLC_DIAG_BMSTRUCT. You must connect this global variable to the base address to the Ethernet diagnostics. In this connection, the base address depends on the slot in which the Ethernet option module is fitted. The following addresses are possible:

Slot	**Base address for Ethernet diagnostics**
G	%MB3.2012320
H	%MB3.3012320
J	%MB3.4012320
K	%MB3.5012320
L	%MB3.6012320
M	%MB3.7012320

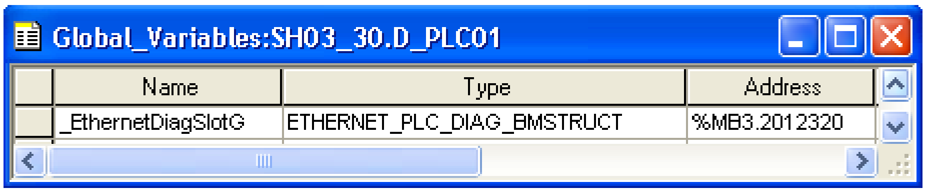
Example if the Ethernet option module is fitted in slot G:

PROPROG wt II:

```
VAR_GLOBAL
   _EthernetDiagSlotG   AT %MB3.3012320   : ETHERNET_PLC_DIAG_BMSTRUCT;
END_VAR
```

Figure 22: Example: Global variable for checking the TCP/IP configuration (PROPROG wt II)

ProProg wt III:


Global_Variables:SH03_30.D_PLC01

Name	Type	Address
_EthernetDiagSlotG	ETHERNET_PLC_DIAG_BMSTRUCT	%MB3.2012320

Figure 23: Example: Global variable for checking the TCP/IP configuration (ProProg wt III)

Where:

`_ EthernetDiagSlotG`	is the variable name with data type short designation "_" for STRUCT
`%MB3.2012320`	is the address for Ethernet diagnostics of slot G
`ETHERNET_PLC_DIAG_BMSTRUCT`	is the data type of the variable

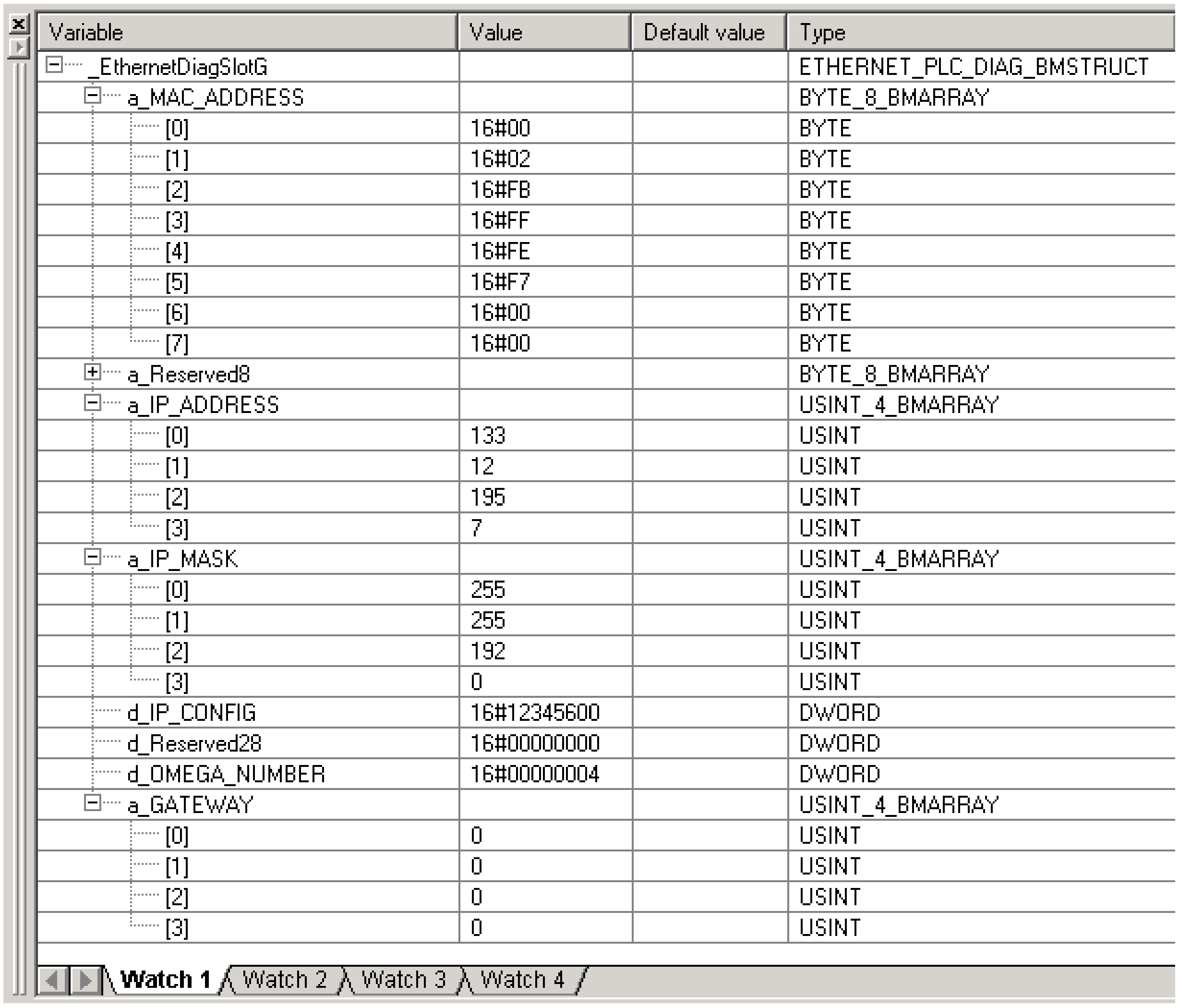
4 Structure elements a_MAC_ADDRESS, a_IP_ADDRESS, a_IP_MASK, d_IP_CONFIG, d_DIP_SWITCH, and a_GATEWAY of the created global variable are available for diagnostics. The individual elements have the following meanings:

Element	**Meaning**
a_MAC_ADDRESS	MAC address of the option module. Field (array) containing 8 entries of data type BYTE in each case. Field indexes 0 to 5 are valid. Indexes 0 to 2 designate the code for manufacturer identification; indexes 3 to 5 designate the manufacturer code for the respective device.
a_IP_ADDRESS, a_IP_MASK, a_GATEWAY	Active IP address, subnet mask and gateway of the option module. All three elements consist of one field (array) containing four entries each of data type USINT. Each address starts in field index 0 with the network section.
d_IP_CONFIG	Designates the type of evaluation of the IP address. 16#00000000 No evaluation 16#12345678 Fixed IP address 16#12345600 IP address + DIP switch 1-5
d_DIP_SWITCH	Settings of DIP switches 1-5

5 Compile the program and load it as a project (even a boot project) on the b maXX drive PLC. You do not need to create a special POU.

6 Start the program and switch to online mode. Add the created global variable to TCP/IP diagnostics in the watch window; display the watch window and open the relevant structure elements. You can now see the set data.

Example:



Variable	Value	Default value	Type
_EthernetDiagSlotG			ETHERNET_PLC_DIAG_BMSTRUCT
a_MAC_ADDRESS			BYTE_8_BMARRAY
[0]	16#00		BYTE
[1]	16#02		BYTE
[2]	16#FB		BYTE
[3]	16#FF		BYTE
[4]	16#FE		BYTE
[5]	16#F7		BYTE
[6]	16#00		BYTE
[7]	16#00		BYTE
a_Reserved8			BYTE_8_BMARRAY
a_IP_ADDRESS			USINT_4_BMARRAY
[0]	133		USINT
[1]	12		USINT
[2]	195		USINT
[3]	7		USINT
a_IP_MASK			USINT_4_BMARRAY
[0]	255		USINT
[1]	255		USINT
[2]	192		USINT
[3]	0		USINT
d_IP_CONFIG	16#12345600		DWORD
d_Reserved28	16#00000000		DWORD
d_OMEGA_NUMBER	16#00000004		DWORD
a_GATEWAY			USINT_4_BMARRAY
[0]	0		USINT
[1]	0		USINT
[2]	0		USINT
[3]	0		USINT

Figure 24: Example: Reading out the TCP/IP configuration in the watch window

3.4 Setting a connection via Ethernet-TCP/IP

3.4.1 Setting a connection via Ethernet-TCP/IP in ProMaster

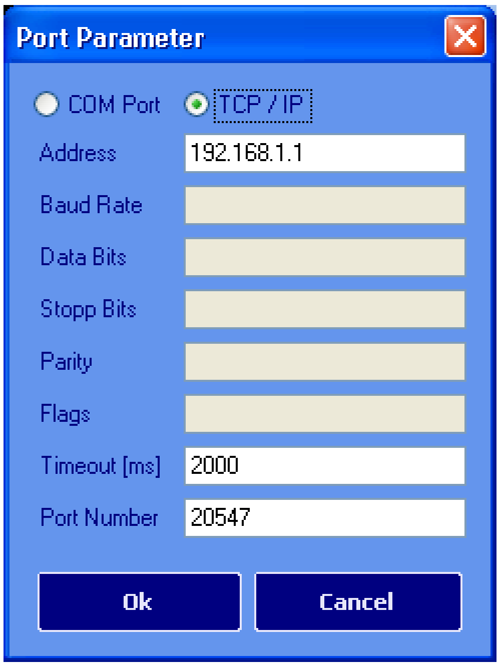
You set the uses of Ethernet-TCP/IP in ProMaster individually for each device.

In this chapter, we will describe how you set the uses of Ethernet TCP/IP in ProMaster for a b maXX drive PLC. To do this, proceed as follows:

- Open your complete configurated ProMaster project (in which the ProProg wt III project must be linked). Assign the device with b maXX drive PLC in ProMaster network view and select it via the context menu „Communication settings“.

The window „Port parameter“ will be opened.

- Activate the radio button „TCP/IP“ and write the TCP/IP address of the b maXX drive PLC resource in the edit box „Address“.


Port Parameter

( ) COM Port [✓ TCP / IP]

Address	192.168.1.1
Baud Rate	
Data Bits	
Stopp Bits	
Parity	
Flags	
Timeout [ms]	2000
Port Number	20547

Figure 25: Setting the TCP/IP configuration in ProMaster for a b maXX drive PLC resource

To be able to access the b maXX drive PLC via Ethernet, an Ethernet compliant module (e.g. BM4-O-ETH-02) must be fitted in the b maXX system in addition to the b maXX drive PLC.

The TCP/IP address "192.168.1.1" corresponds to the address that is preinstalled on the Ethernet module when it leaves the factory, so that the user can contact the module first for the basic initialization to make the adjustments of its TCP/IP address for the TCP/IP network at the machine.

The communication setting of ProMaster will be accepted automatically in the linked ProProg wt III program.

NOTE

If you change the communication settings only in ProProg wt III project, these communication settings will be **not** accepted in ProMaster

3.4.2 Setting a connection via Ethernet-TCP/IP in PROPROG wt II / ProProg wt III

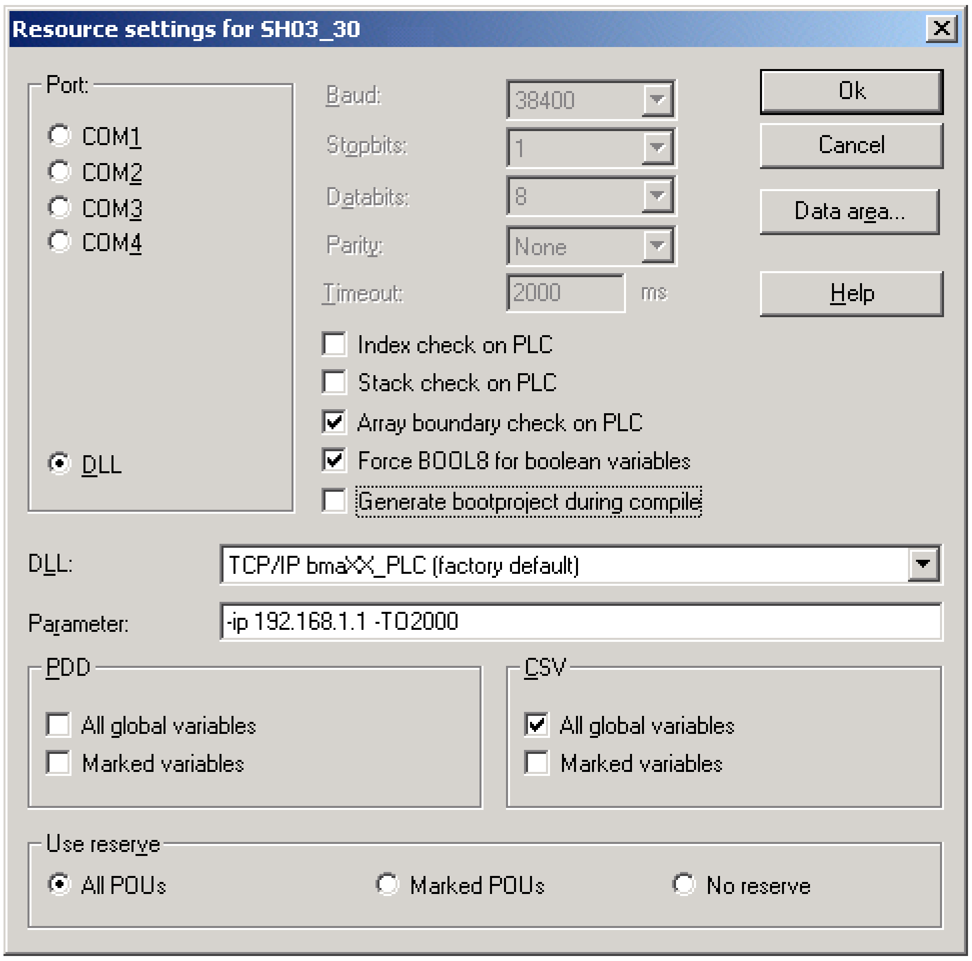
NOTE

If you change the communication settings only in ProProg wt III project, these communication settings will be **not** accepted in ProMaster

You set the uses of Ethernet-TCP/IP in PROPROG wt II / ProProg wt III individually for each resource in the user program. To do this, proceed as follows:

- In PROPROG wt II / ProProg wt III project open the dialog „Resource settings for SH03_30“ via the context menu of the resource b maXX drive PLC
- Set the port to „DLL“

Setting the Ethernet communication source by choosing or stating the TCP/IP address


Resource settings for SH03_30

Port: COM1 / COM2 / COM3 / COM4 / [✓ DLL]

Baud: 38400
Stopbits: 1
Databits: 8
Parity: None
Timeout: 2000 ms

- [ ] Index check on PLC
- [ ] Stack check on PLC
- [x] Array boundary check on PLC
- [x] Force BOOL8 for boolean variables
- [ ] Generate bootproject during compile

DLL: TCP/IP bmaXX_PLC (factory default)
Parameter: -ip 192.168.1.1 -TO2000

PDD:
- [ ] All global variables
- [ ] Marked variables

CSV:
- [x] All global variables
- [ ] Marked variables

Use reserve: [✓ All POUs] Marked POUs No reserve

Figure 26: Setting TCP/IP in PROPROG wt II for one resource

After changing the port to "DLL", you can use the DLL menu to choose applications "Soft-PLC" (=TCP/IP local host (this PC))“ and „b maXX drive PLC - access via Ethernet" (=TCP/IP bmaXX_PLC (brand new)):

- Soft-PLC:
  You do not need to change the preset TCP/IP address in field "Parameter" if the Soft-PLC is installed on the same computer. If installing Soft-PLC on another computer, you must enter here the appropriate TCP/IP target address (or the appropriate network name) to be able to access Soft-PLC via TCP/IP from PROPROG wt II / ProProg wt III.
- b maXX drive PLC:
  To be able to access the b maXX drive PLC via Ethernet, the Ethernet option module must be fitted in the b maXX 4400 basic unit in addition to the b maXX drive PLC. The TCP/IP address "192.168.1.1" that is preset in the parameter field corresponds to the IP address that is preinstalled on the option module when it leaves the factory.

4 CANOPEN

In this chapter, you will find information about data exchange via CANopen. CANopen is only available with the BM4-O-ETH-02 and BM4-O-CAN-04 option modules.

4.1 General information on CANopen and using the CANopen-Master option module

CANopen is a commonly used fieldbus application layer based on the Controller Area Network (CAN) bus system that was published by the international CAN organization CAN in Automation e.V. (CiA). The CANopen mechanisms and functionality are described by different profiles. The communication profile defines how data is exchanged and makes specifications that are generally applicable to all CANopen devices. With the CANopen master for b maXX drive PLC option module, CANopen and CANopen master functions according to the CiA communication profile DS 301 and CiA device profile DS 405, such as

- Exchanging process data via Process Data Objects (PDOs) with high-priority identifiers
- Exchanging service data via Service Data Objects (SDOs) with low-priority identifiers
- Network management (NMT) for implementing network management functions like initializing, starting and resetting network nodes, for example.
- Synchronization (SYNC) for synchronizing real-time data exchange with PDOs
- Error handling (EMERGENCY): for detecting errors of a network node
- Network monitoring (NODE GUARDING): for monitoring network node failures

can easily be realized.

CAN has available nine transfer rates, i.e. 1 Mbps, 800 kbps, 500 kbps, 250 kbps, 125 kbps, 100 kbps, 50 kbps, 20 kbps and 10 kbps. The CANopen-Master option module can communicate with up to 32 CANopen network node (e.g. I/O modules and drives).

There are various methods available to you for achieving a data exchange with the CANopen master module via CANopen.

- ProMaster, ProCANopen, ProProg wt III and Motion Control

  Simple, fast configuration of data exchange in ProMaster and ProCANopen. Simple, fast programming of the machine functions in ProProg wt III with the Motion Control function blocks.

  The evaluation of the network states and the network node states can also be carried out via function blocks from the library CANopen_PLC01_30bd00 (or higher).

4.1 General information on CANopen and using the CANopen-Master option module

For data exchange via CANopen with the CANopen master option module and ProMaster, ProCANopen, ProProg wt III and Motion Control, the CANopen master option module requires a software version of ≥ **01.20** (i.e. ≥ BM4-O-ETH-02/CAN-04-00-00-007-**005**).

Example projects:

ProMaster Project	Example_2_3_2.bmxml
IEC Project in ProProg wt III	Example_BMC_M_CAN04_MA_1.mwt/.zwt

See ▹4.3 ProMaster, ProCANopen, ProProg wt III and Motion Control ◃ from page 57 onward.

- PROPROG wt II

  Programming of the data exchange and the machine functions in PROPROG wt II by using the IEC 61131-3 programming system PROPROG wt II with the library CANop405_PLC01_20bd02 (or higher)

  Example project:

IEC Project in PROPROG wt II	CANopenMaster_Example.mwt/.zwt

  See ▹4.4 Programming data exchange with PROPROG wt II and CANop405_PLC01_20bd03 library ◃ from page 96 onward.

- PROPROG wt II, Motion Configurator and Motion Control

  Simple, fast configuration of the data exchange in the Motion Configurator.

  Simple, fast programming of the machine functions in PROPROG wt II with the Motion Control function blocks.

  See ▹4.5 CANopen and Motion Control with PROPROG wt II and motion configurator ◃ from page 148 onward.

4.2 Basics of CAN and CANopen networks

4.2.1 Basics of CAN

A CAN-based field bus system has a linear structure. The physical basis of data transfer is a three-wire cable with CAN_High, CAN_Low and CAN_Ground connections. CAN uses balanced-to-ground transfer to suppress common-mode interference. This means that the system evaluates differential signals.

Network

CAN is a Multi-Master network. Each node can actively access the bus with the same authorization. CAN uses object-oriented addressing, i.e. an identifier that is specified across the entire network identifies the transferred message. It represents the coded name of the message.

Bus access

Bus access is on the basis of the CSMA/CA procedure (Carrier Sense Multiple Access/ Collision Avoidance). Since each node is authorized to send a message when it detects that the bus is idle, collisions can occur. This is avoided due to bitwise arbitration of the messages to be sent. In this connection, a differentiation is made between two bus levels: a dominant one with a logical bit value of 0, and a recessive one with a logical bit value of 1. In a worst case situation, all the nodes that want to transmit start sending their messages on the bus at the same time. If one node's recessive bit is overwritten by another node's dominant bit, the "recessive" node withdraws from the bus and tries to send its message again when it detects that the bus is idle. This guarantees that the most important, highest priority message (with the lowest identifier) is transferred without a collision and without delay. For this reason, it is of course necessary that each identifier may be assigned only once on the CAN bus.

Identifier

According to the CAN 2.0A 2032 specification, different identifiers are available. Each node can transmit without requesting (Multi-Master capability). A transmitter sends its message to all the CAN nodes (broadcast) that decide for themselves on the basis of the identifier whether to further-process the message or not.

Errors

Up to eight bytes of user data can be transferred in one CAN data message frame. For error or overload signaling, a CAN node can send error or overload message frames. It does this on layer 2 of the OSI/ISO reference model, the Data Link Layer, i.e. independently of the application. Due to high-quality error detection and handling on layer 2, a Hamming distance (measure of error detection) of 6 can be achieved, i.e. the system detects a maximum of five bit errors that occur at the same time in one message frame.

4.2.2 Basics of CANopen

CANopen is an open, non-proprietary field bus system that is based on Layer 1 and 2 definitions of the CAN standard

4.2.2.1 CAL specification and profiles

In the international CAN organization CAN in Automation e.V. (CiA) the CAL application originated that represents a general description language for CAN networks. CAL makes available a collection of communication services without precisely defining their applications. CANopen uses this description language. On the one hand, a subset of the communication services that CAL offers is used to define an open communications interface. These definitions are specified in communications profiles: in this connection, profile DS 301 is particularly important, since it regulates how communication is carried out. On the other hand, CANopen additionally specifies what the data in the respective device classes mean. These definitions are recorded in device profiles, e.g. profile DS 401 for I/Os and profile DSP 402 for drives. Device profile DSP 405 defines elements on programmable interfaces according to IEC 61131-3.

The CANopen master module for b maXX drive PLC supports the communication profile DS 301, DSP 302 (from FW 1.20.mit ProMaster) and the device profile DS 405. The communication profile DSP 302 is supported in conjunction with ProMaster from software version ≥ **01.20** (i.e. ≥ BM4-O-ETH-02/CAN-04-00-00-007-**005**).

4.2.2.2 Communication- and device-specific objects

When accessing across the network the system always accesses the objects of a network node. Objects take on different tasks. Using communication profile-specific objects, you configure the exchange of data across the CANopen field bus. As device profile-specific objects, they are directly connected to the device. Using them, it is possible to use and change the device functions. Objects are always addressed using a 16-bit index and an 8-bit sub-index. Depending on the significance, different areas are defined for objects. The following are the most important user areas:

Index (hex)	Object
1000 - 1FFF	Area for the communication profile. Contains all the objects that are needed for communication, e.g. PDO, SDO.
2000 - 5FFF	Area for manufacturer-specific objects. Contains all the objects that are not defined in a profile and which are manufacturer-dependent.
6000 - 9FFF	Area for the device profile. Contains the objects of the device profile that is supported in each case (DS 401, DSP402, etc.)

The individual profiles precisely define which objects must be present and which ones are optional for a network node.

4.2.2.3 Data exchange and objects of the physical bus system

From the point of view of communications relationships between the CANopen nodes, data exchange is described by

- The Client-server model: Each node is authorized to request data from another node or to transfer it to the other node

and

- The Producer-consumer model: A node sends data without requesting it other nodes monitor it and can evaluate the data

There is no master-slave relationship (i.e. exactly one CANopen node organizes the exchange of data) at the level of communications relationships. This means that a CANopen-Master is considered to be a master at the level of the application rather than a master at the level of communication. This means that CANopen-Master functions are mainly network management tasks such as

- Starting and stopping network nodes
- Monitoring network nodes
- Configuring data exchange

In doing this, the CANopen-Master uses the functions of the communications level. Since functions of this type are closely linked to the actual application, it is sensible for a CANopen-Master to be programmable under IEC 61131-3. In the case of the CANopen-Master option module, this is possible together with library

- CANopen_PLC01_30bd00 (or higher) under ProProg wt III or
- CANop405_PLC01_20bd03 (or higher) under PROPROG wt II.

Data is exchanged on CANopen networks in the form of message frames that transfer the user data or that access the objects of a network node. Each message frame is differentiated by an identifier. The message frame types and the underlying mechanisms for data exchange are divided into groups:

- Process data objects (PDOs): Real-time data exchange with high-priority identifiers and up to 8 bytes per message.
  - ProMaster, ProCANopen, ProProg wt III (possibly the library CANopen_PLC01_30bd00 (or higher)):
    With the CANopen master option module, a maximum of 256 PDOs can be written and 256 PDOs can be read simultaneously.
  - PROPROG wt II, the library CANop405_PLC01_20bd03 (or higher):
    With the CANopen master option module, a maximum of 40 PDOs can be written and 63 PDOs can be read simultaneously.
- Service data objects (SDOs): Parameter data exchange with low-priority identifiers and data addressable by index/subindex.
  - ProMaster, ProCANopen, ProProg wt III (possibly the library CANopen_PLC01_30bd00 (or higher)):
    The CANopen master option module supports the transfer types "expedited" (4 bytes per message) and "segmented" (up to 7 bytes per message). An SDO can be written or read.
  - PROPROG wt II, the library CANop405_PLC01_20bd03 (or higher):
    The transfer type for the CANopen master option module is "expedited", i.e. up to 4 bytes can be transferred per message. Depending on the network configuration, up to eight different SDOs can be read or written simultaneously.

- Network management (NMT): A special type of object for implementing network communication functions like initializing, starting and resetting network nodes, for example. The CANopen-Master option module sends the commands as broadcasts to the respective network nodes.
- Synchronization (SYNC): A special type of object for synchronizing the exchange of data in real time with PDOs. The CANopen-Master option module sends the SYNC commands as broadcasts to the respective network nodes.
- Error handling (EMERGENCY): A special object type for detecting errors of a network node. If an error occurs in a network node, which sends an emergency message frame, this CANopen-Master option module can receive it and evaluate it.
- Network monitoring

  NODE GUARDING:

  - Special object type for the failure monitoring of network nodes. The CANopen master option module can cyclically request the feedback from network nodes via a telegram to detect the failure of network nodes.

  HEARTBEAT:

  - ProMaster, ProCANopen, ProProg wt III (possibly the library CANopen_PLC01_30bd00 (or higher)):
    Special object type for the failure monitoring of network nodes. The CANopen master option module detects the failure of network nodes from the fact that these network nodes no longer transmit heartbeat telegrams. The advantage of the heartbeat compared to node guarding is reduced telegram traffic on the CANopen, and therefore a lower bus load.
  - PROPROG wt II, the library CANop405_PLC01_20bd03 (or higher):
    Heartbeat is not supported.

NOTE

The CiA profile definitions use the term "object" in two different ways. On the one hand as described in the chapter entitled ▹Communication- and device-specific objects ◃ on page 52, as a type of datum that can be accessed from outside, or that describes device properties. On the other hand, it also designates the different types of message frames and the underlying mechanisms for exchanging data. In the second case, you can consider these objects as being objects of the physical CANopen bus system but not as objects that can be accessed via indexes and subindexes. These objects of the physical CANopen bus system transport the communications- and device-specific objects.

By contrast with other field bus systems, these specifications result in the message frames themselves being able to be identified by the individual nodes. Each network node decides for itself when it wants to send data and which message frames it evaluates. There is, however, also the option of using remote message frames to request other network nodes to send data.

4.2.2.4 Predefined identifiers

To be able to establish peer-to-peer communication between the master and the other network nodes directly after booting, there is a predefined identifier assignment for the message frames. Users can reconfigure this identifier assignment. Identifiers that comply with the CAN definition are divided into the function code and the module ID:

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						

The module ID is the same as the node number on the network. The seven bits for the module ID yield a theoretical maximum number of nodes per CANopen network of 127. For physical reasons, you can only operate a maximum of 32 nodes on any one network using the CANopen-Master option module. The total of 11 bits yields the COB-ID (Communication Object Identifier). The following objects are predefined:

Broadcast objects (module ID = 0):

Object name	**Function code, binary**	**Resulting COB-ID, decimal**	**Resulting COB-ID, hexadecimal**
NMT	0000	0	16#0
SYNC	0001	128	16#80
TIME STAMP [1)]	0010	256	16#100

[1)] The CANopen-Master option module does not support the timestamp object.

Peer-to-peer objects:

Object name	**Function code, binary**	**Resulting COB-ID, decimal**	**Resulting COB-ID, hexadecimal**
EMERGENCY	0001	129 - 255	81h - FFh
PDO1 (TX)	0011	385 - 511	181h - 1FFh
PDO1 (RX)	0100	513 - 639	201h - 27Fh
PDO2 (TX)	0101	641 - 767	281h - 2FFh
PDO2 (RX)	0110	769 - 895	301h - 37Fh
PDO3 (TX)	0111	897 - 1023	381h - 3FFh
PDO3 (RX)	1000	1025 - 1151	401h - 47Fh
PDO4 (TX)	1001	1153 - 1279	481h - 4FFh
PDO4 (RX)	1010	1281 - 1407	501h - 57Fh
SDO (TX)	1011	1409 - 1535	581h - 5FFh
SDO (RX)	1100	1537 - 1663	601h - 67Fh
NODE GUARDING	1110	1793 - 1919	701h - 77Fh

TX stands for transmit and RX stands for receive. Note that the direction of transmission is from the point of view of the network node and not of the master. This means, for example, if the option module sends a PDO this is an RxPDO for the evaluating network node; this means that it must be transferred on the CANopen bus with an RX identifier.

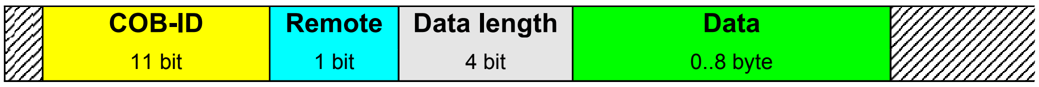
This also applies to the module ID: this must be seen from the network node perspective and not the master's point of view, i.e. the COB-ID contains the module ID of the network node and not of the master.

However, this only applies to predefined identifier assignment. According to communication profile DS 301, users can freely specify COB-IDs for the physical communications objects for each network node using device objects. This makes it possible, for example, to set up process data communication directly between two (or more) network nodes.

4.2.2.5 Message frame structure and priority

In CAN message frames, the most important CANopen elements are the COB-ID, the remote bit, the data length and the data. Since the COB-ID is the same as the CAN identifier, this means that the CSMA/CA procedure (see the chapter entitled ▹4.2.1 Basics of CAN ◃ from page 51 onward) dictates that less significant COB-IDs have higher priority. This means that the predefined identifier assignment results in NMT message frames having the highest priority followed by SYNC message frames. Since the COB-ID also contains the network node number, this also results in message frames of network nodes with lower node numbers having higher priority.

The most important elements in a CANopen message frame are as follows:



If the remote bit is set, another network node requests data. Up to 8 bytes are available for data. The number of bytes depends on the type of message frame. In the case of SDO, this is a maximum of four bytes; the other four bytes are needed to identify the data type. PDOs can use all eight bytes. SYNC message frames contain no data.

4.3 ProMaster, ProCANopen, ProProg wt III and Motion Control

In this chapter we explain how you can easily and quickly put a CANopen network into operation with the b maXX drive PLC and the CANopen master option module. This chapter is accompanied by a ProMaster example project.

A CANopen network consists of a CANopen master and one or several CANopen slaves.

The machine configuration is created from these devices with the Baumüller ProMaster engineering framework. The CANopen communication between the individual devices is configured with the ProCANopen field bus configurator. By using Motion Control, the configuration settings are reduced to a minimum and the user can concentrate completely on the actual application (machine functionality) in the IEC 61131-3 programming system ProProg wt III.

The data and settings for CANopen which result during the configuration of CANopen communication are stored on the CANopen master option module. This concerns both the data for the CANopen master itself and the data for the CANopen slaves. The CANopen slaves do not receive their data from the CANopen master until after the machine is switched on.

The data and settings for ProProg wt III, which are stored during the configuration of CANopen communication, are stored in the IEC project (ProProg wt III project) of the device with the CANopen master option module. This IEC project is loaded on the b maXX drive PLC, to which the CANopen master is connected.

The Ethernet with CANopen master option module (BM4-O-ETH-02) enables Ethernet communication with Single Axis Motion Control. Ethernet communication should not be used with Multi Axis Motion Control, as this reduces the synchronization accuracy in the CANopen network.

NOTE

Do not use Ethernet communication and Multi Axis Motion Control together on an Ethernet with CANopen master option module (BM4-O-ETH-02).

No "not Multi Axis Motion Control" CANopen slaves should be used in a CANopen network in which the Multi Axis Motion Control is used. This especially applies to IO modules. The "not Multi Axis Motion Control" CANopen slaves can interfere with the synchronicity in the CANopen network. Use a second CANopen master module for "not Multi Axis Motion Control" CANopen slaves.

NOTE

Do not use "not Multi Axis Motion Control" CANopen slaves and "Multi Axis Motion Control" CANopen slaves together on one (Ethernet with) CANopen master option module (BM4-O-ETH-02 / BM4-O-CAN-04).

4.3.1 Requirements

The requirements are

- PC with

  - Engineering Framework ProMaster
  - IEC 61131-3 programming system ProProg wt III
    with the firmware libraries
    - Bit_UTIL_30bd00
    - OmegaOS_30bd00
    - SYSTEM2_PLC01_30bd00

    and the libraries
    - BM_TYPES_30bd00
    - SYSTEM1_PLC01_30bd00
    - CANopen_PLC01_30bd00 (for devices that do not support Motion Control)
    - MOTION_TYPES_30bd00
    - MOTION_CONTROL_30bd00
    - MOTION_MULTI_AXIS_30bd00
  - Drive control program for b maXX ProDrive (included in ProMaster) or WinBASS II (for commissioning the CANopen (controller) slaves).
- Physical commissioning of the components of the CANopen network; in our example, these are the CANopen master
  - b maXX 4400 with
    - b maXX drive PLC (BM4-O-PLC-0x-...) with
    - CANopen master option module (BM4-O-CAN-04-... or BM4-O-ETH-02-...)

  and the CANopen slave
  - b maXX 4400 with
    - CANopen slave option module (BM4-O-CAN-03-...)

See the respective operating instructions for more information on this topic.

4.3.2 Steps to be carried out

To use the CANopen master for b maXX drive PLC option module for Motion Control in a CANopen network, the following steps must be carried out.

1 Commissioning of the CANopen network

2 Creation of an IEC 61131-3 project for ProProg wt III with a Motion Control template

3 Creation of a machine configuration in ProMaster

4 Configuration of the CANopen communication with ProCANopen and subsequent downloading to the CANopen master
(testing of the CANopen network with this configuration if necessary)

5 Configuration of the PLC with connection of the IEC project and the ProMaster and configuration of the cam disk data

6 Programming of the IEC 61131-3 projects for ProProg wt III (application) and subsequent downloading to the b maXX drive PLC

7 Running of the application in the CANopen network

Of course, you can also begin with step 2 and wait until step to carry out the physical commissioning (step 1).

In the process, the ProMaster project **Example_2_3_2.bmxml** and the IEC 61131-3 project (ProProg wt III project) **Example_BM4_O_CAN04_MA_2.mwt/.zwt** are created.

4.3.3 Commissioning of the CANopen network

For details on commissioning the CANopen network, please see the operating instructions of the BM4-O-ETH-02 / BM4-O-CAN-04 option module and the operating instructions of the other CANopen network nodes.

After physically commissioning the network and switching in the power supply for the b maXX 4400 unit, the CANopen master option module is ready for operation after approximately 5 seconds. This is shown with two CANopen LEDs at the front side of the option module. Two combinations are possible:

H5 (green): 200 ms On / Off H6 (red): Off	CANopen: The option module waits for the initialization by an application program on the b maXX drive PLC
H5 (green): Off H6 (red): On	CANopen: The CAN is in the Bus-Off state

All other states of the LEDs H5 and H6 indicate errors. For more information on this topic, please see the related operating instructions for the CANopen master option module BM4-O-ETH-02 / BM4-O-CAN-04.

4.3.4 Creating an IEC project with ProProg wt III

In the following we will explain how a Motion Control IEC project is created.

Multi Axis Motion Control is set in the ProMaster default settings for Motion Control. Therefore, it is practical to first create an IEC project for Multi Axis Motion Control.

Open ProProg wt III and create a new project by selecting an IEC template with the "File\New Project" menu.

For the example we use the IEC template project „Tmpl_PLC01_MA_2_0103.zwt“ (for BM4_O_PLC01). You will find this project in the ProMaster installation directory \\Baumueller\ProMaster.NET in the subdirectory \\...\projects\IEC-Templates. Unzip the IEC template „Tmpl_PLC01_MA_2_0103.zwt“ as „Example_BM4_O_CAN04_MA_2.mwt“.

The template contains the libraries

- Bit_UTIL_30bd00
- BM_TYPES_30bd00
- MC_SYS_30bd00
- MOTION_TYPES_30bd00
- MOTION_CONTROL_30bd00
- MOTION_MULTI_AXIS_30bd00

If you do not want to use all axes as Motion Control axes, or if you want to integrate other CANopen devices in your machine configuration, you must now also integrate the libraries

- BM_TYPES_30bd00
- CANopen_PLC01_30bd00

for the evaluation of the network states and the network node states in your project.

Save the project. In our example we use the project name **"Example_BM4_O_CAN04_MA_2.mwt"**.

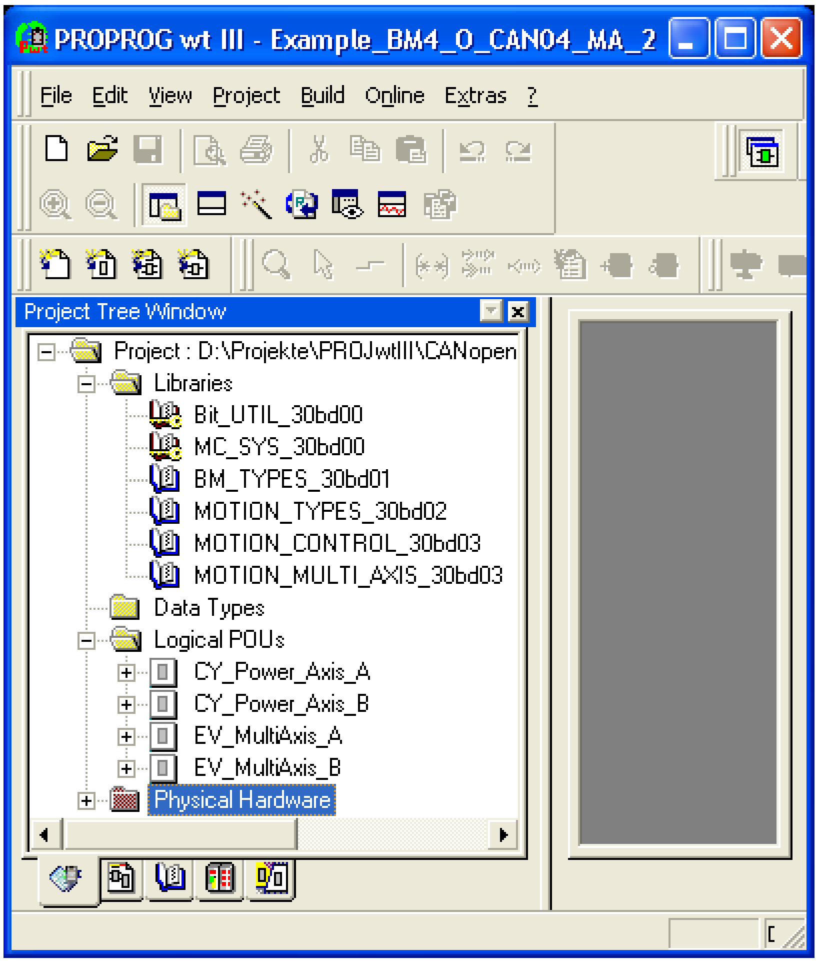


Figure 27: ProProg wt III - Creating the project "Example_BM4_O_CAN04_MA_2.mwt"

Now the communication settings must also be configured. The serial port COM1 is the default setting. If you want to use the Ethernet, see the chapter ▹3 Ethernet ◃ from page 17 onward.

For the use of an IEC project in ProMaster, it is absolutely necessary that the checkbox "Generate boot project during compiling" be activated for ProProg wt III in the "Resource - Settings" window (Mark resource and select "Settings" context menu). This generates the file "bootfile.pro", which is required for downloading onto the b maXX drive PLC.

NOTE

In ProProg wt III the checkbox "Generate boot project during compiling" must be activated under "Resource - Settings".

The "Example_BM4_O_CAN04_MA_2.mwt" project is connected later to the ProMaster Project in the chapter ▹4.3.7 Configuring PLC ◃ from page 82 onward.

Now close ProProg wt III via the "File\Exit" menu.

4.3.5 Creation of a machine configuration in ProMaster

In this section we explain how a project is created in ProMaster and how a machine configuration is created.

Open ProMaster:

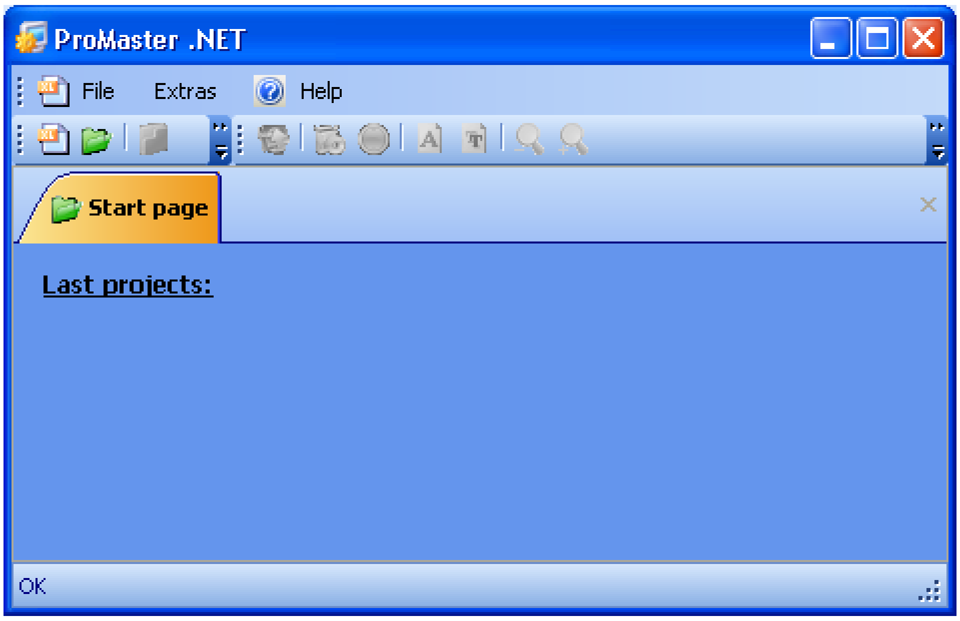
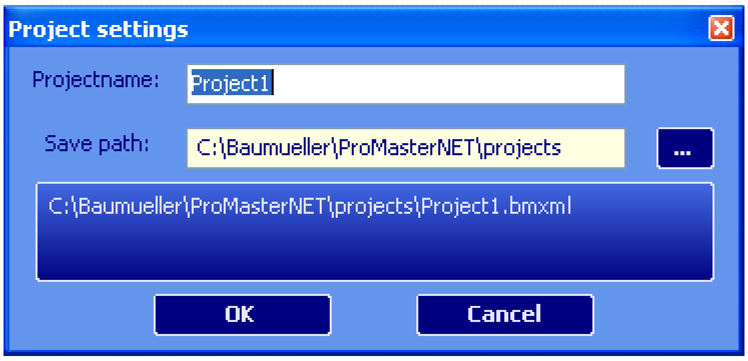


Figure 28: ProMaster without projects

Create a new ProMaster project by opening the "Project Settings" window with the "File\New Project" menu.


Project settings

Projectname: Project1

Save path: C:\Baumueller\ProMasterNET\projects

C:\Baumueller\ProMasterNET\projects\Project1.bmxml

Figure 29: ProMaster - Project name

Specify the name (in the edit box) and the storage location (via the path selection) of the project. We'll use the name "Example_2_3_2.bmxml" for our project.

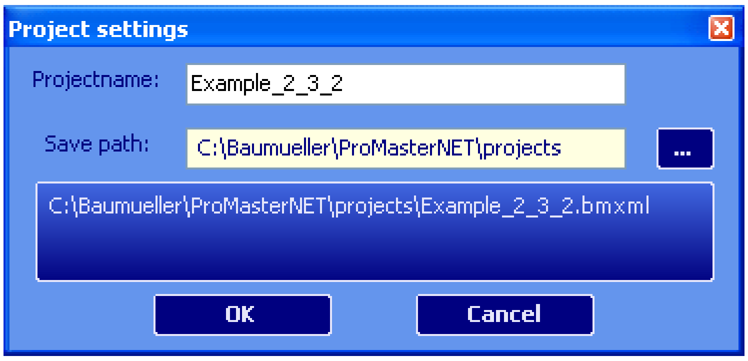

Project settings
Projectname:
Example_2_3_2
Save path:
C:\Baumueller\ProMasterNET\projects
...
C:\Baumueller\ProMasterNET\projects\Example_2_3_2.bmxml
OK
Cancel

Figure 30: ProMaster - Assign project name

Apply the settings from the "Project Settings" window by clicking the "OK" button. ProMaster now changes to the network view.

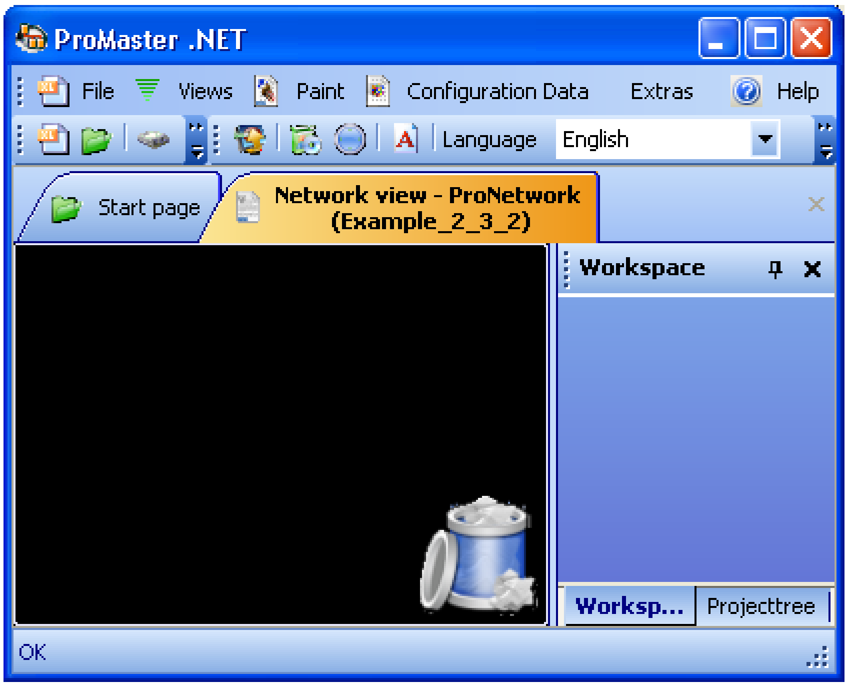


Figure 31: ProMaster - Assign project name

Now open the Baumüller catalog via the "Views\Catalog" menu. The Baumüller catalog contains the devices, bus system and components Baumüller provides you with in the default configuration.

For our example we use

- the devices
  - "bmaXX 4000 PLC CANopen master (2-row)“ (from the group "b maXX 4000 Drive“) b maXX 4000 with two slots,
    Ethernet with CANopen master option module in slot G,
    b maXX drive PLC in slot H

- "b maXX 4000 CANopen slave (2-row)" (from the group "b maXX 4000 Drive");
  b maXX 4000 with two slots,
  Ethernet with CANopen slave option module in slot G
- the bus system
  - "bus_canopen" (CANopen), is automatically created.

Drag the device "bmaXX 4000 PLC CANopen master (2-row)" out of the Baumüller catalog and into the ProMaster network view with the Drag&Drop function. The bus system has been connected automatically. Now drag the device "bmaXX 4000 CANopen slave (2-row)" onto the CANopen bus with the Drag&Drop function (dropping is not possible until the bus changes color).

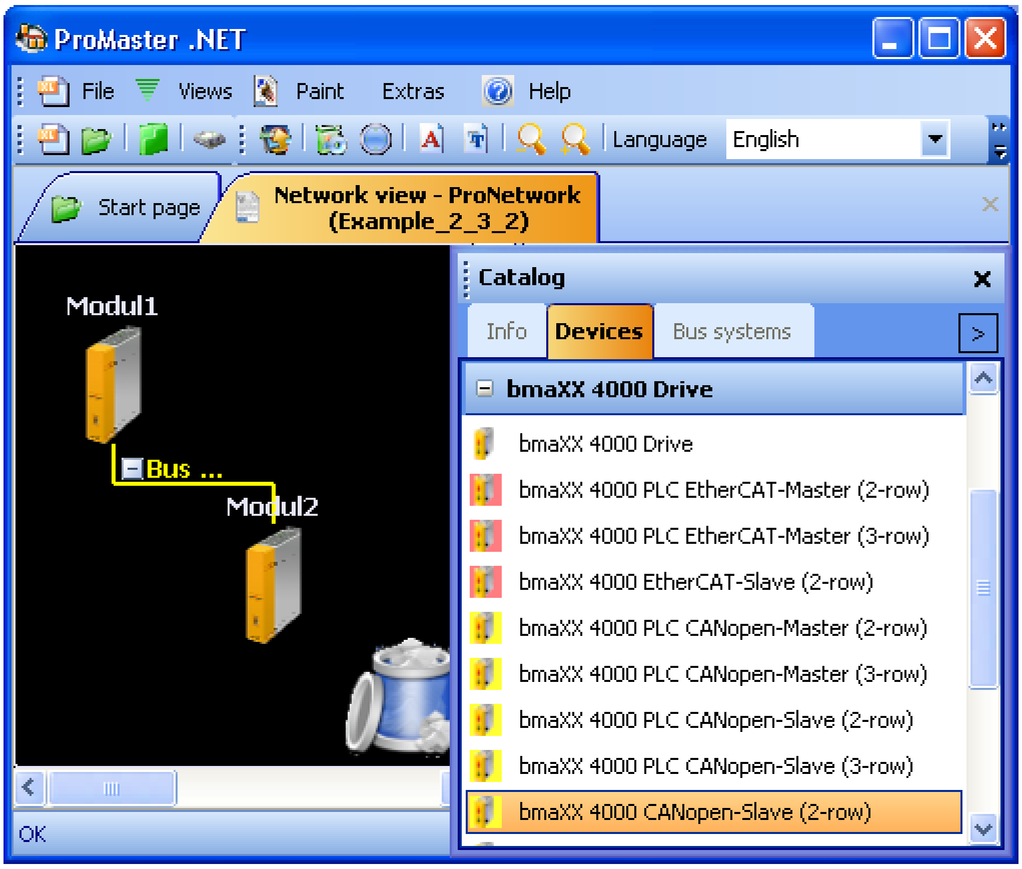
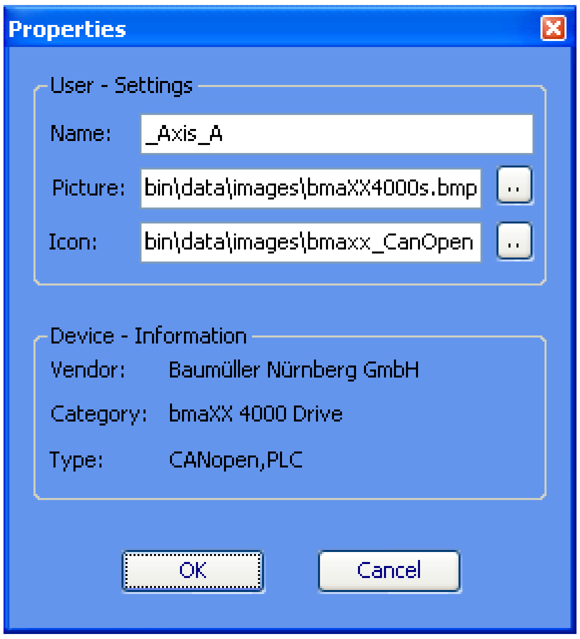

ProMaster .NET
File
Views
Paint
Extras
Help
Language
English
Start page
Network view - ProNetwork (Example_2_3_2)
Modul1
Bus ...
Modul2
OK
Catalog
Info
Devices
Bus systems
bmaXX 4000 Drive
bmaXX 4000 Drive
bmaXX 4000 PLC EtherCAT-Master (2-row)
bmaXX 4000 PLC EtherCAT-Master (3-row)
bmaXX 4000 EtherCAT-Slave (2-row)
bmaXX 4000 PLC CANopen-Master (2-row)
bmaXX 4000 PLC CANopen-Master (3-row)
bmaXX 4000 PLC CANopen-Slave (2-row)
bmaXX 4000 PLC CANopen-Slave (3-row)
bmaXX 4000 CANopen-Slave (2-row)

Figure 32: ProMaster - Machine configuration created

In the picture you now see the CANopen master (Modul1) and the CANopen slave (Modul2). Additional information on the modules is available via the respective tooltip or the "Project Tree Window". The "Catalog" window is closed for improved clarity.

The names (Modul1 and Modul2) are changed by clicking on the respective module and then opening the properties window of the respective module via the context menu and "Properties".


Properties

User - Settings

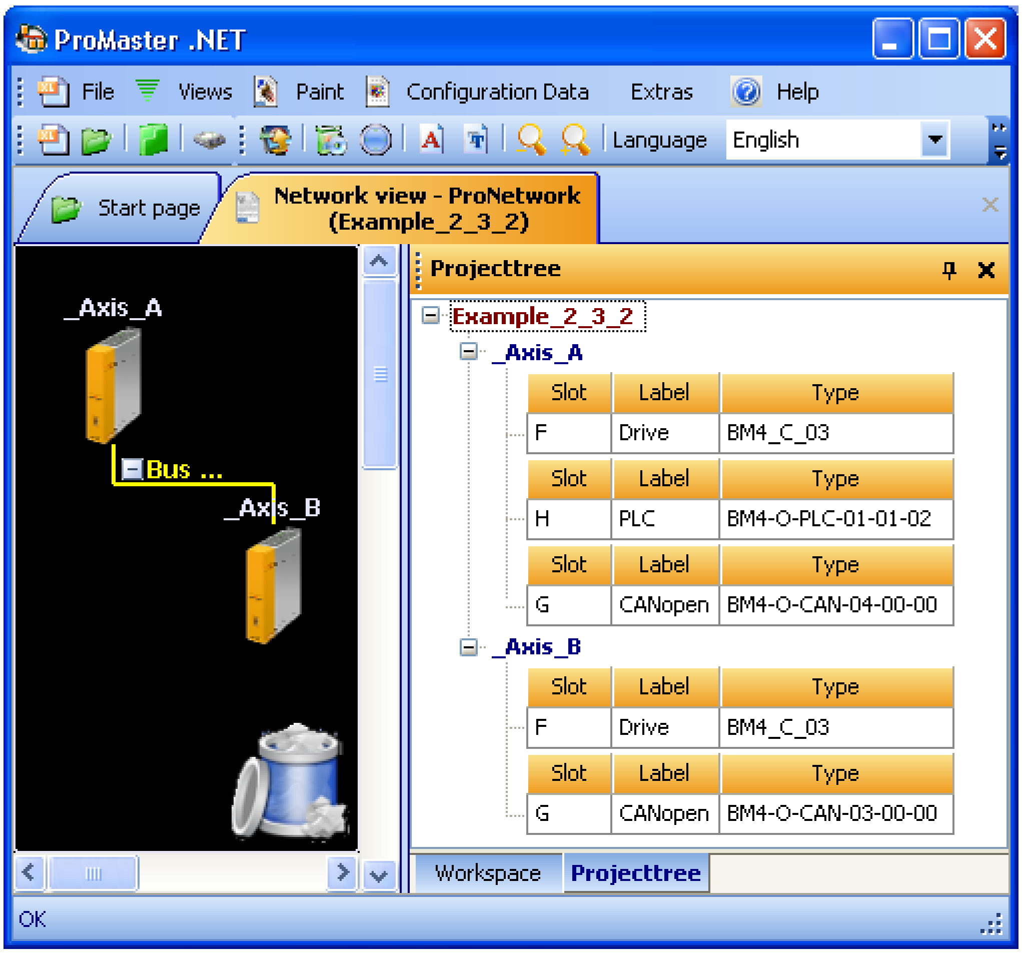
Name:	_Axis_A
Picture:	bin\data\images\bmaXX4000s.bmp
Icon:	bin\data\images\bmaxx_CanOpen

Device - Information

Vendor:	Baumüller Nürnberg GmbH
Category:	bmaXX 4000 Drive
Type:	CANopen,PLC

Figure 33: ProMaster - Module properties

For our example we assign the CANopen master the name _Axis_A, and for the CANopen slave the name Namen _Axis_B (_Axis_A and _Axis_B are the default axis variable names of the Motion Control axes in our IEC project). We keep the respective images (*.bmp). The new names now appear in the ProMaster network view and in the project tree.



Example_2_3_2

_Axis_A

Slot	Label	Type
F	Drive	BM4_C_03
H	PLC	BM4-O-PLC-01-01-02
G	CANopen	BM4-O-CAN-04-00-00

_Axis_B

Slot	Label	Type
F	Drive	BM4_C_03
G	CANopen	BM4-O-CAN-03-00-00

Figure 34: ProMaster - Machine configuration - Modules renamed

You can set the communication settings for the communication of ProMaster to the b maXX drive PLC by clicking on the device "_Axis_A" and selecting "Communication Settings" via the context menu. Then save the project with the "File\Save Project" menu.

In ProMaster, Motion Control is activated for the components with CANopen (CANopen master drive, CANopen slave drive) in the default configuration. This reduces the settings to be configured to a minimum.

If you want to use the default settings (e.g. CANopen baud rate 1 MBit/s and the automatically assigned CANopen node-ID), you do not need to carry out any configuration of the CANopen communication with ProCANopen.

In this case, you continue with ▷4.3.7 Configuring PLC ◁ from page 82 onward.

4.3.6 Configuring CANopen communication with ProCANopen

In ProMaster, Motion Control is activated for the components with CANopen (CANopen master drive, CANopen slave drive) in the default configuration. This reduces the settings to be configured to a minimum.

If you want to use the default settings (e.g. CANopen baud rate 1 MBit/s and the automatically assigned CANopen node-ID), you do not need to carry out any configuration of the CANopen communication with ProCANopen.

In this case, you continue with ▷4.3.7 Configuring PLC ◁ from page 82 onward.

We explain how you can change the settings of the CANopen communication in the following section.

First, the CANopen slaves are configured, then the CANopen master is configured.

4.3.6.1 Configuring CANopen slave communication

The CANopen slave communication is configured for each CANopen slave individually.

Open ProCANopen for our CANopen slave by clicking on the device "_Axis_B" and then select "Configuration Data (Components)\CANopen Slave (Slot G)\Configure Slave Bus (ProCANopen)" via the context menu.

The CANopen slave was already set with the default setting when creating the machine configuration so that the CANopen settings for Motion Control are configured (Drag&Drop of the slave on a master for which the Motion Control is activated).

The window for the CANopen slave communication configuration is opened.

Ident tab

For Motion Control, you can set the slot of the CANopen slave option module in the b maXX 4400, the name of this module (device name) in ProMaster and the baud rate and the node-ID for CANopen in the "Ident" tab. The baud rate and node-ID were applied from the CANopen configuration of the CANopen master module.

Changing the device name, baud rate and node-ID affects the ProMaster project.

NOTE

Reducing the baud rate with an unchanged "Cycle time of SYNC" (Network tab) can result in an excessively high bus load on the CANopen and negatively affect the synchronicity.

Changing the device name also has a negative effect on the Motion Control axis name in the IEC 61131-3 project in ProProg wt III.

NOTE

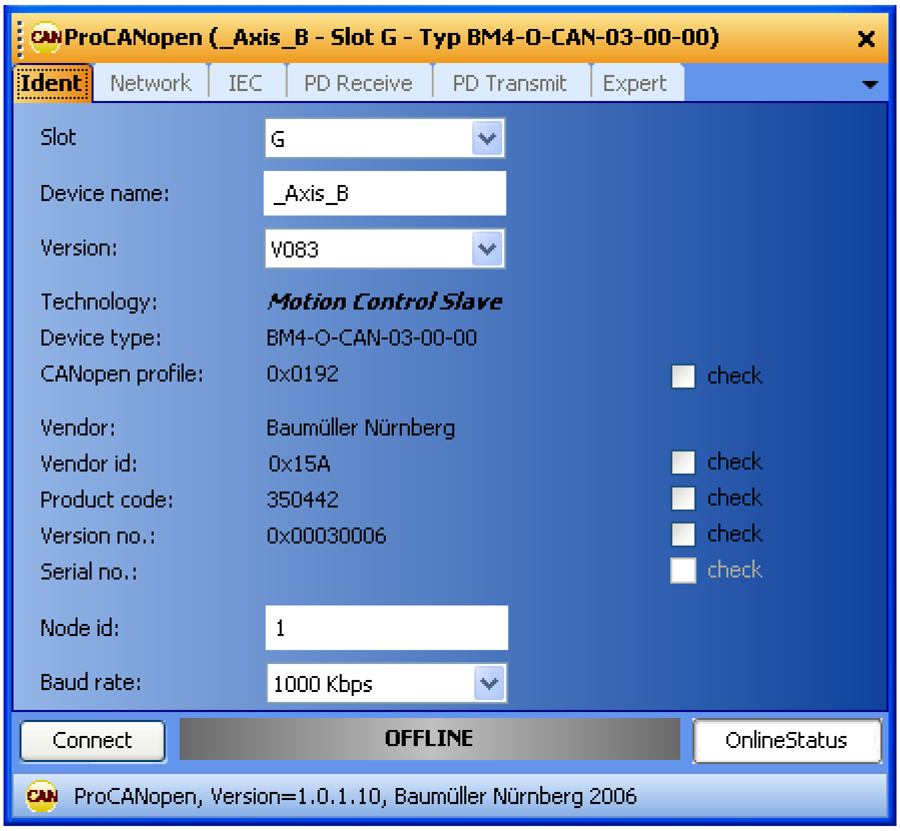
Changes to the device name also have an effect on the Motion Control axis name in the global variable worksheet in the IEC 61131-3 project in ProProg wt III.

This change is **not** automatically carried out in the local variable worksheets and in the code worksheets in the IEC 61131-3 project in ProProg wt III.

This remains the explicit task of the user (for safety reasons).

In addition, CANopen-specific information like the device type, CANopen profile, manufacturer/vendor-ID etc. are also displayed.

With series production of machines, it can be checked with the "Check" checkboxes during start-up, whether the correct devices are connected.


ProCANopen (_Axis_B - Slot G - Typ BM4-O-CAN-03-00-00)

Ident | Network | IEC | PD Receive | PD Transmit | Expert

Slot	G	
Device name:	_Axis_B	
Version:	V083	
Technology:	*Motion Control Slave*	
Device type:	BM4-O-CAN-03-00-00	
CANopen profile:	0x0192	☐ check
Vendor:	Baumüller Nürnberg	
Vendor id:	0x15A	☐ check
Product code:	350442	☐ check
Version no.:	0x00030006	☐ check
Serial no.:		☐ check
Node id:	1	
Baud rate:	1000 Kbps	

Connect | OFFLINE | OnlineStatus

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 35: ProCANopen - CANopen slave communication configuration Ident tab

Network tab

You can set the SYNC cycle time and guarding or heartbeat in the "Network" tab.

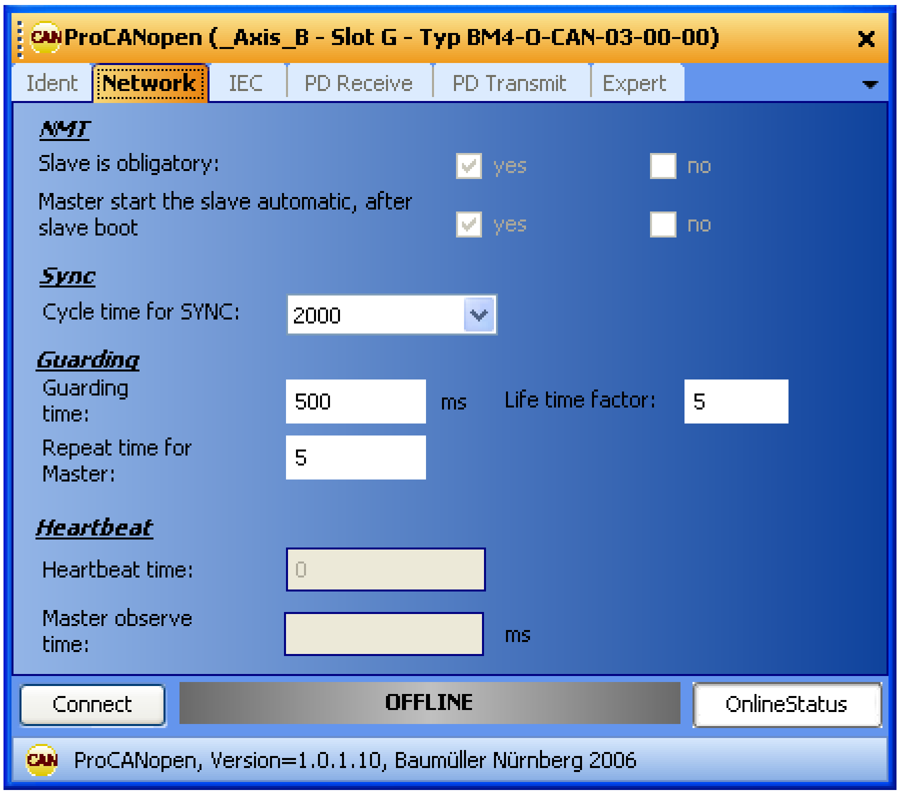
Changing the SYNC cycle time and the guarding or heartbeat settings affects the ProMaster project. The SYNC cycle time is the time interval in which the CANopen master transmits the SYNC telegram on the CANopen bus.

NOTE

A CANopen slave can only support guarding or heartbeat.

NOTE

Reducing the "Cycle time of SYNC" with an unchanged "Baud rate" (Ident tab) can result in an excessively high bus load on the CANopen, and therefore negatively affect communication and synchronicity.



NMT

Slave is obligatory: [✓ yes] no

Master start the slave automatic, after slave boot [✓ yes] no

Sync

Cycle time for SYNC: 2000

Guarding

Guarding time: 500 ms Life time factor: 5

Repeat time for Master: 5

Heartbeat

Heartbeat time: 0

Master observe time: ms

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 36: ProCANopen - CANopen slave communication configuration Network tab

IEC tab

During the configuration of the CANopen slave communication, the CANopen slave objects (via CANopen master objects) are assigned to the network variables for the IEC project on the PLC.

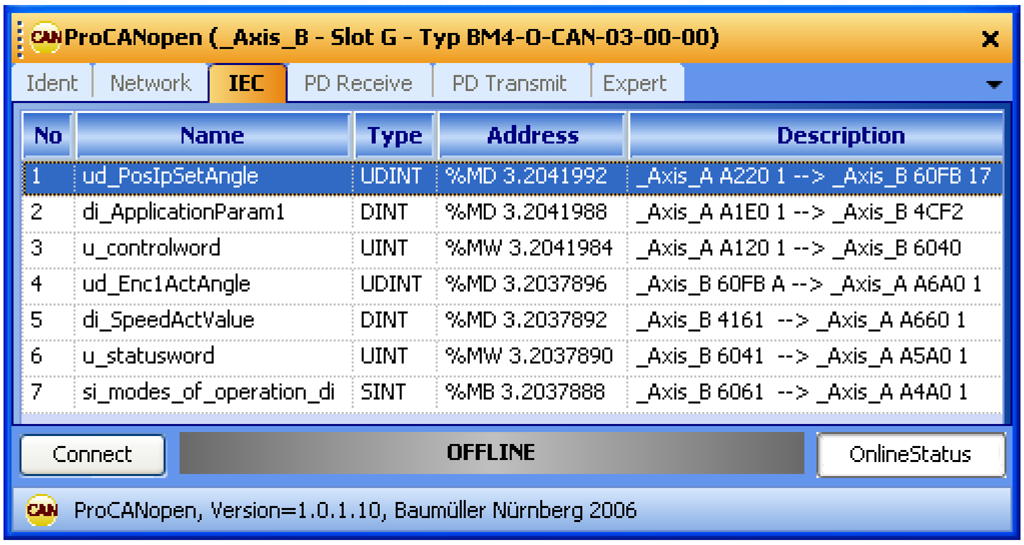
The variables shown in the IEC tab are also called network variables. The process data are written and read in the IEC project via the network variables.

When using Motion Control, some of the assignments are created with the default settings.

The network variables and their data type in the IEC project on the PLC are shown in the "IEC" register.

The calculation of the (IEC) address of the network variables is not carried out until after "Update list" in the chapter ▷4.3.6.2 Configuring CANopen master communication ◁ from page 74 onward.

Other network variables are configured in the PD Receive tab and the PD Transmit tab.


ProCANopen (_Axis_B - Slot G - Typ BM4-O-CAN-03-00-00)
Ident
Network
IEC
PD Receive
PD Transmit
Expert
Connect
OFFLINE
OnlineStatus
ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

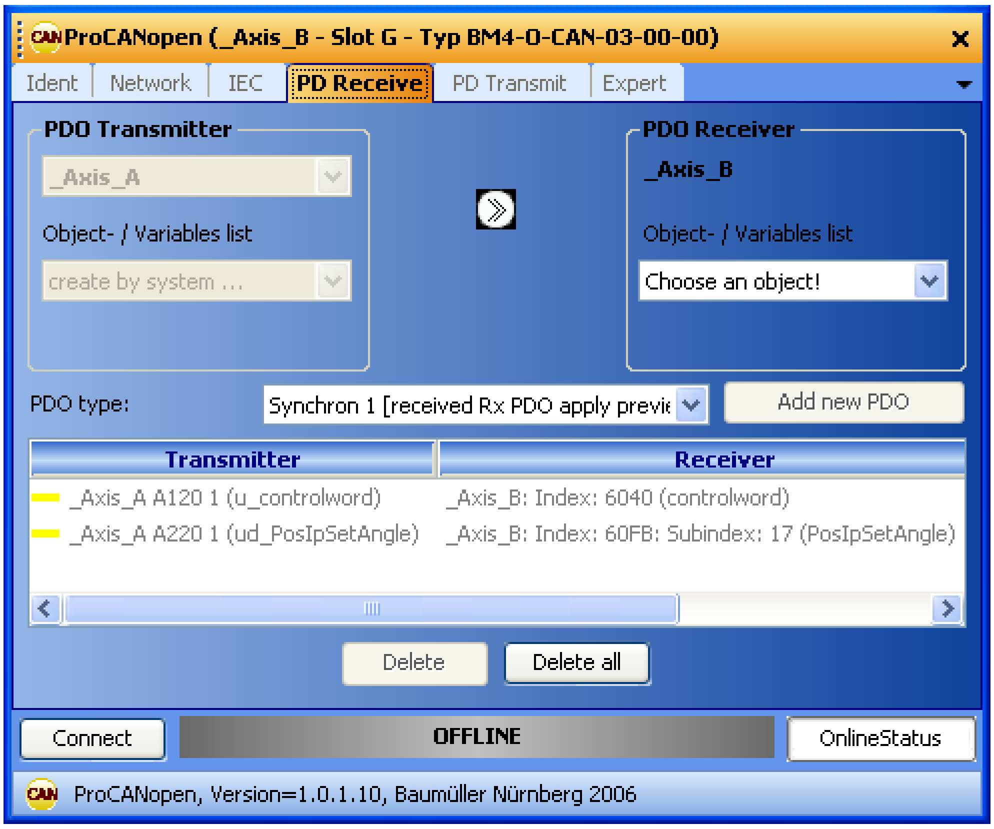
No	Name	Type	Address	Description
1	ud_PosIpSetAngle	UDINT	%MD 3.2041992	_Axis_A A220 1 --> _Axis_B 60FB 17
2	di_ApplicationParam1	DINT	%MD 3.2041988	_Axis_A A1E0 1 --> _Axis_B 4CF2
3	u_controlword	UINT	%MW 3.2041984	_Axis_A A120 1 --> _Axis_B 6040
4	ud_Enc1ActAngle	UDINT	%MD 3.2037896	_Axis_B 60FB A --> _Axis_A A6A0 1
5	di_SpeedActValue	DINT	%MD 3.2037892	_Axis_B 4161 --> _Axis_A A660 1
6	u_statusword	UINT	%MW 3.2037890	_Axis_B 6041 --> _Axis_A A5A0 1
7	si_modes_of_operation_di	SINT	%MB 3.2037888	_Axis_B 6061 --> _Axis_A A4A0 1

Figure 37: ProCANopen - CANopen slave communication configuration IEC tab following configuration of the CANopen slaves and "Update list" (CANopen master communication configuration Download tab)

PD Receive tab

The communication from the master to the slave is configured in the "PD Receive" tab.

The default configuration is already set for Motion Control. You can add to the configuration. When doing so, note that the Motion Control configuration has a higher priority than the configuration by the user.


ProCANopen (_Axis_B - Slot G - Typ BM4-O-CAN-03-00-00)

Ident | Network | IEC | PD Receive | PD Transmit | Expert

PDO Transmitter

_Axis_A

Object- / Variables list: create by system ...

PDO Receiver

_Axis_B

Object- / Variables list: Choose an object!

PDO type: Synchron 1 [received Rx PDO apply previ

Add new PDO

Transmitter	Receiver
_Axis_A A120 1 (u_controlword)	_Axis_B: Index: 6040 (controlword)
_Axis_A A220 1 (ud_PosIpSetAngle)	_Axis_B: Index: 60FB: Subindex: 17 (PosIpSetAngle)

Delete | Delete all

Connect | OFFLINE | OnlineStatus

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 38: ProCANopen - Motion Control CANopen slave communication configuration "PD Receive" tab with default Motion Control configuration

The PDO transmitter is the CANopen master with Motion Control.

The linking of an object in the CANopen slave to an object in the CANopen master takes place here. This object in the CANopen master is written from within the IEC program via a network variable.

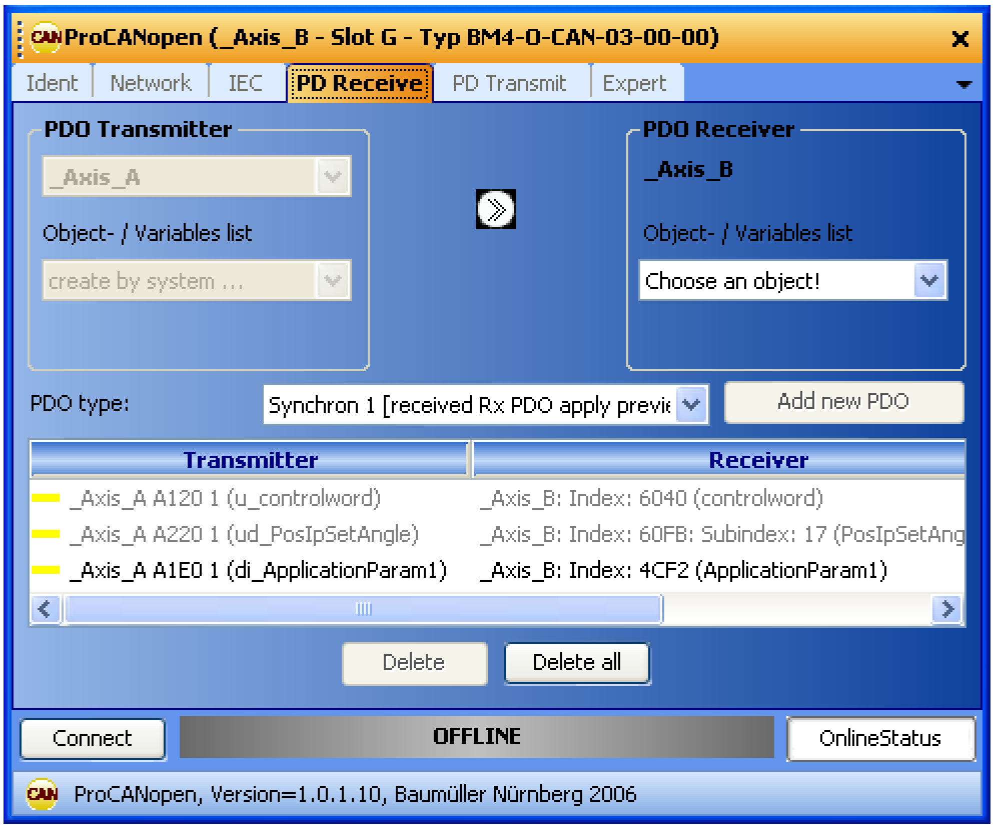
Procedure:

- Selection of an object of the CANopen slave in the "PDO Receiver" area in the "Object/ Variables list" combo box.

  Example: We select CANopen slave object 0x4CF2 (Application parameter 1) in the "PDO Receiver" area in the "Object/Variables list" combo box.
- Selection of the PDO transmission type from the "PDO transmission type" combo box. You can choose between Synchronous 1 to Synchronous 240 and Asynchronous 254 and Asynchronous 255.

  Example: For our example we select "Synchronous 1", i.e. the Receive PDO (or its content) received before the last SYNC telegram is applied when the new SYNC telegram has been received.
- The link just set with the "Add new PDO" button is checked for conformance with the PDO entries and entered in the list.

  The CANopen slave object 0x4CF2 (Application parameter 1) is now linked to the CANopen master object 0xA1E0, subindex 0x01. ProMaster and ProCANopen automatically generate an IEC variable (network variable) for the IEC project for this CANopen master object. In our case the name of the network variable in the IEC project is "di_ApplicationParam1". In the IEC project this is written to this network variable. The written value is written to the CANopen slave object 0x4CF2 via the CANopen master object 0xA1E0 - 0x01. See ▹IEC tab◃ in Chapter "4.3.6.2 Configuring CANopen master communication" from page 79 onward.



Transmitter	Receiver
_Axis_A A120 1 (u_controlword)	_Axis_B: Index: 6040 (controlword)
_Axis_A A220 1 (ud_PosIpSetAngle)	_Axis_B: Index: 60FB: Subindex: 17 (PosIpSetAng
_Axis_A A1E0 1 (di_ApplicationParam1)	_Axis_B: Index: 4CF2 (ApplicationParam1)

Figure 39: ProCANopen - Motion Control CANopen slave communication configuration "PD Receive" tab with default Motion Control configuration and additional object

NOTE

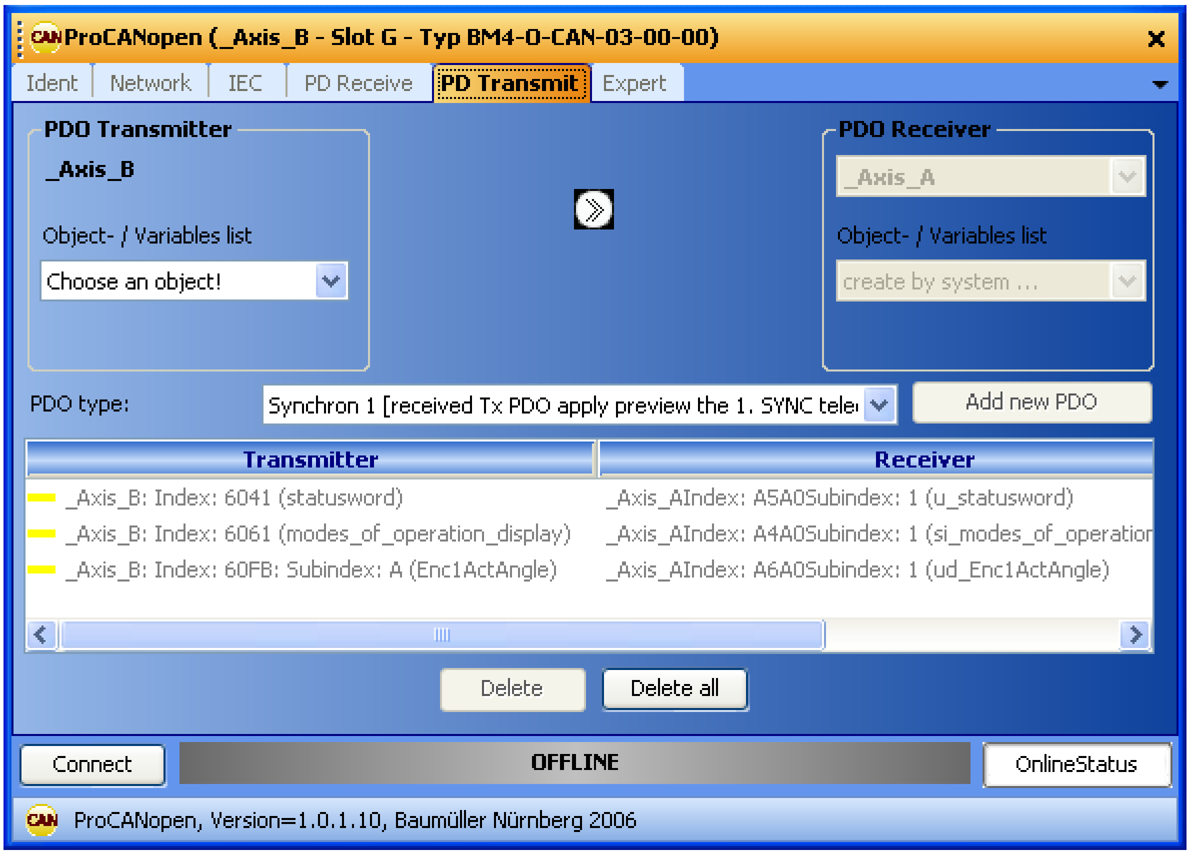
ProMaster automatically takes the limit of 30 characters for a variable name in the IEC project in ProProg wt III into account.

The name of the variables from the additional configuration can be changed later during the configuration of the PLC, in the chapter ▹4.3.7.4 CANopen ◃ from page 88 onward.

PD Transmit tab

The communication from the slave to the master is configured in the "PD Transmit" tab.

The default configuration is already set for Motion Control. You can carry out an additional configuration. When doing so, note that the Motion Control configuration has a higher priority than the configuration by the user.



Transmitter	Receiver
_Axis_B: Index: 6041 (statusword)	_Axis_AIndex: A5A0Subindex: 1 (u_statusword)
_Axis_B: Index: 6061 (modes_of_operation_display)	_Axis_AIndex: A4A0Subindex: 1 (si_modes_of_operation
_Axis_B: Index: 60FB: Subindex: A (Enc1ActAngle)	_Axis_AIndex: A6A0Subindex: 1 (ud_Enc1ActAngle)

Figure 40: ProCANopen - Motion Control CANopen slave communication configuration "PD Transmit" tab with default Motion Control configuration

The PDO receiver is the CANopen master with Motion Control.

The linking of an object in the CANopen slave to an object in the CANopen master takes place here. This object in the CANopen master is read via a network variable in the IEC program.

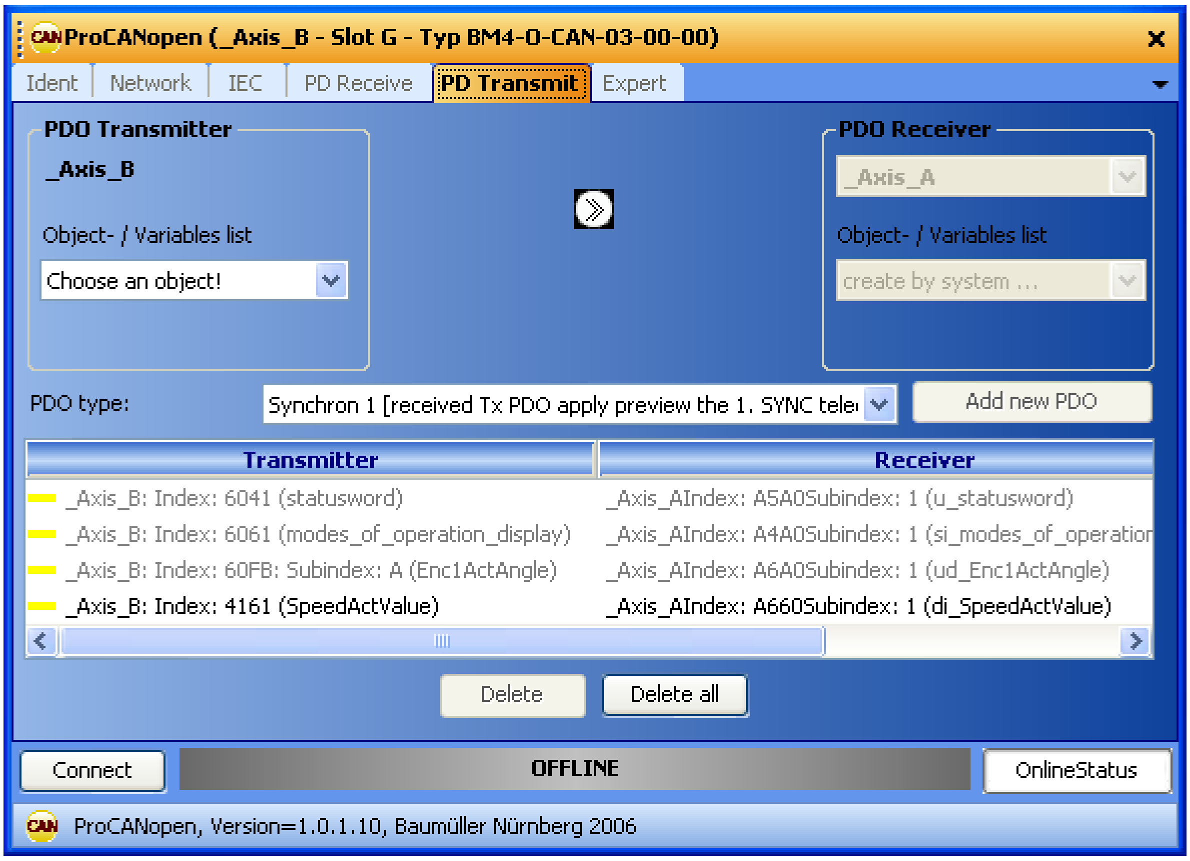
Procedure:

- Selection of the object to be transmitted in the "PDO Transmitter" area in the "Object/ Variables list" combo box.

  Example: We select CANopen slave object 0x4161 (Actual speed value) in the "PDO Transmitter" area in the "Object/Variable List" combo box.
- Selection of the PDO transmission type from the "PDO transmission type" combo box. You can choose between Synchronous 1 to Synchronous 240 and Asynchronous 254 and Asynchronous 255.

  Example: We select "Synchronous 1", i.e. the transmission is carried out in each case after the reception of the SYNC telegram (transmission after the first SYNC telegram since the last transmission).
- The link just set with the "Add new PDO" button is checked for conformance with the PDO entries and entered in the list.

  The CANopen slave object 0x4161 (Actual speed value) is now linked to the CANopen master object 0xA660, subindex 0x01. ProMaster and ProCANopen automatically generate an IEC variable (network variable) for the IEC project for this CANopen master object. In our case the name of the network variable in the IEC project is "di_SpeedActValue". In the IEC project this network variable is read. The value is read from the CANopen slave object 0x4161 via the CANopen master object 0xA660 - 0x01. See ▷IEC tab◁ in Chapter "4.3.6.2 Configuring CANopen master communication" from page 79 onward.



PDO Transmitter
_Axis_B
Object- / Variables list: Choose an object!

PDO Receiver
_Axis_A
Object- / Variables list: create by system ...

PDO type: Synchron 1 [received Tx PDO apply preview the 1. SYNC tele

Transmitter	Receiver
_Axis_B: Index: 6041 (statusword)	_Axis_AIndex: A5A0Subindex: 1 (u_statusword)
_Axis_B: Index: 6061 (modes_of_operation_display)	_Axis_AIndex: A4A0Subindex: 1 (si_modes_of_operatior
_Axis_B: Index: 60FB: Subindex: A (Enc1ActAngle)	_Axis_AIndex: A6A0Subindex: 1 (ud_Enc1ActAngle)
_Axis_B: Index: 4161 (SpeedActValue)	_Axis_AIndex: A660Subindex: 1 (di_SpeedActValue)

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 41: ProCANopen - Motion Control CANopen slave communication configuration "PD Transmit" tab with default Motion Control configuration and additional object

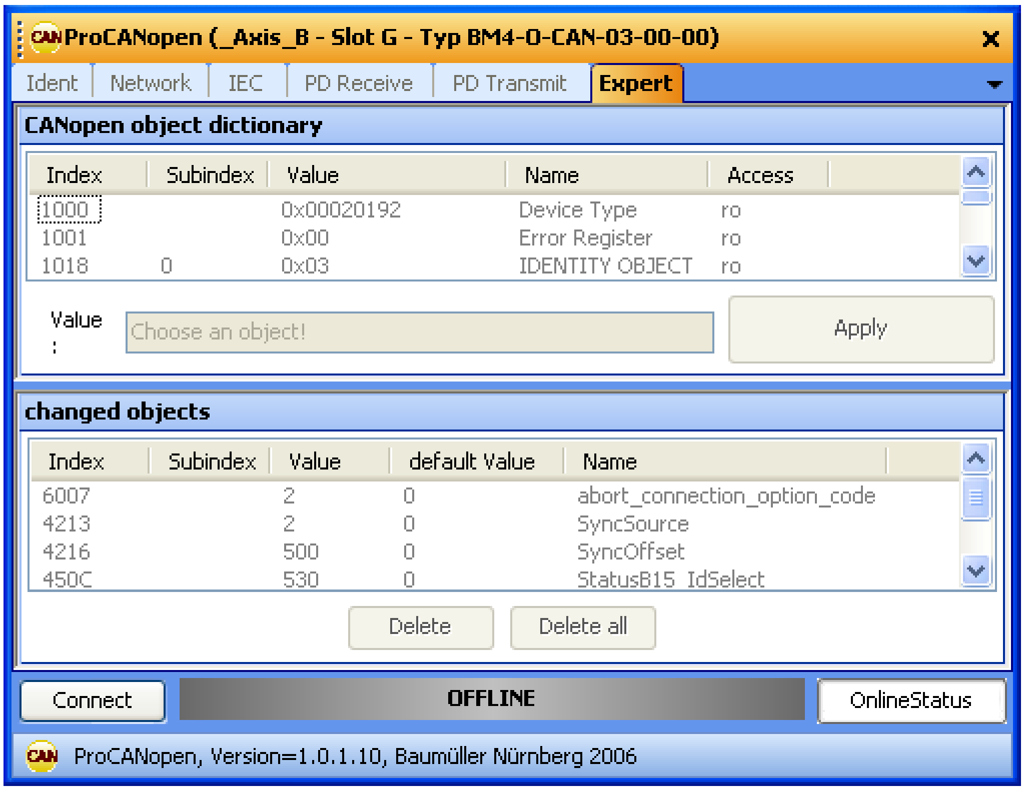
NOTE

ProMaster automatically takes the limit of 30 characters for a variable name in the IEC project in ProProg wt III into account.

The name of the variables from the additional configuration can be changed later during the configuration of the PLC, in the chapter ▷4.3.7.4 CANopen ◁ from page 88 onward.

Expert tab

When Motion Control is used, the "Expert" tab is used to display the objects available in the CANopen slave and their values.


ProCANopen (_Axis_B - Slot G - Typ BM4-O-CAN-03-00-00)

CANopen object dictionary

Index	Subindex	Value	Name	Access
1000		0x00020192	Device Type	ro
1001		0x00	Error Register	ro
1018	0	0x03	IDENTITY OBJECT	ro

Value : Choose an object!

changed objects

Index	Subindex	Value	default Value	Name
6007		2	0	abort_connection_option_code
4213		2	0	SyncSource
4216		500	0	SyncOffset
450C		530	0	StatusB15_IdSelect

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 42: ProCANopen - CANopen slave communication configuration "Expert" tab

4.3.6.2 Configuring CANopen master communication

Open ProCANopen for our CANopen master by clicking on the device "_Axis_A" and then select "Configuration Data (Components)\CANopen Master (Slot G)\Configure Master Bus (ProCANopen)" via the context menu.

The CANopen master was already set with the default setting when creating the machine configuration so that the CANopen settings for Motion Control are configured.

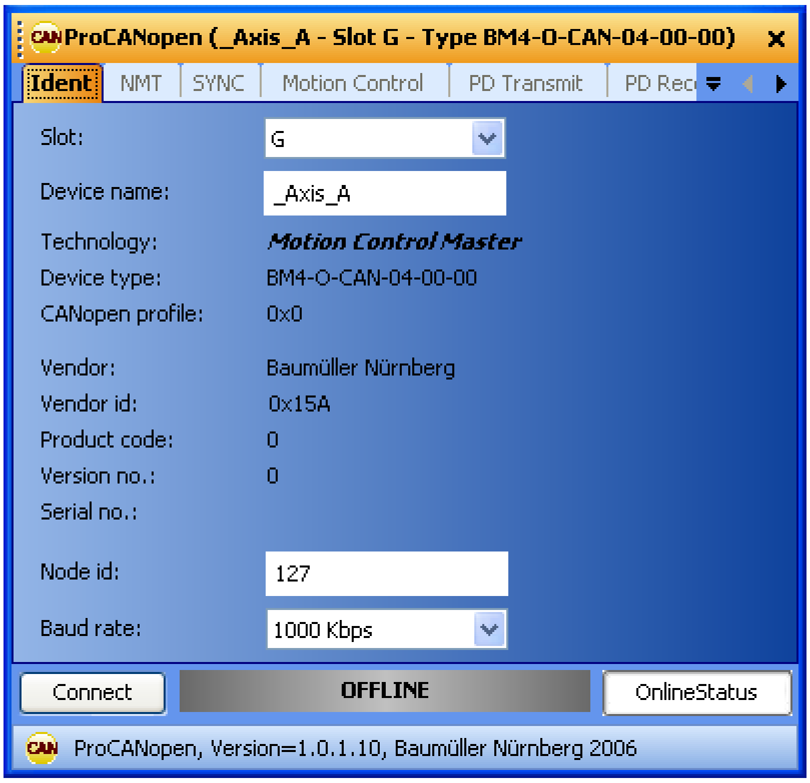
The window for the CANopen master communication configuration is opened.

Ident tab

For Motion Control, you can set the slot of the CANopen master module on the b maXX drive PLC, the name of this module (device name) in ProMaster and the baud rate and the node-ID for CANopen in the "Ident" tab.

Changing the slot, device name, baud rate and node-ID affects the ProMaster project.

In addition, the technology and the CANopen-specific information like the device type, CANopen profile, manufacturer/vendor-ID etc. are also displayed.


ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)

Slot:	G
Device name:	_Axis_A
Technology:	*Motion Control Master*
Device type:	BM4-O-CAN-04-00-00
CANopen profile:	0x0
Vendor:	Baumüller Nürnberg
Vendor id:	0x15A
Product code:	0
Version no.:	0
Serial no.:	
Node id:	127
Baud rate:	1000 Kbps

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 43: ProCANopen - CANopen master communication configuration Ident tab

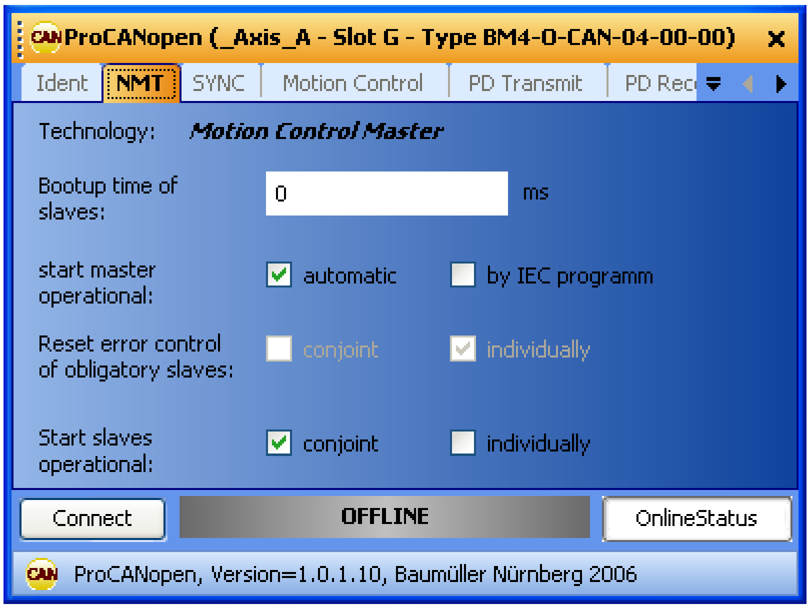
NMT tab

When using Motion Control, the settings on the "NMT" tab remain unchanged.

In the edit box "Bootup time of slaves", it can be set how long the CANopen master waits for the boot-up telegram of the CANopen slaves following switch-on. The entry 0 ms means that the master waits until all slaves have transmitted their boot-up message. This means the master will not signal a timeout.

NOTE

There are CANopen slaves that do not transmit their boot-up telegram until over 30 seconds after being switched on.



ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)

Ident | NMT | SYNC | Motion Control | PD Transmit | PD Rec

Technology: *Motion Control Master*

Bootup time of slaves: 0 ms

start master operational: [✓ automatic] by IEC programm

Reset error control of obligatory slaves: conjoint [✓ individually]

Start slaves operational: [✓ conjoint] individually

Connect | OFFLINE | OnlineStatus

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

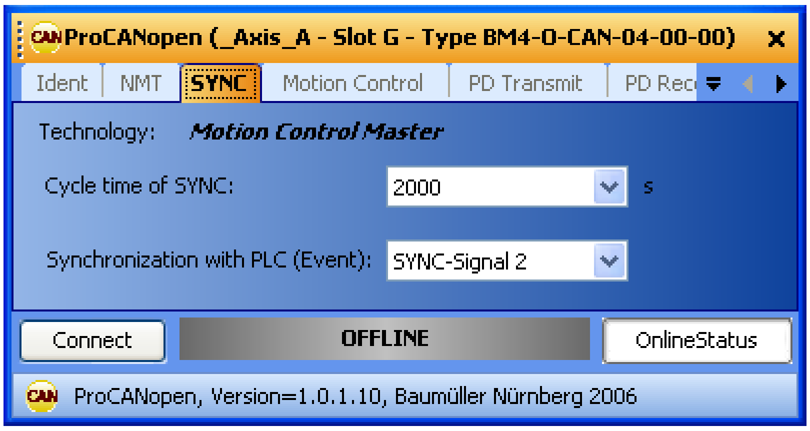
Figure 44: ProCANopen - CANopen master communication configuration NMT tab

SYNC tab

When using Motion Control, you can set the SYNC cycle time and the synchronization signal between the CANopen master option module and the b maXX drive PLC in the "SYNC" tab.

The change from the SYNC cycle time and the synchronization signal affects the ProMaster project.

It must be noted that the CANopen master transmits the SYNC telegram on the CANopen bus as a SYNC generator. The transmission interval is specified for the CANopen master in our example via the signal in the "Synchronization with PLC (Event)" combination box. For the CANopen master option module you can choose from the signals "SYNC-Signal 1" and "SYNC-Signal 2". Other signals are not permitted. The signal must be generated on the b maXX drive PLC. With Motion Control, the signal "SYNC-Signal 1" is used and automatically generated by the b maXX drive PLC in the default configuration.



ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)

Ident | NMT | SYNC | Motion Control | PD Transmit | PD Rec

Technology: *Motion Control Master*

Cycle time of SYNC: 2000 s

Synchronization with PLC (Event): SYNC-Signal 2

Connect | OFFLINE | OnlineStatus

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 45: ProCANopen - CANopen master communication configuration SYNC tab

Motion Control tab

The settings for Motion Control are made in the "Motion Control" tab.

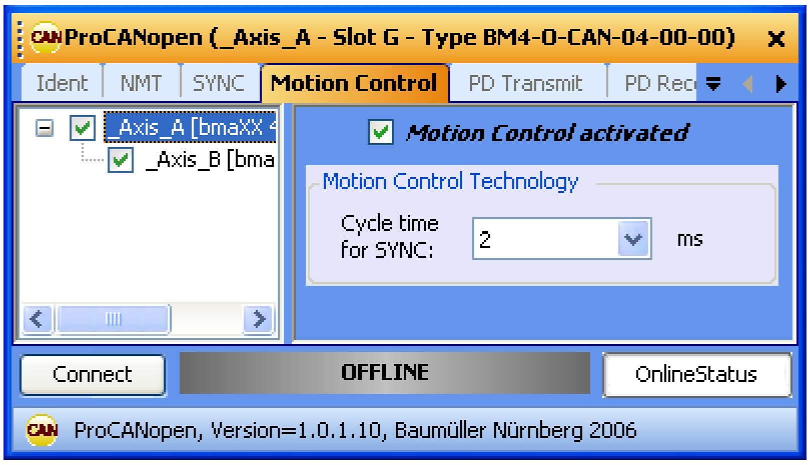
For the machine configuration, you can specify whether Motion Control will be used. The selection is enabled for you when the master is selected from the project tree.

In addition, you can specify for individual drives that these are not controlled via Motion Control and, for example, change the names of individual drives. The selection is enabled for you when the respective drive is selected in the project tree. The changes affect the ProMaster project.

NOTE

If you use drives without Motion Control in the machine configuration, these must always be configured (see ▷Configuring CANopen slave communication◁ in Chapter "4.3 ProMaster, ProCANopen, ProProg wt III and Motion Control" from Page 66.

In addition, you are shown Motion Control-specific information, such as the „Cycle time of SYNC“ and the CANopen-specific information, such as the node-ID of the respective drive.



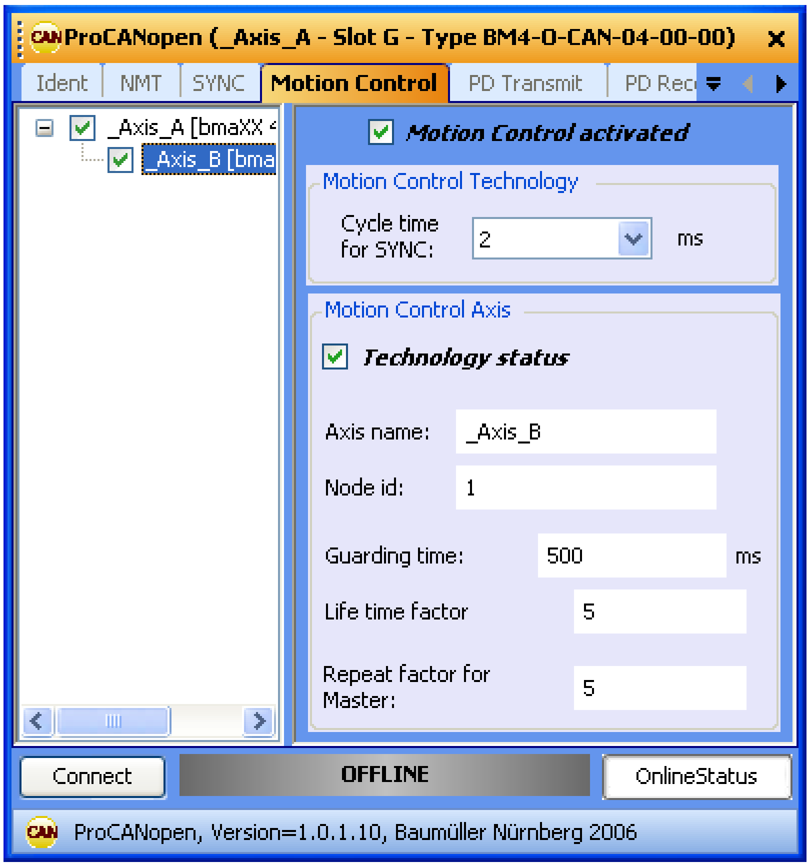
☑ Motion Control activated

Motion Control Technology

Cycle time for SYNC: 2 ms

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 46: ProCANopen - CANopen master communication configuration Motion Control tab


ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)
Ident
NMT
SYNC
Motion Control
PD Transmit
PD Rec
_Axis_A [bmaXX
_Axis_B [bma
Connect
OFFLINE
OnlineStatus
ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

☑ Motion Control activated

Motion Control Technology

Cycle time for SYNC: 2 ms

Motion Control Axis

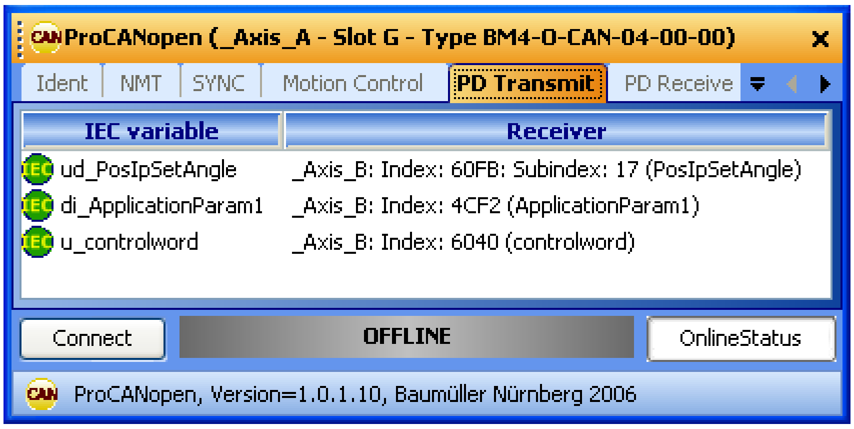
☑ Technology status

Axis name:	_Axis_B	
Node id:	1	
Guarding time:	500	ms
Life time factor	5	
Repeat factor for Master:	5	

Figure 47: ProCANopen - CANopen master communication configuration Motion Control tab

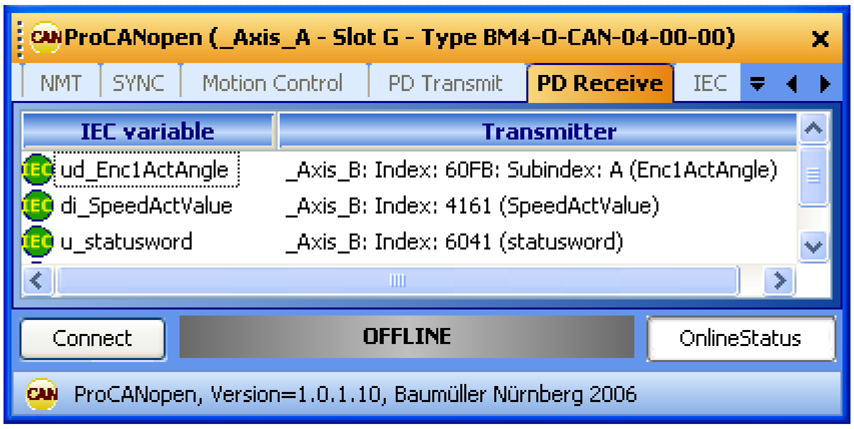
PD Transmit and PD Receive tab

The data on these pages is used for display.


ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)
Ident
NMT
SYNC
Motion Control
PD Transmit
PD Receive
Connect
OFFLINE
OnlineStatus
ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

IEC variable	Receiver
ud_PosIpSetAngle	_Axis_B: Index: 60FB: Subindex: 17 (PosIpSetAngle)
di_ApplicationParam1	_Axis_B: Index: 4CF2 (ApplicationParam1)
u_controlword	_Axis_B: Index: 6040 (controlword)

Figure 48: ProCANopen - CANopen master communication configuration PD Transmit tab


ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)

IEC variable	Transmitter
ud_Enc1ActAngle	_Axis_B: Index: 60FB: Subindex: A (Enc1ActAngle)
di_SpeedActValue	_Axis_B: Index: 4161 (SpeedActValue)
u_statusword	_Axis_B: Index: 6041 (statusword)

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 49: ProCANopen - CANopen master communication configuration PD Receive tab

In the "PD Transmit" tab and "PD Receive" tab the IEC variables (network variables), are each shown for transmission (PD Transmit) and reception (PD Receive) respectively.

The network variables (via the CANopen master objects) are assigned to the CANopen slave objects automatically with ProMaster and ProCANopen.

Additional CANopen slave objects are assigned to network variables during the configuration of the CANopen slave communication of the CANopen slaves. After this configuration of the CANopen slaves, the corresponding data are shown in the „PD Transmit" tab and the „PD Receive“ tab of the configuration of the CANopen master communication.

To configure the CANopen communication of the CANopen slaves, see the chapter ▹4.3.6.1 Configuring CANopen slave communication ◃ from page 66 onward.

IEC tab

During the configuration of the CANopen slave communication, the CANopen slave objects (via CANopen master objects) are assigned to the network variables for the IEC project on the PLC.

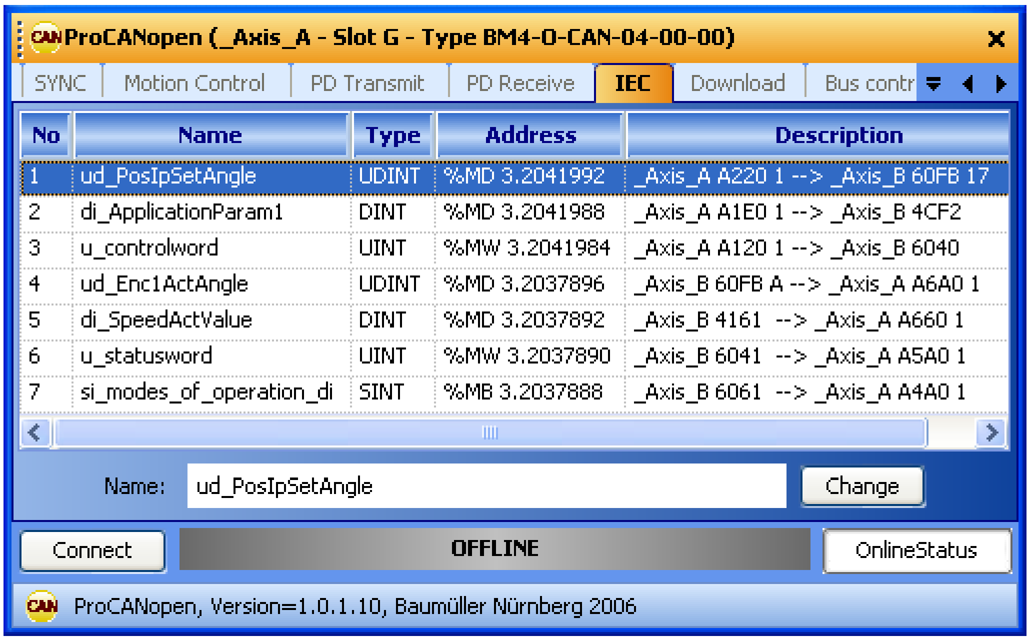
The variables shown in the IEC tab are also called network variables. The process data are written and read in the IEC project via the network variables.

When using Motion Control, some of the assignments are created with the default settings.

The network variables and their data type in the IEC project on the PLC are shown in the "IEC" tab.

The calculation of the (IEC) address of the network variables is not carried out until after "Update list" in the chapter ▹Download tab◃ in Chapter "4.3.6.2 Configuring CANopen master communication" from page 80 onward.

To configure the CANopen communication of the CANopen slaves, see ▹4.3.6.1 Configuring CANopen slave communication ◃ from page 66 onward.


ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)

No	Name	Type	Address	Description
1	ud_PosIpSetAngle	UDINT	%MD 3.2041992	_Axis_A A220 1 --> _Axis_B 60FB 17
2	di_ApplicationParam1	DINT	%MD 3.2041988	_Axis_A A1E0 1 --> _Axis_B 4CF2
3	u_controlword	UINT	%MW 3.2041984	_Axis_A A120 1 --> _Axis_B 6040
4	ud_Enc1ActAngle	UDINT	%MD 3.2037896	_Axis_B 60FB A --> _Axis_A A6A0 1
5	di_SpeedActValue	DINT	%MD 3.2037892	_Axis_B 4161 --> _Axis_A A660 1
6	u_statusword	UINT	%MW 3.2037890	_Axis_B 6041 --> _Axis_A A5A0 1
7	si_modes_of_operation_di	SINT	%MB 3.2037888	_Axis_B 6061 --> _Axis_A A4A0 1

Name: ud_PosIpSetAngle

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 50: ProCANopen - CANopen master communication configuration IEC tab following configuration of the CANopen slaves and "Update list" (Download tab)

You can change the name of the network variables by selecting the network variable name from the "Name" column, editing the new network name in the "Name" edit box and then clicking on the "Change" button.

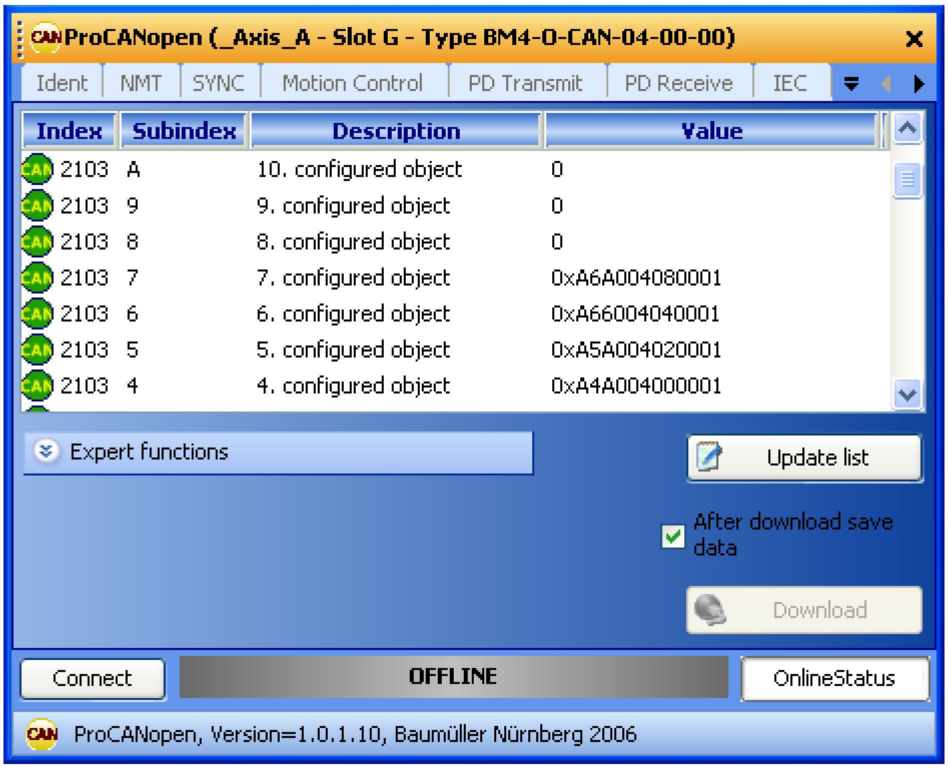
NOTE

Please take the limit of 30 characters for a variable name in the IEC project in ProProg wt III into account.

Download tab

In the Download tab, the objects are shown which are loaded to the CANopen master (Download) following the configuration of the CANopen communication of the CANopen master and of the CANopen slave. As long as the configuration is not completed, no data are shown in the Download tab, i.e. no data can be loaded to the CANopen master.

Now press the "Update list" button (lower right). This generates the configuration data (both master and slave) for the download on the CANopen master.



Index	Subindex	Description	Value
2103	A	10. configured object	0
2103	9	9. configured object	0
2103	8	8. configured object	0
2103	7	7. configured object	0xA6A004080001
2103	6	6. configured object	0xA66004040001
2103	5	5. configured object	0xA5A004020001
2103	4	4. configured object	0xA4A004000001

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 51: ProCANopen - CANopen master communication configuration Download tab

Now press the "Download" button (press the „Connect“ button before if necessary). The connection from ProMaster to the CANopen master (BM4-O-CAN-04 module) is established via the b maXX drive PLC (BM4-O-PLC-01 module). Then the CANopen configuration data (both master and slave) are transmitted on the CANopen master.

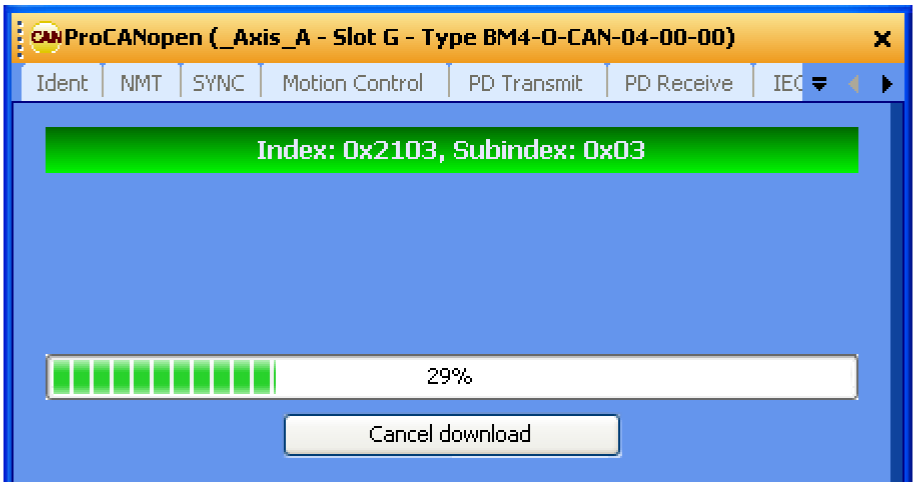

ProCANopen (_Axis_A - Slot G - Type BM4-O-CAN-04-00-00)
Ident
NMT
SYNC
Motion Control
PD Transmit
PD Receive
IEC
Index: 0x2103, Subindex: 0x03
29%
Cancel download

Figure 52: ProCANopen - CANopen communication configuration Download

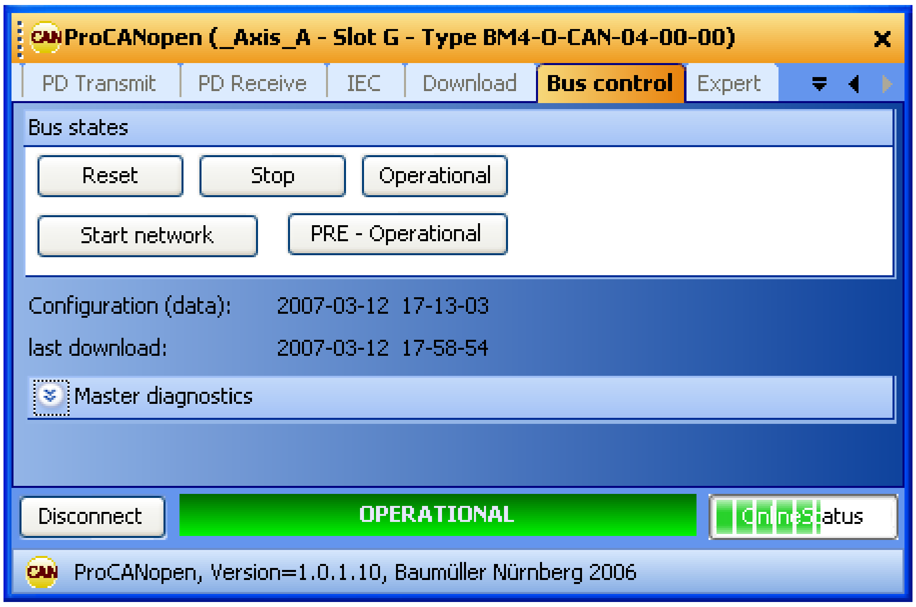
If errors are reported here, it may be necessary to switch the CANopen master into the CANopen "Reset" state with the "Bus Control" tab, "Connect" button and "Reset" button, and then to start the download again in the "Download" tab.

After the download is completed, the Download object list is shown again.

Now both the CANopen configuration data for the CANopen master and the CANopen configuration data for the CANopen slave, which the master transmits to the slave when the CANopen network is started up, are located on the CANopen master.

Bus control tab

In the "Bus control" tab, experienced CANopen users can control the CANopen network "by hand" following the download of the CANopen configuration data. This can, for example, be helpful during the physical commissioning of the CANopen network.



Configuration (data): 2007-03-12 17-13-03

last download: 2007-03-12 17-58-54

ProCANopen, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 53: ProCANopen - CANopen master communication configuration "Bus control" tab

4.3.7 Configuring PLC

After the configuration of the CANopen network has been configured and the IEC project has been created, the data for the b maXX drive PLC are now configured.

In the process,

- the IEC project is integrated in the ProMaster project
- the cam data records are selected.

In addition, it is possible to

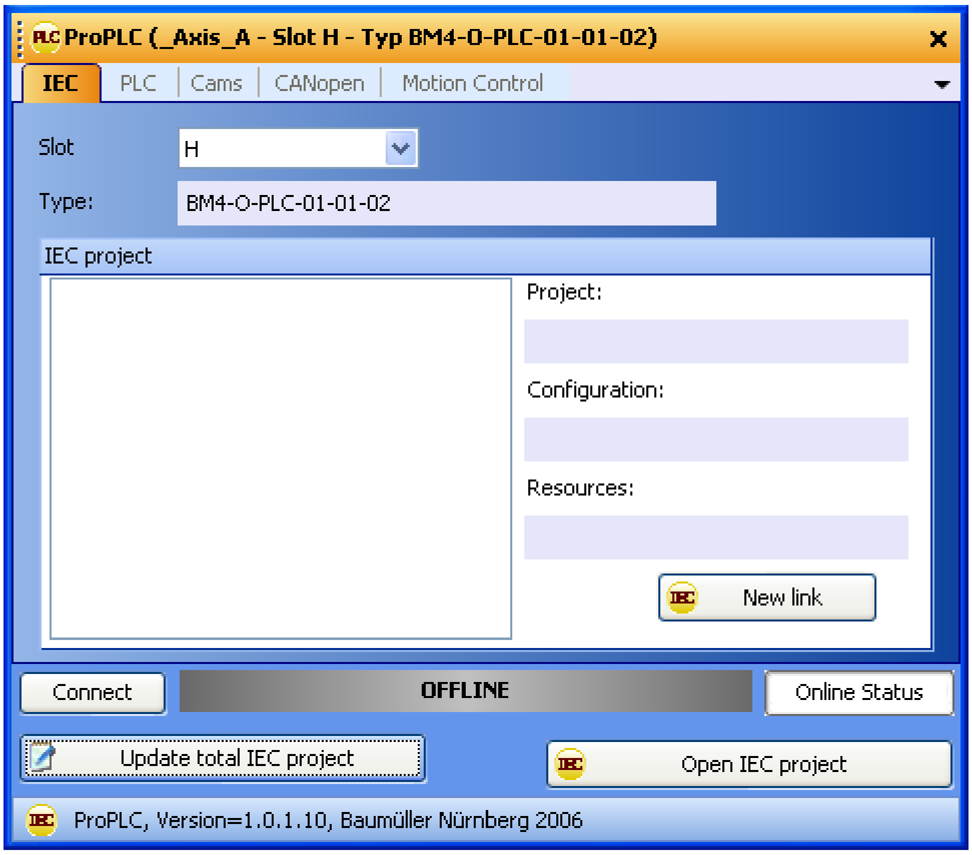
- create your own cam disk data with the ProCAM cam disk editor

Then you click on the "Update total IEC project" button in the "ProPLC" window. The data for the machine configuration, both for the CANopen network and for the b maXX drive PLC, are then generated.

4.3.7.1 IEC

In this section we explain how an IEC project is connected to ProMaster.

Open the window "ProPLC" in the ProMaster project in the network view for our CANopen master by clicking on the device "_Axis_A", selecting "Configuration Data (Components)\PLC (Slot H)\PLC - Configuration (ProPLC)" via the context menu and then selecting the "IEC" tab there.



Slot: H

Type: BM4-O-PLC-01-01-02

IEC project

Project:

Configuration:

Resources:

OFFLINE

ProPLC, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 54: ProPLC - Connecting to example IEC project

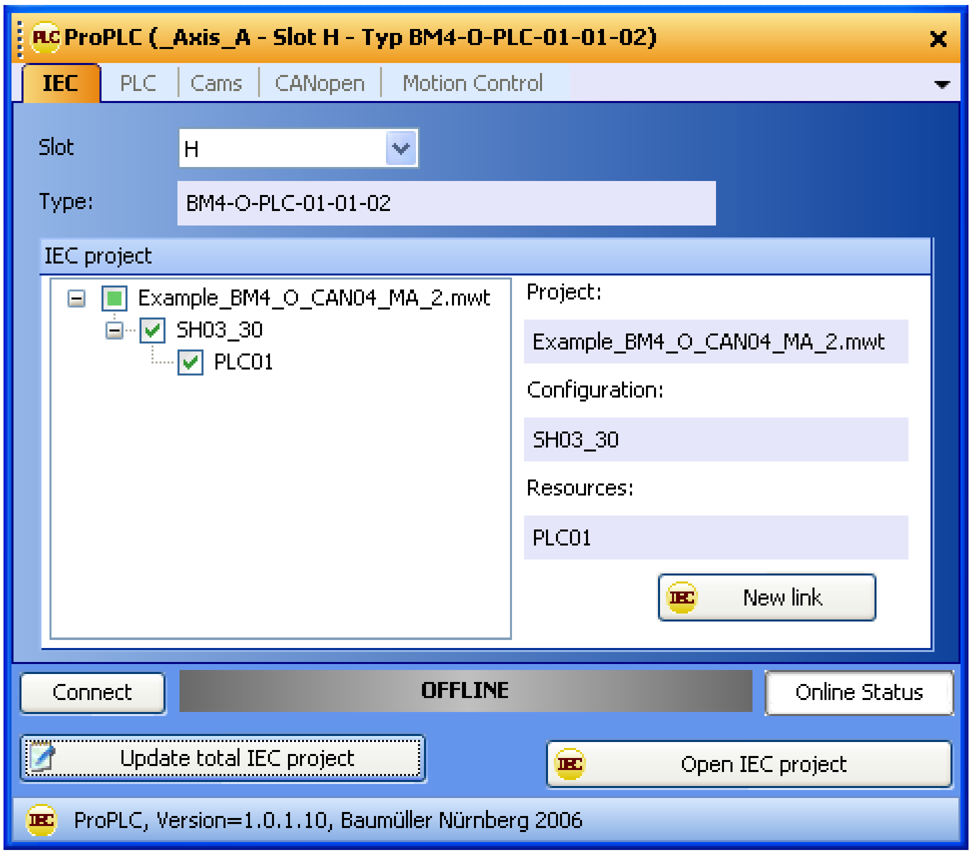
NOTE

Only ProProg wt III projects can be opened with ProMaster. If you want to use and existing PROPROG wt II project, you must first open it with ProProg wt III (and convert it in the process) and can then open and use it in ProMaster.

Please note that your PROPROG wt II libraries are also converted in the process!

Click on the "New link" button and select our example IEC project "Example_BM4_O_CAN04_MA_2.mwt", which we created in the chapter ▹4.3.4 Creating an IEC project with ProProg wt III ◃ from page 59 onward.

The IEC project just selected is shown in the "IEC" tab.


ProPLC (_Axis_A - Slot H - Typ BM4-O-PLC-01-01-02)

Slot: H

Type: BM4-O-PLC-01-01-02

IEC project

- Example_BM4_O_CAN04_MA_2.mwt
  - SH03_30
    - PLC01

Project: Example_BM4_O_CAN04_MA_2.mwt

Configuration: SH03_30

Resources: PLC01

ProPLC, Version=1.0.1.10, Baumüller Nürnberg 2006

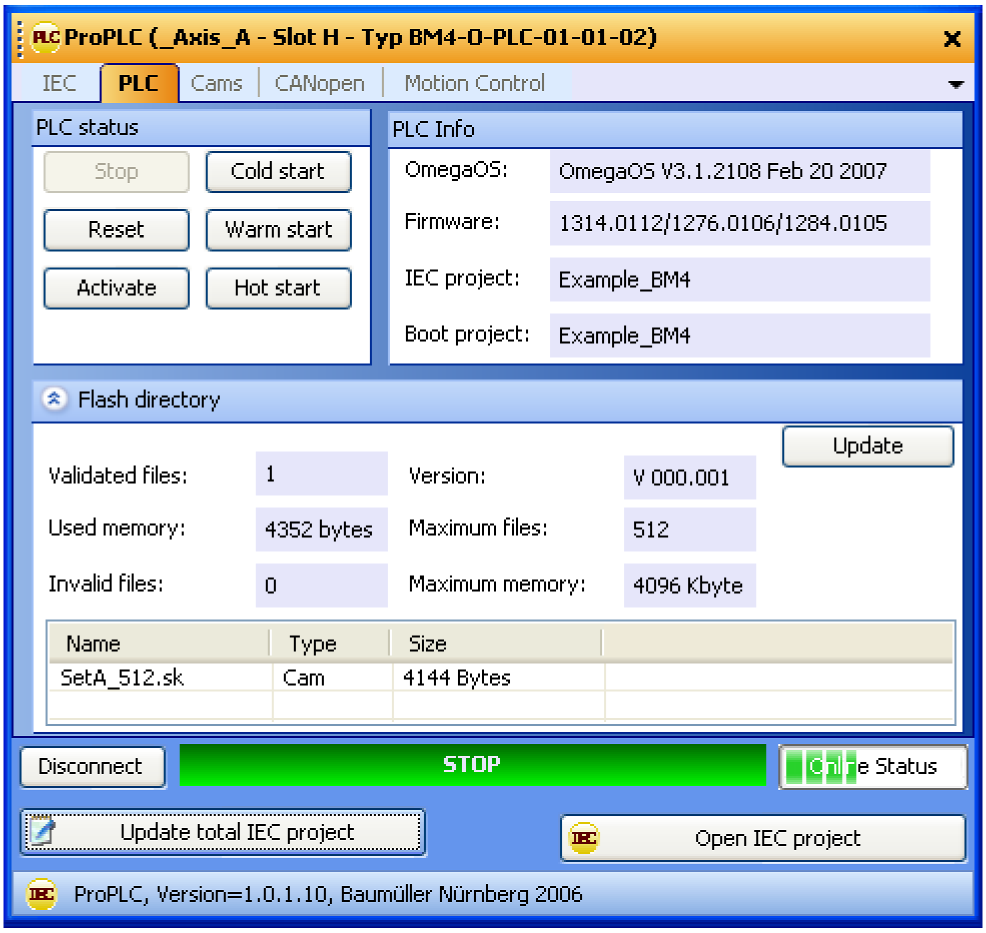
Figure 55: ProPLC - IEC tab

If you connect an existing application (or a template) from ProProg wt III to the ProMaster project, the names of the devices in ProMaster and in the IEC project may not be identical. In this case, see the chapter ▷4.3.9 Programming IEC project ◁ from page 92 onward.

4.3.7.2 PLC

Following downloading data to CANopen master and to b maXX drive PLC (see ▷4.3.8 Downloading data to CANopen master and to b maXX drive PLC ◁ from page 91 onward), the IEC project in the PLC tab can be activated and started on the b maXX drive PLC (area „PLC status“).

Furthermore data of the b maXX drive PLC, such as version of operating system, Firmware version, IEC project and boot project are shown in the area „PLC Info“. Information to the files in the Flash memory of the b maXX drive PLC are shown in the area „Flash directory“ (following „Connect“ and „Update“), e.g. cam data set (see Cams).


ProPLC (_Axis_A - Slot H - Typ BM4-O-PLC-01-01-02)

IEC | PLC | Cams | CANopen | Motion Control

PLC status: Stop, Cold start, Reset, Warm start, Activate, Hot start

PLC Info

OmegaOS:	OmegaOS V3.1.2108 Feb 20 2007
Firmware:	1314.0112/1276.0106/1284.0105
IEC project:	Example_BM4
Boot project:	Example_BM4

Flash directory

Validated files:	1	Version:	V 000.001
Used memory:	4352 bytes	Maximum files:	512
Invalid files:	0	Maximum memory:	4096 Kbyte

Name	Type	Size
SetA_512.sk	Cam	4144 Bytes

ProPLC, Version=1.0.1.10, Baumüller Nürnberg 2006

Figure 56: ProPLC - PLC tab following download of cam disk (cam disk tab)

4.3.7.3 Cams

In this section we explain how a cam-disk data record is created and connected to ProMaster.

Open the window "ProPLC" in the ProMaster project in the network view for our CANopen master by clicking on the device "_Axis_A", selecting "Configuration Data (Components)\PLC (Slot H)\PLC - Configuration (ProPLC)" via the context menu and then selecting the "Cams" tab there.

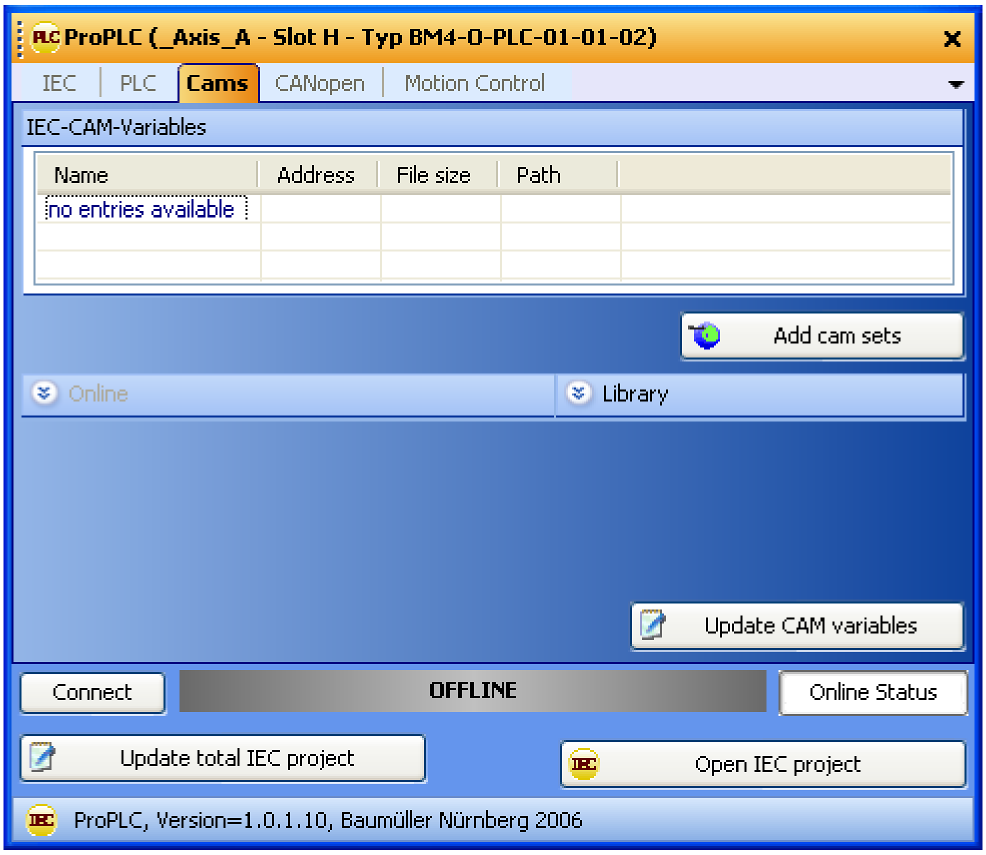


Figure 57: ProPLC - "Cam Disks" tab without cam data record

Press the "Add cam sets" button. The "ProMaster.NET - Cams" window is opened.

Press the "Add" button in the "Hard disk" area and select the path on your hard disk in which your cam data are located.

NOTE

Example cam data are contained in the installation directory of ProCAM in the sub-folder "examples" (e.g. C:\...\Baumueller\ProCam 2\examples).

If you want to generate your own cams, click on the "ProCAM" button to open the cam disk editor. You can edit your cams there.

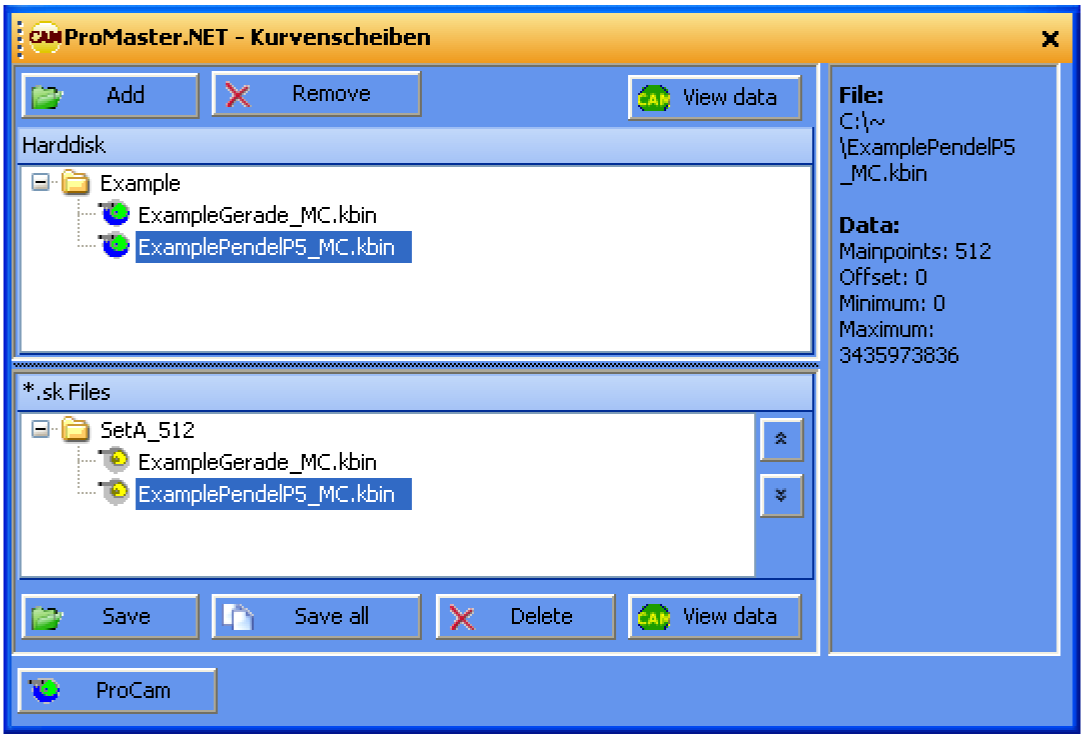
Now put together your cam data record (in the area "*.sk files") from the various cam disk data (in the area "Hard disk") with Drag&Drop.

For our example we select the cam disk data "ExampleGerade_MC.kbin" and "ExamplePendelP5_MC.kbin", which are grouped in the cam-disk data record "SetA_512".

NOTE

You can graphically display the cam disk data (in the respective area) with the "View data" button. This simplifies the cam selection.

Then click on the "Save" button. This creates the cam data record and makes it available to ProMaster.



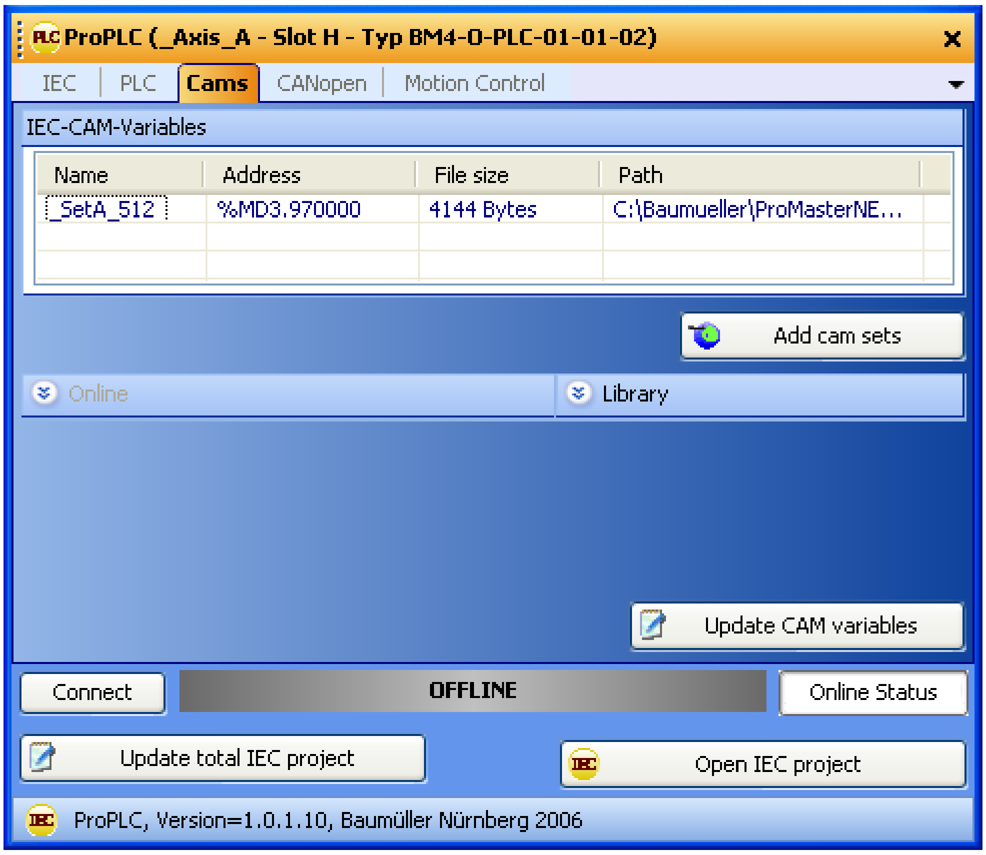
File:
C:\~
\ExamplePendelP5
_MC.kbin

Data:
Mainpoints: 512
Offset: 0
Minimum: 0
Maximum:
3435973836

Figure 58: ProMaster (- PLC configuration - Cam Disks tab -) Create cam data record

Now close the "ProMaster.NET - Cams" window by clicking on "x" at the upper right.

In the "Cams" tab you now see the name of the variables of the cam-disk data record in the IEC project, their address in the IEC project and the file name of the cam-disk data record on the hard disk.



Name	Address	File size	Path
_SetA_512	%MD3.970000	4144 Bytes	C:\Baumueller\ProMasterNE...

Figure 59: ProPLC - "Cams" tab with cam data record

If you connect an existing application (or a template) from ProProg wt III to the ProMaster project, the cam data-record name in ProMaster and in the IEC project may not be identical. In this case, see the chapter ▹4.3.9 Programming IEC project ◃ from page 92 onward.

From within the "Cams" tab you can clear the flash memory on the b maXX drive PLC with the "Delete data" button. This may be necessary if the flash memory is "full" due to various downloads (of both cam-disk data records and IEC projects).

In addition, you can download the cam-disk data records individually onto the b maXX drive PLC from within the "Cams" tab via the "Download Cams" button.

Now load the cam-disk data records onto the b maXX drive PLC via the "Download Cams" button.

4.3.7.4 CANopen

In the "CANopen" tab the name, address and comments of the network variables of the CANopen slaves in the IEC project are shown.

When using Motion Control, the communication is carried out via the axis variable. After the "Update total IEC project" button is pressed, see the chapter ▹4.3.8 Downloading data to CANopen master and to b maXX drive PLC ◃ from page 91 onward, ProMaster creates the new axis variable with the device name of the CANopen slave in the IEC project. In addition, ProMaster creates network variables in the IEC project. These are divided into the standard network variables for Motion Control and, if additional objects have been created during the CANopen slave configuration, the additional network variables for CANopen.

These network variables are shown in the "CANopen" tab, i.e. both the standard network variables for Motion Control and the additional network variables for CANopen.

The following must always be observed here:

The standard network variables for Motion Control may be read by the user, but not written.

These are the standard Motion Control network variables for the set values:

"u_controlword.."	for the axis control word
"ud_PosIpSetAngel.."	for the synchronized position angle set value of the axis

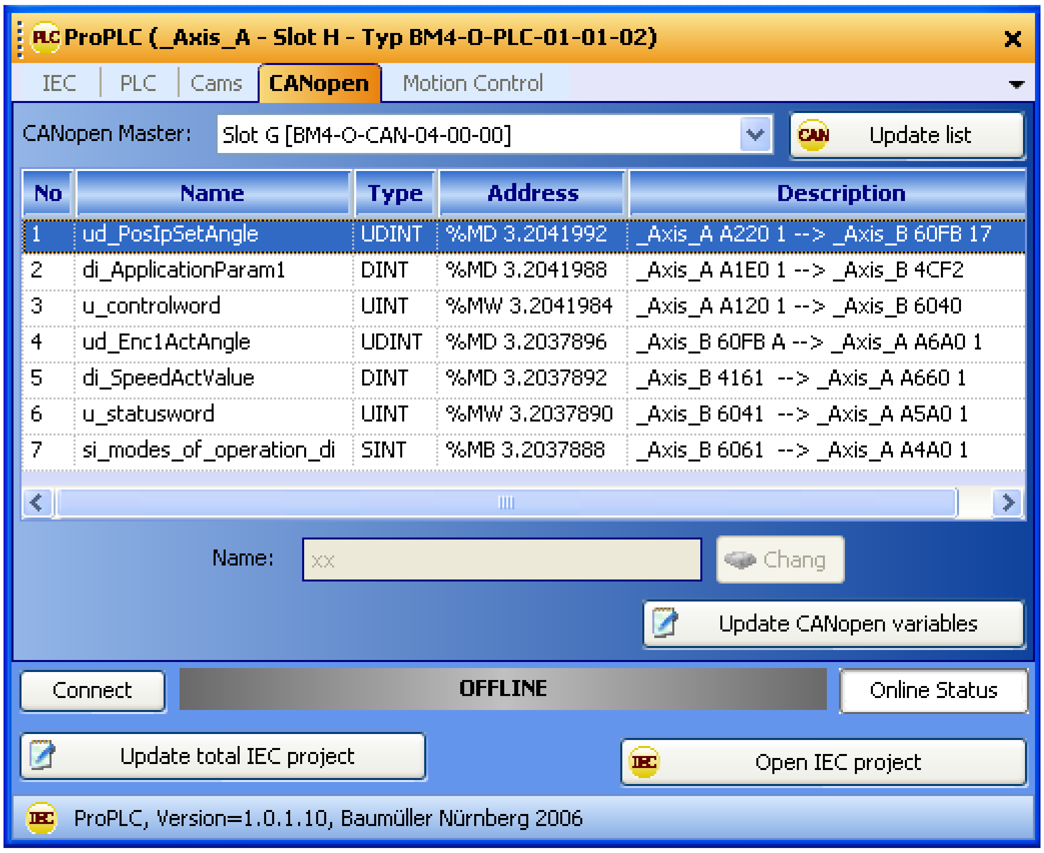
These are the standard Motion Control network variables for the actual values:

"u_statusword.."	for the axis status word
"ud_Enc1ActAngle.."	for the synchronized position angle actual value of the axis
"si_modes_of_operation_display.."	for the axis operating mode

NOTE

The standard network variables for Motion Control may be read by the user, but not written.

The number after the network variable name is an internal number to distinguish automatically generated network variables with the same name.



No	Name	Type	Address	Description
1	ud_PosIpSetAngle	UDINT	%MD 3.2041992	_Axis_A A220 1 --> _Axis_B 60FB 17
2	di_ApplicationParam1	DINT	%MD 3.2041988	_Axis_A A1E0 1 --> _Axis_B 4CF2
3	u_controlword	UINT	%MW 3.2041984	_Axis_A A120 1 --> _Axis_B 6040
4	ud_Enc1ActAngle	UDINT	%MD 3.2037896	_Axis_B 60FB A --> _Axis_A A6A0 1
5	di_SpeedActValue	DINT	%MD 3.2037892	_Axis_B 4161 --> _Axis_A A660 1
6	u_statusword	UINT	%MW 3.2037890	_Axis_B 6041 --> _Axis_A A5A0 1
7	si_modes_of_operation_di	SINT	%MB 3.2037888	_Axis_B 6061 --> _Axis_A A4A0 1

Figure 60: ProPLC - CANopen tab

If during the configuration of the CANopen slave communication you have created the additional network variables (and their link to CANopen slaves) suggested in the sections ▹PD Receive tab ◃ from page 69 onward and ▹PD Transmit tab ◃ from page 71 onward, you see the additional network variables.

The additional network variable for the setpoint values is:

"di_ApplicationParam1.."	for Application parameter 1 of the axis

The additional network variable for the actual values is:

"di_SpeedActValue.."	for the actual speed value of the axis

When changing network variable names, please note that each variable name can only be assigned once in the IEC project. This must especially be observed when using several CANopen master option modules on the b maXX drive PLC.

NOTE

The additional network variables for set values must be written in the motion control event task.

The additional network variables for set values must be written at every call of the motion control event task.

NOTE

Each variable name my only be used once in the IEC project.

4.3.7.5 Motion Control

In the future, the general settings for Motion Control will be are configured in the "Motion Control" tab.

If you want to use the default Motion Control settings, you need make no settings here.

4.3.8 Downloading data to CANopen master and to b maXX drive PLC

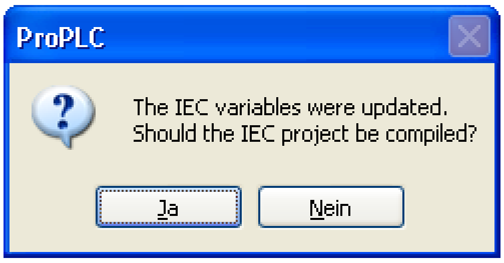
The data for the CANopen master is currently downloaded via ▹Download tab◃ in Chapter "4.3.6.2 Configuring CANopen master communication" from page 80 onward.

The cam data records for the b maXX controller PLC are currently downloaded via the "Cams" tab of the PLC configuration. See the chapter ▹4.3.7.3 Cams ◃ from page 85 onward.

The IEC project for the b maXX drive PLC is currently downloaded as usual via the ProProg wt III.

Now click on the "Update total IEC project" button in the "PLC Configuration" window. The data for the b maXX drive PLC are now generated. This process can take some time, as among other things ProProg wt III is opened and the configured IEC variables are written to the global variable worksheet in the IEC project.

Afterwards you can compile the IEC project by confirming the respective request with „Yes“.


ProPLC

The IEC variables were updated.
Should the IEC project be compiled?

Figure 61: ProMaster - Compiling the updated IEC project

You can alternatively compile the IEC project also in ProProg wt III via the ProProg wt III menu "Build\Rebuild Project".

Now download the IEC project to the b maXX drive PLC with (ProProg wt III - "Online\ Project Control..." menu ¨ "Download" ¨ "Download" boot project).

Now carry out a reset on the b maXX drive PLC and then switch the PLC into the "RUN" state (as an alternative, you can also switch the entire CANopen network off and then on again). Now you can control the local axis _Axis_A and the CANopen slave axis _Axis_B via the IEC project in ProProg wt III.

4.3.9 Programming IEC project

4.3.9.1 General information

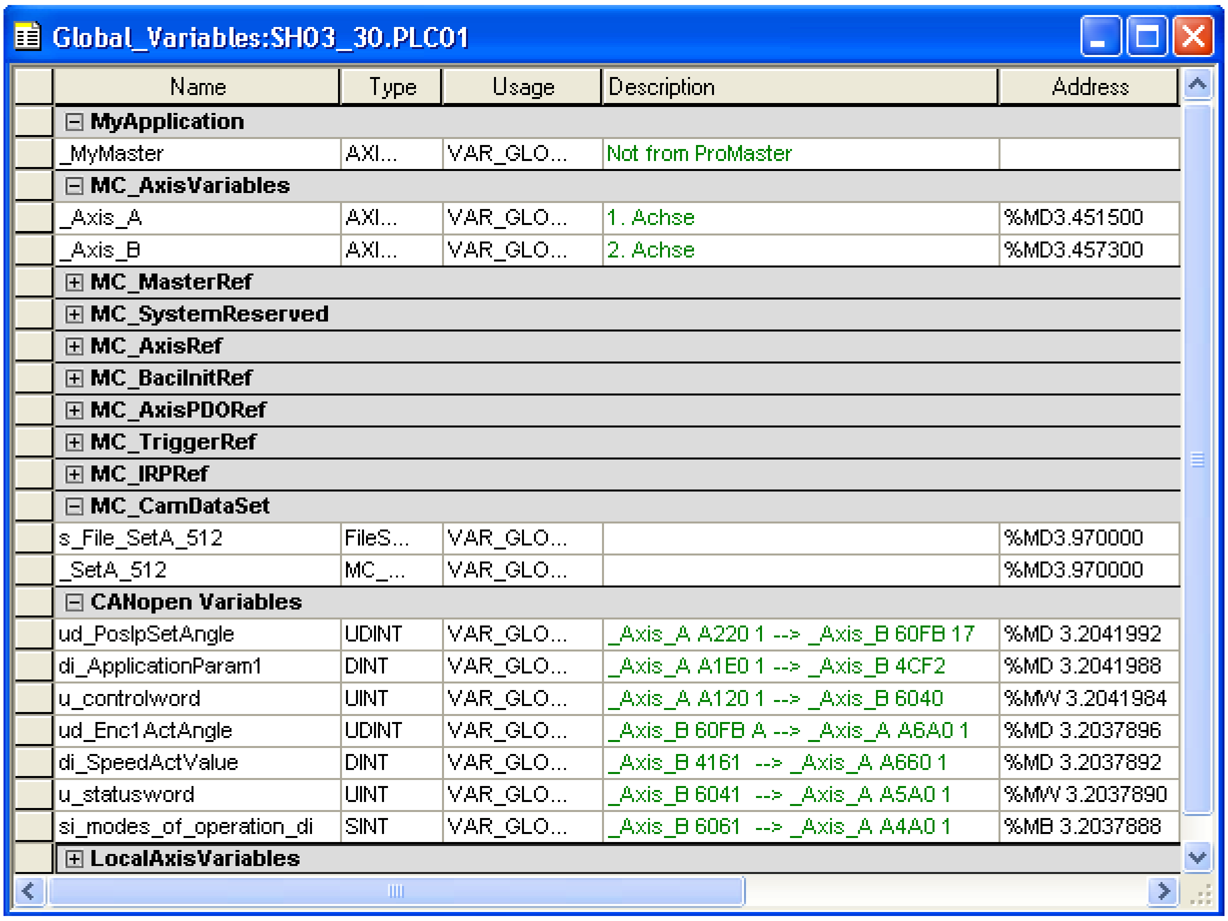
For information on how to program a Motion Control application in the IEC project in ProProg wt III, please see the Motion Control application manual and the ProProg wt III Online Help System.

ProProg wt III with our IEC project "Example_BM4_O_CAN04_MA_2.mwt" is opened via the context menu on _Axis_A "Configuration Data\PLC\IEC - Programming". You can edit the IEC project in the accustomed manner.

ProMaster has written

- the Motion Control axis variable (in the section MC_AxisVariables)
- the cam-disk data record (in the section MC_CamDataSet)
- the network variables (in the section CANopenVariables)
  (standard network variables for Motion Control and additional network variables for CANopen; see ▷4.3.9.2 Data exchange ◁ from page 93 onward)
- the local axis variables (in the section LocalAxisVariables)

to the global variable worksheet "Global_Variables".


Global_Variables:SH03_30.PLC01

Name	Type	Usage	Description	Address
⊟ MyApplication				
_MyMaster	AXI...	VAR_GLO...	Not from ProMaster	
⊟ MC_AxisVariables				
_Axis_A	AXI...	VAR_GLO...	1. Achse	%MD3.451500
_Axis_B	AXI...	VAR_GLO...	2. Achse	%MD3.457300
⊞ MC_MasterRef				
⊞ MC_SystemReserved				
⊞ MC_AxisRef				
⊞ MC_BacInitRef				
⊞ MC_AxisPDORef				
⊞ MC_TriggerRef				
⊞ MC_IRPRef				
⊟ MC_CamDataSet				
s_File_SetA_512	FileS...	VAR_GLO...		%MD3.970000
_SetA_512	MC_...	VAR_GLO...		%MD3.970000
⊟ CANopen Variables				
ud_PosIpSetAngle	UDINT	VAR_GLO...	_Axis_A A220 1 --> _Axis_B 60FB 17	%MD 3.2041992
di_ApplicationParam1	DINT	VAR_GLO...	_Axis_A A1E0 1 --> _Axis_B 4CF2	%MD 3.2041988
u_controlword	UINT	VAR_GLO...	_Axis_A A120 1 --> _Axis_B 6040	%MW 3.2041984
ud_Enc1ActAngle	UDINT	VAR_GLO...	_Axis_B 60FB A --> _Axis_A A6A0 1	%MD 3.2037896
di_SpeedActValue	DINT	VAR_GLO...	_Axis_B 4161 --> _Axis_A A660 1	%MD 3.2037892
u_statusword	UINT	VAR_GLO...	_Axis_B 6041 --> _Axis_A A5A0 1	%MW 3.2037890
si_modes_of_operation_di	SINT	VAR_GLO...	_Axis_B 6061 --> _Axis_A A4A0 1	%MB 3.2037888
⊞ LocalAxisVariables				

Figure 62: ProProg wt III - Global variable worksheet with data from ProMaster

In particular, the following must be observed:

The device name of the device on the CANopen bus in ProMaster is simultaneously the name of the axis in the IEC project in ProProg wt III. That means that if you have connected an existing application (or a template) from ProProg wt III to the ProMaster project, the names of the devices in ProMaster and in the IEC project may not be identical. In this case, there are two possible solutions:

1. ProMaster changes the axis names to the device names in ProMaster in ProProg wt III in the global variable worksheet "Global_Variables" in the section "MC_AxisVariables", and the user changes the axis names in the POEs to the device names in ProMaster (via the ProProg wt III "Global Replacement" function).
2. You change the device names in ProMaster to the axis names in ProProg wt III.

In our example we have given the local axis the device name _Axis_A and the CANopen slave the device name _Axis_B in ProMaster. These are also the axis names from our Motion Control template, which we have used for our IEC project.

The same also applies to the cam-data record name in the IEC project in ProProg wt III. That means that if you connect an existing application (or a template) from ProProg wt III to the ProMaster project, the cam data-record names in ProMaster and in the IEC project may not be identical. In this case, the following applies:

ProMaster changes the cam data set names to the cam data set names in ProMaster in the "MC_CamDataSet" section of the "Global_Variables" variable worksheet in ProProg wt III, and the user changes the cam data set names in the POEs to the cam data set names in ProMaster (via the ProProg wt III function "Global Replace").

4.3.9.2 Data exchange

A global variable is required in each case for exchanging data between

- the function blocks in the IEC project on the b maXX drive PLC (BM4-O-PLC-0x) and the local axis and
- the function blocks in the IEC project on the b maXX drive PLC (BM4-O-PLC-0x), the CANopen master option module (BM4-O-ETH-02 / BM4-O-CAN-04) and (via the CANopen field bus) the CANopen slave.

These variables are also called axis variables and have an axis name in the IEC project.

Our axis variable for the local axis has the axis name _Axis_A.

Our axis variable for the CANopen slave has the axis name _Axis_B.

We have given our local axis the device name _Axis_A and our CANopen slave the (same-sounding) name _Axis_B in ProMaster, which is why we do not need to make any adjustments here.

The same applies to the variable for the cam data that has the cam data-record name _SetA_512 in the IEC project and ProMaster.

The axis variable and the cam data variable are connected to the Motion Control function blocks in the IEC project. The machine function is programmed with the Motion Control function blocks.

Information on using the Motion Control function blocks is contained in the Motion Control application manual.

When using Motion Control, the communication is carried out via the axis variable. ProMaster creates network variables in the IEC project. These are divided into the standard network variables for Motion Control and, if additional objects have been created during the CANopen slave configuration, the additional network variables for CANopen.

The following must always be observed here:

The standard network variables for Motion Control may be read by the user, but not written.

These are the standard Motion Control network variables for the set values:

"u_controlword.."	for the axis control word
"ud_PosIpSetAngel.."	for the synchronized position angle set value of the axis

These are the standard Motion Control network variables for the actual values:

"u_statusword.."	for the axis status word
"ud_Enc1ActAngle.."	for the synchronized position angle actual value of the axis
"si_modes_of_operation_display.."	for the axis operating mode

The additional network variables for CANopen are written (set values) and read (actual values) by the user.

If during the configuration of the CANopen slave communication you have created the additional network variables (and their link to CANopen slaves) suggested in the sections ▹PD Receive tab ◃ from page 69 onward and ▹PD Transmit tab ◃ from page 71 onward, the additional network variable for the set values is:

"di_ApplicationParam1.."	for Application parameter 1 of the axis

The additional network variable for the actual values is:

"di_SpeedActValue.."	for the actual speed value of the axis

NOTE

The standard network variables for Motion Control may be read by the user, but not written.

The additional network variables for CANopen are written (reference values) and read (actual values) by the user.

NOTE

You use the CANopen master module together with Motion Control.

No Motion Control axis (from the CANopen standpoint it is a network node) may be operated via the function blocks from the CANopen_PLC01_30bd00 library (or higher) under ProProg wt III together with Motion Control.

This means a network node is either operated via the function blocks from the CANopen_PLC01_30bd00 library (or higher) under ProProg wt III or by Motion Control.

NOTE

When using Motion Control, the initialization of the CANopen master module is assumed by Motion Control. The function block COM405_KERNEL_INIT from the CANopen_PLC01_30bd00 library (or higher) under ProProg wt III may not be used.

NOTE

When using Motion Control, the evaluation of the network states and the network node states is assumed by Motion Control.

The COM405_NETWORK_CONTROL function block from the CANopen_PLC01_30bd00 library (or higher) may not be used to control the network states and the network node states.

The COM405_NETWORK_CONTROL function block from the CANopen_PLC01_30bd00 library (or higher) may be used to evaluate the network states and the network node states.

NOTE

When using Motion Control, the network is started up automatically by Motion Control.

NOTE

The additional network variables for set values must be written in the Motion Control event task.

The additional network variables for set values must be written at every call of the motion control event task.

Exception: The PDO transmission type of the network variables is set asynchronous. Network variables of CANopen bus coupler IOs, because their transmission type is asynchron.

4.4 Programming data exchange with PROPROG wt II and CANop405_PLC01_20bd03 library

The data exchange with PROPROG wt II and the CANop405_PLC01_20bd03 library (or higher) can be programmed with

- the CANopen master module, software version **< 01.20** (i.e. < BM4-O-ETH-02/CAN-04-01-00-001-**005**)
- the CANopen master option module, software version ≥ **01.20** (i.e. ≥ BM4-O-ETH-02/CAN-04-01-00-001-**005**).

4.4.1 Overview

In the next few chapters, we will explain in detail the various options for using a CANopen-Master option module on a CANopen network. We will show you both the basic mechanisms that are defined in profile DS 301 as well as the related use of function blocks from library CANop405_PLC01_20bd00 (or higher) according to profile DS 405. The representations will be accompanied by a step by step sample project for PROPROG wt II. In this sample project, we will program a simple network containing a CANopen-Master option module for the b maXX drive PLC and a CANopen-Slave option module for the b maXX controller from the device series of the b maXX drives. For information on commissioning the CANopen-Slave option module, refer to the appropriate Operating Instructions. Observe any b maXX controller settings that may be necessary to make possible communication with the CANopen-Slave option module.

NOTE

The sample project is intended to explain in more detail the CANopen functions that are possible with the CANopen-Master option module and to make them easier to understand. You should not think of it as a fully functional application. The values of the data that is written to the CANopen-Slave in the examples have no functional meaning. For this reason, you must ensure that a drive that may be connected is not ready for operation!

The CANopen-Master option module and the function blocks obviously offer many more options than we can describe in the scope of this manual. it is also not possible to go into every detail of the function blocks. For more details about the function blocks, refer to the online help system in PROPROG wt II.

4.4.2 Steps to be carried out

To be able to use the CANopen-Master option module for data exchange on a CANopen network, you must carry out the following steps:

1 physically commission the CANopen network (see the Operating Instructions of option module BM4-O-ETH-02 / BM4-O-CAN-04 and the Operating Instructions of the CANopen network nodes)

2 create a PROPROG wt II project using library CANop405_PLC01_20bd00 (or higher)

3 implement function blocks for CANopen communication, compile and send the project

You can of course start with step 2 and carry out physical commissioning later.

4.4.3 Commissioning the CANopen network

For details on commissioning the CANopen network, refer to the Operating Instructions for option module BM4-O-ETH-02 / BM4-O-CAN-04 and the Operating Instructions for the other CANopen network nodes.

After physically commissioning the network and switching in the power supply for the b maXX 4400 unit, the CANopen-Master option module is ready for operation after approximately 5 seconds. The CANopen LEDs on the front of the option module indicate this. Two combinations are possible:

H5 (green):
H6 (red):
	Flashes
	Off
		CANopen:
		The module waits for the initialization by an application program on the b maXX drive PLC

H5 (green):
H6 (red):
	OFF
	ON
		CANopen: The CAN is in the bus OFF status

Flashes:ton = 200 ms, toff = 200 ms

All the other status conditions of LEDs H5 and H6 indicate errors. For more information, refer to the Operating Instructions for the CANopen-Master option module BM4-O-ETH-02 / BM4-O-CAN-04.

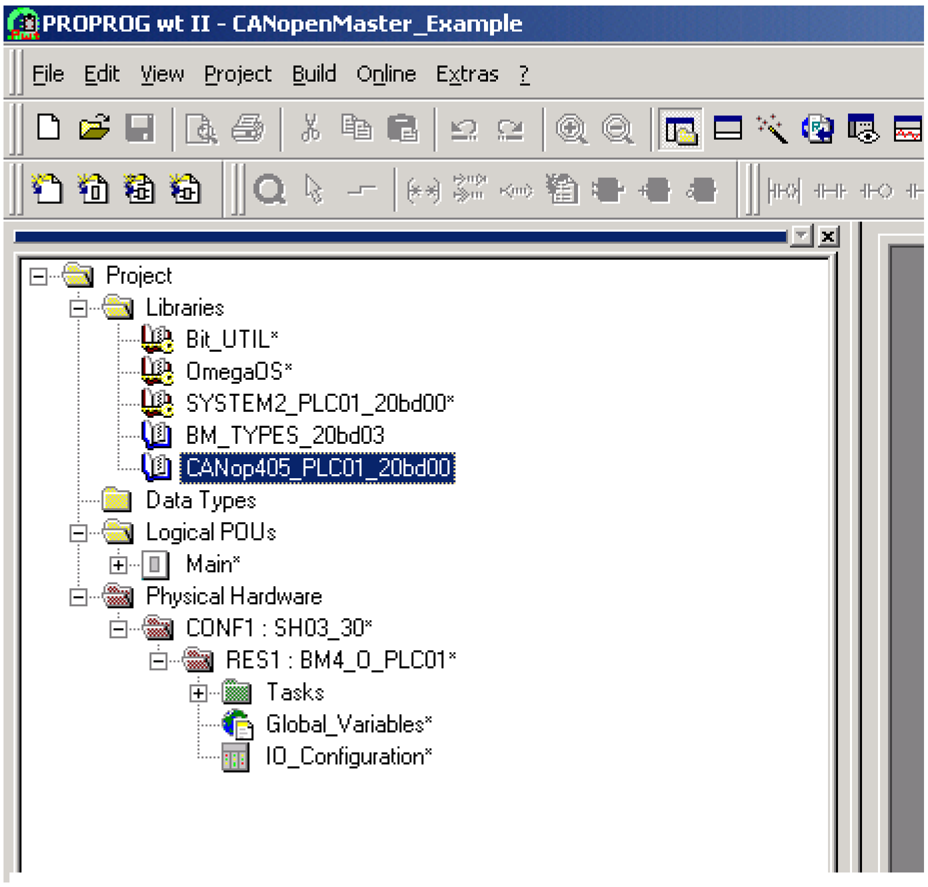
4.4.4 Creating a project and linking library CANop405_PLC01_20bd00

4.4.4.1 Procedure when creating a project

To be able to use the CANopen-Master option module using CANopen function blocks according to profile DSP 405, you need a PROPROG wt II project for the b maXX PLC01. If you have not yet done so, create a separate project for your application using template *BM4_O_PLC01*. For this, you need PROPROG wt II Version 3.0 Build 261 or higher. You can find the version number of PROPROG wt II on the jewel case of the PROPROG wt II installation CD or in the program itself under menu item *? \ About*. Also check whether library BM_TYPES_20bd03 (or higher) is present in your PROPROG wt II project. If this is not the case, link this library in your project. It contains important data types for CANopen. Then link library CANop405_PLC01_20bd00 in your project.

4.4.4.2 Example: Creating project "CANopenMaster_Example"

We created the sample project called "CANopenMaster_Example" using template BM4_O_PLC01 and linked library CANop405_PLC01_20bd00.


PROPROG wt II - CANopenMaster_Example
File Edit View Project Build Online Extras ?

Project
Libraries
Bit_UTIL*
OmegaOS*
SYSTEM2_PLC01_20bd00*
BM_TYPES_20bd03
CANop405_PLC01_20bd00
Data Types
Logical POUs
Main*
Physical Hardware
CONF1 : SH03_30*
RES1 : BM4_O_PLC01*
Tasks
Global_Variables*
IO_Configuration*

Figure 63: Example: Creating project "CANopenMaster_Example"

4.4.5 Creating a global variable for data exchange

A global variable is needed for data exchange between option module BM4-O-ETH-02 / BM4-O-CAN-04 and the function blocks. The variable has no further meaning for users. This global variable has already been created in your project if you created the project using template BM4_O_PLC01. This global variable is connected to the function blocks for CANopen at input/output _CANop405_CTRL.

Depending on the option module's slot (G to M), you have available global variables _CANop405Ma_Ctrl_Slot_G to _CANop405Ma_Ctrl_Slot_M. You can also find them in the "Global_Variables" worksheet:

```
(* Option module CANopen-Master (BM4-O-ETH-02) *)
   _CANop405Ma_Ctrl_Slot_G      AT %MB3.2016384   :   CANop405_PLC_BMSTRUCT;
   _CANop405Ma_Ctrl_Slot_H      AT %MB3.3016384   :   CANop405_PLC_BMSTRUCT;
   _CANop405Ma_Ctrl_Slot_J      AT %MB3.4016384   :   CANop405_PLC_BMSTRUCT;
   _CANop405Ma_Ctrl_Slot_K      AT %MB3.5016384   :   CANop405_PLC_BMSTRUCT;
   _CANop405Ma_Ctrl_Slot_L      AT %MB3.6016384   :   CANop405_PLC_BMSTRUCT;
   _CANop405Ma_Ctrl_Slot_M      AT %MB3.7016384   :   CANop405_PLC_BMSTRUCT;
```

Figure 64: Global variables for the CANopen-Master option module depending on the slot

If the global variable that is necessary for the option module's slot is not present in the project, create – in dependence on the slot (G to M) – the global variable _CANop405Ma_Ctrl_Slot_G (to _CANop405Ma_Ctrl_Slot_M) of data type CANop405_PLC_BMSTRUCT. This variable must be declared as a global one and must be connected to the base address to the CANopen-Master communication of option module BM4-O-ETH-02 / BM4-O-CAN-04. The base address depends on the slot:

Slot	Base address for CANopen-Master communication
G	%MB3.2016384
H	%MB3.3016384
J	%MB3.4016384
K	%MB3.5016384
L	%MB3.6016384
M	%MB3.7016384

4.4.6 Initializing the CANopen-Master option module

4.4.6.1 Procedure for initializing the CANopen-Master option module

You initialize the CANopen-Master option module using FB CANop405_INIT. To use this function block, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cold boot and warm restart task.
- Place FB CANop405_INIT in this POU.
- Interconnect the function block with variables of the correct data type.
  At input us_BAUDRATE, set the baud rate for the CANopen network. The following values are possible:

us_BAUDRATE	Baud rate of the network
0	1 Mbps
1	800 kbps
2	500 kbps
3	250 kbps
4	125 kbps
5	100 kbps
6	50 kbps
7	20 kbps
8	10 kbps

4.4.6.2 Example: Initializing the CANopen-Master option module

Interconnection of the FB CANop405_INIT for initialization at 500 kbps can look like this:

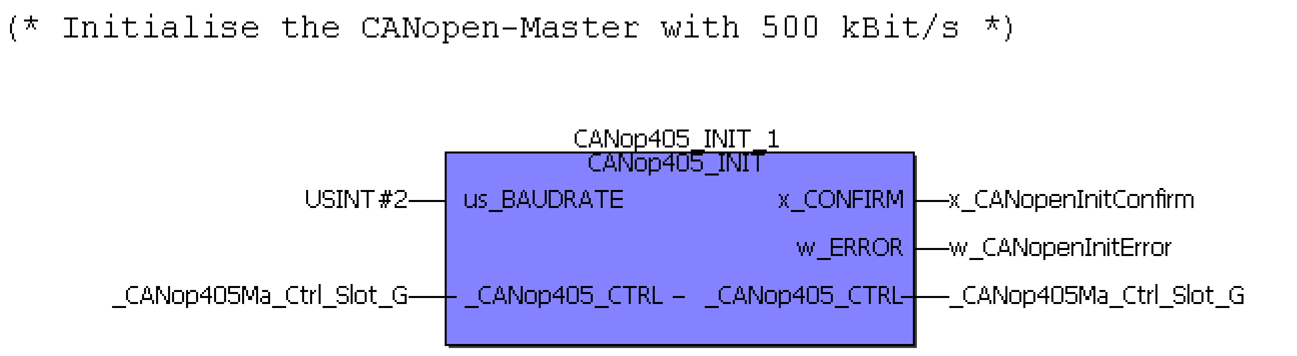


Figure 65: Initializing the CANopen-Master with FB CANop405_INIT

- If they are not yet present in your project, create tasks for cold booting and warm re-starting the PLC. Link the created POU for initializing the option module in both tasks.
- Compile the project and load it as a boot project on the PLC. Switch the b maXX 4400 unit off and back on again.

FB CANop405_INIT signals successful initialization with x_CONFIRM = 1 and w_ERROR = 16#0.

Example:

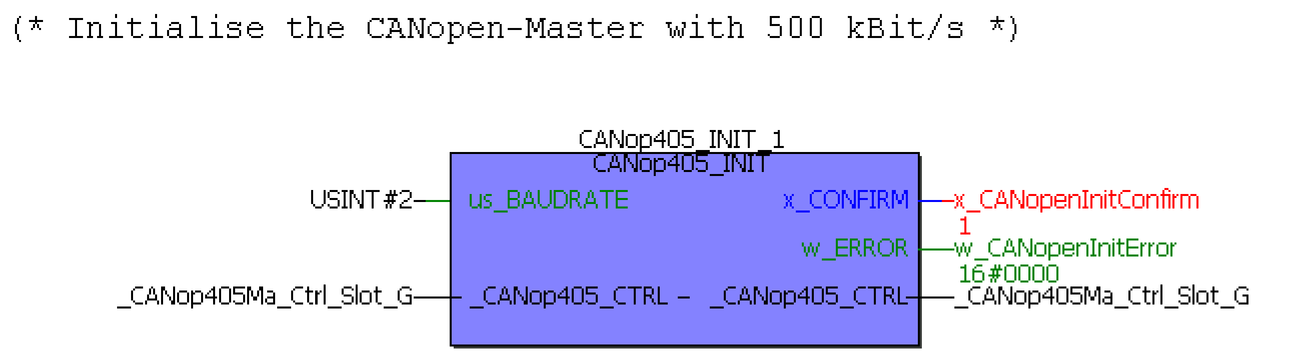


Figure 66: Initializing the CANopen-Master with FB CANop405_INIT - online

The LEDs on the CANopen socket on the option module indicate successful initialization:

H5 (green): ON H6 (red): OFF	CANopen: The CANopen-Master option module is initialized and ready for data transfer

Set the correct baud rate on the other CANopen nodes too. For information on this topic, refer to the relevant documentation.

4.4.7 Starting the individual network node NMT

4.4.7.1 Definition according to the CANopen specification

Using NMT (Network Management), a CANopen-Master controls the communication status conditions of network nodes. A device can have the following communication status conditions:

- Initialization:
- PRE-OPERATIONAL
- STOPPED
- OPERATIONAL

The behavior of individual network nodes is described by the following status transitions:

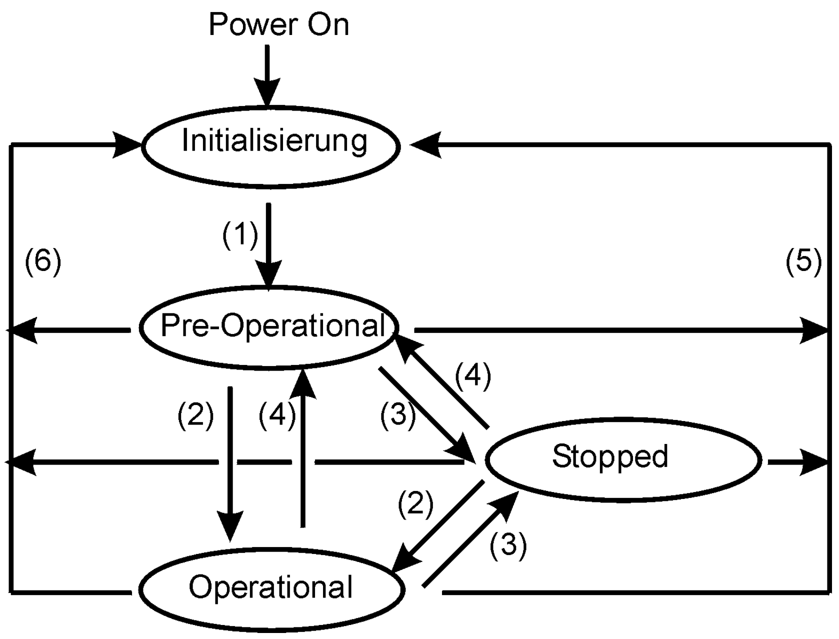


Figure 67: Status transitions

After INITIALIZATION (triggered by switching on the network node), the PRE-OPERATIONAL status is reached automatically. If a network node is in this status, you can configure it by means of SDOs. Data exchange via PDOs is not possible.

In the STOPPED status, only Node Guarding is possible. It is not possible to send or receive either SDOs or PDOs. There are six possible transitions between the individual status conditions. The transition is triggered either automatically (Power On) or by an internal command from the master or node (e.g. in the case of a fault).

(1) Automatic transition from INITIALIZATION to PRE-OPERATIONAL

(2) Start Remote Node

(3) Stop Remote Node

(4) Enter Pre-Operational State

(5) Reset Node

(6) ResetCommunication

The CANopen-Master option module itself has no status conditions and status transitions according to the CANopen definition.

The CANopen-Master can trigger status transitions (2) to (6) by means of a message frame. This message frame is not confirmed, i.e. the CANopen-Master has no information telling it whether a network node has received the message frame and whether it has carried out the change of status. On the CANopen-Master, it is only possible to read out the status of a network node via Node Guarding (see the chapter entitled ▷4.4.11 Monitoring network nodes by Node Guarding ◁ from page 140 onward).

NOTE

Note the effects of the individual status transitions! A Reset Node can trigger a resetting of the entire network node, which can have an undesirable effect on your application. Refer to the descriptions of the individual network nodes to get more information about possible effects.

4.4.7.2 Issuing NMT commands

The CANopen-Master option module issues NMT commands by means of FB CANop405_NMT. To use this function block, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cyclical task.
- Place FB CANop405_NMT in this POU.
- Interconnect the function block with variables of the correct data type. At input us_DEVICE, state the number of the network node that the NMT command is to evaluate. A value of USINT#0 means that all the network nodes evaluate the command. You state the command for the status transitions at us_TRANSITION_STATE. The individual values are assigned to the commands as follows:

us_TRANSITION_STATE	Command
1	Start Remote Node
2	Stop Remote Node
3	Enter Pre-Operational
4	Reset Node
5	Reset Communication

- If it is not yet present in your project, create a cyclical task with medium to low priority. Link the created POU to FB CANop405_NMT in this task.
- Compile the project and load it as a (boot) project on the PLC.
- Start the project.

The NMT command is issued on x_ENABLE = 1. FB CANop405_NMT signals successful execution with x_CONFIRM = 1 and w_ERROR = 16#0.

4.4.7.3 Example: Network Management using NMT

To be able to use FB CANop405_NMT, we create a POU and place the function block in it. We assign the created POU to a cyclical task with a 200 ms call interval. We want to switch all the network node to the OPERATIONAL status, which means that us_DEVICE must have a value of 0 and us_TRANSITION_STATE must have a value of 1. After interconnecting the function block, compile the project and send it as a project to the PLC. Start the project. x_ENABLE = 1 activates the OPERATIONAL status on all the network nodes:

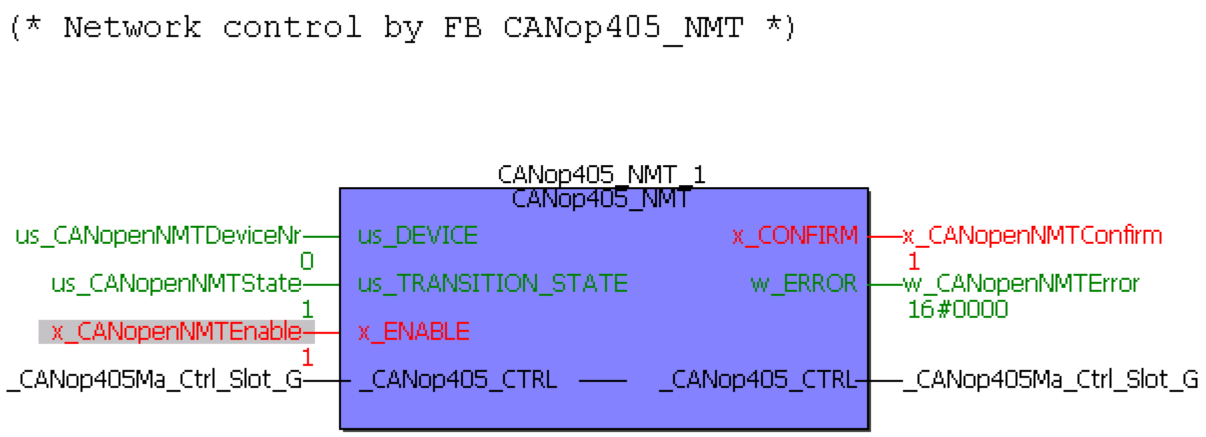


Figure 68: Network Management using FB CANop405_NMT

4.4.8 Exchanging requirements data with SDO

4.4.8.1 Definition according to the CANopen specification

a) Objects of a network node

When accessing via SDO (Service Data Object), the system always read or write accesses the objects of a network node. Depending on the profile, the objects can represent configuration data, device functions or parameters, for example. A network node has an object directory that contains all the device's objects. It is possible to directly access this object directory via SDO. To find out the objects that a network node really supports, refer to its documentation, since not all the objects that are defined in a profile must be supported. In addition, manufacturer-specific objects may also be defined for a device. The three most important areas in a device's object directory are as follows:

- Objects of the communication profile (especially DS 301)
- Manufacturer-specific objects
- Objects of the device profile

Objects are always addressed using a 16-bit index and an 8-bit sub-index. In addition to the index and the sub-index, the object directory also contains information about an object's name, data type, attribute, default value and meaning.

Examples of object entries include:

Object according to DS 301:

Index	Subindex	Name	Data type	Attribute	Default value	Meaning
1000 h	00	Device Type	Unsigned 32	ro	402	Supported device profile

This entry in the object directory of option module BM4-O-CAN-03 (CANopen-Slave for b maXX controller) indicates that the object with index 1000h and sub-index 0h contains the number of the supported device profile. Entry 402 means that device profile DSP 402 (drives) is supported. The value is read only (ro) and is transferred as a 32-bit value. In the case of an I/O terminal module, the same object would have a default value of 401 for device profile DS 401 (I/O modules).

Object according to DS 402:

Index	Subindex	Name	Data type	Attribute	Default value	Meaning
6040 h	00	Control-word	Unsigned 16	rw	-	Control word

This entry in the object directory of option module BM4-O-CAN-03 (CANopen-Slave for b maXX controller) indicates that the object with index 6040h and sub-index 0300h represents the control word (parameter 300). The value can be read and written (rw) and is transferred as a 16-bit value. Since this is a device profile-specific object, it has a different meaning with an I/O terminal module, namely the strainer for digital inputs.

Manufacturer-specific object:

Index	Subindex	Name	Data type	Attribute	Default value	Meaning
41BAh	00	SVG set value 1	Signed 16	rw	16384	Specified value generator specified value 1

This entry in the object directory of option module BM4-O-CAN-03 (CANopen-Slave for b maXX controller) indicates that the object with index 41BAh and sub-index 0h represents specified value 1 of the specified value generator (parameter 442). The value can be read and written (rw) and is transferred as a signed 16-bit value.

NOTE

The CANopen-Master option module itself has neither communication profile-specific objects nor device profile-specific ones. Using CANopen-Master option module and the function blocks from library CANop405_PLC01_20bd00, you can directly access individual SDO and PDO message frames.

b) Sequence of SDO communication

SDO communication corresponds to the client/server- communication model, i.e. the CANopen-Master option module is the client and sends a message frame telling a network node to take data or to send it. The network node acts as the server: it takes the data and confirms this with a message frame or by sending the requested data. In the other direction, it is not possible for the CANopen-Master option module to respond as the server to a client's request. Using the CANopen-Master, it is possible to carry out expedited SDO transfer according to DS 301 with up to 4 bytes of data per job. Segmented and block SDO transfer according to DS 301 are not supported.

4.4.8.2 Writing objects using SDO

The system writes to objects via SDO by means of function blocks CANop405_SDO1_WRITE to CANop405_SDO8_WRITE. Using these function blocks, it is possible to start up to eight SDO jobs at the same time. One FB is available for each SDO job.

NOTE

You must not multi-instantiate function blocks CANop405_SDO1_WRITE to CANop405_SDO8_WRITE.

To use function blocks CANop405_SDO1_WRITE to CANop405_SDO8_WRITE, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cyclical task. You can also use the POU with FB CANop405_NMT.
- Place one or more of FBs CANop405_SDO1_WRITE to CANop405_SDO8_WRITE in this POU as required.
- Interconnect the function blocks with variables of the correct data type and assign them the desired values. At input us_DEVICE, state the number of the network node to which you want to write an object. w_INDEX and b_SUBINDEX are the addresses of the object; us_DATALENGTH indicates the size of the object in bytes and you state the data that is to be written at d_DATA.
- If it is not yet present in your project, create a cyclical task with medium to low priority. Link the created POU in this task. If you are using the POU with FB CANop405_NMT, you do not need to create another task.
- Compile the project and load it as the (boot) project on the PLC. Start the PLC.

On x_ENABLE = 1, the system sends the object via SDO. FBs CANop405_SDO1_WRITE to CANop405_SDO8_WRITE indicate that successful execution with x_CONFIRM = 1 and w_ERROR = 16#0. Since the other network node confirms writing of objects via SDO, it may well be that the system issues an error message in w_ERROR and ud_ERRORINFO. One reason can be that the object is read-only, for example. For details of error messages, refer to the online documentation for the function blocks.

NOTE

You must not use the same node number (us_DEVICE) for FBs CANop405_SDOx_READ (see the next chapter) and FBs CANop405_SDOx_WRITE (where x = 1 to 8) at the same time. FBs CANop405_SDOx_READ and CANop405_SDOx_WRITE must not be active at the same time.

4.4.8.3 Example: Writing an object via SDO

We want to write parameter 1172 "ramp function generator ramp-up time" of the b maXX controller with 56 s. In the CANopen-Slave option module's object directory, parameter 1172 has the following object data:

Index	Subindex	Name	Data type	Attribute	Default value	Meaning
604Fh	00 h	vl_ramp_ function_ time	Unsigned 32	rw	0	Ramp function, generator ramp-up time

You set the CANopen-Slave option module to module number 5 using its DIP switch. We use FB CANop405_SDO1_WRITE. We implement FB CANop405_SDO1_WRITE in the POU using function block CANop405_NMT. After interconnecting the function block, you must compile the project and send it as a project to the PLC. Start the project. We start data transfer via SDO with x_ENABLE = 1.

In the online representation, the function block should look like this after sending the SDO:

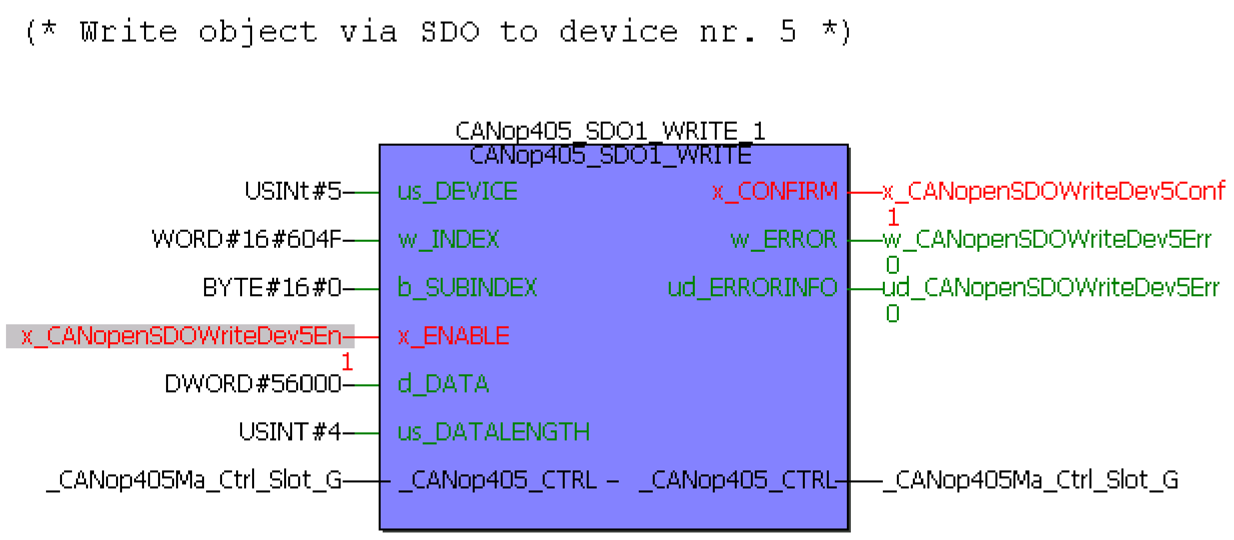


Figure 69: Writing the ramp-up time using FB CANop405_SDO1_WRITE

When you check parameter 1172 "ramp function generator ramp-up time" using WinBASS II, it should have a value of 56 s:

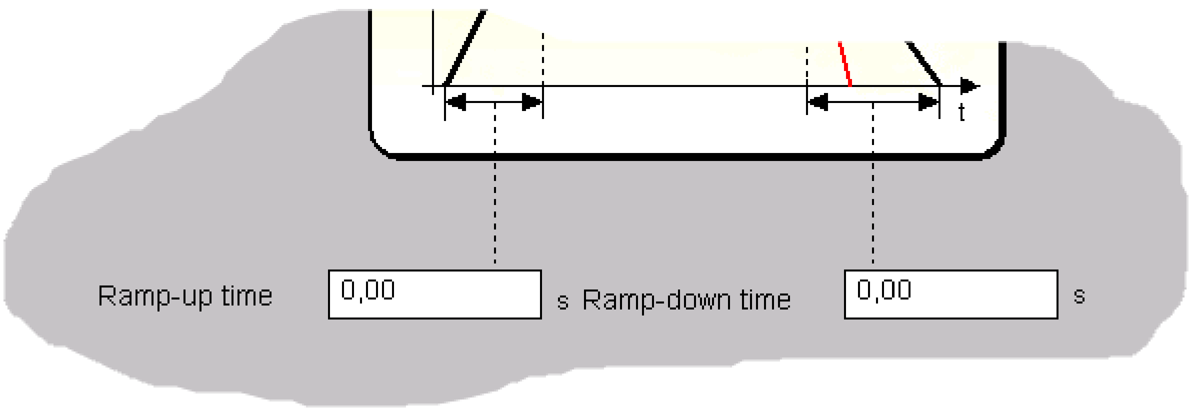


Figure 70: Ramp-up time in WinBASS II

4.4.8.4 Reading objects using SDO

The system reads from objects via SDO by means of function blocks CANop405_SDO1_READ to CANop405_SDO8_READ. Using these function blocks, it is possible to start up to eight SDO jobs at the same time. One FB is available for each SDO job.

NOTE

You must not multi-instantiate function blocks CANop405_SDO1_READ to CANop405_SDO8_READ.

To use function blocks CANop405_SDO1_READ to CANop405_SDO8_READ, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cyclical task. You can also use the POU with FB CANop405_NMT or the function blocks for writing objects via SDO.
- Place one or more of FBs CANop405_SDO1_READ to CANop405_SDO8_READ in this POU as required.
- Interconnect the function blocks with variables of the correct data type and assign them the desired values. At input us_DEVICE, state the number of the network node from which you want to read an object. w_INDEX and b_SUBINDEX are the addresses of the object. At us_DATALENGTH the system displays the size of the read object in bytes and displays the read data at d_DATA.
- If it is not yet present in your project, create a cyclical task with medium to low priority. Link the created POU in this task.
- Compile the project and load it as the (boot) project on the PLC. Start the PLC.

On x_ENABLE = 1, the slave queries the object via SDO. FBs CANop405_SDO1_READ to CANop405_SDO8_READ indicate that successful execution with x_CONFIRM = 1 and w_ERROR = 16#0. Since the slave confirms reading of objects via SDO, it may well be that the system issues an error message in w_ERROR and ud_ERRORINFO. One reason may be that an object is not present, for example. For details of error messages, refer to the online documentation for the function blocks.

NOTE

You must not use the same node number (us_DEVICE) for FBs CANop405_SDOx_READ and FBs CANop405_SDOx_WRITE (see the chapter entitled ▹4.4.8.2 Writing objects using SDO ◃ from page 105 onward) (where x = 1 to 8) at the same time. FBs CANop405_SDOx_READ and CANop405_SDOx_WRITE must not be active at the same time.

4.4.8.5 Example: Reading an object via SDO

We now want to read back parameter 1172 "ramp function generator ramp-up time" of the b maXX controller that we just wrote. Assuming that the parameter has not been overwritten and you did not switch off the device, its value must be 56 s. In the CANopen-Slave option module's object directory, parameter 1172 has the following object data:

Index	Subindex	Name	Data type	Attribute	Default value	Meaning
604Fh	00 h	vl_ramp_function_time	Unsigned 32	rw	0	Ramp function generator ramp-up time

The CANopen-Slave option module is set to module number 5. We will implement FB CANop405_SDO1_READ in the same POU as function block CANop405_SDO1_WRITE. After interconnecting the function block, compile the project and send it as a project to the PLC. Start the project. We start data transfer via SDO with x_ENABLE = 1. In the online representation, the function block should look like this after sending the SDO:

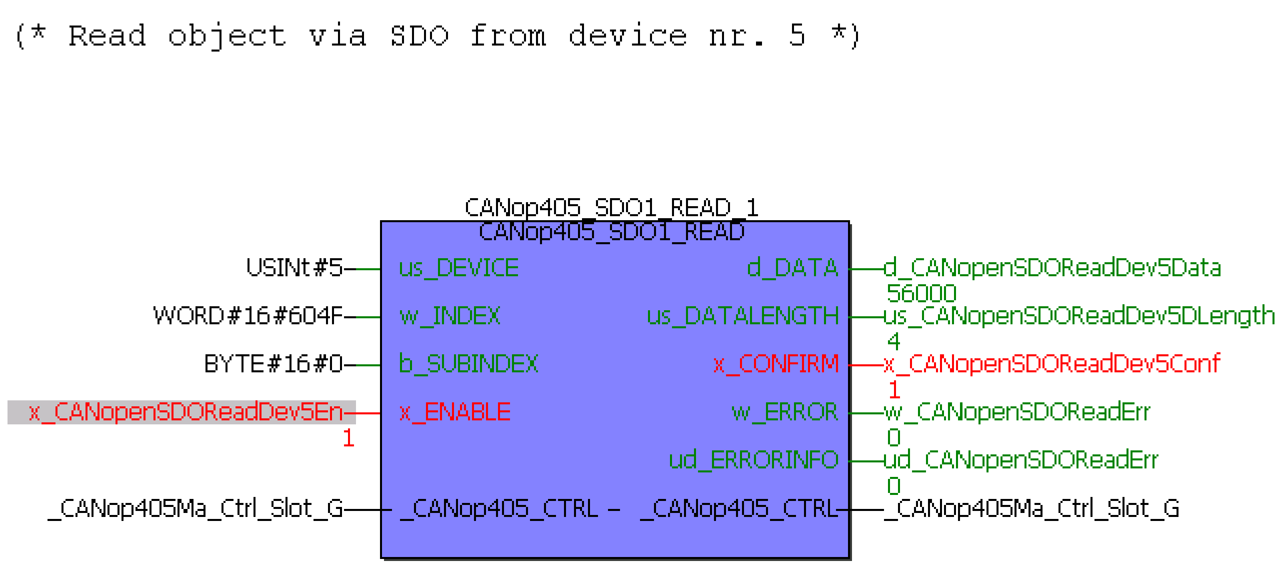


Figure 71: Reading the ramp-up time using FB CANop405_SDO1_READ

4.4.9 Exchanging process data with PDOs

4.4.9.1 Definition according to the CANopen specification.

PDO communication (PDO = Process Data Object) corresponds to the Consumer/Producer communications model, i.e. a network node generates a PDO message frame that one or more other network node can process. They do not confirm receiving the message frame. The CANopen-Master option module can also generate and process PDO message frames. You must configure the actual transfer. This means, in particular, that you must specify when the data in a PDO is to be taken or the instant at which the data is to be entered in a PDO. By contrast with data exchange via SDOs, the system does not directly access a network node's object with PDOs. Rather, the system informs a network node which objects, with the index, sub-index and data length are contained in a PDO message frame. The system then removes these objects from a PDO when writing to it and enters them in the PDO when reading. Up to 8 bytes of data per message frame are available.

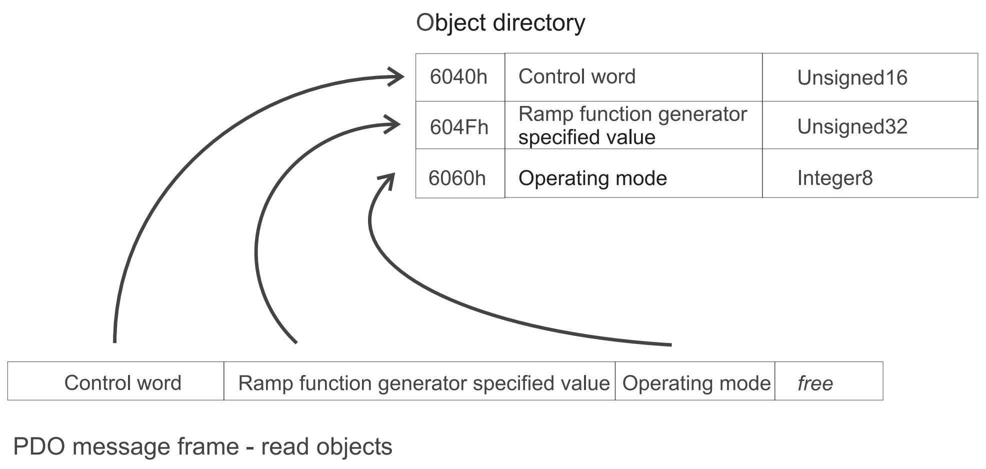


Figure 72: Reading objects from a PDO message frame

The PDO message frame itself contains only user data; there is no more information about the object index or sub-index, since the system already knows it due to the configuration that you carried out previously. This is also known as mapping the objects to a PDO.

There are three phases of data exchange via PDOs:

1 Configuring the transfer behavior of a PDO on each network node

2 Configuring PDO mapping on each network node

3 Cyclical data exchange via PDOs

The configuration of the PDOs is stored in a device's communication profile-specific objects. The system reads and writes these objects via SDOs (see the chapter entitled ▹4.4.8 Exchanging requirements data with SDO ◃ from page 103 onward). You must only configure PDOs in the PRE-OPERATIONAL status of a network node. Below, we will explain the details of configuration objects.

4.4.9.2 Configuring the transfer properties of a PDO

The communication profile-specific objects of a network node describe the properties for transferring a PDO. Below, we will describe the configuration object for the first receive PDO (1400h) and the first transmit PDO (1800h). To configure all other PDOs, you must use the next highest object index, i.e. the second receive PDO is configured via object 1401h, the third one via object 1402h, etc. For transmit PDOs, these are objects 1801h, 1802h, etc. A maximum of 512 receive PDOs and 512 transmit PDOs are possible. According to communication profile DS 301, the values of the objects have the following meanings:

Receive PDOs

Object	Meaning
1400h (- 15FF)	Communications parameter for the first receive PDO of a device.
Subindex 0	Number of the sub-index that contains the last valid entry
Subindex 1	COB-ID of the PDO and validity
Subindex 2	Transfer type of the PDO: synchronous or asynchronous
Subindex 3	*Not used*
Subindex 4	Entry for compatibility. Optional.
Subindex 5	*Not used*

Transmit PDOs

Object	Meaning
1800h (- 19FF)	Communications parameter for the first send PDO of a device.
Subindex 0	Number of the sub-index that contains the last valid entry
Subindex 1	COB-ID of the PDO and validity
Subindex 2	Transfer type of the PDO: synchronous, asynchronous, event-driven.
Subindex 3	Send delay time of the PDO. Optional
Subindex 4	Entry for compatibility. Optional.
Subindex 5	Timer value if PDO transfer is time-controlled. Optional.

Let us look at the meanings of the sub-indexes in more detail.

Object 1400h / 1800h Sub-index 0 - Number of valid sub-indexes

Subindex 0 indicates the number of the last valid subindex. The value is at least 2 and increases in accordance with the other entries (sub-indexes) that are to be supported. In general, the value is read only, i.e. it is only informative.

Object 1400h / 1800h Sub-index 1 - COB-ID of the PDO

Sub-index 1 contains various information. The value has data type Unsigned32 and is divided as follows:

Bit	Value / meaning
31 (MSB)	0: PDO exists and is valid 1: PDO does not exist or is not valid According to the predefined identifier assignment (see the chapter entitled ▹4.2.2.3 Data exchange and objects of the physical bus system ◃ from page 53 onward, only 4 PDOs are possible for writing and 4 PDOs for reading. If a device supports more than these four PDOs, you must generally assign the COB-IDs for the additional PDOs themselves (bits 10 - 0 in this subindex). For this reason, bit 31 of the additional PDOs usually has a value of 1 and users do not set it to 0 until they have assigned a COB-ID.
30	0: The PDO can be requested by means of a remote request 1: The PDO cannot be requested by means of a remote request
29	0: 11-bit CAN ID 1: 29-bit CAN ID (is not supported by the CANopen-Master option module)
28 - 11	Not supported by the CANopen-Master option module
10 - 0 (LSB)	COB-ID of the PDO. In the default settings of a device, the first four PDOs for writing and reading have the COB-IDs after the predefined identifier assignment (see the chapter entitled ▹4.2.2.3 Data exchange and objects of the physical bus system ◃ from page 53 onward).

If you do not need more than four PDOs for writing and four for reading, this value is generally just an informative one.

Object 1400h / 1800h Sub-index 2 - transfer type of the PDO

This value is very important, since it states when a PDO is transferred. The type of transfer is configured in Subindex 2. When you do this, the following assignment applies to send and receive PDOs:

Subindex 2 / type	Effect transmit PDO 1800h	Receive PDO 1400h
0 SYNCHRO-NIZED	Sending is carried out after each received SYNC message frame (see the chapter entitled ▹4.4.10 Synchronizing data exchange ◃ from page 135 onward)	The system applies the PDO with an appropriate COB-ID that was received before the last SYNC message frame
1 - 240 SYNCHRO-NIZED	The system sends after receiving the set number of SYNC message frames	The system applies the PDO with an appropriate COB-ID that was received before the last SYNC message frame
252 Remote	On receiving a SYNC message frame, the system updates the PDO. The system does not send until it receives a remote request.	*Not possible*
253 Remote	The system updates and sends after receiving a remote request.	*Not possible*

Subindex 2 / type	Effect transmit PDO 1800h	Receive PDO 1400h
254 Asynchronous	The system sends on a manufacturer-specific basis. Note: With this type, most devices transfer the PDO on a time (when a timer runs down). The time is set under subindex 5.[1)]	The system applies each PDO with an appropriate COB-ID when it is received
255 Asynchronous	The system sends in dependence on the supported device profile. Note: With this type, most devices transfer the PDO as soon as one of the mapped values has changed.[2)]	The system applies each PDO with an appropriate COB-ID when it is received

1) Time-controlled sending means that the send condition is linked to a timer. This timer is set for the first transmit PDO using sub-index 5 in object 1800h (16-bit). The resolution is milliseconds. The system starts the timer on the change of status to OPERATIONAL. The system sends the PDO cyclically at the interval set in the timer. The timer is cleared by writing a value of "0" to sub-index 5. There is no time-controlled receiving, rather the system takes all the PDOs with an appropriate COB-ID.

2) Event-driven sending means that the send condition is linked to a change in the values of one of the mapped objects. If, for example, three objects are mapped (the status word, the RPM speed actual value and the actual operating mode), the system sends the PDO as soon as at least one of the three values changes. If the values stay constant, no PDO is sent. This makes it possible to reduce bus loading (message frames are only transferred if they contain new information). Event-driven receiving means that the system takes all the PDOs with an appropriate COB-ID.

NOTE

Many devices only have specific transfer types implemented and can have different default settings. If you do not receive a PDO from a network node or a network node does not take a PDO, this is often due to the setting of the transfer type.

Object 1400h / 1800h Sub-index 3 - send inhibit time of the PDO

Using subindex 3, you can set a PDO inhibit time that does not extend the response time at the relative first value change; rather, it is active on the changes that directly follow this one. The inhibit time describes the minimum period of time that the system must wait between sending two of the same message frames. The value is meaningless for receive PDOs.

Object 1400h / 1800h Sub-index 4

The value of sub-index 4 has no relevant meaning.

Object 1400h / 1800h Sub-index 5 - Time for event-driven PDO transfer

The value of sub-index 5 states the time in ms for sending the PDO (see sub-index 2).

4.4.9.3 Mapping objects in a PDO

The second step in configuring data exchange via PDOs is specifying mapping of objects in a PDO. There are special, communication profile-specific objects for mapping too that contain a description of mapping for one network node. Below, we will describe the configuration object for the first receive PDO (1600h) and the first transmit PDO (1A00h). You must use the next highest object index to configure all the other PDOs, i.e. you configure mapping of the second receive PDO via object 1601h and the third one via object 1602h. The detailed meanings of the objects according to DS 301 are as follows:

Receive PDOs

Object	Meaning
1600h (- 17FF)	Mapping parameter for the first receive PDO of a device.
Subindex 0	Number of mapped objects
Subindex 1 - Subindex 64	Information about the mapped object

Transmit PDOs

Object	Meaning
1A00h (- 1BFF)	Mapping parameter for the first transmit PDO of a device.
Subindex 0	Number of mapped objects
Subindex 1 - Subindex 64	Information about the mapped object

Let us look at the meanings of the sub-indexes in more detail.

Object 1600h / 1A00h Sub-index 0 - Number of mapped objects

Subindex 0 indicates the number of mapped objects in this PDO. As soon as you want to change the mapping of a PDO, you must first set the number of mapped objects to zero. You can then change the mapping and then set the number of mapped objects in sub-index 0. Note that there can be device-dependent limitations on the number of objects to be mapped.

Object 1600h / 1A00h Sub-index 1 to 64 - Mapping information

Sub-indexes 1 to 64 indicate the mapping information of the objects in a PDO. Sub-index 1 for the first mapped object, sub-index 2 for the second mapped object, etc. The data type of these sub-indexes is Unsigned32. The contents have the following meaning:

Bits 31 - 16	Bits 15 - 8	Bits 7 - 0
Index of the mapped object	Sub-index of the mapped object	Number of bits of the mapped object

The system maps the objects successively in one PDO.

Note that not every object in a PDO can be mapped.

NOTE

Default mappings are defined in the device profiles. If the default mapping values are adequate for your purposes, you do not need to make any further settings here to the network node. The appendix contains an overview of the default mappings for devices according to profiles DS 401 and DSP 402.

4.4.9.4 Cyclical data exchange using PDO

After configuring the transfer properties and the mapping of the PDO, you can start actual data exchange via PDOs. The CANopen-Master option module sends PDOs that the network nodes respond to in accordance with their settings. In the same way, the CANopen-Master option module can receive PDOs that the network nodes send. Data can only be exchanged via PDOs in the OPERATIONAL status of the network node. In the OPERATIONAL status, you can no longer configure PDOs. With data exchange via PDOs too, you must always remember that data is exchanged from the point of view of the network nodes, i.e. if the CANopen-Master sends a PDO this is a receive PDO (RxPDO) on the master side, since the network node receives this PDO. The same applies to a transmit PDO (TxPDO) that the master receives.

4.4.9.5 Writing objects via PDOs

a) Mechanism on the CANopen-Master option module

The system writes objects via PDOs in the case of the CANopen-Master option module using FB CANop405_PDO_WRITE. The mechanism that runs in this connection looks like this:

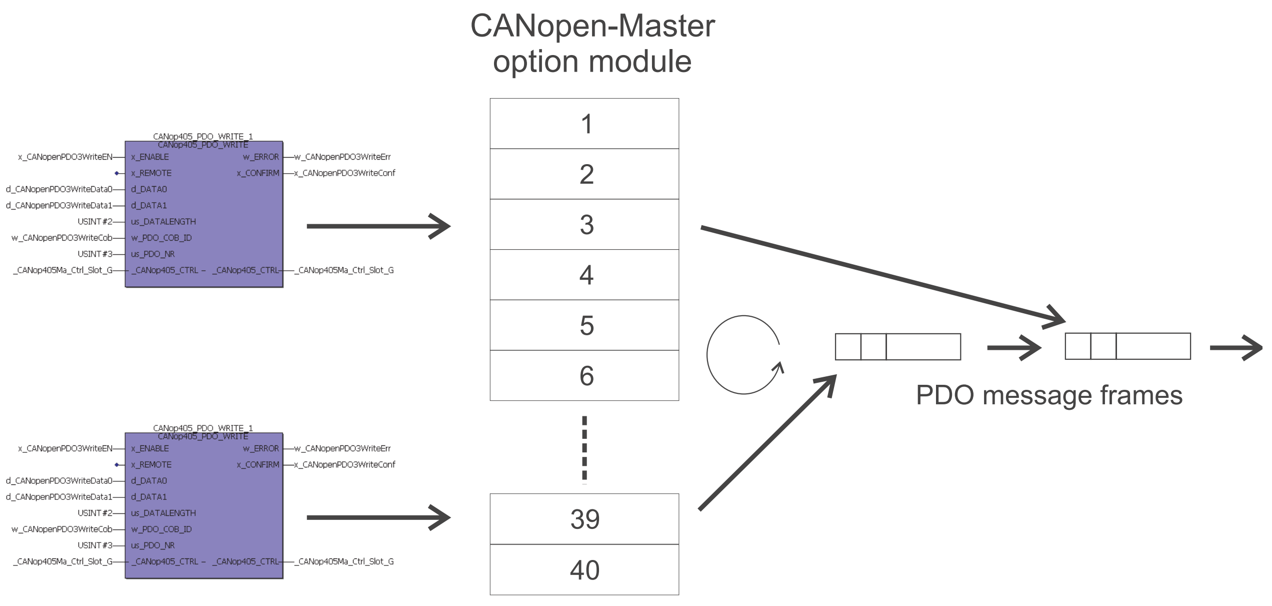


Figure 73: Mechanism on the option module for writing PDO

An FB CANop405_PDO_WRITE is assigned to each sending PDO. This is done by this function block forming appropriate instances and stating the COB-ID of the PDO at the block. Each of these function blocks must be assigned with a transmission slot on the CANopen-Master option module via an input on the function block. A total of 40 transmission slots are available. This means that 40 send jobs for PDOs can be entered at the same time. The option module checks the transmission slots cyclically for whether a transmit job is entered in the slot for a PDO. If this is the case, the system generates a PDO message frame from it and sends it.

b) Preparation and configuration

Before you can send a PDO, you must make some preparations as well as some configuration settings on the network node that the PDO is to evaluate. To do this, proceed as follows:

Determine important data:

- Determine the number of the network node on the CANopen network that is to process the PDO to be sent.
- Determine which of the network node's PDOs you want to transmit. When doing this, refer to the network node's documentation to be sure of how many PDOs it supports.
- Determine an unassigned transmission slot for the PDO. You should start with transmission slot 1 and assign the next ones continuously without gaps.
- Specify the PDO's type of transfer. If a network node receives PDOs, you only have two options: Immediate acceptance or acceptance after receiving a SYNC message frame.
- Now specify the objects that are mapped in the PDO. Observe the network node's, default mapping; you may possibly be able to use it for your application.
- Specify the time raster in which the PDO is to be transmitted. You later create a task in PROPROG wt II, which matches the time raster, to enter the PDO in the transmission slot. Observe the maximum CANopen bus loading. When operating CANopen at 1 Mbps, you can theoretically transmit 7 PDOs with 8 bytes of data each per ms. This is, however, conditional on there being no other data traffic. This is very unlikely, since you will probably be receiving PDOs too as well as running other CANopen functions like Node Guarding and Sync. This means that your application will for the most part determine the actual maximum possible bus loading. In actual fact, you will only be able to send less than 7 PDOs per ms.

Repeat these steps for all the PDOs that you want to write.

Programming the configuration and run it:

- Initialize sending of PDOs using FB CANop405_PDO_INIT. You must place this function block before FB CANop405_INIT. At FB CANop405_PDO_INIT, you must state the highest number of the transmission slot in use. You can use a maximum of 40 transmission slots.
- Use FB CANop405_NMT to set the network node to PRE-OPERATIONAL status.
- Use function blocks CANop405_SDO1_WRITE to CANop405_SDO8_WRITE that were described in the chapter entitled "Writing objects using SDO" to check and set the network node's configuration data (objects 1400h, 1401h, etc. and 1600, 1601, etc.)

- To determine the PDO's COB-ID, you can use function block CANop405_COB_ID. The FB calculates the COB-ID from the node number that you determined and the PDO number. You should use the FB in an initialization task.

c) Example: Configuration of transmit PDO

We want to write in a 2-ms cycle to ramp function generator input P1171 and RPM speed additional reference value P1040 of the BM4-O-CAN-03 option module. Receive PDO 2 of the BM4-O-CAN-03 option module is to be used. The PDO is to be applied when it arrives. Since we have not yet set up a PDO for transmitting, we will use transmission slot 1. After all, we do know the node number from the previous chapters: 5. For parameters 1171 and 1040, we still need data from the object directory to configure mapping:

Parameter	Index	Subindex	Data length
1171 Ramp function generator input	6042	0	16-bit
1040 RPM speed additional reference value	4410	0	32-bit

In the case of P1171, the value to be written directly represents according to the object directory the speed in RPM. Standardization of P1040 via CANopen is: -100% to +100% RPM speed corresponds to -230 to +230.

NOTE

Parameters P1171 and P1040 should only be written to as examples and a change in the values should be visible in WinBASS II. There is no functionality behind the written values – some of them change abruptly. This means that to avoid damage, you should ensure that a drive that may be connected to the b maXX 4400 is not ready for operation!

First, you must configure FB CANop405_PDO_INIT. To do this, you place this FB in the initialization POU before FB CANop405_INIT. At us_MAX_PDO_WRITE, we state the number of the transmission slot that was used last. Since we are now using the first transmission slot, this is USINT#1. After compiling and downloading the project, the FB in online mode should look like this:

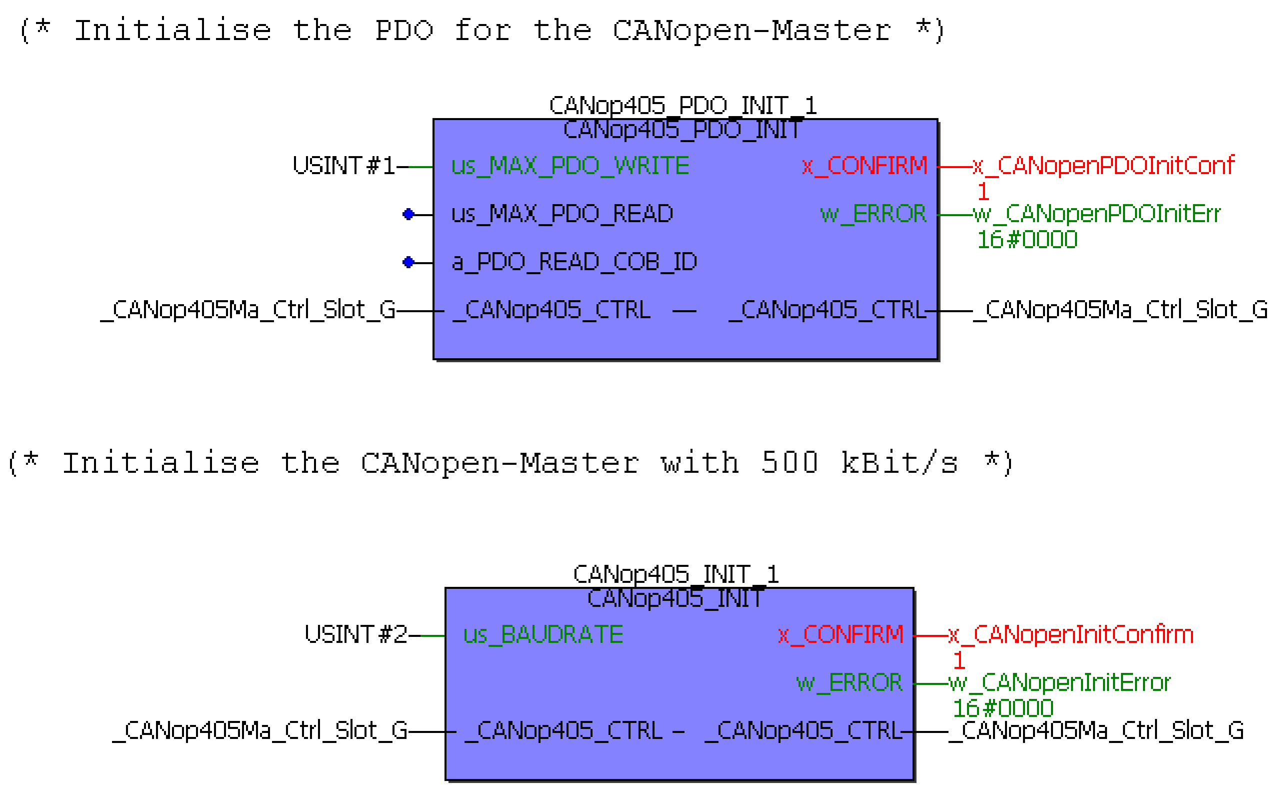


Figure 74: Initializing PDO transfer using FB CANop405_PDO_INIT

You must first place the BM4-O-CAN-03 option module in the PRE-OPERATIONAL status to be able to configure the PDO. To do this, we again use FB CANop405_NMT. You should not do this directly after switching on the b maXX 4400 unit, since the option module automatically goes into the PRE-OPERATIONAL status.

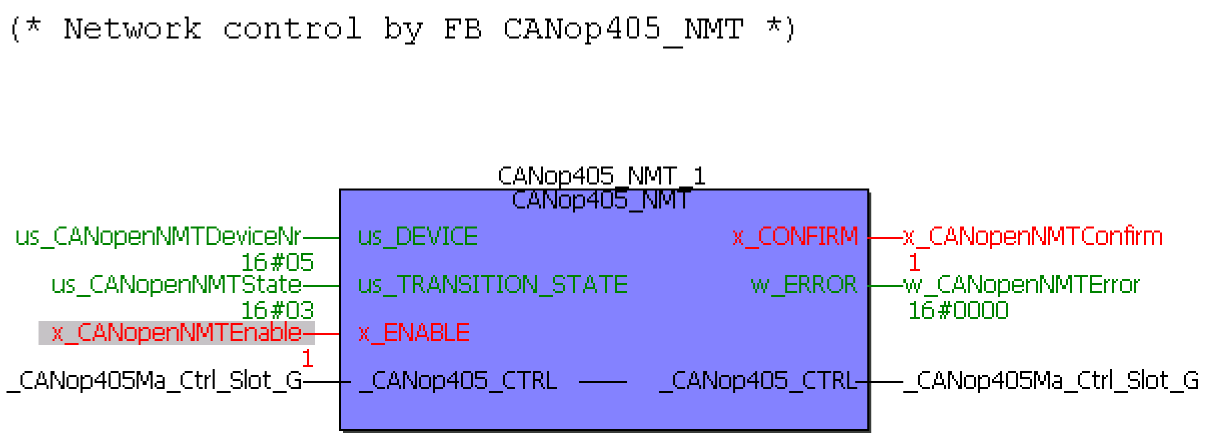


Figure 75: Switching the CANopen-Slave to the PRE-OPERATIONAL status

We now use FB CANop405_COB_ID to determine the COB-ID for the PDO. We implement this block in the existing initialization task. We connect node number 5 to us_DEVICE; at us_PDO_NR we connect node number 2 (we want to write to the second receive PDO). Since option module BM4-O-CAN-03 receives this PDO – i.e. it is an Rx-PDO – we must interconnect input x_RX with 1. We output the COB-ID to a global variable. After compiling and transmitting the project, we get a value of 305h for the COB-ID in the online representation.

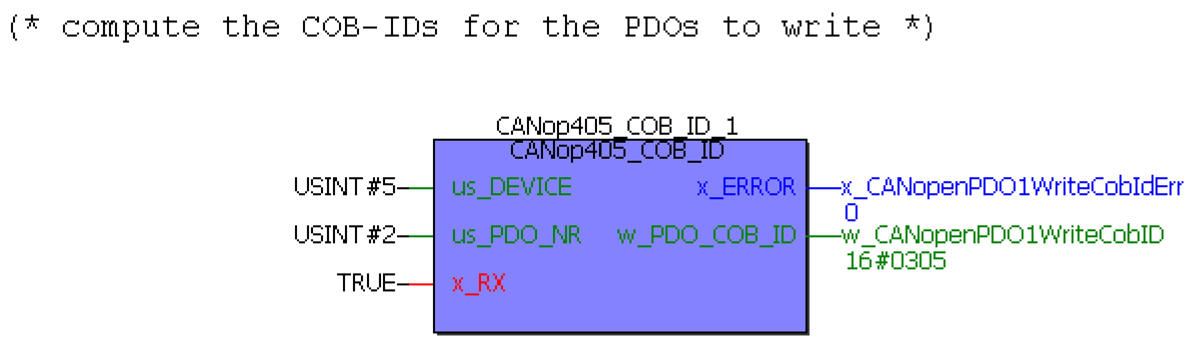


Figure 76: Determining the COB-ID for the second receive PDO using FB CANop405_COB_ID

We now check the settings for this PDO on the BM4-O-CAN-03 option module. The first receive PDO is configured for communication in object 1400h. After this, we find the configuration for the second receive PDO in object 1401h. We use the existing FB CANop405_SDO1_READ but with interconnection for object 1401h. This time, we use variables and not constants to easily switch to the next object online.

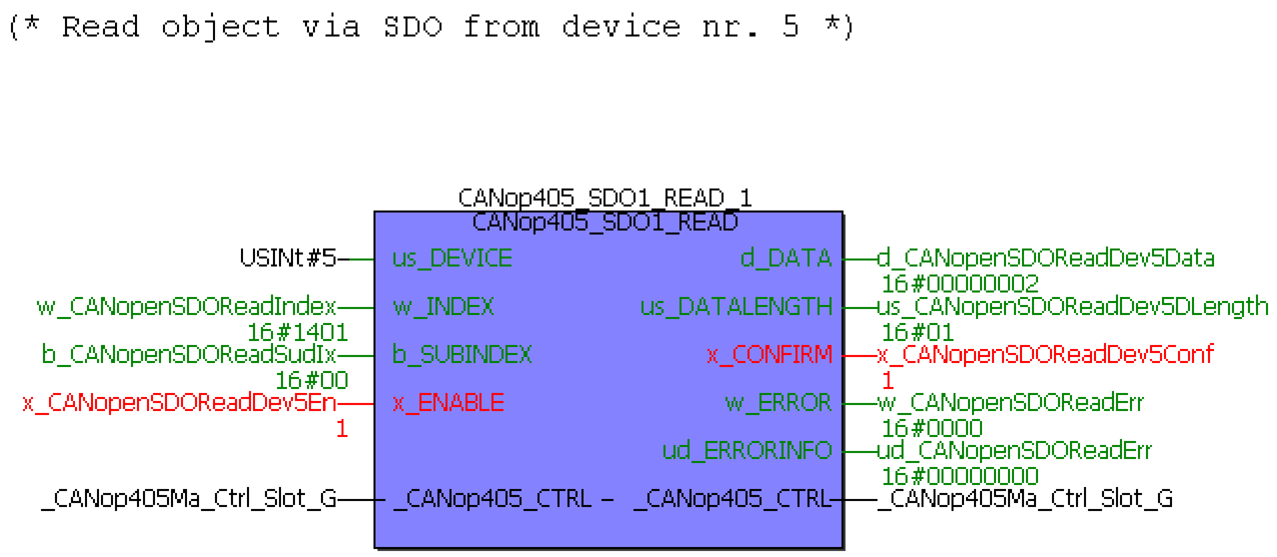


Figure 77: Reading out the configuration of the second receive PDO of the CANopen-Slave – sub-index 0

For sub-index 0, we get a value of 2. This means that object 1401h has valid entries in subindexes 0 - 2. Since we are interested in the set COB-ID and the validity of the PDO (both sub-index 1) and in the transfer type (sub-index 2), we must read sub-indexes 1 and 2. We get:

```
(* Read object via SDO from device nr. 5 *)
```

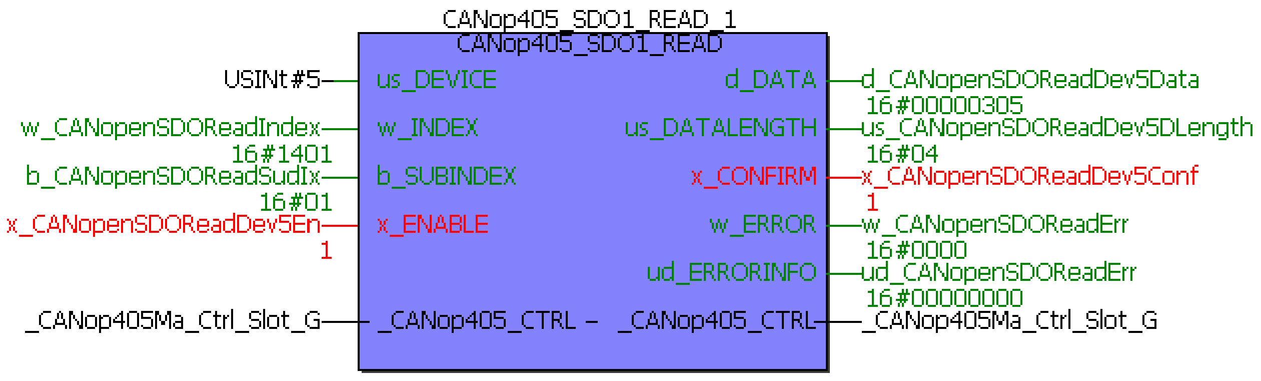


Figure 78: Reading out the configuration of the second receive PDO of the CANopen-Slave – sub-index 1

The value 305h means that this PDO's COB-ID is 305h, that it uses an 11-bit CAN-ID and that the PDO is valid. This means that the value of sub-index 1 is correct.

```
(* Read object via SDO from device nr. 5 *)
```

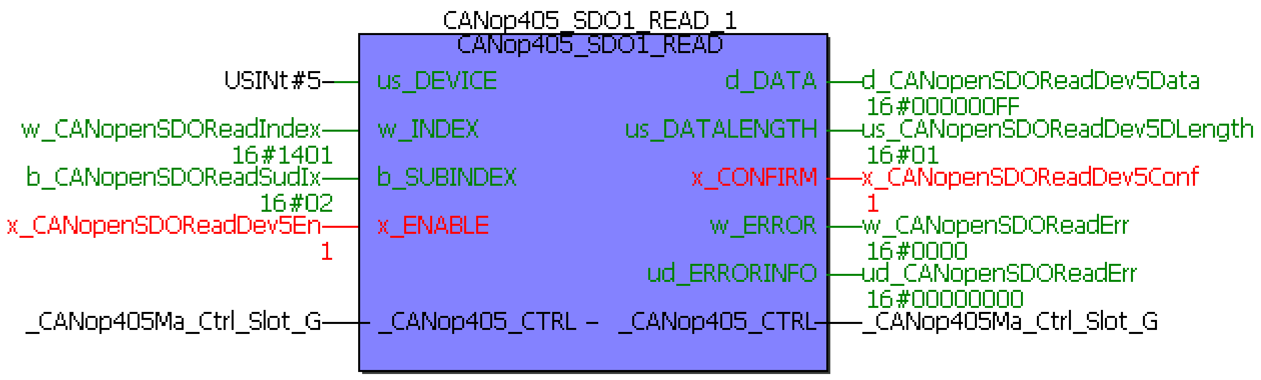


Figure 79: Reading out the configuration of the second receive PDO of the CANopen-Slave – sub-index 2

The value FFh means that the transfer type for this PDO is 255; this means that the system takes every PDO with an appropriate COB-ID (305h) when it is received. This value is correct too. If you get different values for the object contents here, you must use one of FBs CANop405_SDOx_WRITE (x = 1 to 8) to change the object contents.

Now we check the mapping for the second receive PDO in object 1601h. We want to map parameter 1171 first. This means that we should get the values listed below for the entries in 1601h, taking into account the object index, the sub-index and the data length:

Object 1601h

Subindex 0: 2h	Two mapped values
Subindex 1: 6042 00 10h	Index, sub-index, length for P1171
Subindex 2: 4410 00 20h	Index, sub-index, length for P1040

Let us check the values of object 1601h according to the procedure above using FB CANop405_SDO1_READ. In the sample project, we get the following values:

Object 1601h

Subindex 0: 2h	Two mapped values
Subindex 1: 6040 00 10h	
Subindex 2: 6060 00 08h	

Apart from sub-index 0, none of the values are correct. We must change the mapping. We use FB CANop405_SDO1_WRITE to do this. Instead of the constants, we connect global variables again to be able to easily switch to the next object online.

First of all, we delete the mapping of the second receive PDO by setting the value of sub-index 0 to zero.

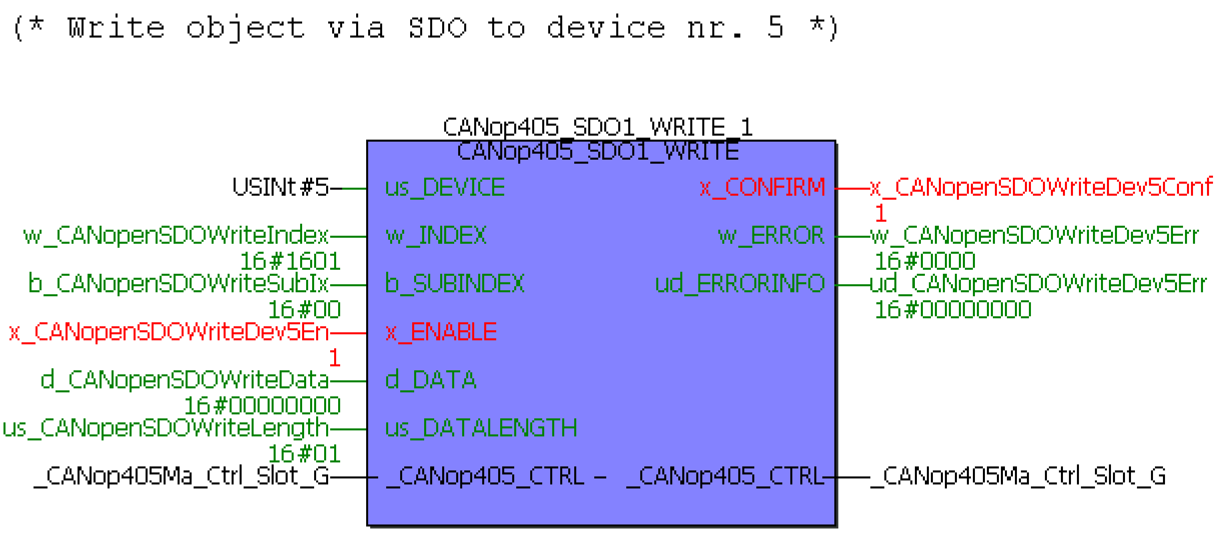


Figure 80: Deleting the mapping of the second receive PDO

Now, we enter the mapping of the first object P1171 in sub-index 1:

(* Write object via SDO to device nr. 5 *)

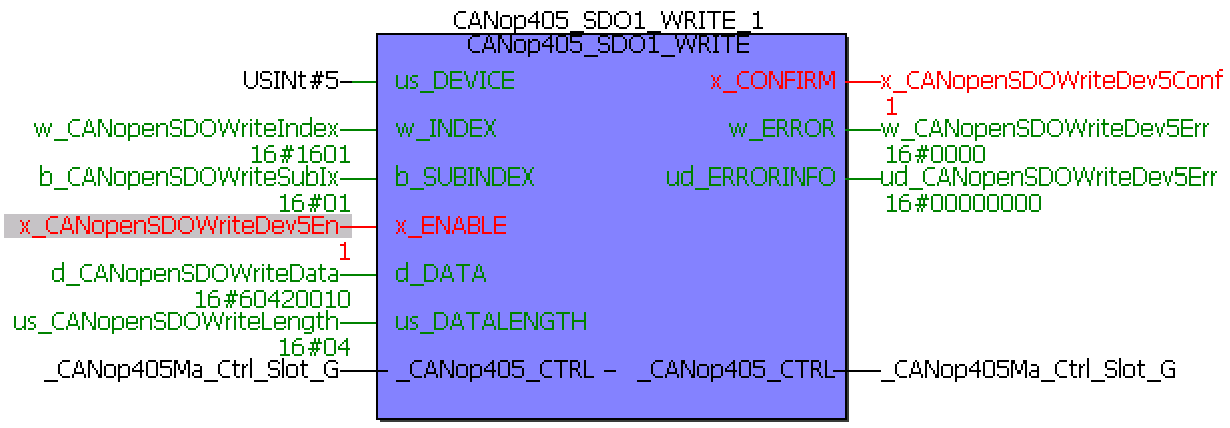


Figure 81: Entering the first mapped object of the second receive PDO

And then we delete the mapping of object P1040 in sub-index 2:

(* Write object via SDO to device nr. 5 *)

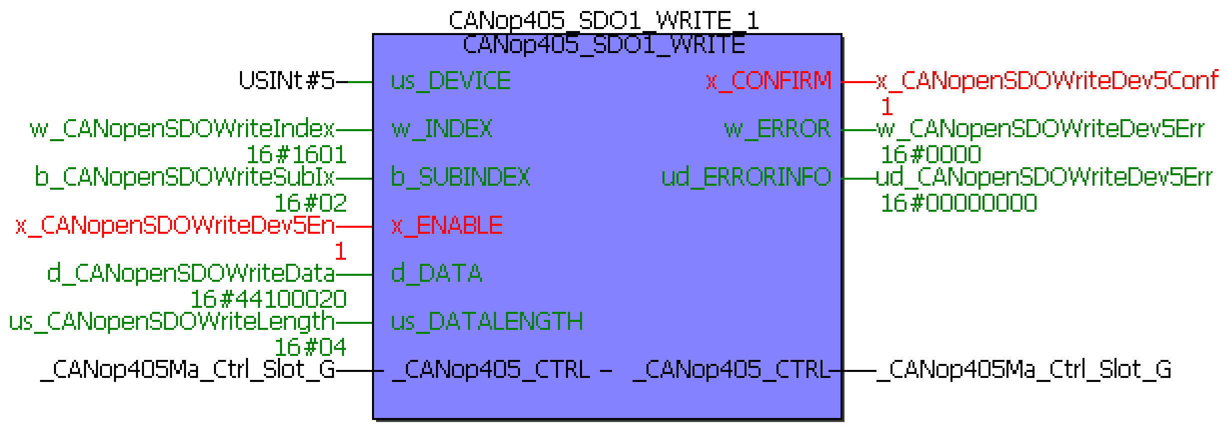


Figure 82: Entering the second mapped object of the second receive PDO

Now, all the objects are entered and we must release the mapping again by entering the number of mapped objects in sub-index 0:

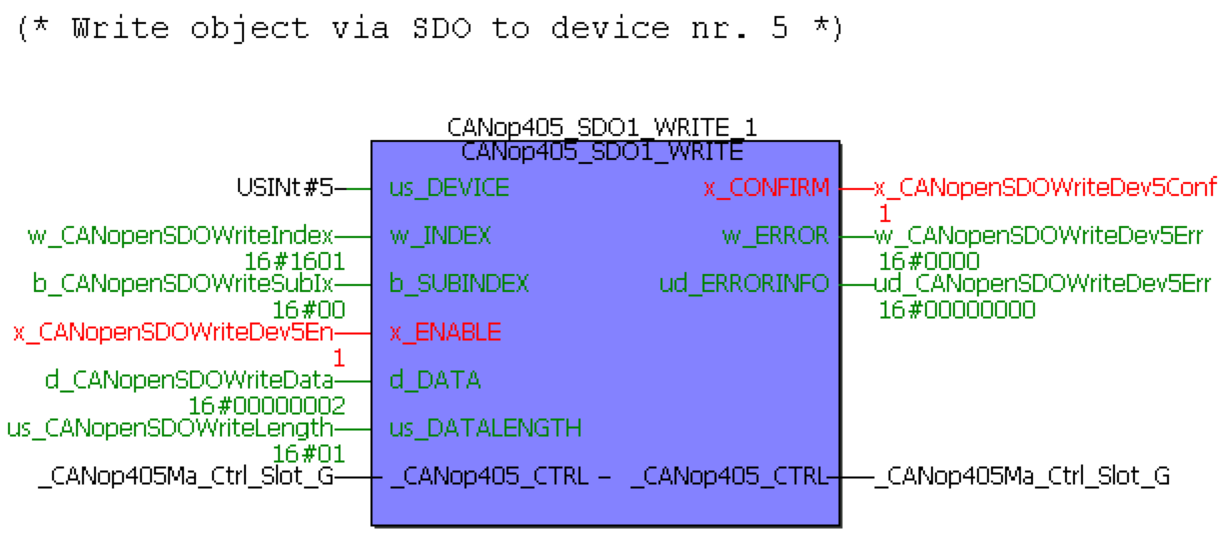


Figure 83: Entering the number of mapped objects of the second receive PDO

The BM4-O-CAN-03 option module gives you the option of saving the set communications parameters. The next time that you switch the b maXX 4400 unit off and on again, the system then automatically applies the saved communications parameters. Storing is triggered by writing objects 1010h, sub-index 2 with a value of 65766173h. We want to carry this out now:

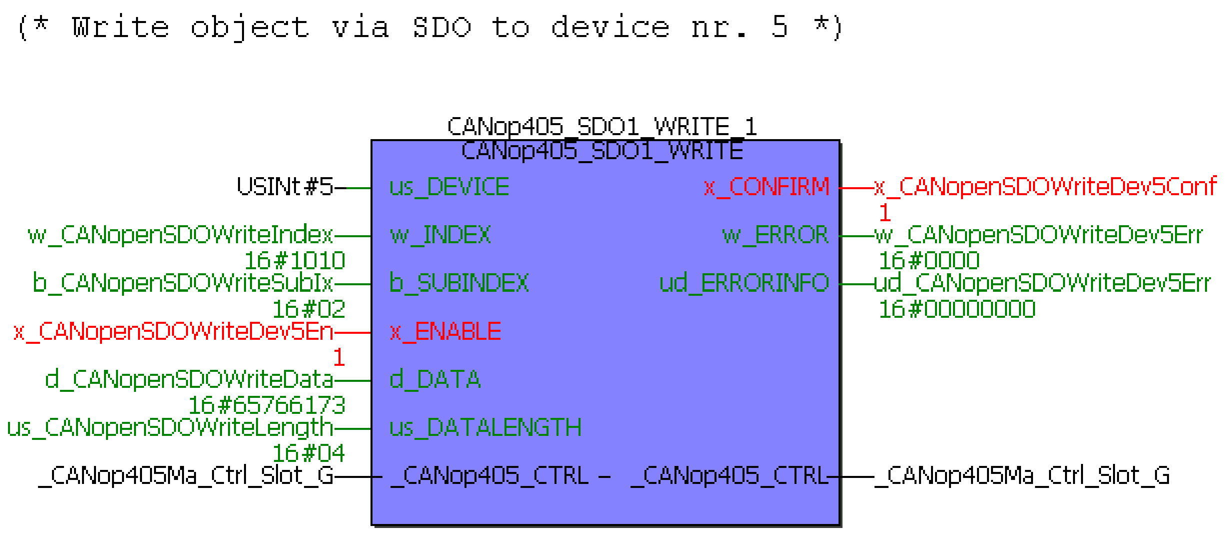


Figure 84: Storing the communications parameters of the CANopen-Slave

The second receive PDO on the BM4-O-CAN-03 option module is now completely configured and we can now proceed to actual transmission of the PDO.

d) Transmitting a PDO to a network node

You transmit a PDO via the instances of function block CANop405_PDO_WRITE. Using these function blocks, it is possible to start up to 40 PDO jobs at the same time. In this connection, the number 40 corresponds to the number of transmission slots on the CANopen-Master option module that are available for this purpose. You must not use a transmission slot twice. It is, however, possible to use one transmission slot successively for different PDOs. For a network node to take a PDO, it must be in the OPERATIONAL status.

To use function block CANop405_PDO_WRITE, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cyclical IRP task.
- Place one or more FBs CANop405_PDO_WRITE in this POU as required.
- Interconnect the function blocks with variables of the correct data type and assign them the desired values. At input us_PDO_NR, state the number of the transmission slot that you chose. w_PDO_COB_ID is the COB-ID of the PDO that you want to write to. us_DATALENGTH designates the valid bytes in the PDO. It is the total of all the bytes of the mapped objects. You connect the data for the objects at d_DATA0 and d_DATA1. If you want to request a remote PDO, set input x_REMOTE to 1.
- If it is not yet present in your project, create a cyclical bypass IRP task. If the PDO is not time-critical, you can also use a simple cyclical task with the priority of your choice. Link the created POU in this task.
- Compile the project and load it as the (boot) project on the PLC. Start the PLC.
- If necessary, configure the network nodes and place them in the OPERATIONAL status.

x_ENABLE = 1 transmits the PDO. FB CANop405_PDO_WRITE indicates successful execution with x_CONFIRM = 1 and w_ERROR = 16#0. Since the slave cannot write a PDO, x_CONFIRM = 1 only means that the CANopen-Master option module has sent the PDO but not that the slave has taken it.

e) Example: Transmitting a PDO

We will continue the example. The second receive PDO of the BM4-O-CAN-03 option module is now configured and we can now proceed to actual data exchange via PDOs. First of all, you must place the network node in the OPERATIONAL status. Again, we use FB CANop405_NMT to do this.

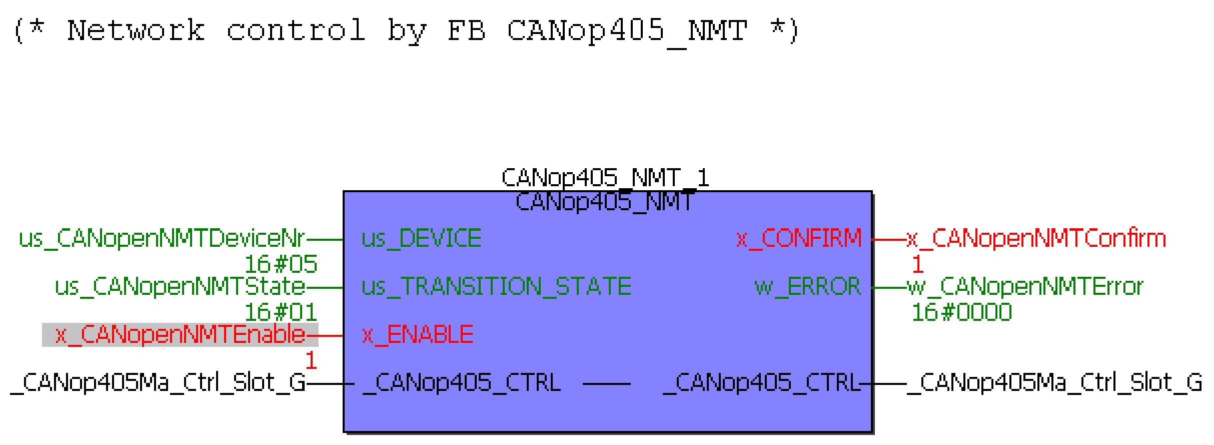


Figure 85: Switching the CANopen-Slave to the OPERATIONAL status

After this, we create a POU for later linking in an IRP task. We place FB CANop405_PDO_WRITE in this POU and interconnect it to corresponding constants and variables. As was specified before, you must use the first transmission slot: this means that us_PDO_NR = 1. We fetch COB-ID from FB CANop405_COB_ID from the initialization task via global variable w_CANopenPDO1WriteCobID. P1171 is two bytes long, P1040 is four bytes long; this means that us_DATALENGTH gets a value of USINT#6. Due to the mapping, P1171 is located in the first word of d_DATA0 and P1040 is in the second word of d_DATA0, as well as in the first word of d_DATA1. To detect data traffic, both parameters should just count up or count down; in this connection, P1171 is limited to a maximum of 3000 RPM. The finished code looks like this:

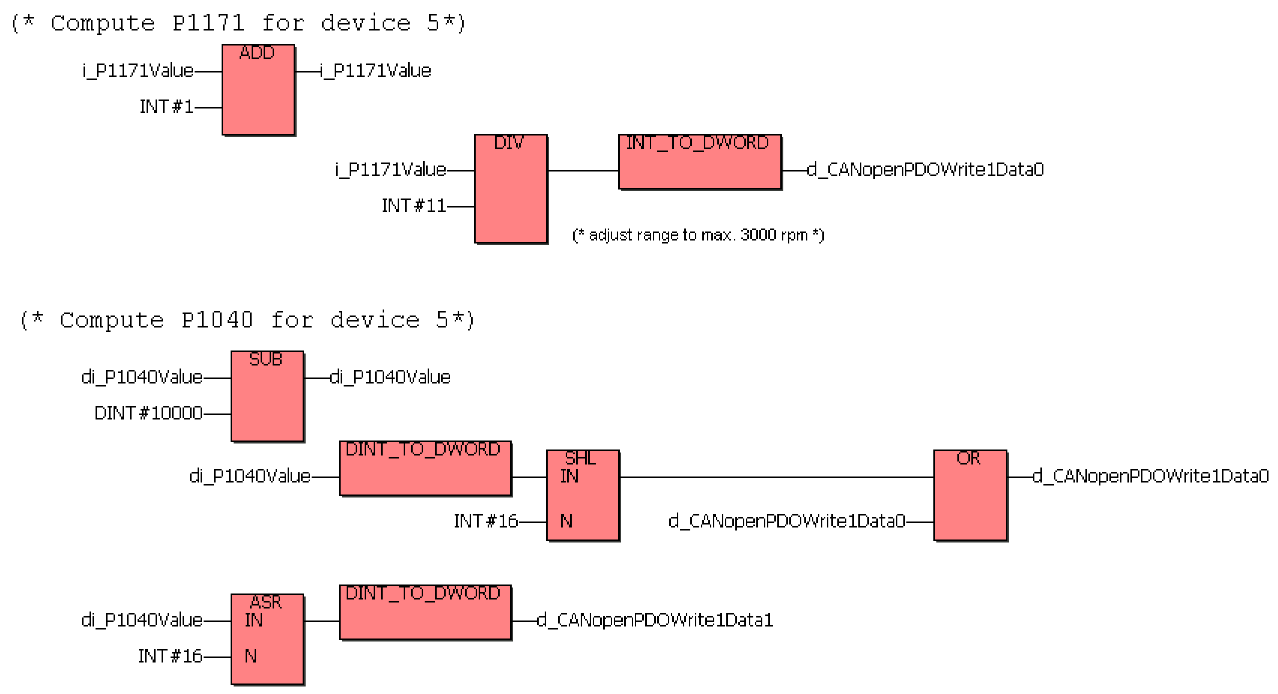


Figure 86: Changing the data to be written

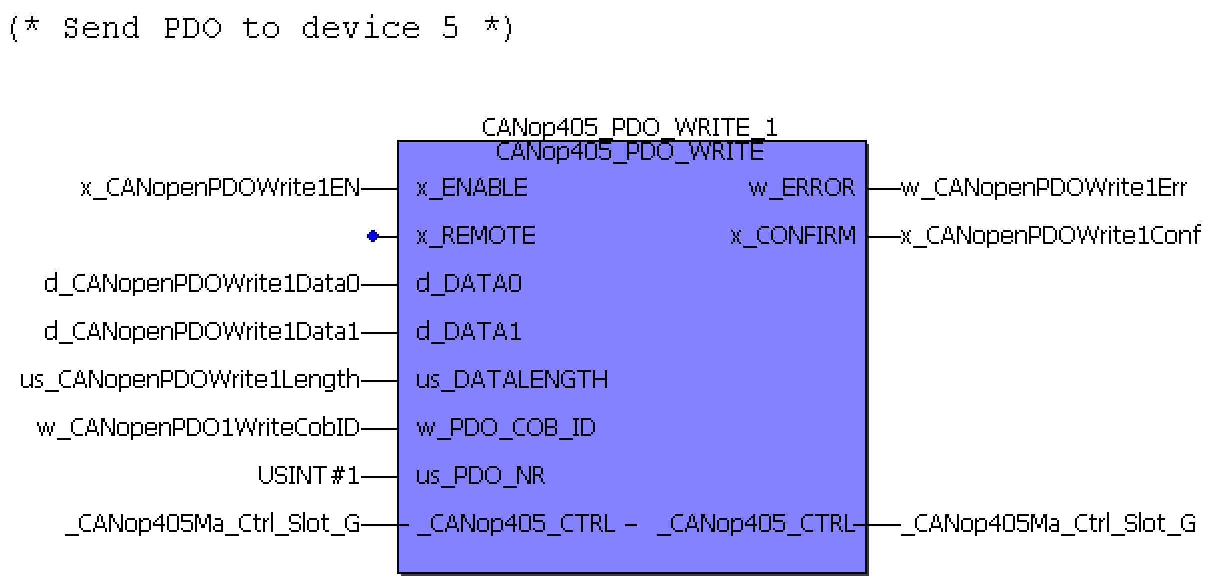


Figure 87: Writing data using FB CANop405_PDO_WRITE

To be able to call FB CANop405_PDO_WRITE every 2 ms, we need a timer IRP task. To set it up, we need FB INTR_SET from library SYSTEM_PLC01_20bd00 (or higher). We configure FB INTR_SET for a 2-ms CPU timer 1 interrupt and implement this function block in the CANopen initialization POU before the FB CANop405_PDO_INIT:

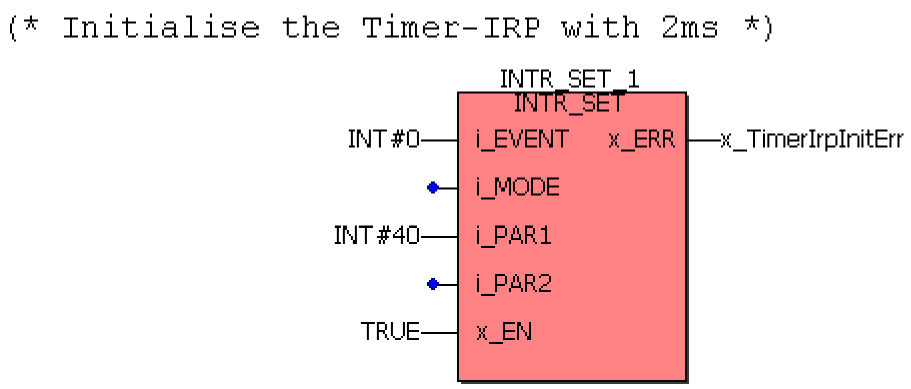


Figure 88: Setting up a 2-ms timer IRP task for data transfer

You can now set up the corresponding event task with "CPU timer 1" event and bypass operation and assign the POU containing FB CANop405_PDO_WRITE to this event task. Compile the project and send it as a project (not as a boot project) to the PLC. Start the project. If you sent a boot project and restarted the b maXX 4400 unit, you will lose the previously set PDO configuration if you did not save it. In this case, you must carry out configuration again, since the BM4-O-CAN-03 option module takes the saved PDO configuration at starting. In the case of a cold restart, you must also put the BM4-O-CAN-03 option module in the OPERATIONAL status.

Using x_ENABLE = 1 on FB CANop405_PDO_WRITE, you can start data transfer:

```
(* Send PDO to device 5 *)
```

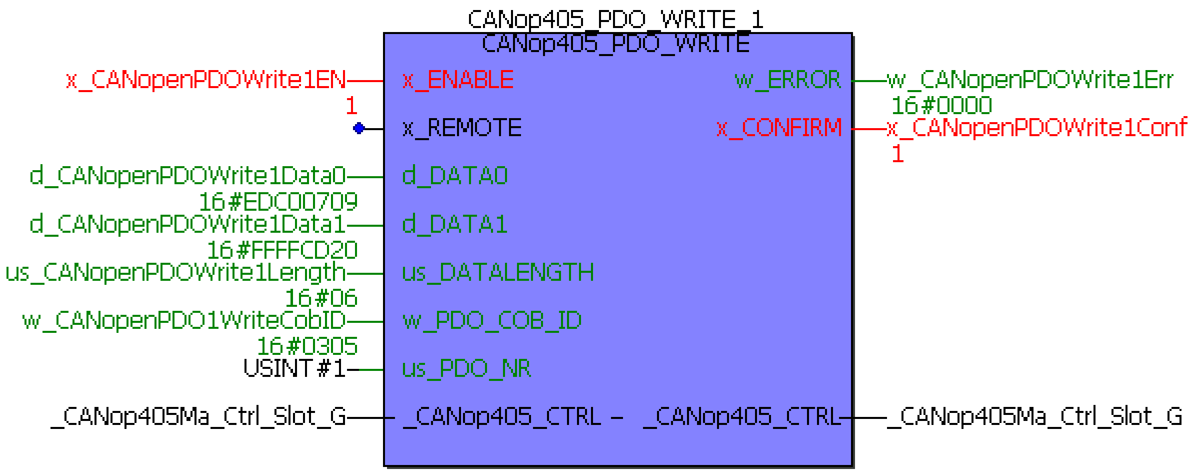


Figure 89: Writing data using FB CANop405_PDO_WRITE online

The system confirms correct sending of the PDO with x_CONFIRM = 1. You should be able to detect a change in parameters 1171 and 1040 in WinBASS II.

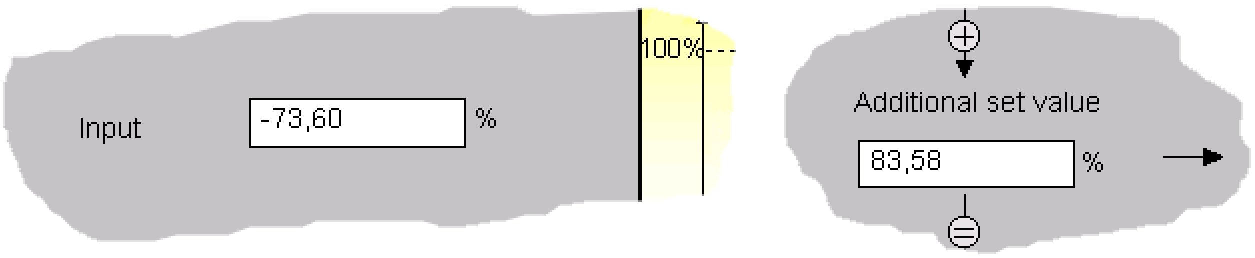


Figure 90: Written data in WinBASS II

Note that these are snapshots in online mode and that your values may be different.

4.4.9.6 Reading objects via PDOs

a) Mechanism on the CANopen-Master option module

The system reads objects via PDOs in the case of the CANopen-Master option module using FB CANop405_PDO_READ. The mechanism that runs in this connection looks like this:

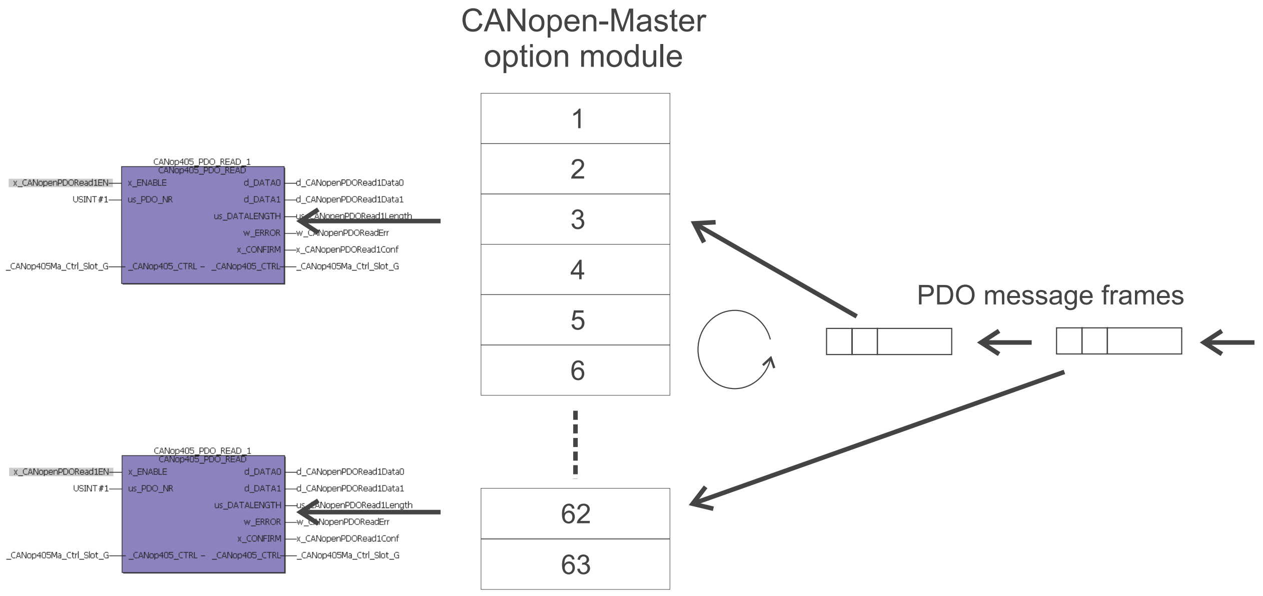


Figure 91: Mechanism on the option module for reading PDO

An FB CANop405_PDO_READ is assigned to each PDO to receive by this function block forming appropriate instances. Each of these function blocks must be assigned with a reception slot on the CANopen-Master option module via an input on the function block. A total of 63 reception slots are available. This means that 63 receive jobs for PDOs can be evaluated at the same time. When a PDO arrives, the option module enters it in a reception slot. The option module chooses the correct reception slot via the PDO's COB-ID. To do this, the option module must be informed at the initialization stage of which COB-ID is assigned to which reception slot. To do this, you use FB CANop405_PDO_INIT. It is no longer possible to reconfigure during operation. At calling, FB CANop405_PDO_READ checks whether an entry is present for it in its assigned reception slot.

NOTE

Received PDOs may be lost if more PDOs arrive than FB CANop405_PDO_READ reads out or if FB CANop405_PDO_READ is just taking a PDO from the reception slot. To minimize possible losses, FB CANop405_PDO_READ should have a longer call interval than data can arrive. Apart from this, you should use mechanisms for controlled PDO transfer (e.g. SYNC or remote request, see the appropriate chapter).

b) Preparation and configuration

Before you can receive a PDO, you must make some preparations as well as some configuration settings on the network node that the PDO is to transmit. To do this, proceed as follows:

Determine important data:

- Determine the number of the network node on the CANopen network that is to send the PDO.

- Determine which of the network node's PDOs is to be sent. When doing this, refer to the network node's documentation to be sure of how many PDOs it supports.
- Determine an unassigned reception slot for the PDO. You should start with reception slot 1 and assign the next ones continuously without gaps.
- Specify the PDO's type of transfer. When a network node sends PDOs, you have various options; get more information on this topic in the chapter entitled ▷Configuring the transfer properties of a PDO ◁ on page 110, and in the documentation on the network node.
- Now specify the objects that are to be mapped in the PDO. Observe the network node's, default mapping; you may possibly be able to use it for your application.
- Specify the time raster in which the PDO is to be evaluated. Later, create a task in PROPROG wt II, which matches the time raster, to check reception in the reception slot. The time raster should match the PDO's transfer type. Observe the maximum CANopen bus loading. When operating CANopen at 1 Mbps, you can theoretically receive 7 PDOs with 8 bytes of data each per ms. This is, however, conditional on there being no other data traffic. This is very unlikely, since you will probably be sending PDOs too as well as running other CANopen functions like Node Guarding and Sync. This means that your application will for the most part determine the actual maximum possible bus loading. In actual fact, you will only be able to receive less than 7 PDOs per ms.

Repeat these steps for all the PDOs that you want to receive.

Programming the configuration and run it:

- Initialize reception of PDOs using FB CANop405_PDO_INIT. You must place this function block before FB CANop405_INIT. At FB CANop405_PDO_INIT, you must state the highest number of the reception slot in use. You can use a maximum of 63 reception slots. At this FB, you assign the COB-IDs of the PDOs to the reception slots. To do this, you use an array whose index place matches in each case a reception slot and whose contents are the COB-ID. As a maximum, you can enter values up to index 63. If you make entries above this number and at index 0, the system issues an error message.
- To determine the PDO's COB-ID, you can use function block CANop405_COB_ID. This FB calculates the COB-ID from the node number that you determined and the PDO number. This means that the FB is called before FB CANop405_PDO_INIT.
- Use FB CANop405_NMT to set the network node to PRE-OPERATIONAL status.
- Use function blocks CANop405_SDO1_WRITE to CANop405_SDO8_WRITE that were described in the chapter entitled "Writing objects using SDO" to check and set the network node's configuration data (objects 1800h, 1801h, etc. and 1A00, 1A01, etc.)

c) Example: Configuration of receive PDO

The BM4-O-CAN-03 option module is to send in a 5-ms cycle status word P301 and speed additional reference value P1040. We want to use the BM4-O-CAN-03 option module's transmit PDO 1. On the CANopen-Master option module side, we want to check receipting of the PDO every 2 ms. Since we have not yet set up a PDO for receiving, we will use reception slot 1. After all, we do know the node number from the previous chapters: 5. For parameters 301 and 1040, we still need some data from the object directory to configure mapping:

Parameter	Index	Subindex	Data length
301 status word	6041	0	16 bit
1040 RPM speed additional reference value	4410	0	32 bit

Standardization of P1040 via CANopen is: -100% to +100% RPM speed corresponds to -2^{30} to $+2^{30}$.

First, you must configure FB CANop405_PDO_INIT. This FB is already present in the initialization POU due to configuration of transmission. At us_MAX_PDO_READ, we state the number of the reception slot that was used last. Since we are only using the first reception slot, this is USINT#1. You must still assign a COB-ID to reception slot 1. You make the assignment via input a_PDO_READ_COB_ID. Here, you must connect an array of data type WORD_64_BMARRAY that contains the COB-ID. Since we are using reception slot 1, you must enter the COB-ID at index 1 of the array. We use FB CANop405_COB_ID for generating the COB-ID. At us_DEVICE, you connect node number 5 with node number 1 being connected to an us_PDO_NR. Since option module BM4-O-CAN-03 transmits this PDO – i.e. it is a TxPDO – we must interconnect input x_RX with 0.

After compiling and downloading the project, initialization in online mode should look like this:

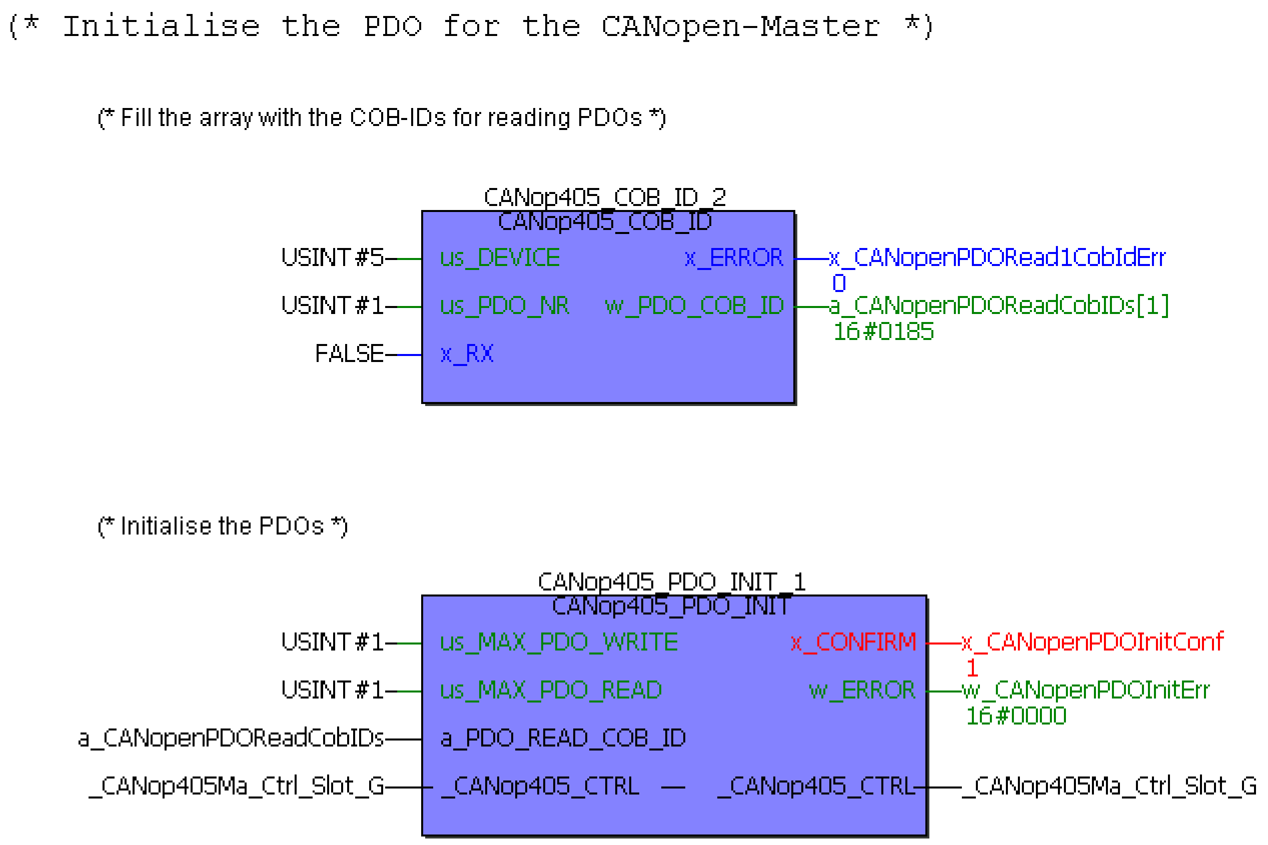


Figure 92: Initializing PDO transfer using FB CANop405_PDO_INIT

For the COB-ID we get a value of 185h. Put the BM4-O-CAN-03 option module with FB CANop405_NMT in PRE-OPERATIONAL status to be able to configure the PDO.

We now check the settings for this PDO on the BM4-O-CAN-03 option module. The first transmit PDO is configured for communication in object 1800h. We use the existing FB CANop405_SDO1_READ, but with interconnection for object 1800h.

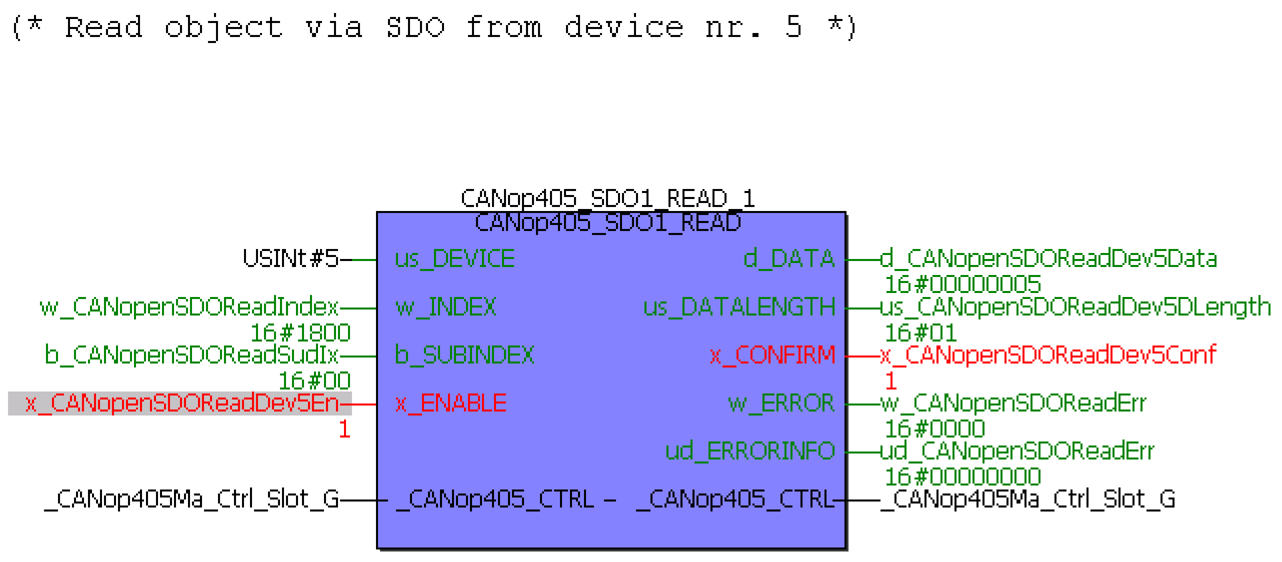


Figure 93: Reading out the configuration of the first transmit PDO of the CANopen-Slave – sub-index 0

For sub-index 0, we get a value of 5. This means that object 1800h has valid entries in subindexes 0 - 5. Since we are interested in the set COB-ID and the validity of the PDO (both sub-index 1), in the transfer type (sub-index 2) and in the transfer time (sub-index 5), we must read sub-indexes 1, 2 and 5.
We obtain values below by the same procedure as for sub-index 0:

Object 1800h

Subindex 0:	5h	Five valid sub-index entries
Subindex 1:	185h	COB-ID of the first transmit PDO
Subindex 2:	FFh	Transfer type 255
Subindex 5:	0h	Timer value for transfer time

The value of this PDO's COB-ID is 185h; it uses an 11-bit CAN-ID and the PDO is valid. This means that the value of sub-index 1 is correct. The value FFh of sub-index 2 means that the transfer type for this PDO is 255, i.e. that the PDO is transferred in the case of a change in the value of a mapped object. We must change this value to 254 using one of FBs CANop405_SDO1_WRITE to CANop405_SDO8_WRITE, since we need time event-driven transfer.

```
(* Write object via SDO to device nr. 5 *)
```

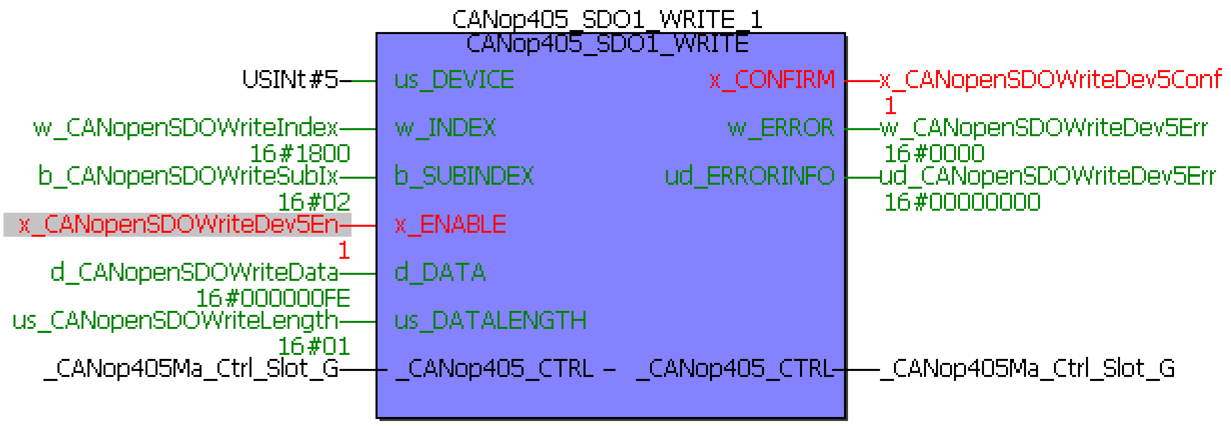


Figure 94: Writing the configuration of the first transmit PDO of the CANopen-Slave – sub-index 2

Sub-index 5 contains the timer value of the time event. 0h is entered; however, we want to transfer every 5 ms. This means that you must change sub-index 0 to a value of 5 ms:

```
(* Write object via SDO to device nr. 5 *)
```

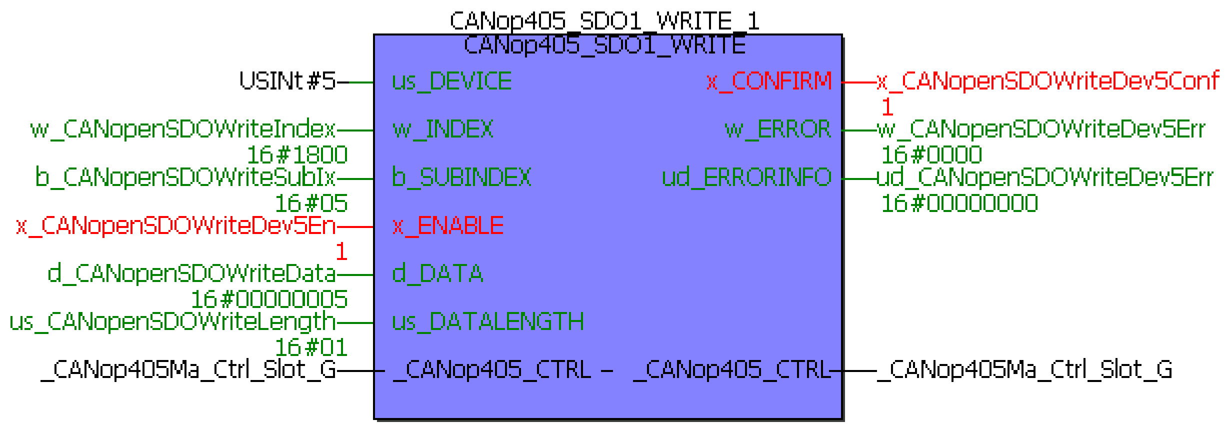


Figure 95: Writing the configuration of the first transmit PDO of the CANopen-Slave – sub-index 5

Now we check the mapping for the first transmit PDO in object 1A00h. We want to map parameter 301 first. This means that we should get the values listed below for the entries in 1A00h, taking into account the object index, the sub-index and the data length:

Object 1A00h

Subindex 0:	2h	Two mapped values
Subindex 1:	6041 00 10h	Index, sub-index, length for P301
Subindex 2:	4410 00 20h	Index, sub-index, length for P1040

Let us check the values of object 1A00h according to the procedure above using FB CANop405_SDO1_READ. In the sample project, we get the following values:

Object 1A00h

Subindex 0:	1h	One mapped value
Subindex 1:	6041 00 10h	

Apart from sub-index 1, none of the values are correct. We must change the mapping. We again use FB CANop405_SDO1_WRITE to do this.

First of all, we must delete the mapping of the first transmit PDO by setting the value of sub-index 0 to zero. In actual fact, the mapping is not deleted; rather, the system ignores it for this PDO. The value of sub-index 1's status word, which we can use, is retained. This means that after deleting sub-index 0 we only need to set the mapping in sub-index 2:

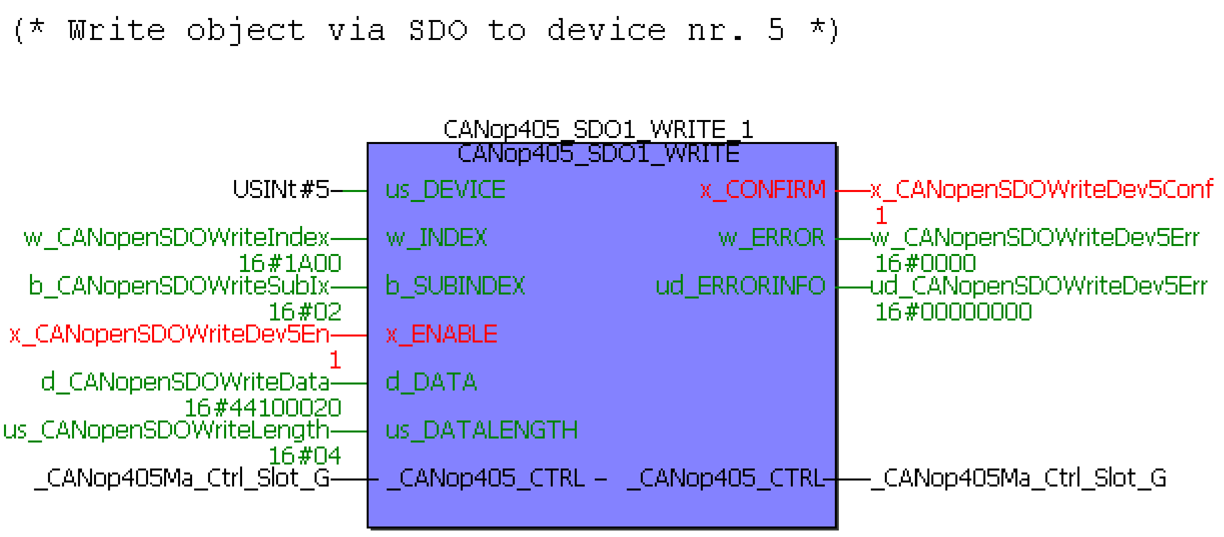


Figure 96: Entering the second mapped object of the second transmit PDO

Now, all the objects are entered and we must release the mapping again by entering the number of mapped objects in sub-index 0: You must write sub-index 0 of object 1A00h with a value of 2 using the same procedure. Finally, we save the set communications parameters by using an SDO to write object 1010h, sub-index 2 with a value of 65766173h.

The first transmit PDO on the BM4-O-CAN-03 option module is now completely configured and we can now proceed to actual reception of the PDO.

d) Receiving a PDO from a network node

You receive a PDO via the instances of function block CANop405_PDO_READ. Using these function blocks, it is possible to receive up to 63 PDO message frames at the same time. In this connection, the number 63 corresponds to the number of reception slots on the CANopen-Master option module that are available for this purpose. You must not use a reception slot twice. For a network node to transmit a PDO, it must be in the OPERATIONAL status.

To use function block CANop405_PDO_READ, proceed as follows:

- Create a POU with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a cyclical IRP task.
- Place one or more FBs CANop405_PDO_READ in this POU as required.
- Interconnect the function blocks with variables of the correct data type and assign them the desired values. At input us_PDO_NR, state the number of the reception slot you selected. Using FB CANop405_PDO_INIT, you have already assigned COB_IDs to reception slots at initialization. us_DATALENGTH designates the valid bytes of the received PDO. It is the total of all the bytes of the mapped objects. The system outputs the data of the received objects at d_DATA0 and d_DATA1. With FB CANop405_PDO_READ, you also receive transmit PDOs that you requested by means of a remote request using FB CANop405_PDO_WRITE.
- If it is not yet present in your project, create a cyclical bypass IRP task. If the PDO is not time-critical, you can also use a simple cyclical task with the priority of your choice. Link the created POU in this task.
- Compile the project and load it as the (boot) project on the PLC. Start the PLC.
- If necessary, configure the network nodes and place them in the OPERATIONAL status.

Using x_ENABLE = 1, you can receive the PDO. FB CANop405_PDO_READ reports that a PDO has arrived using x_CONFIRM = 1 and w_ERROR = 16#0. x_CONFIRM is zeroed again in the next cycle and the next PDO can receive it with x_ENABLE = 1.

e) Example: Receiving a PDO

We will continue the example. The first transmit PDO of the BM4-O-CAN-03 option module is now configured and we can now proceed to actual data exchange via PDOs. First of all, you must place the network node in the OPERATIONAL status. Again, we use FB CANop405_NMT to do this.

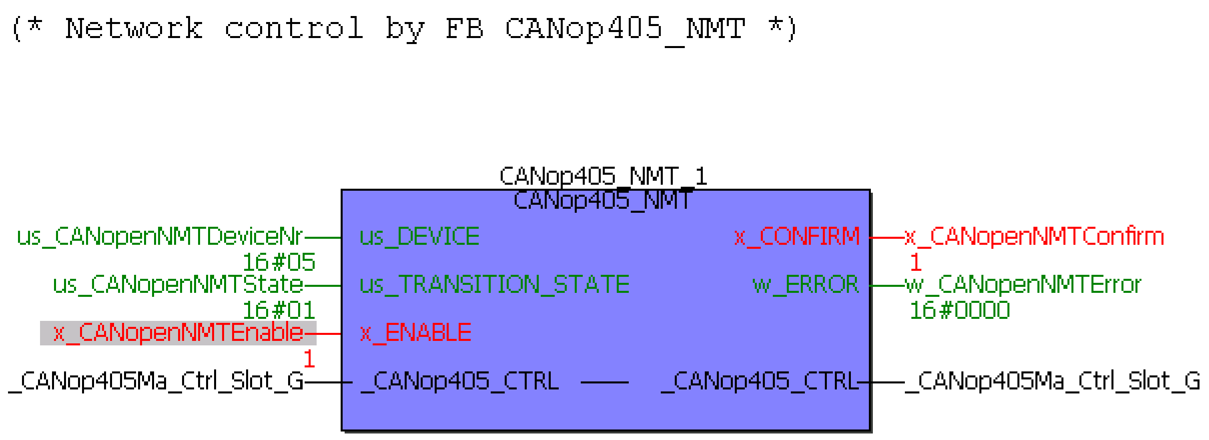


Figure 97: Switching the CANopen-Slave to the OPERATIONAL status

We place FB CANop405_PDO_RERAD in the existing POU using FB CANop405_PDO_WRITE. This means that the system calls FB CANop405_PDO_READ every 2 ms. As was previously specified, the first reception slot must be used: this means that us_PDO_NR = 1. Due to mapping, P301 is in the first word of d_DATA0 and P1040 is in the second word of d_DATA0, as well as in the first word of d_DATA1. Compile the project and send it as a project (not as a boot project) to the PLC. Start the project. If you sent a boot project and restarted the b maXX 4400 unit, you will lose the previously set PDO configuration if you did not save it. In this case, you must also carry out configuration again, since the BM4-O-CAN-03 option module takes the saved PDO configuration at starting. In the case of a cold restart, you must also put the BM4-O-CAN-03 option module in the OPERATIONAL status.

To detect data traffic, we additionally start the existing FB CANop405_PDO_WRITE, which changes P1040. We can influence the status word via WinBASS II. Using x_ENABLE = 1 on FB CANop405_PDO_READ, you can start PDO reception. PDOs should arrive immediately:

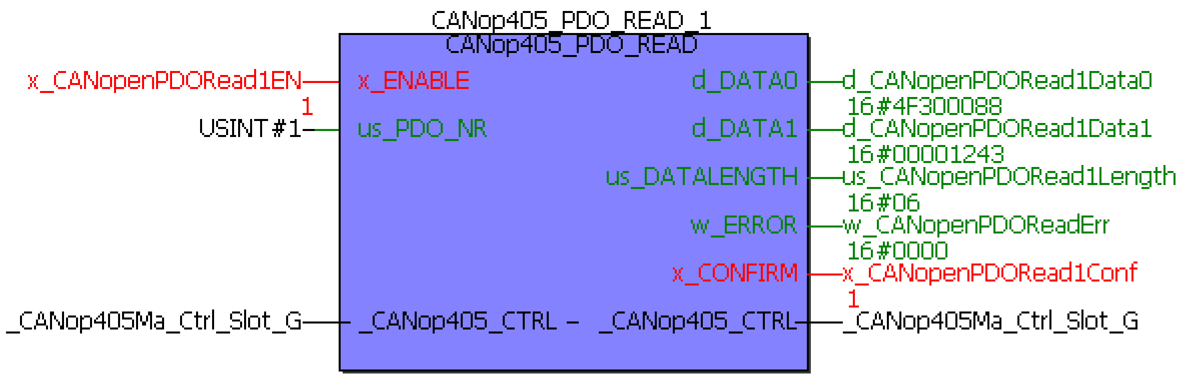


Figure 98: Reading data using FB CANop405_PDO_READ online

Note that these are snapshots in online mode and that your values may be different. Output x_CONFIRM will not have a constant value of 1, since the PDOs are transmitted at a 5-ms time interval and FB CANop405_PDO_READ is called at a 2-ms time interval.

4.4.10 Synchronizing data exchange

4.4.10.1 Definition according to the CANopen specification

In many applications it is sensible to synchronize taking input data and transmitting output data. CANopen makes available SYNC message frames for this. After NMT message frames, SYNC message frames have the highest priority and no user data. SYNC message frames are transferred according to the consumer/producer communication model, i.e. a network node generates a SYNC message frame that is processed by one or more other network node(s). They do not confirm receiving the message frame. The network node uses reception to trigger reading of the received PDOs or transmission of PDOs, assuming that a PDO is configured for this type of transfer (see the chapter entitled ▹4.4.9.2 Configuring the transfer properties of a PDO ◃ from page 110 onward). You use object 1005h to set whether a network node is a consumer or a producer:

SYNC message frame

Object	Meaning
1005 h	Configuration of SYNC message frame.
Subindex 0	COB-ID and validity

The value of sub-index 0 has data type Unsigned32 and is divided as follows:

Bit	Value / meaning
31 (MSB)	*Not relevant*
30	0: Network node is not generating a SYNC message frame 1: Network node is generating a SYNC message frame
29	0: 11-bit CAN ID 1: 29-bit CAN ID (is not supported by the CANopen-Master option module)
28 - 11	Not supported by the CANopen-Master option module
10 - 0 (LSB)	COB-ID of the SYNC message frame. Since the CANopen-Master option module uses the COB-ID in accordance with the predefined identifier assignment (see the chapter entitled ▷4.2.2.3 Data exchange and objects of the physical bus system ◁ from page 53 onward), this must be 80h.

The default setting is virtually always SYNC consumer, i.e. you do not have to separately set the network node to SYNC consumer.

You can use the CANopen-Master option module as a SYNC producer. Since the CANopen-Master option module has no objects of its own, you only need to use function block CANop405_SYNC to generate a SYNC message frame. The SYNC consumer function is not supported.

Taking or transferring PDO contents on a SYNC consumer runs according to the following scheme:

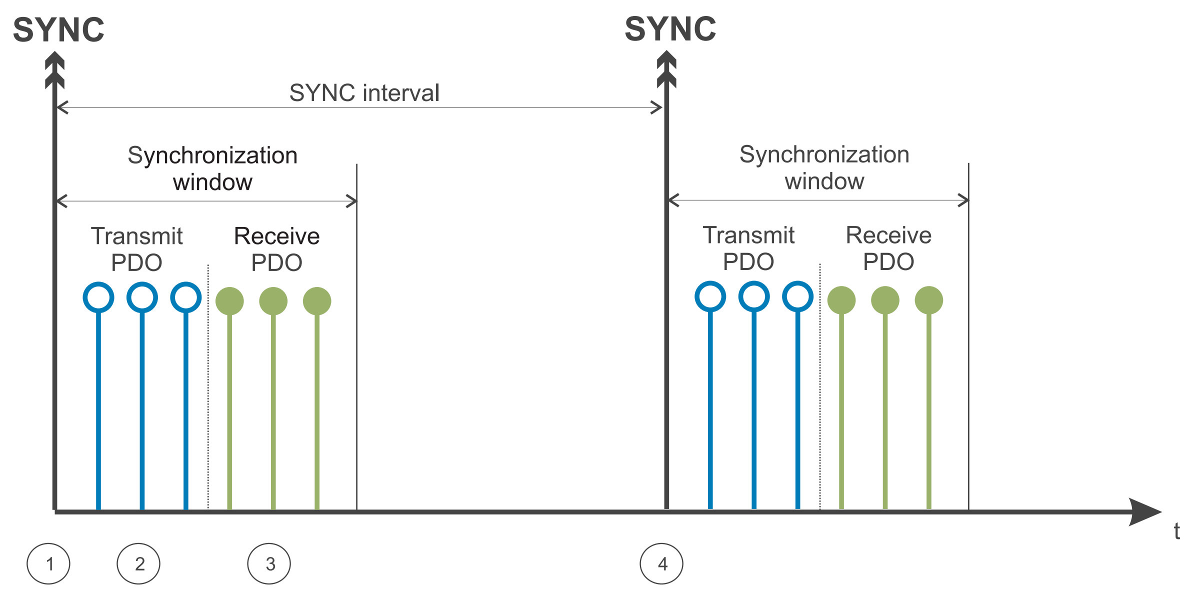


Figure 99: Synchronizing mechanism

After receiving SYNC message frame ①, a network node initially sends the transmit PDOs (TxPDOs) ②. After this, PDOs (RxPDOs) ③, which have been received up to this point are buffered. The system does not process the RxPDOs until the next SYNC message frame ④ is received. Synchronous PDOs must only occur within the synchronization window. Many network nodes offer the option of monitoring the SYNC interval and of responding appropriately if errors occur. In the same way, there are network nodes that do not just synchronize data transfer but also internal processes to the SYNC message frame. In both cases, you must set the SYNC interval in object 1006h on the network node:

SYNC interval:

Object	**Meaning**
1006 h	SYNC interval / duration of communication cycle.
Subindex 0	0: Not used n: Length of the SYNC interval in µs.

Note that many devices do not accept every value of object 1006h; rather this is limited to specific values (e.g. 1000, 2000, 4000, etc.)

By means of the transfer type of the PDO (see the chapter entitled ▹Configuring the transfer properties of a PDO◃ on page 110), you can also configure that a PDO is not taken until after the nth SYNC message frame. This allows you to set up short cycles for important data and long cycles for less important data.

4.4.10.2 Sending SYNC message frames

The CANopen-Master option module issues SYNC message frames by means of FB CANop405_SYNC. To use this function block, proceed as follows:

- You should implement FB CANop405_SYNC in a POU that is called at the same time interval as type CANop405_PDO_WRITE or CANop405_PDO_READ function blocks which transmit and receive synchronized PDOs. If a POU of this type is not present, create it with PLC type SH03-30 and processor BM4_O_PLC01. This POU will be called later in a timer event task.
- Interconnect the function block with variables of the correct data type. Generating the SYNC message frame is started with x_ENABLE = 1.
- If it is not yet present in your project, create a cyclical bypass IRP task. Link the created POU to FB CANop405_SYNC in this task.
- Compile the project and load it as a (boot) project on the PLC.
- Start the project.

The SYNC message frame is issued on x_ENABLE = 1. FB CANop405_SYNC indicates successful execution with x_CONFIRM = 1 and w_ERROR = 16#0. The system generates the SYNC message frame with x_ENABLE = 1 each time the function block is run through. It is not necessary to reset using x_ENABLE = 0.

4.4.10.3 Example: Sending SYNC message frames

We want to use option module BM4-O-CAN-03 to synchronize data exchange via PDOs that we created in chapters ▹4.4.9.6 Reading objects via PDOs ◃ from page 127 onward and ▹4.4.9.5 Writing objects via PDOs ◃ from page 115 onward. For this, the BM4-O-CAN-03 option module must initially be in the PRE-OPERATIONAL status. Set the type of transfer for both the second receive PDO (object 1401h) and the first transmit PDO (object 1800h) to synchronous. We want the PDOs to be taken or transmitted in every tenth SYNC message frame, i.e. you must enter the value 10 in sub-index 2 of objects 1401h and 1800h. As soon as you change back to OPERATIONAL status and enable transmission or reception of the PDOs, you will notice that parameters 1171 and 1040 in WinBASS II no longer change even though FB CANop405_PDO_WRITE is transmitting data. FB CANop405_PDO_READ no longer outputs x_CONFIRM = 1 either. This is normal, since synchronization is not taking place because you have not yet implemented FB CANop405_SYNC an it is not yet active.

You must, however, first configure a few values. For this, the BM4-O-CAN-03 option module must be back in the PRE-OPERATIONAL status. First of all, you must use parameter 531 to set the source of the SYNC signal on the b maXX controller. We "Use Sync 1 signal from the BACI". For this, the system must write to object 4213h (P531), sub-index 0 a value of 2 (data length 1 byte):

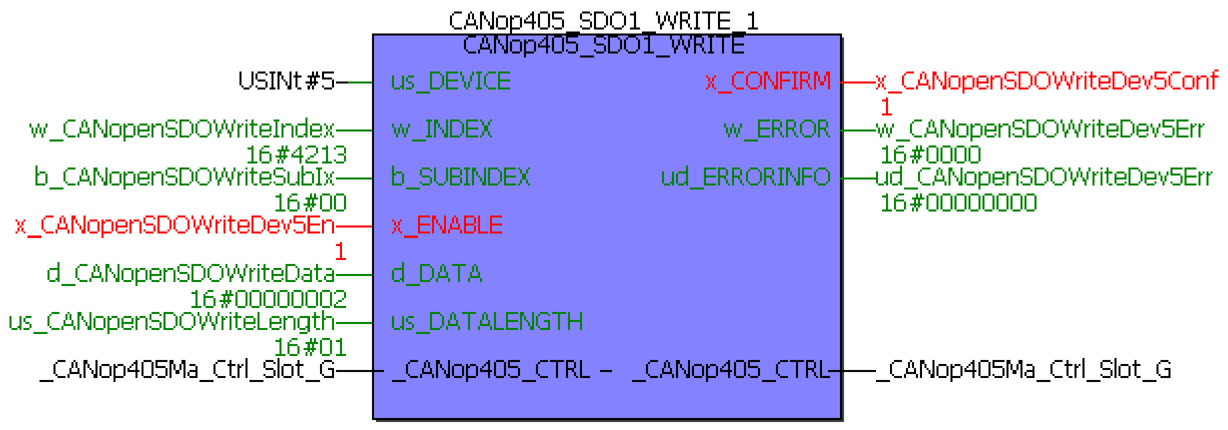


Figure 100: Setting the source of the SYNC signal on the b maXX controller

FB CANop405_SYNC is to be called every 2 ms (= 2000 µs = 7D0h µs). You must also set object 1006h (duration of communication cycle) to this value:

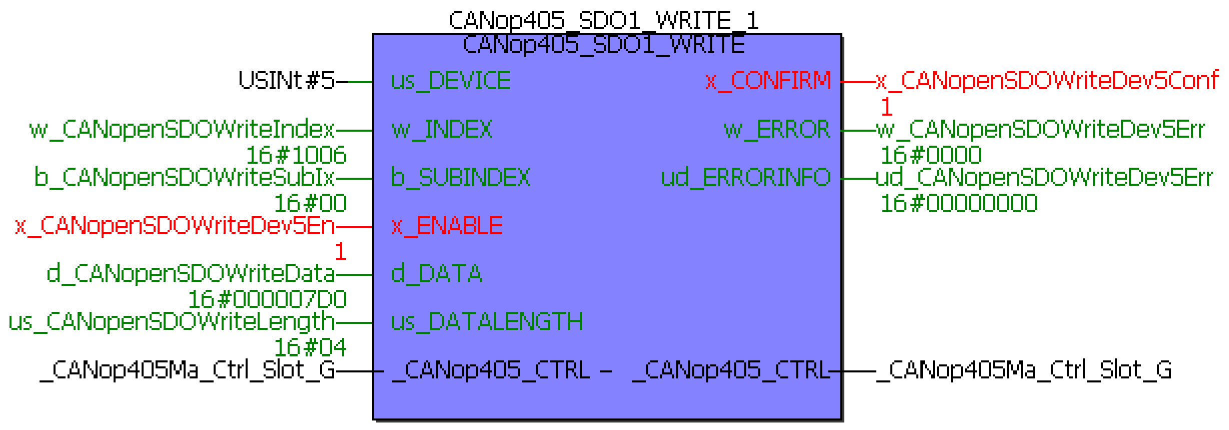
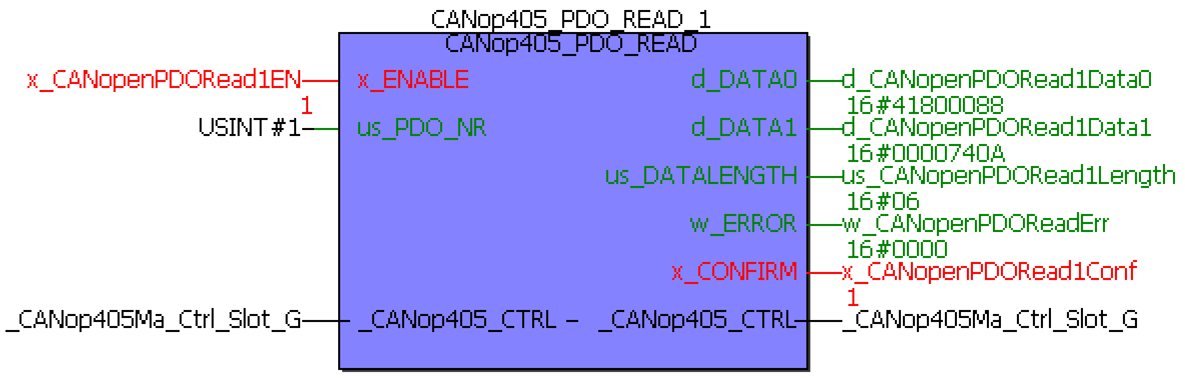


Figure 101: Setting the communication cycle

The system automatically applies this value to parameter 532 "SYNC interval". Using WinBASS II, save the b maXX controller's data record so that you do not lose the set configuration when you switch the unit off.

We will implement FB CANop405_SYNC below FB CANop405_PDO_READ. This means that it is automatically called every 2 ms. Compile the project and send it as a project (not as a boot project) to the PLC. Start the project. If you sent a boot project and restarted the b maXX 4400 unit, you will lose the previously set SYNC configuration of the b maXX controller if you did not save the controller's data record. In this case, you must carry out configuration again.

After starting, you must put the BM4-O-CAN-03 option module back in the OPERATIONAL status. Enable FBs CANop405_PDO_READ and CANop405_PDO_WRITE. FB CANop405_PDO_READ should not yet receive any data. However, as soon as FB CANop405_SYNC is also enabled, it receives PDOs.



```
(* Generation of the SYNC command *)
```

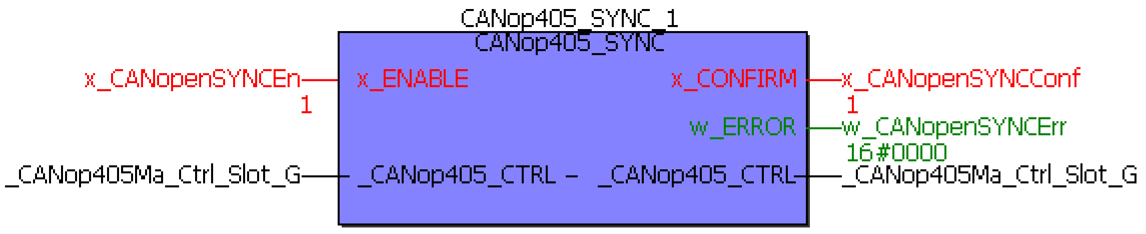


Figure 102: Generating the SYNC message frame using FB CANop405_SYNC

Note that these are snapshots in online mode and that your values may be different.

4.4.11 Monitoring network nodes by Node Guarding

4.4.11.1 Definition according to the CANopen specification

There are two mechanisms for monitoring network node failures, i.e. Node Guarding and Life Guarding. By means of Node Guarding, the CANopen-Master monitors the network nodes, which can use Life Guarding to detect a CANopen-Master failure. Both mechanisms are based on the same functionality:

In the case of Node Guarding, the master uses a remote message frame to request a response from the slave at specific time intervals (the Guard Time). The slave responds with the guarding message. This contains the status of the slave's node (see the chapter entitled ▹4.4.7 Starting the individual network node NMT ◃ from page 101 onward) as well as a toggle bit that must change after each message. If the status or the toggle bit does not match what the master expects or if there is no response within the lifetime period the master of a slave issues an error. The slave can detect a master failure. If the slave does not receive a message request from the master within the set lifetime, it assumes that the master has failed and sets its outputs to the error status, for example, and issues an emergency message frame.

Two objects are important on the network node for setting the Guarding functionality:

Guarding Time:

Object	Meaning
100Ch	Configuration of Guarding Time.
Subindex 0	0: Guarding is not active n: The distance between two guarding message frames in ms

Life Time Factor:

Object	Meaning
100Dh	Configuration of the Life Time Factor.
Subindex 0	0: Life Guarding is not active n: The Life Time results from n x Guard Time

Node Guarding is available at all stages of communication (STOPPED, PRE-OPERATIONAL, OPERATIONAL). The toggle bit is only reset to its default value at the INITIALIZATION stage. This means that the toggle mechanism is continued in the case of changes of status too. Node Guarding is started in the slave after receiving the first Guarding request message frame. From this time onward, the monitoring time runs in the slave that is parameterized in objects 100Ch and 100Dh.

4.4.11.2 Node Guarding

Node Guarding by the CANopen-Master option module is carried out by means of FB CANop405_NODE_GUARDING. To use this function block, proceed as follows:

- You must implement FB CANop405_NODE_GUARDING in a POU with PLC type SH03-30 and processor BM4_O_PLC01.
- Interconnect the function block with variables of the correct data type. At us_DEVICE, enter the number of the node to be monitored; at. t_NODE_GUARD_TIME, enter the Guarding time; and at us_LIFE_TIME_FACTOR enter the Life Time factor.
- Create a task in which you link the POU with FB CANop405_NODE_GUARDING. This task's cycle time depends on how quickly you want to detect a network node failure. The Node Guarding mechanism itself runs on the CANopen-Master option module. This means that it is not necessary to call FB CANop405_NODE_GUARDING in the time interval of the Guard Time.
- Compile the project and load it as a (boot) project on the PLC.
- Configure the Guard Time (object 100Ch) and the Life Time Factor (object 100Dh) on the individual network nodes. To do this, use the function blocks for SDO communication.

Node Guarding is started on x_ENABLE = 1. FB CANop405_NODE_GUARDING signals x_NODE_OK = 1 as soon as the first valid response arrives from the slave. While valid responses keep coming from the slave, x_NODE_OK stays = 1. The slave's status is output at us_NODE_STATE. FB CANop405_ NODE_GUARDING is the only way to read out a node status. If an invalid response comes from the slave or there is no response within the lifetime, the system issues an error message and x_NODE_OK changes to 0.

4.4.11.3 Example: Monitoring a network node using Node Guarding

We want to monitor the BM4-O-CAN-03 option module for failure at 500-ms intervals using Node Guarding. To be able to set the Node Guarding time and the Life Time Factor, the BM4-O-CAN-03 option module must initially be in the PRE-OPERATIONAL status. We now set object 100Ch to 500 ms:

```
(* Write object via SDO to device nr. 5 *)
```

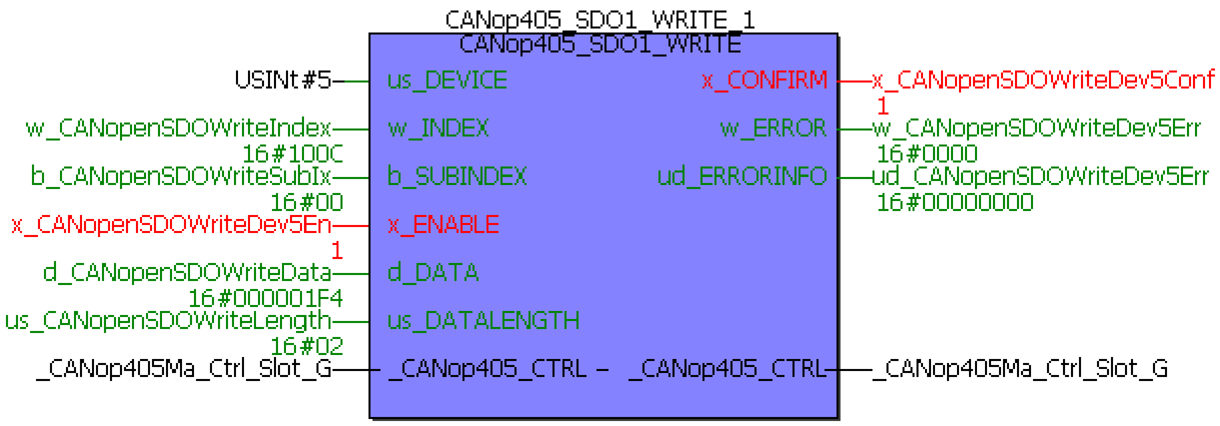


Figure 103: Setting the Node Guarding time on a CANopen-Slave

For the Life Time Factor, object 100Dh, we choose a value of 3:

```
(* Write object via SDO to device nr. 5 *)
```

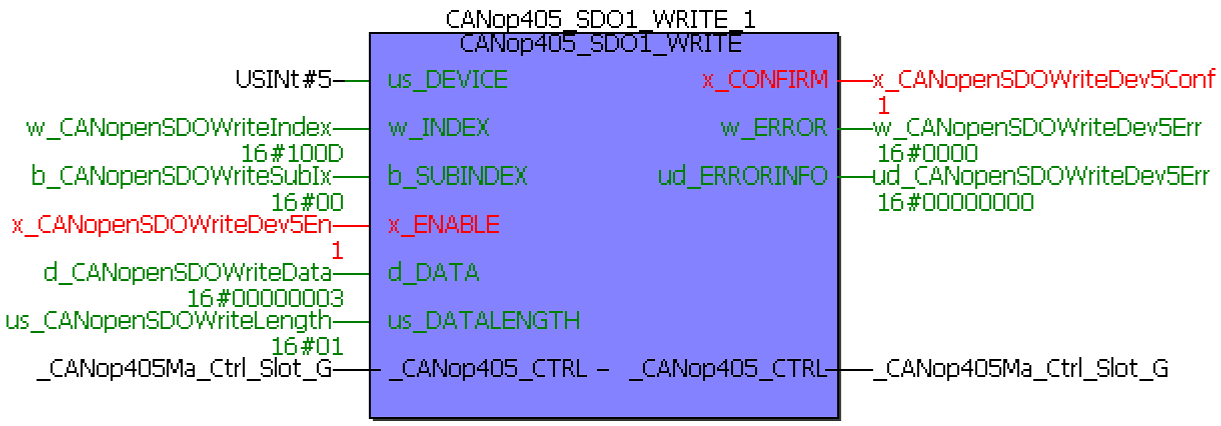


Figure 104: Setting the Life Time Factor on a CANopen-Slave

We implement FB CANop405_NODE_GUARDING in the POU using the function blocks for SDO communication. The time interval of the task that is assigned to this POU is more than adequate for our purposes to detect a node failure. After interconnecting the function block, compile the project and send it as a project (not as a boot project) to the PLC. Start the project. If you sent a boot project and restarted the b maXX 4400 unit, you will lose the previously set Node Guarding configuration. In this case, you must carry out configuration again, since the BM4-O-CAN-03 option module sets the Guard Time and the Life Time Factor to the default values at starting. Node Guarding is possible in the PRE-OPERATIONAL and OPERATIONAL status. In online mode, we can start Node Guarding with x_ENABLE = 1.

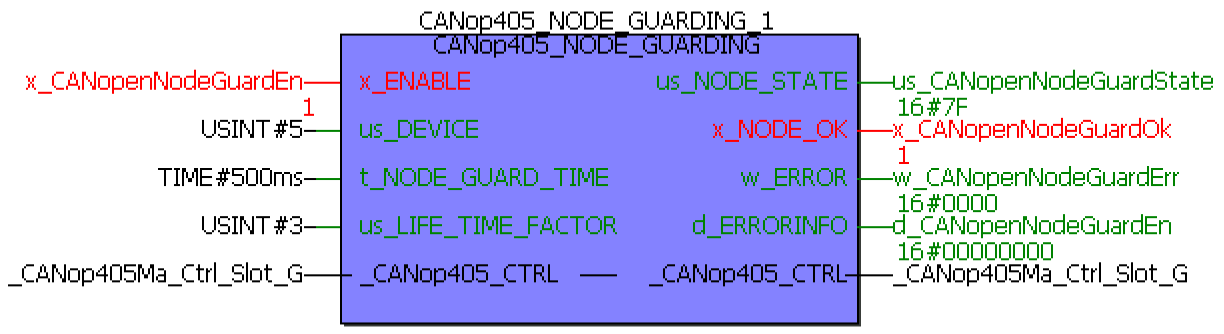


Figure 105: Node Guarding using FB CANop405_NODE_GUARDING

In this case, BM4-O-CAN-03 option module reports with the PRE-OPERATIONAL status (7Fh). By unplugging the CAN cable, we can generate the "Toggle bit error" message:

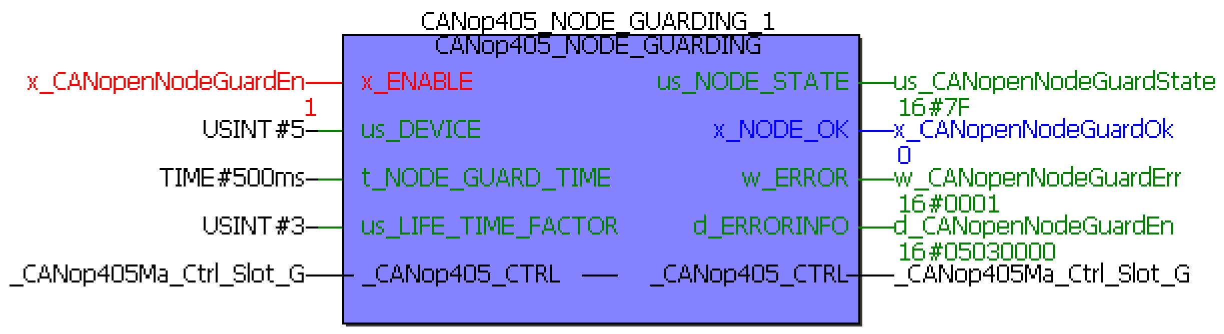


Figure 106: Node Guarding "Toggle bit error"

4.4.12 Receiving emergency message frames

4.4.12.1 Definition according to the CANopen specification

Network nodes use emergency message frames to report errors. Emergency message frames are transferred according to the consumer/producer communication model, i.e. a network node generates an emergency message frame that is processed by one or more other network node(s). They do not confirm receiving the message frame. A new emergency message frame is generated on a one-time basis for each additional error. The message frame is not repeated. The CANopen-Master option module can evaluate emergency message frames from network nodes. It is not possible to send an emergency message frame.

The user data area of an emergency message frame is divided into three sections:

1 Emergency Error Code, 2 bytes

2 Error Register, 1 byte

3 Manufacturer-specific error code, 5 bytes

The *Emergency Error Code* has the following meaning according to the CiA:

Error Code	**Meaning**
00xxh	Error Reset or No Error
10xxh	Generic error
20xxh	Current
21xxh	Current, device input side
22xxh	Current inside the device
23xxh	Current, device output side
30xxh	Voltage
31xxh	Mains Voltage
32xxh	Voltage inside the device
33xxh	Output Voltage
40xxh	Temperature
41xxh	Ambient Temperature
42xxh	Device Temperature
50xxh	Device Hardware
60xxh	Device Software
61xxh	Internal Software
62xxh	User Software
63xxh	Data Set
70xxh	Additional Modules
80xxh	Monitoring
81xxh	Communication
8110 h	CAN Overrun (Objects lost)
8120 h	CAN in Error Passive Mode
8130 h	Life Guard Error or Heartbeat Error
8140 h	Recovered from bus off
8150 h	Transmit COB-ID
82xxh	Protocol Error
8210 h	PDO not processed due to length error
8220 h	PDO length exceeded

Error Code	Meaning
90xxh	External Error
F0xxh	Additional Functions
FFxxh	Device-specific

xx = Not defined

The Error Register contains object 1001h. A network node can enter an internal error in object 1001h. According to CiA, the coding looks like this:

Bit	Meaning
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error
5	Device profile-specific
6	Reserved
7	Manufacturer-specific

Five bytes are available for a manufacturer-specific error code to further-characterize errors. Due to the vast range of different options, coding of errors is highly device-specific; this means that you should refer to the device's documentation in each case. Note also that most of the entries in Emergency Error Code and Error Register are not binding. In this case, too, refer to the device's documentation to find out which entries are supported.

4.4.12.2 Evaluating emergency message frames

Emergency message frames due to the CANopen-Master option module are evaluated by FB CANop405_EMERGENCY. To use this function block, proceed as follows:

- You must implement FB CANop405_EMERGENCY in a POU with PLC type SH03-30 and processor BM4_O_PLC01.
- Interconnect the function block with variables of the correct data type. At us_DEVICE, enter the number of the node to be monitored. If an emergency message frame is received, the system outputs the received Emergency Error Code at w_EMCY_ERROR_CODE, the Error Register at b_ERROR_REGISTER and the manufacturer-specific error information in array a_ERROR_FIELD. If the received error code >< 0, the system sets output x_EMERGENCY to 1. x_RESET = 1 resets the outputs.
- Create a task in which you link the POU with FB CANop405_EMERGENCY. This task's cycle time depends on how quickly you want to detect a received emergency message frame.
- Compile the project and load it as a (boot) project on the PLC.

The system indicates a received emergency message frame with x_EMERGENCY = 1. You do not need to separately enable the function block: it is automatically active.

4.4.12.3 Example: Receiving an emergency message frame

We want the BM4-O-CAN-03 option module to transmit an emergency message frame and then evaluate it. One easy way to do this is simply to deactivate active Node Guarding. After this, the BM4-O-CAN-03 option module must generate a Life Guarding error. According to the CiA definition (DS 301) and the documentation of the option module BM4-O-CAN-03, a Life Guarding error returns the following values:

Emergency Error Code:	8130h
Error Register:	At least bits 0 and 4 set
Manufacturer Code:	0h, since not a b maXX unit error

We implement FB CANop405_EMERGENCY in the POU using the function blocks for SDO communication. After interconnecting the function block, compile the project and send it as a boot project to the PLC. Restart the b maXX 4400 unit. Activate Node Guarding as described in the chapter entitled "Node Guarding". Do not forget to set objects 100Ch (Guard Time) and 100Dh (Life Time Factor) via SDO. Activate Node Guarding. It may well be that FB CANop405_EMERGENCY displays an earlier emergency message frame. If this is the case, reset the function block by setting x_RESET briefly to 1. Now switch off Node Guarding at FB CANop405_NODE_GUARDING using x_ENABLE = 0. Directly after this, FB CANop405_EMERGENCY should display that it has received the emergency message frame:

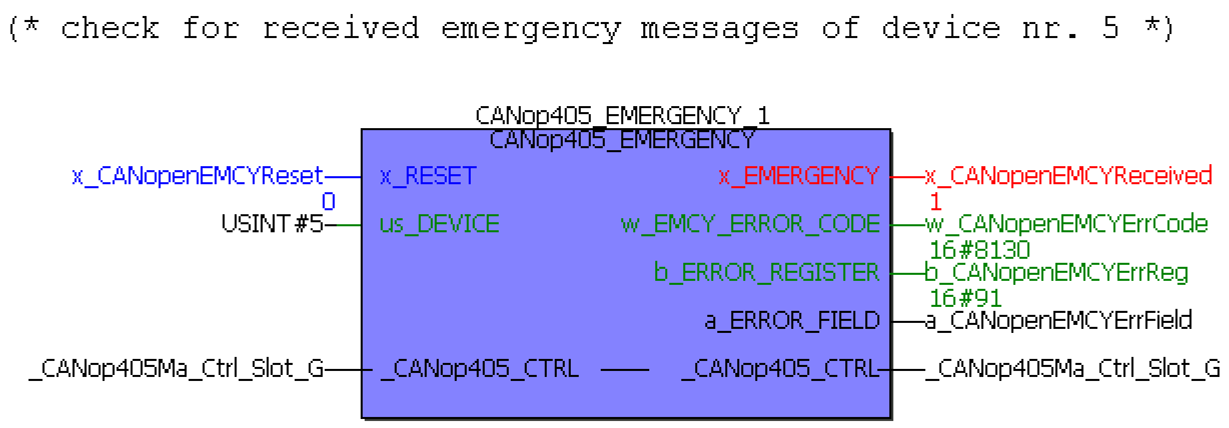
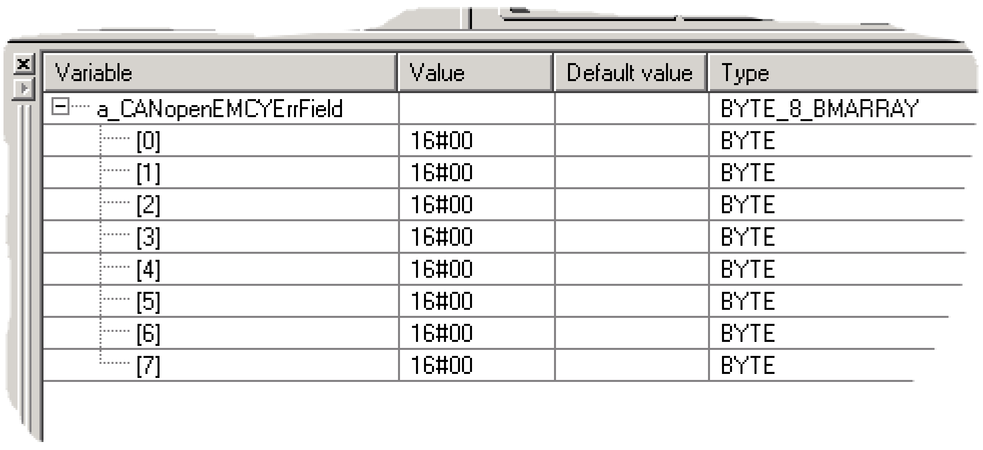


Figure 107: Evaluating emergency message frames using FB CANop405_EMERGENCY



Variable	Value	Default value	Type
a_CANopenEMCYErrField			BYTE_8_BMARRAY
[0]	16#00		BYTE
[1]	16#00		BYTE
[2]	16#00		BYTE
[3]	16#00		BYTE
[4]	16#00		BYTE
[5]	16#00		BYTE
[6]	16#00		BYTE
[7]	16#00		BYTE

Figure 108: a_ERROR_FIELD in PROPROG wt II's watch window

We observe the value of a_ERROR_FIELD in PROPROG wt II's watch window. The displayed values match the ones that we expected. Using x_RESET = 1, you can reset the function block and with this receive the next emergency message frame:

```
(* check for received emergency messages of device nr. 5 *)
```

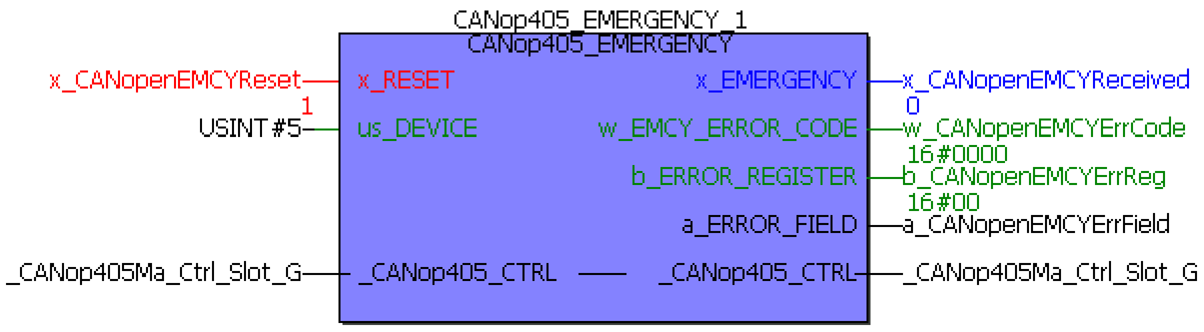


Figure 109: Resetting FB CANop405_EMERGENCY

4.5 CANopen and Motion Control with PROPROG wt II and motion configurator

4.5.1 General information on the CANopen-Master option module and Motion Control

If you intend using the CANopen-Master option module together with Motion Control, you must never operate a node on the CANopen network by means of function blocks from library CANop405_PLC01_20bd00 (or higher) together with Motion Control. This means that a network node is operated either via the function blocks from library CANop405_PLC01_20bd00 (or higher) or by Motion Control.

4.5.2 Initializing the CANopen-Master option module

When using Motion Control, it initializes CANopen-Master option module. You must no longer use function block CANop405_INIT.

4.5.3 Data exchange via PDOs

When you use the CANopen-Master option module together with Motion Control, the number of PDOs to be written is reduced from 40 by the number of axes that Motion Control operates:

PDOs to be written $\leq$ 40 - number of Motion Control nodes

This also reduces to this value the number of transmit slots that can be assigned. The number of PDOs to be received is also reduced. The maximum number of PDOs to be received reduces from 63 by double the number of axes that Motion Control operates:

PDOs to be received $\leq$ 63 - 2 × number of Motion Control nodes

This also reduces to this value the number of receive slots that can be assigned.

The Motion Control configurator considers this reduction in the PDOs. Note this value when configuring your application.

4.5.4 Synchronizing data exchange via PDOs

When using Motion Control, it generates the SYNC message frame for data exchange of PDOs. You must no longer use function block CANop405_SYNC. You can of course integrate non-Motion Control nodes in synchronized data exchange by configuring appropriately.

4.6 Structure of CANopen message frames

4.6.1 General

Due to the CANopen-Master option module and the function blocks from library CANop405_PLC01_20bd00 (and higher), message frame traffic is completely isolated from you and you need virtually no knowledge of the structure of the individual CANopen message frames. However, if you want to use a CAN analysis tool to inspect data traffic, you will find the structures of the individual message frames in the next few chapters. The message frames are explained in the following format

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

You will find this structure in most CAN analysis tools; in this connection, data range byte 0 to byte 7 may be adapted to the respective CANopen message frame. RTR designates the bit for a Remote Transmit Request. Field DLC consists of four bits and designates the number of data bytes (DLC = Data Length Code).

4.6.2 NMT message frame

Two data bytes are transferred per message frame. Data byte 0 contains the Command Specifier, CS, with data byte 1 containing the device address. If address 0 is entered, the corresponding command is directed to all the nodes (broadcast).

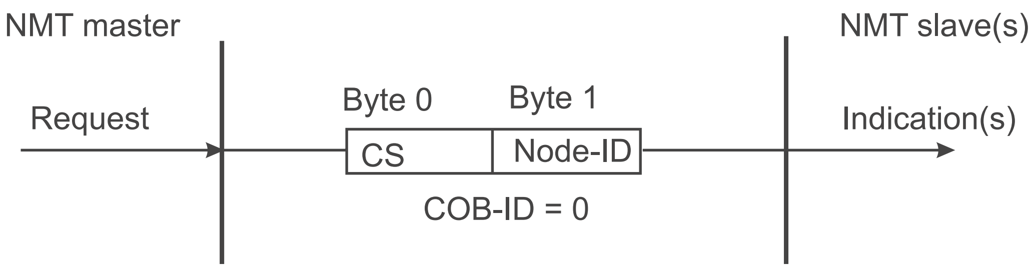


Figure 110: NMT communications relationship

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

A message frame that sets node 16 to configuration mode looks like this:

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00 h	0	2 h	80 h	10 h						

These message frames are not confirmed, i.e. no NMT slave acknowledges the correctly received message to the NMT master.

4.6.3 SDO message frame

4.6.3.1 Structure of an SDO message frame

The COB-ID of the request SDO (request from the master) results from 600h + the address; in the case of response SDOs (response from the slave), it results from 580h + the address. The data field of the CAN data message frame (8 bytes) for an SDO is divided into three sections: a Command Specifier, CS (1 byte), a Multiplexor M (3 bytes) and the actual user data area D0 - D4 (4 bytes).

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
600h/ 580h + address	0	8 h	CS	M			D0	D1	D2	D3

The Multiplexor M consists of the 16-bit index of an object and the associated eight-bit wide sub-index. In the case of segmented message frames (not supported by the CANopen-Master option module), the user data area is extended by the Multiplexor's three bytes, that can transfer seven bytes of user data per message frame. Command Specifier CS classifies the different SDO types.

4.6.3.2 Types of SDO transfer

The CANopen-Master option module supports Expedited Transfer according to DS 301. It is possible to read or write objects whose data comprises a maximum of four bytes. Only two message frames are necessary, i.e. a request and a response.

4.6.3.3 Writing an object via SDO

The CANopen-Master option module transmits (client) a write request to the slave (server). The slave carries out the request and acknowledges it with the response.

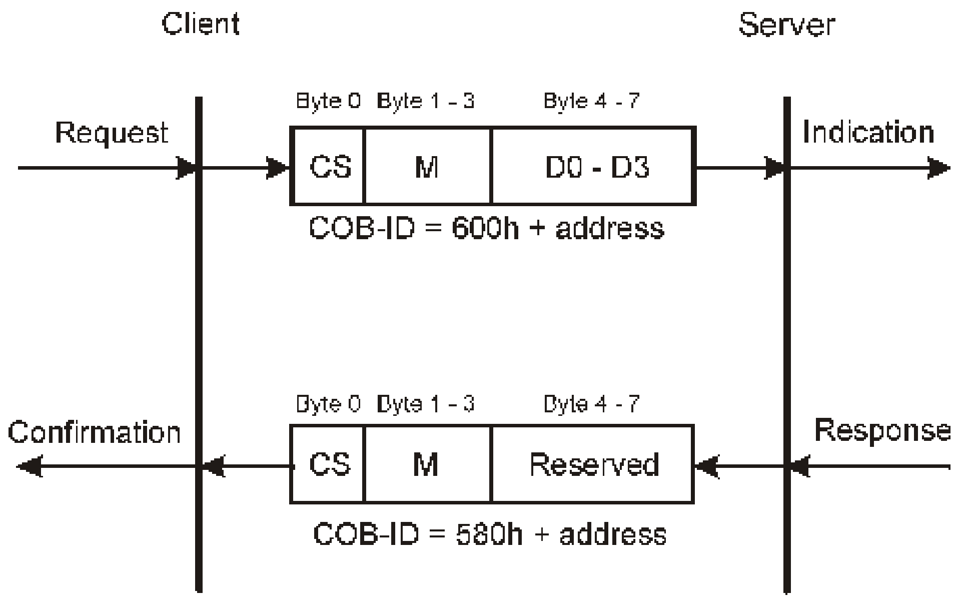


Figure 111: Writing a communications relationship SDO

The Command Specifier CS for the request depends on the length of the user data. D0 is the LSB, D3 is the MSB.

Data length in D0 - D3	Command Specifier CS
1 byte	2Fh
2 bytes	2Bh
4 bytes	23 h

The Command Specifier CS for the response is 60h, the Multiplexor M is identical with that of the request, the data field is meaningless (reserved).

Example:

The value "-3" (FDh) is to be written to object 6060h, sub-index 00h of the slave with address 4. This object's data width is eight bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	2Fh	60 h	60 h	00 h	FDh	00 h	00 h	00 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	2Fh	60 h	60 h	00 h	00 h	00 h	00 h	00 h

The value "12" (0Ch) is to be written to object 43E9h, sub-index 00h of the slave with address 4. This object's data width is 16 bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	2Bh	E9h	43 h	00 h	0Ch	00 h	00 h	00 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	60 h	E9h	43 h	00 h	00 h	00 h	00 h	00 h

The value "60610008h" is to be written to object 1800h, sub-index 02h of the slave with address 4. This object's data width is 32 bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	23 h	00 h	18 h	02 h	08 h	00 h	61 h	60 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	60 h	00 h	18 h	02 h	00 h	00 h	00 h	00 h

4.6.3.4 Reading an object via SDO

The CANopen-Master option module transmits (client) a read request to the slave (server). The slave carries out the request and acknowledges it with the response that also contains the data.

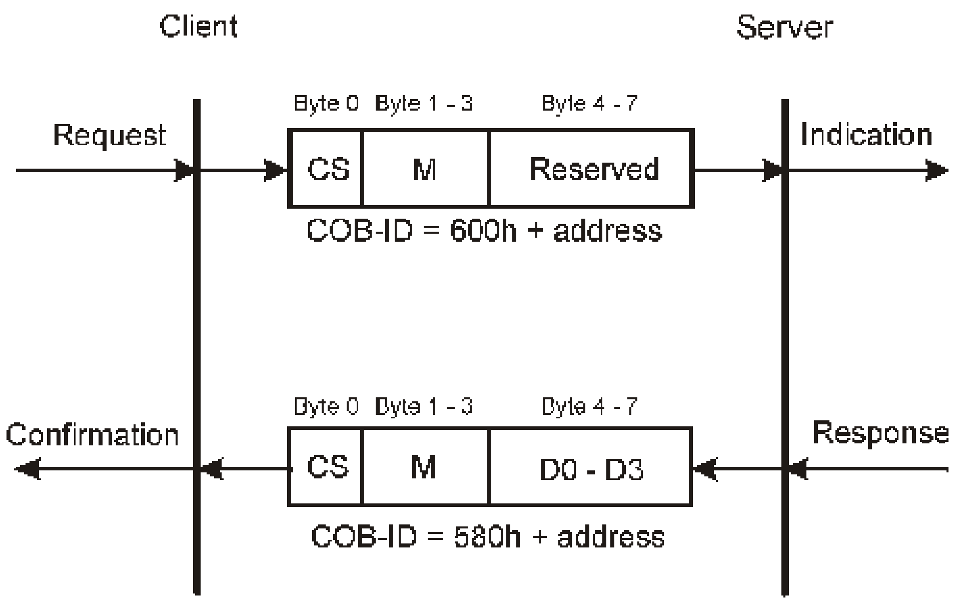


Figure 112: Reading communications relationship SDO

An SDO server (master) transmits a read request to the slave. This slave carries out the request and transmits the requested data in the response message frame. Command Specifier CS for the request is always 40h. the Command Specifier CS for the response depends on the length of the user data. D0 is the LSB, D3 is the MSB.

Data length in D0 - D3	Command Specifier CS
1 byte	4Fh
2 bytes	4Bh
4 bytes	43 h

The Multiplexor of the request and the response matches.

Example:

Object 6061h, sub-index 00h of the slave with address 4 is to be read. This object's data width is eight bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	40 h	61 h	60 h	00 h	00 h	00 h	00 h	00 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	4Fh	61 h	60 h	00 h	D0	00 h	00 h	00 h

Object 6041h, sub-index 00h of the slave with address 4 is to be read. This object's data width is 16 bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	40 h	41 h	60 h	00 h	00 h	00 h	00 h	00 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	4Bh	41 h	60 h	00 h	D0	D1	00 h	00 h

Object 1400h, sub-index 01h of the slave with address 4 is to be read. This object's data width is 32 bits.

Request:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
604 h	0	8 h	40 h	00 h	14 h	01 h	00 h	00 h	00 h	00 h

Response:

			CS	Multiplexor			D0	D1	D2	D3
COB-ID	**RTR**	**DLC**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
584 h	0	8 h	43 h	00 h	14 h	01 h	D0	D1	D2	D3

4.6.4 PDO message frame

A PDO producer transmits the process data in a PDO to the CANopen network. Depending on the configuration, one or more PDO consumers evaluate(s) the PDO.

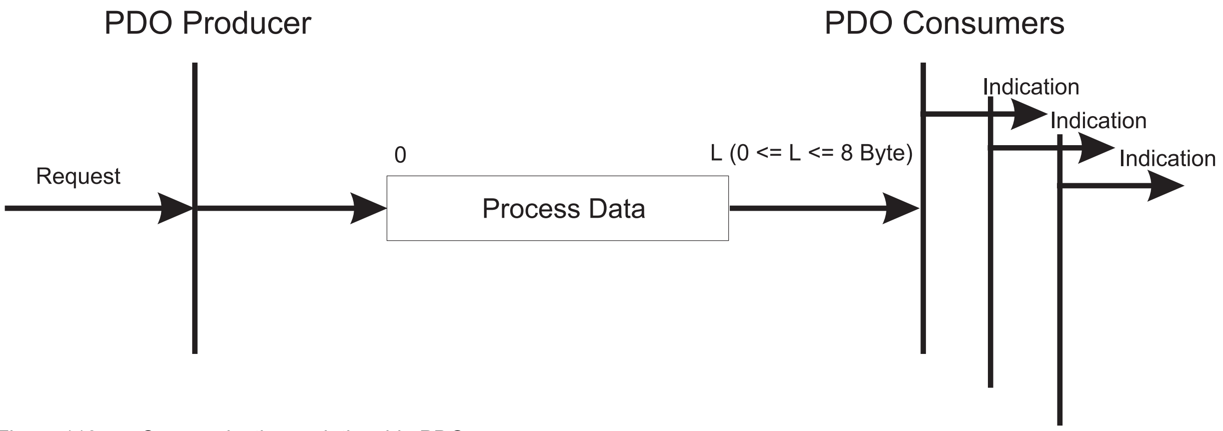


Figure 113: Communications relationship PDO

All eight bytes of the data range in the CAN data message frame are available to PDO message frames. The contents result from the settings of the Mapping objects. For the COB-ID, values result from 181h to 57Fh, assuming that the COB-ID was assigned according to the predefined CiA identifier assignment. Apart from this, you can freely set the COB-ID too.

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
181h - 57Fh or any	0	0h - 8h	D0	D1	D2	D3	D4	D5	D6	D7

In the case of remote requests from PDO message frames, the COB-ID of the PDO request and of the PDO response have the same value. In the PDO request, the RTR bit (which is a component of the DLC) is set. The data bytes have no contents and this means that the DLC field is 0.

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
181h - 57Fh or any	1	0 h	D0	D1	D2	D3	D4	D5	D6	D7

4.6.5 SYNC message frame

A SYNC producer transmits the SYNC message frame to the CANopen network. Depending on the configuration, one or more PDO consumers evaluate(s) the message frame.

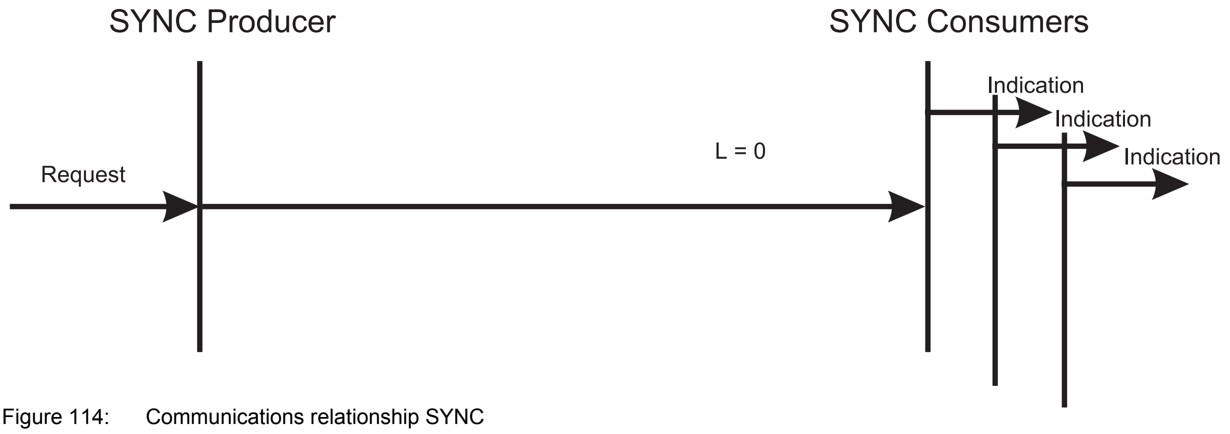


Figure 114: Communications relationship SYNC

The SYNC message frame contains no data. The COB-ID is 80h according to the predefined CiA identifier assignment. Apart from this, you can freely set the COB-ID too.

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80h or any	0 h	0 h	-	-	-	-	-	-	-	-

4.6.6 Node Guarding message frame

The master uses remote frames to poll the slaves at specific intervals.

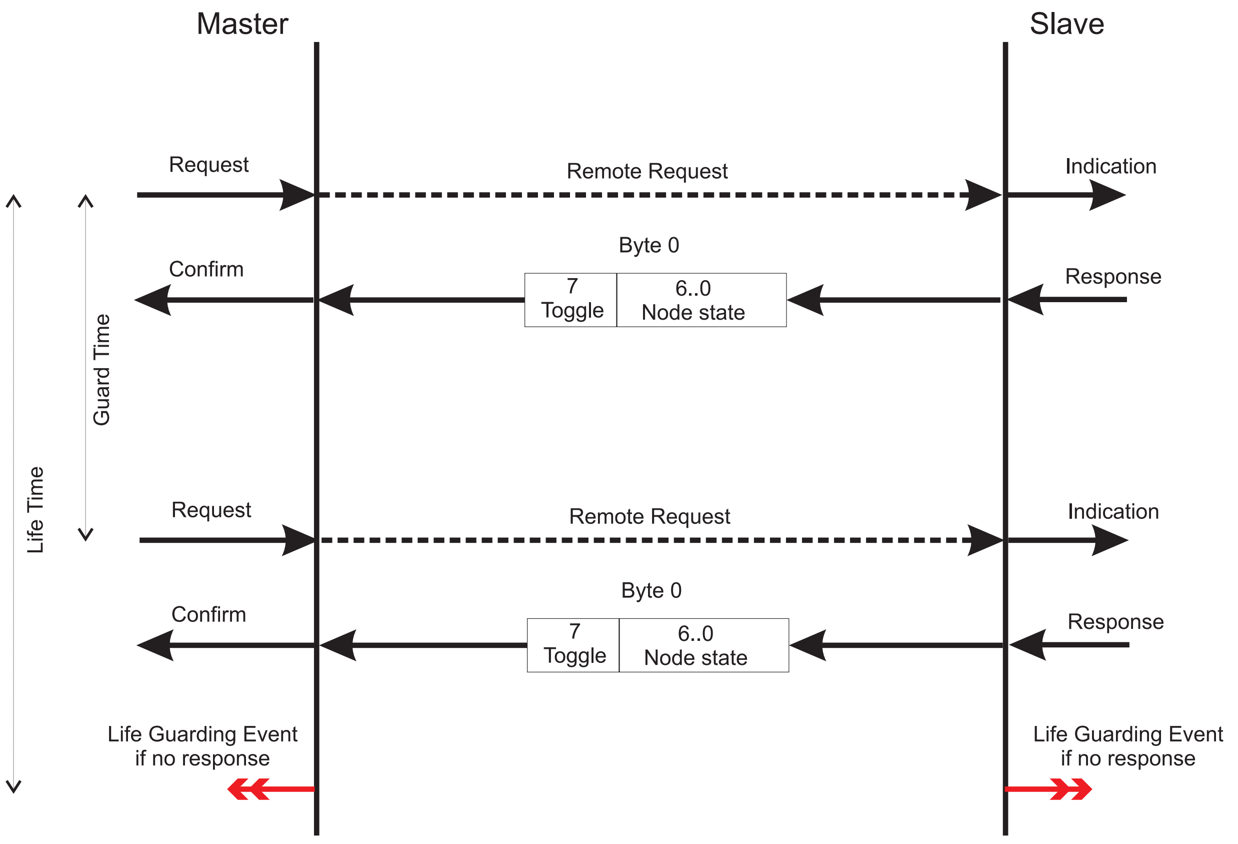


Figure 115: Communications relationship Node Guarding

The COB-ID results from 700h + the address. In the response message frame, the data field of the CAN data message frame consists of one byte, i.e. byte 0. Byte 0 is divided into the toggle bit (bit 7) and the node state (bits 6..0).

COB-ID	**RTR**	**DLC**	**Byte 0**
700h + address	0	1 h	Toggle bit + node state

The remote request does not contain any data. The RTR bit is set.

COB-ID	**RTR**	**DLC**	**Byte 0**
700h + address	1	0 h	-

4.6.7 Emergency message frame

An emergency producer transmits the emergency message frame to the CANopen network. Depending on the configuration, one or more emergency consumers evaluate(s) the message frame.

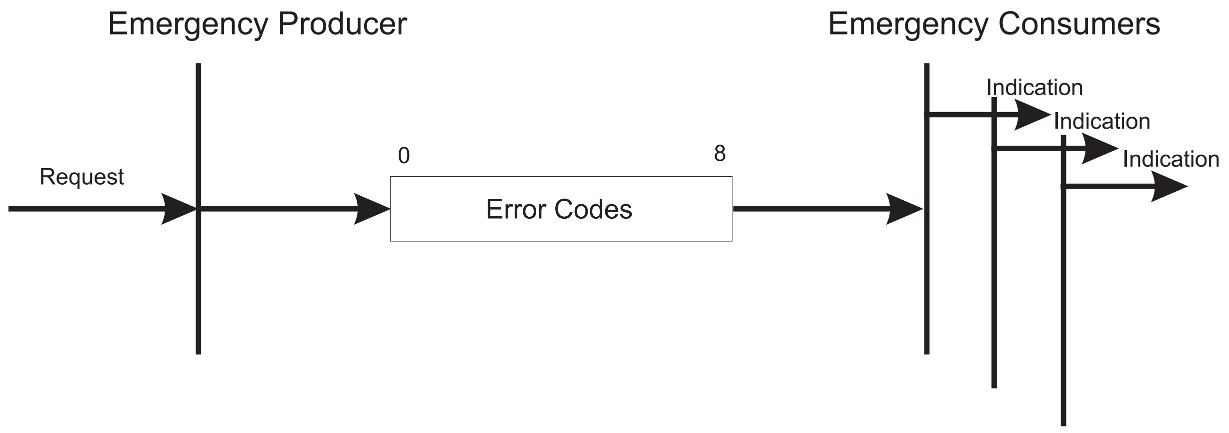


Figure 116: Communications relationship emergency

The COB-ID results from 80h + the address. All eight bytes of the CAN data message frame are available to emergency message frames. The eight bytes are divided as follows:

COB-ID	RTR	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
80h + address	0 h	8 h	Emergency Error Code		Error Reg.	Manufacturer-specific				

Appendix A - Abbreviations

API	Applications Programming Interface
ARP	Address Resolution Protocol
BACI	Baumüller Component Interface
BUB	Ballast unit
BUC	Baumüller feed/return feed unit
BUG	Baumüller converter basic feed unit
BUM	Baumüller individual power unit
BUS	Baumüller power module
CAL	CAN Application Layer
CAN	Controller Area Network
CiA	CAN in Automation
COB	Communication Object
COB-ID	Communication Object Identifier
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CPU	Central Processing Unit
DC	d.c. current
DCF	Device Configuration File
DHCP	Dynamic Host Configuration Protocol
DIN	Deutsches Institut für Normung e.V. (German Standards Institute)
DP-RAM	Dual-port RAM
DR	Draft Recommendation
DS	Draft Standard
DSP	Draft Standard Proposal
EDS	Electronic Data Sheet
EMC	Electromagnetic compatibility
EN	European standard
EPROM	Erasable Programmable Read Only Memory
ESD	Electrostatic Sensitive Device
FTP	File Transfer Protocol
HD	Hamming distance
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IANA	Internet Assigned Numbers Authority (www.iana.org)
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IRP	Interrupt
ISO	International Organization for Standardization
LAN	Local Area Network
LED	Light Emitting Diode
LSS	Layer Setting Services
MAC	Media Access Control
OSI	Open Systems Interconnect
PDD	Process Data Directory
PDO	Process Data Object
PLC	Process Loop Controller (PLC)
RAM	Random Access Memory
SAP	Service Access Point
SDO	Service Data Object
SMS	Short Message System
SMTP	Simple Mail Transfer Protocol
PLC	Programmable Logic Controller
SRD	SDO Requesting Device

SRDO	Safety Relevant Data Object
TCP	Transport Control Protocol
Telnet	Terminal over Network
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USS	USS protocol function module
USS®	Trademark of Siemens, universal serial interface
VDE	Verband deutscher Elektrotechniker (German Association of Electrical Engineers)
WWW	World Wide Web
16#	Prefix for hexadecimal numbers

B

APPENDIX B - TABLES

B.1 CANopen objects according to DS 301

The following is a list of frequently used objects pursuant to the CANopen communication profile DS 301. Note that the CANopen master option module with software version < **01.20** (i.e. BM4-O-ETH-02/CAN-04-01-00-001-**000**) itself does not contain these object.

In conjunction with ProMaster, the CANopen master option module has its own CANopen objects. An overview of the objects of the CANopen master option module with software version ≥ **01.20** (i.e. ≥ BM4-O-ETH-02/CAN-04-01-00-001-**001**) is found in ▷B.3 CANopen objects of the CANopen masters (software version ≥ 01.20)◁ from page 177 onward.

Note that the CANopen master option module with software version < **01.20** itself does not contain any objects.

Overview of the objects:

Index	Object content / meaning
1000h	Device type: Supported device profile and additional information
1001h	Error register
1003h	Error memory contains an error list of the unit
1005h	SYNC message frame COB-ID
1006h	SYNC interval
100Ch	Guarding Time
100Dh	Life Time Factor
1010h	Store parameter
1011h	Load default parameter
1400h - 15FFh	Receive PDO (RxPDO) communications parameter
1600h - 17FFh	Receive PDO (RxPDO) mapping parameter
1800h - 19FFh	Send PDO (TxPDO) communications parameter
1A00h - 1BFFh	Send PDO (TxPDO) mapping parameter

CANopen objects according to DS 301

1000h - Unit type

Displays the supported unit profile and additional information about the unit.

Index	Subindex	Name	Data type	Access	Category	Mappable
1000h	0 h	Device Type	Unsigned32	ro	Mandatory	No

The 32-bit value is divided into two 16-bit values:

Byte:	MSB	LSB
	Additional information, manufacturer-dependent	Device-profile

1001h - Error register

Contains a bit strip with pending device errors

Index	Subindex	Name	Data type	Access	Category	Mappable
1001h	0 h	Error Register	Unsigned8	ro	Mandatory	Optional

The individual bits of the 8-Bit values have the following meanings:

Bit	Meaning	Category
0	Generic error	Mandatory
1	Current	Optional
2	Voltage	Optional
3	Temperature	Optional
4	Communication error	Optional
5	Device profile-specific	Optional
6	Reserved	Optional
7	Manufacturer-specific	Optional

If a bit is set, the error occurred.

1003h - Error memory

Contains an error list of the device in descending time order.

Index	Subindex	Name	Data type	Access	Category	Mappable
1003h	-	Pre-defined error field	ARRAY	-	Optional	-
	0 h	Number of errors	Unsigned8	rw	Mandatory	No
	1h - FEh	Standard error field	Unsigned32	ro	Optional	No

Subindex 0h contains the number of entered errors. You can clear the error list by writing a value of 0 to subindex 0h. Subindex 1h contains the current errors. If a new error occurs, the system enters it at subindex 1h and all the others go down by one subindex. The 32-bit value of the standard error field is divided into two 16-bit values:

Byte:	MSB	LSB
	Additional information, manufacturer-dependent	Error code

The error code results from the definition of the *Emergency Error Code* in the emergency message frame:

Error Code	Meaning
00xxh	Error Reset or No Error
10xxh	Generic error
20xxh	Current
21xxh	Current, device input side
22xxh	Current inside the device
23xxh	Current, device output side
30xxh	Voltage
31xxh	Mains Voltage
32xxh	Voltage inside the device
33xxh	Output Voltage
40xxh	Temperature
41xxh	Ambient Temperature
42xxh	Device Temperature
50xxh	Device Hardware
60xxh	Device Software
61xxh	Internal Software
62xxh	User Software

Error Code	Meaning
63xxh	Data Set
70xxh	Additional Modules
80xxh	Monitoring
81xxh	Communication
8110 h	CAN Overrun (Objects lost)
8120 h	CAN in Error Passive Mode
8130 h	Life Guard Error or Heartbeat Error
8140 h	Recovered from bus off
8150 h	Transmit COB-ID
82xxh	Protocol Error
8210 h	PDO not processed due to length error
8220 h	PDO length exceeded
90xxh	External Error
F0xxh	Additional Functions
FFxxh	Device-specific

xx = Not defined

1005h - SYNC message frame COB-ID

Contains the COB-ID of the SYNC message frame and indicates whether the unit generates the SYNC message frame.

Index	Subindex	Name	Data type	Access	Category	Mappable
1005h	0 h	COB-ID SYNC	Unsigned32	rw	Conditional	No

The object has the Mandatory category if synchronous PDO transfer is supported.

The individual bits of the 32-bit values have the following meanings:

Bit	Value / meaning
31 (MSB)	Not relevant
30	0: Network node is not generating a SYNC message frame 1: Network node is generating a SYNC message frame
29	0: 11-bit CAN ID 1: 29-bit CAN ID (is not supported by the CANopen-Master option module)

Bit	Value / meaning
28 - 11	Bits 28 - 11 with 29-bit CAN ID (is not supported by the CANopen-Master option module)
10 - 0 (LSB)	COB-ID of the SYNC message frame. Since the CANopen-Master option module uses the COB-ID in accordance with the predefined identifier assignment, this must be 80h.

1006h - SYNC interval

Length of the SYNC interval in µs.

Index	Subindex	Name	Data type	Access	Category	Mappable
1006h	0 h	Communication cycle period	Unsigned32	rw	Conditional	No

The object has the Mandatory category for network nodes that generate the SYNC message frame. In this case, the period is entered in µs. Network nodes that evaluate the SYNC message frame, can use this value for monitoring purposes and to synchronize internal communication cycles.

100Ch - Guarding Time

The distance between two guarding message frames in ms.

Index	Subindex	Name	Data type	Access	Category	Mappable
100Ch	0 h	Guard time	Unsigned16	rw	Conditional	No

The object contains the distance between two guarding message frames in ms. The object has the Mandatory category for network nodes that do not support the heartbeat. Since it is not possible to use the CANopen-Master option module for monitoring network nodes by means of the heartbeat, the object must be present.

100Dh - Life Time Factor

Multiplied by the guarding time (object 100Ch) yields the life time in ms.

Index	Subindex	Name	Data type	Access	Category	Mappable
100Dh	0 h	Life time factor	Unsigned8	rw	Conditional	No

The object has the Mandatory category for network nodes that do not support the heartbeat. Since it is not possible to use the CANopen-Master option module for monitoring network nodes by means of the heartbeat, the object must be present.

1010h - Store parameters

Makes it possible to store parameters in non-volatile memory.

Index	Subindex	Name	Data type	Access	Category	Mappable
1010h	-	Store parameters	ARRAY	-	Optional	-
	0 h	Largest sub-index supported	Unsigned8	ro	Mandatory	No
	1 h	Save all parameters	Unsigned32	rw	Mandatory	No
	2 h	Save communication parameters	Unsigned32	rw	Optional	No
	2 h	Save application parameters	Unsigned32	rw	Optional	No
	4h - 7h	Save manufacturer defined	Unsigned32	rw	Optional	No

Writing the "save" command to the respective subindex saves a specific parameter selection. The "save" command is represented by the following 32-bit value:

Byte: MSB LSB

e	V	A	C
65 h	76 h	61 h	73 h

The scope of the parameter selection results from the name of the subindex.

Reading a subindex yields the following information:

Bit	Value / meaning
31 - 2	*Reserved*
1	0: Unit does not automatically save the parameters (e.g. when you switch off the unit) 1: Unit automatically saves the parameters (e.g. when you switch off the unit)
0	0: Parameters cannot be saved on command 1: Parameters can be saved on command

1011h - Load Default Parameters

Allows you to activate default parameters.

Index	Subindex	Name	Data type	Access	Category	Mappable
1011h	-	Restore default parameters	ARRAY	-	Optional	-
	0 h	Largest sub-index supported	Unsigned8	ro	Mandatory	No
	1 h	restore all parameters	Unsigned32	rw	Mandatory	No
	2 h	restore communication parameters	Unsigned32	rw	Optional	No
	2 h	restore application parameters	Unsigned32	rw	Optional	No
	4h - 7h	restore manufacturer defined	Unsigned32	rw	Optional	No

Writing the "load" command to the respective subindex loads a specific default parameter selection and activates it after the next device reset. The "load" command is represented by the following 32-bit value:

Byte: MSB LSB

d	A	o	L
64 h	61 h	6Fh	6Ch

The scope of the parameter selection results from the name of the subindex.

Reading a subindex yields the following information:

Bit	Value / meaning
31 - 1	*Reserved*
0	0: Default parameters cannot be activated 1: Default parameters can be activated

1400h-15FFh - Receive PDO (RxPDO) Communications Parameter

Objects 1400h to 15FFh determine the properties for transferring RxPDO 1 to RxPDO 512. Object 1400h writes to RxPDO 1 and – assuming the unit supports it – object 15FFh writes to RxPDO 512.

Index	Subindex	Name	Data type	Access	Category	Mappable
1400h - 15FFh	-	RxPDO parameter	CiA defined	-	Conditional	-
	0 h	Largest sub-index supported	Unsigned8	rw	Mandatory	No
	1 h	COB-ID Receive RxPDO	Unsigned32	ro / rw	Mandatory	No
	2 h	Transmission type RxPDO	Unsigned8	ro / rw	Mandatory	No

Subindex 0 indicates the number of the last valid subindex. The individual bits of the 32-bit value of **subindex 1** have the following meanings:

Subindex 1, Bit	Value / meaning
31 (MSB)	0: PDO exists and is valid 1: PDO does not exist or is not valid According to the predefined identifier assignment, only 4 PDOs are possible for writing and 4 PDOs for reading. If a device supports more than these four PDOs, the COB-IDs for the additional PDOs must be assigned themselves (bits 10 - 0 in this subindex). For this reason, bit 31 of the additional PDOs usually has a value of 1 and users do not set it to 0 until they have assigned a COB-ID.
30	0: The PDO can be requested by means of a remote request 1: The PDO cannot be requested by means of a remote request
29	0: 11-bit CAN ID 1: 29-bit CAN ID (is not supported by the CANopen-Master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (is not supported by the CANopen-Master option module)
10 - 0 (LSB)	COB-ID of the PDO. In the default settings of a device, the first four PDOs for reading have the COB-IDs after the predefined identifier assignment.

Subindex 2 determines the PDO's transfer type:

Subindex 2, value	Type	Meaning
0	Synchronized	The system applies the PDO with an appropriate COB-ID that was received before the last SYNC message frame
1 - 240	Synchronized	The system applies the PDO with an appropriate COB-ID that was received before the last SYNC message frame

Subindex 2, value	Type	Meaning
252	Remote	*Not possible*
253	Remote	*Not possible*
254	Asynchro-nous	The system applies each PDO with an appropriate COB-ID when it is received
255	Asynchro-nous	The system applies each PDO with an appropriate COB-ID when it is received

1600h-17FFh - Receive PDO (RxPDO) Mapping Parameter

Objects 1600h to 17FFh determine the mapped objects of RxPDO 1 to RxPDO 512. Object 1600h writes to RxPDO 1 and – assuming the unit supports it – object 17FFh writes to RxPDO 512.

Index	Subindex	Name	Data type	Access	Category	Mappable
1600h - 17FFh	-	RxPDO mapping	CiA defined	-	Conditional	-
	0 h	Number of mapped objects	Unsigned8	rw	Mandatory	No
	1h - 40h	Object to be mapped	Unsigned32	rw	Conditional	No

Subindex 0 indicates the number of mapped objects in the PDO. As soon as the mapping of a PDO is to be changed, you must first set the number of mapped objects to zero. After this, you can set the mapping and finally enter the number of mapped objects in subindex 0. In subindex 1h to 40h, the mapping information of the objects in the PDO is stated. Subindex 1h contains the information for the first mapped object; subindex 2h contains the information for the second one, etc. A maximum of 64 (= 40h) objects can be mapped; in this connection, you cannot exceed the maximum data range of 8 bytes of the PDO. The contents of the 32-bit values of subindexes 1h - 40h have the following meanings:

Byte:	MSB		LSB
	Index of the mapped object (16-bit)	Subindex of the mapped object (8-bit)	Number of bits of the mapped object (8-bit)

The system maps the objects successively in one PDO.

1800h-19FFh - Send PDO (TxPDO) Communications Parameter

Objects 1800h to 19FFh determine the properties for transferring TxPDO 1 to TxPDO 512. Object 1800h writes to TxPDO 1 and – assuming the unit supports it – object 19FFh writes to TxPDO 512.

Index	Subindex	Name	Data type	Access	Category	Mappable
1800h - 19FFh	-	TxPDO parameter	CiA defined	-	Conditional	-
	0 h	Largest sub-index supported	Unsigned8	rw	Mandatory	No
	1 h	COB-ID Receive TxPDO	Unsigned32	ro / rw	Mandatory	No
	2 h	Transmission type TxPDO	Unsigned8	ro / rw	Mandatory	No
	3 h	Inhibit time TxPDO	Unsigned16	rw	Optional	No
	4 h	Reserved	-	-	-	-
	5 h	Event Timer TxPDO	Unsigned16	rw	Optional	No

Subindex 0 indicates the number of the last valid subindex. The individual bits of the 32-bit value of **subindex 1** have the following meanings:

Subindex 1, Bit	Value / meaning
31 (MSB)	0: PDO exists and is valid 1: PDO does not exist or is not valid According to the predefined identifier assignment, only 4 PDOs are possible for writing and 4 PDOs for reading. If a device supports more than these four PDOs, the COB-IDs for the additional PDOs must be assigned themselves (bits 10 - 0 in this subindex). For this reason, bit 31 of the additional PDOs usually has a value of 1 and users do not set it to 0 until they have assigned a COB-ID.
30	0: The PDO can be requested by means of a remote request 1: The PDO cannot be requested by means of a remote request
29	0: 11-bit CAN ID 1: 29-bit CAN ID (is not supported by the CANopen-Master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (is not supported by the CANopen-Master option module)
10 - 0 (LSB)	COB-ID of the PDO. In the default settings of a device, the first four PDOs for writing have the COB-IDs after the predefined identifier assignment.

Subindex 2 determines the PDO's transfer type:

Subindex 2, value	Type	Meaning
0	Synchro-nized	The system sends after each received SYNC message frame
1 - 240	Synchro-nized	The system sends after receiving the set number (1 - 240) of SYNC message frames
252	Remote	On receiving a SYNC message frame, the system updates the PDO. The system does not send until it receives a remote request.
253	Remote	The system updates and sends after receiving a remote request.
254	Asynchro-nous	The system sends on a manufacturer-specific basis. Note: With this type, most devices transfer the PDO on a time (when a timer runs down). The time is set under subindex 5.
255	Asynchro-nous	The system sends in dependence on the supported device profile. Note: With this type, most devices transfer the PDO as soon as one of the mapped values has changed.

Using **subindex 3,** you set a PDO inhibit time in multiples of 100 µs. This does not extend the response time at the relative first value change; rather, it is active on the changes that directly follow this one. The inhibit time describes the minimum period of time that the system must wait between sending two of the same message frames.

In **subindex 5,** you set the time in ms for event-driven PDO transfer assuming that a device supports event-driven PDO transfer (subindex 2 = 254 or 255 depending on the device profile).

1A00h-1BFFh - Send PDO (TxPDO) Mapping Parameter

Objects 1A00h to 1BFFh determine the mapped objects of TxPDO 1 to TxPDO 512. Object 1A00h writes to TxPDO 1 and – assuming the unit supports it – object 1AFFh writes to TxPDO 512.

Index	Subindex	Name	Data type	Access	Category	Mappable
1A00h - 1BFFh	-	TxPDO mapping	CiA defined	-	Conditional	-
	0 h	Number of mapped objects	Unsigned8	rw	Mandatory	No
	1h - 40h	Object to be mapped	Unsigned32	rw	Conditional	No

Subindex 0 indicates the number of mapped objects in the PDO. As soon as the mapping of a PDO is to be changed, you must first set the number of mapped objects to zero. After this, you can set the mapping and finally enter the number of mapped objects in subindex 0. In **Subindex 1h to 40h** the mapping information of the objects in the PDO is stated. Subindex 1h contains the information for the first mapped object; subindex 2h contains the information for the second one, etc. A maximum of 64 (= 40h) objects can be mapped; in this connection, you cannot exceed the maximum data range of 8 bytes of the PDO. The contents of the 32-bit values of **subindexes 1h to 40h** have the following meanings:

Byte:	MSB		LSB
	Index of the mapped object (16-bit)	Subindex of the mapped object (8-bit)	Number of bits of the mapped object (8-bit)

The system maps the objects successively in one PDO.

B.2 Default Mapping of the PDO

Default mappings are defined in the device profiles. Below, there is an overview of the default mapping values for device profiles DS 401 and DSP 402. The assignment that is shown below refers to the maximum configuration of a device. Depending on the hardware, fewer objects may be mapped. However, this does not mean that objects of a different type replace other ones in the mapping, e.g. analog inputs instead of digital inputs. If objects are dropped, the associated locations in the default mapping stay vacant. It is, of course, your choice whether you want to assign other objects to the vacant locations.

B.2.1 Default Mapping for I/O Modules According to DS 401

Object	Meaning	Mapping value index subindex length
1600h	First receive PDO.	
Subindex 0	Number of mapped objects	8
Subindex 1	Digital outputs 0 - 7	6200 01 08 h
Subindex 2	Digital outputs 8 - 15	6200 02 08 h
Subindex 3	Digital outputs 16 - 23	6200 03 08 h

Object	Meaning	Mapping value index subindex length
Subindex 4	Digital outputs 24 - 31	6200 04 08 h
Subindex 5	Digital outputs 32 - 39	6200 05 08 h
Subindex 6	Digital outputs 40 - 47	6200 06 08 h
Subindex 7	Digital outputs 48 - 55	6200 07 08 h
Subindex 8	Digital outputs 56 - 63	6200 08 08 h
1601h	Second receive PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog output 1 (16-bit resolution)	6411 01 10 h
Subindex 2	Analog output 2 (16-bit resolution)	6411 02 10 h
Subindex 3	Analog output 3 (16-bit resolution)	6411 03 10 h
Subindex 4	Analog output 4 (16-bit resolution)	6411 04 10 h
1602h	Third receive PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog output 5 (16-bit resolution)	6411 05 10 h
Subindex 2	Analog output 6 (16-bit resolution)	6411 06 10 h
Subindex 3	Analog output 7 (16-bit resolution)	6411 07 10 h
Subindex 4	Analog output 8 (16-bit resolution)	6411 08 10 h
1603h	Fourth receive PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog output 9 (16-bit resolution)	6411 09 10 h
Subindex 2	Analog output 10 (16-bit resolution)	6411 0A 10h
Subindex 3	Analog output 11 (16-bit resolution)	6411 0B 10h
Subindex 4	Analog output 12 (16-bit resolution)	6411 0C 10h

Object	Meaning	Mapping value index subindex length
1A00h	First send PDO.	
Subindex 0	Number of mapped objects	8
Subindex 1	Digital inputs 0 - 7	6000 01 08 h
Subindex 2	Digital inputs 8 - 15	6000 02 08 h
Subindex 3	Digital inputs 16 - 23	6000 03 08 h
Subindex 4	Digital inputs 24 - 31	6000 04 08 h
Subindex 5	Digital inputs 32 - 39	6000 05 08 h
Subindex 6	Digital inputs 40 - 47	6000 06 08 h

Object	Meaning	Mapping value index subindex length
Subindex 7	Digital inputs 48 - 55	6000 07 08 h
Subindex 8	Digital inputs 56 - 63	6000 08 08 h
1A01h	Second send PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog input 1 (16-bit resolution)	6401 01 10 h
Subindex 2	Analog input 2 (16-bit resolution)	6401 02 10 h
Subindex 3	Analog input 3 (16-bit resolution)	6401 03 10 h
Subindex 4	Analog input 4 (16-bit resolution)	6401 04 10 h
1A02h	Third send PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog input 5 (16-bit resolution)	6401 05 10 h
Subindex 2	Analog input 6 (16-bit resolution)	6401 06 10 h
Subindex 3	Analog input 7 (16-bit resolution)	6401 07 10 h
Subindex 4	Analog input 8 (16-bit resolution)	6401 08 10 h
1A03h	Fourth send PDO.	
Subindex 0	Number of mapped objects	4
Subindex 1	Analog input 9 (16-bit resolution)	6401 09 10 h
Subindex 2	Analog input 10 (16-bit resolution)	6401 0A 10h
Subindex 3	Analog input 11 (16-bit resolution)	6401 0B 10h
Subindex 4	Analog input 12 (16-bit resolution)	6401 0C 10h

B.2.2 Default Mapping for Drives According to DSP 402

Object	Meaning	Mapping value index subindex length
1600h	First receive PDO.	
Subindex 0	Number of mapped objects	1
Subindex 1	Control word (16-bit)	6040 00 10 h
1601h	Second receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	Operating mode (8-bit)	6060 00 08 h
1602h	Third receive PDO.	
Subindex 0	Number of mapped objects	2

Object	Meaning	Mapping value index subindex length
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	Target position (32-bit)	607A 00 20h
1603h	Fourth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	RPM speed reference value (32-bit)	60FF 00 20h
1604h	Fifth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	Target torque (32-bit)	6071 00 20 h
1605h	Sixth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	RPM speed reference value (16-bit)	6042 00 10 h
1606h	Seventh receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	Digital outputs (32-bit)	6042 00 20 h
1607h	Eighth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Control word (16-bit)	6040 00 10 h
Subindex 2	Operating mode (8-bit)	6060 00 08 h

Object	Meaning	Mapping value index subindex length
1A00h	First receive PDO.	
Subindex 0	Number of mapped objects	1
Subindex 1	Status word (16-bit)	6041 00 10 h
1A01h	Second receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	Operating mode display (8-bit)	6061 00 08 h
1A02h	Third receive PDO.	

Object	Meaning	Mapping value index subindex length
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	Position actual value (32-bit)	6064 00 20 h
1A03h	Fourth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	RPM speed actual value (32-bit)	606C 00 20h
1A04h	Fifth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	Torque actual value (32-bit)	6077 00 20 h
1A05h	Sixth receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	RPM speed actual value (16-bit)	6044 00 10 h
1A06h	Seventh receive PDO.	
Subindex 0	Number of mapped objects	2
Subindex 1	Status word (16-bit)	6041 00 10 h
Subindex 2	Digital inputs (32-bit)	60FD 00 20h

B.3 CANopen objects of the CANopen masters (software version $\geq$ 01.20)

B.3.1 Overview of the objects

Index	Object contents/Meaning
1000h	Device type: supported device profile and additional information
1001h	Error register
1003h	Error memory contains a device error list
1005h	SYNC telegram COB-ID
1006h	SYNC interval
1008h	Manufacturer device name
1009h	Manufacturer hardware version
100Ah	Manufacturer software version
1010h	Save parameters
1011h	Load default parameters
1014h	EMCY telegramm COB-ID
1016h	Heartbeat times of network nodes
1017h	Heartbeat time of CANopen master
1018h	Identification of option module
1020h	Identification of configuration
1200h	Server SDO parameters
1280h	Client SDO parameters
1400h - 14FFh	Reception PDO (RxPDO) communication parameters
1600h - 16FFh	Reception PDO (RxPDO) mapping parameters
1800h - 18FFh	Transmission PDO (TxPDO) communication parameters
1A00h - 1AFFh	Transmission PDO (TxPDO) communication parameters
1F22h	Concise DCF of network nodes
1F25h	Network node configuration request
1F80h	NMT Startup
1F81h	Network node assignment
1F82h	NMT command request
1F84h	Network node identification, device type
1F85h	Network node identification, manufacturer ID
1F86h	Network node identification, product code
1F87h	Network node identification, revision number
1F88h	Network node identification, serial number

B.3 CANopen objects of the CANopen masters (software version ≥ 01.20)

Index	Object contents/Meaning
1F89h	Wait time for network node boot procedure
2003h	User device name
2004h	Synchronization with PLC
2005h	Reaction to PLC status
2020h	Network bootup setting
2022h	SDO wait time during the boot procedure of a node
2023h	Wait time for restoring the default settings
2024h	Wait time after network reset before network scanning begins
2025h	Wait time until the next node scan begins
2100h	Process map configuration
2101h	Asynchronous and synchronous areas in process map
2102h	Configured asynchronous objects in process map
2103h	Configured synchronous objects in process map

B.3.2 Objects of the CANopen master pursuant to CiA DS 301 and DSP 302

1000h - Device type

Specifies the supported device profile of the CANopen master and additional information.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1000h	0h	device type	Unsigned32	ro	no	No

The 32-bit value is divided into two 16-bit values:

Byte:	MSB	LSB
Con-tents	Additional information, manufacturer-specific	Device profile
Value	0000h	0195h

The CANopen master supports the device profile CiA DSP 405 V 2.0 “Interface and Device Profile for IEC 61131-3 Programmable Devices”. Additional information is not available.

1001h - Error register

Contains a bit list with active errors of the CANopen master.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1001h	0h	error register	Unsigned8	ro	no	No

The individual bits of the 8-bit value have the following meaning:

Bit	Meaning	Category
0	Generic error	Mandatory
1	Current	Optional
2	Voltage	Optional
3	Temperature	Optional
4	Communication error	Optional
5	Device profile specific	Optional
6	Reserved	Optional
7	Manufacturer specific	Optional

If a bit is set, the error is active. This object corresponds to the *error register* of the emergency telegram and thus contains the value of the *error register* from the last emergency telegram sent by the CANopen master. Since the CANopen master does not generate emergency telegrams itself, the object 1001h only specifies the errors set by the user via the COM405_SEND_EMCY function block.

1003h - Error memory

Contains an error list of the CANopen master starting with the newest.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1003h	-	pre-defined error field	Array	-	-	-
	0h	number of errors	Unsigned8	rw	0	No
	1h - 5h	standard error field	Unsigned32	ro	no	No

Subindex 0h contains the number of entered errors. The error list can be deleted by writing the value 0 to subindex 0h. Subindex 1h contains the current error. If a new error occurs, it is entered at subindex 1h and all others move down one subindex. A fault history of max. five entries is constructed.

The 32-bit value of the standard error field is divided into two 16-bit values:

Byte:	MSB	LSB
	Additional information, manufacturer-specific	Error code

The 16-bit error code is yielded after the *Emergency Error Code* is defined in the emergency telegram:

Error code	Meaning
00xxh	Error Reset or No Error
10xxh	Generic Error
20xxh	Current
21xxh	Current, device input side
22xxh	Current inside the device
23xxh	Current, device output side
30xxh	Voltage
31xxh	Mains Voltage
32xxh	Voltage inside the device
33xxh	Output Voltage
40xxh	Temperature
41xxh	Ambient Temperature
42xxh	Device Temperature
50xxh	Device Hardware

Error code	Meaning
60xxh	Device Software
61xxh	Internal Software
62xxh	User Software
63xxh	Data Set
70xxh	Additional Modules
80xxh	Monitoring
81xxh	Communication
8110h	CAN Overrun (Objects lost)
8120h	CAN in Error Passive Mode
8130h	Life Guard Error or Heartbeat Error
8140h	recovered from bus off
8150h	Transmit COB-ID
82xxh	Protocol Error
8210h	PDO not processed due to length error
8220h	PDO length exceeded
90xxh	External Error
F0xxh	Additional Functions
FFxxh	Device specific

The 16-bit additional information for the CANopen master contains the first two bytes of the *Manufacturer specific Error Field* from the emergency telegram.

Since the CANopen master does not generate emergency telegrams itself, the object 1003h only specifies the errors generated by the user via the COM405_SEND_EMCY function block.

1005h - SYNC telegram COB-ID

Contains the COB-ID of the SYNC telegram and specifies whether the CANopen master generates the SYNC telegram.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1005h	0h	COB-ID SYNC	Unsigned32	rw	40000080h	No

The individual bits of the 32-bit value have the following meaning:

Bit	Value/Meaning
31 (MSB)	Not relevant
30	0: CANopen master does not generate SYNC telegram 1: CANopen master generates SYNC telegram With CANopen master, generation of the SYNC telegram is activated by default.
29	0: 11-bit CAN ID 1: 29-bit CAN ID (not supported by CANopen master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (not supported by CANopen master option module)
10 - 0 (LSB)	COB-ID of the SYNC telegram. Since the CANopen master option module uses the COB-ID after predefined issuing of the identifier, 80h is used by default.

Please be aware that further configuration of the SYNC mechanisms is carried out with object 2004h.

1006h - SYNC interval

Length of the SYNC interval in µs.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1006h	0h	Communication cycle period	Unsigned32	rw	0	No

The object specifies the time interval during which the CANopen master sends the SYNC telegram. The specified value in µs is converted to the internal resolution of 1 ms by the CANopen master, i.e. it is divided by 1,000. The result is evaluated as a whole number. SYNC intervals in multiples of 1 ms are therefore possible. If the object 1006h has the value 0, no SYNC telegram is generated and no data is exchanged with synchronous PDOs. Writing the object is not possible in the OPERATIONAL state.

NOTE

The synchronization type used with the PLC is specified via object 2004h. Object 2004h must be configured before object 1006h!

1008h - Manufacturer device name

Contains the device name of the optional CANopen master option module issued by the Baumüller company.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1008h	0h	manufacturer device name	VISIBLE STRING	ro	no	No

The object specifies the name of the optional module as a character string, depending on the type of execution, e.g. "BM4-O-CAN-04".

1009h - Manufacturer hardware version

Contains the number of the hardware version of the optional CANopen master option module issued by the Baumüller company.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1009h	0h	manufacturer hardware version	VISIBLE STRING	ro	no	No

The object specifies the number of the hardware version of the optional module as a character string, depending on the type of execution, e.g. "3.0107".

100Ah - Manufacturer software version

Contains the number of the software version of the optional CANopen master option module issued by the Baumüller company.

Index	Subindex	Name	Data type	Access	Default value	Map-able
100Ah	0h	manufacturer software version	VISIBLE STRING	ro	no	No

The object specifies the number of the software version of the optional module as a character string, depending on the type of execution, e.g. "6.1307.0201".

1010h - Save parameters

Enables the saving of object contents in nonvolatile memory. The object contents remain intact when the power is switched off and are reloaded when the power is switched on. The object contents are also loaded from the nonvolatile memory of the CANopen master after a RESET.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1010h	-	store parameters	Array	-	-	-
	0h	largest subindex supported	Unsigned8	ro	no	No
	1h	save all parameters	Unsigned32	rw	no	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
	2h	save communication parameters	Unsigned32	rw	no	No
	4h	save manufacturer defined	Unsigned32	rw	no	No

By writing the command "save" in ASCII code to the respective subindex, the contents of a specific selection of objects is saved. The "save" command is represented by the following 32-bit value:

Byte: MSB LSB

e	v	a	s
65h	76h	61h	73h

The scope of object selection is yielded from the subindex:

Subindex	Memory scope
1	Saves the contents of all objects of the CANopen master
2	Saves the contents of the objects 1000h to 1FFFh of the CANopen master
4	Saves the contents of the objects 2000h to 5FFFh of the CANopen master

Reading a subindex 1 - 4 provides the following information:

Bit	Value/Meaning
31 - 2	*Reserved*
1	0: The device does not save the parameters automatically (e.g. when it is switched off) 1: The device saves the parameters automatically (e.g. when it is switched off)
0	0: Parameters cannot be saved by command 1: Parameters can be saved by command

1011h - Load default parameters

Enables activation of default parameters. The object contents saved in the nonvolatile memory are deleted.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1011h	-	restore parameters	Array	-	-	-
	0h	largest subindex supported	Unsigned8	ro	no	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
	1h	restore all parameters	Unsigned32	rw	no	No
	2h	restore communication parameters	Unsigned32	rw	no	No
	4h	restore manufacturer defined	Unsigned32	rw	no	No

By writing the command "load" in ASCII code to the respective subindex, a specific selection of object contents and their default values are loaded and activated after the next RESET of the CANopen master. The object contents in nonvolatile memory are deleted. The "load" command is represented by the following 32-bit value:

Byte: MSB LSB

d	a	o	l
64h	61h	6Fh	6Ch

The scope of object selection is yielded from the subindex:

Subindex	Memory scope
1	Loads default values of all objects of the CANopen master
2	Loads default values of the objects 1000h to 1FFFh of the CANopen master
4	Loads default values of the objects 2000h to 5FFFh of the CANopen master

Reading a subindex 1 - 4 provides the following information:

Bit	Value/Meaning
31 - 1	Reserved
0	0: Default parameters cannot be activated 1: Default parameters can be activated

1014h - EMCY telegram COB-ID

Contains the COB-ID of the emergency telegram and specifies whether the EMCY telegram is valid.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1014h	0h	COB-ID Emergency message	Unsigned32	rw	80h + node ID	No

The individual bits of the 32-bit value have the following meaning:

Bit	Value/Meaning
31	0: EMCY telegram is valid/activated 1: EMCY telegram is invalid/deactivated
30	*Reserved (always 0)*
29	0: 11-bit CAN ID 1: 29-bit CAN ID (not supported by CANopen master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (not supported by CANopen master option module)
10 - 0 (LSB)	COB-ID of the EMCY telegram. Since the CANopen master option module uses the COB-ID after predefined issuing of the identifier, 80h + node ID is used by default. If a new COB-ID is issued, the node ID is no longer taken into account!

The EMCY telegram can only be generated by the user via the COM405_SEND_EMCY function block.

1016h - Heartbeat times of network nodes

Contains a list with the heartbeat times of the individual network nodes to be monitored by the CANopen master.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1016h	-	Consumer Heartbeat Time	Array	-	-	-
	0h	number entries	Unsigned8 (1 – 127)	ro	no	No
	1h - 7Fh	Consumer Heartbeat Time	Unsigned32	rw	no	No

The object 1016h contains the expected heartbeat times of the individual network nodes in list form in the subindices 1 - 127. This list contains consecutive "Consumer Heartbeat Time" entries consisting of node number and the accompanying heartbeat time. A maximum of 127 nodes can be monitored. The "Consumer Heartbeat Time" entry has the following format:

Bits:	31 - 24	23 - 16	15 - 0
Contents	*Reserved (= 0)*	Node ID	Heartbeat time in ms

The heartbeat time in ms is converted to the internal resolution of 10 ms by the CANopen master, i.e. it is divided by 10. The result is evaluated as a whole number. Monitoring times in multiples of 10 ms are therefore possible. If a heartbeat time of 0 is entered, monitoring is not carried out.

CANopen master begins monitoring a node with the first heartbeat received. If a heartbeat is not received from a node within the specified time, the CANopen master can respond to this. The type of reaction is determined by objects 1F80h and 1F81h. It makes sense to set the heartbeat time on the network node to be monitored (object 1017h of the network node) lower than with the CANopen master in object 1016h.

1017h - Heartbeat time of CANopen master

Specifies the heartbeat time of the CANopen master.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1017h	0h	Producer Hearbeat Time	Unsigned16	rw	0	No

The CANopen master sends heartbeat telegrams at the "Producer Hearbeat Time" time interval set in this object. The heartbeat time is specified in ms, whereby only multiples of 10 ms are possible, i.e. 10 ms, 20 ms, 30 ms etc.. Intermediate values are rounded up internally. If the time interval is 0, no heartbeat is sent. Heartbeat transmission begins as soon as a value different from zero is entered.

1018h - Identification of option module

Specifies the option module.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1018h	-	Identity Object	Record	-	-	-
	0h	number of entries	Unsigned8	ro	no	No
	1h	Vendor ID	Unsigned32	ro	no	No
	2h	Product code	Unsigned32	ro	no	No
	3h	Revision number	Unsigned32	ro	no	No
	4h	Serial number	Unsigned32	ro	no	No

The object 1018h contains data for more precise identification of the CANopen master option module. Subindices 1 - 3 have the following meaning:

Subindex	Name	Meaning
1	Vendor ID	Manufacturer ID. The Baumüller Nürnberg GmbH company has the ID 346
2	Product code	Not used, 0
3	Revision number	Not used, 0
4	Serial number	Serial number of the option module

1020h - Identification of configuration

Enables the identification of the configuration of the CANopen master via the date and time.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1020h	-	Verify Configuration	Array	-	-	-
	0h	number of entries	Unsigned8	ro	no	No
	1h	Configuration date	Unsigned32	rw	no	No
	2h	Configuration time	Unsigned32	rw	no	No

The object 1020h contains the date and time for more precise identification of the configuration of the CANopen master. A date can be entered in subindex 1 by the configuration tool or user, and a time can be entered in subindex 2. The contents of subindices 1 and 2 are deleted by CANopen master as soon as a new object is written. Exceptions include objects 1010h, 1020h, 1F25h, 1F82h and the objects of the process image (A000h etc.); writing of these objects does not lead to deletion of the contents of subindices 1 and 2.

Once the date and time are entered, the entire configuration is to be saved via object 1010h.

In greater detail, subindices 1 - 3 have the following meaning:

Subindex	Name	Meaning
1	Configuration date	The number of days that have passed since January 1st, 1984 is to be entered for the configuration date
2	Configuration time	The number of milliseconds that have passed since midnight is to be entered for the configuration date

1200h - Server SDO parameters

Contains the parameters for the access to the object directory of the CANopen master via SDO.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1200h	-	Server SDO parameters	Record	-	-	-
	0h	number of entries	Unsigned8	ro	no	No
	1h	COB-ID Client -> Server (rx)	Unsigned32	ro	600h + node ID	No
	2h	COB-ID Server -> Client (tx)	Unsigned32	ro	580h + node ID	No

This object contains the parameters for SDO access to the object directory of the CANopen master. The CANopen master is the server in this case. The client can be another network node or the function blocks for SDO communication from the xx.xx library if the master itself is accessed. The entries in subindices 1 and 2 have the following structure:

Bit	Value/Meaning
31	0: SDO is valid/activated 1: SDO is invalid/deactivated
30	*Reserved (always 0)*
29	0: 11-bit CAN ID 1: 29-bit CAN ID (not supported by CANopen master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (not supported by CANopen master option module)
10 - 0 (LSB)	COB-ID of the SDO telegram. Since the CANopen master option module uses the COB-ID after predefined issuing of the identifier, 600h + node ID is used for the client -> server direction and 580h + node ID for the server -> client direction. The node ID is the node number of the CANopen master.

1280h - Client SDO parameters

The CANopen master manages SDO access to the object directories of other network nodes internally. Configuration of the parameters of this object (e.g. COB-ID) by the user is not possible. The COB-IDs result from issuing via object 1200h, i.e. for inquiries via SDOs sent to a network node, 600h + node ID is used, and COB-ID 580h + node ID is the awaited answer.

1400h to 14FFh - Reception PDO (RxPDO) communication parameters

The objects 1400h to 14FFh determine the properties for transmission of RxPDO 1 to RxPDO 256 of the CANopen master. Object 1400h describes RxPDO 1, object 1401h describes RxPDO 2, object 14FFh describes RxPDO 256 etc..

Index	Subindex	Name	Data type	Access	Default value	Map-able
1400h	-	RxPDO 1 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	2	No
	1h	COB-ID Receive RxPDO 1	Unsigned32	rw	NODEID+0x200	No
	2h	Transmission type RxPDO 1	Unsigned8	rw	255	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1401h	-	RxPDO 2 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	2	No
	1h	COB-ID Receive RxPDO 2	Unsigned32	rw	NODEID+0x300	No
	2h	Transmission type RxPDO 2	Unsigned8	rw	255	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1402h	-	RxPDO 3 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	2	No
	1h	COB-ID Receive RxPDO 3	Unsigned32	rw	NODEID+0x400	No
	2h	Transmission type RxPDO 3	Unsigned8	rw	255	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1403h	-	RxPDO 4 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	2	No
	1h	COB-ID Receive RxPDO 4	Unsigned32	rw	NODEID+0x500	No
	2h	Transmission type RxPDO 4	Unsigned8	rw	255	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1404h – 14FFh	-	RxPDO 5 – 256 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	2	No
	1h	COB-ID Receive RxPDO 5 – 256	Unsigned32	rw	0x80000000	No
	2h	Transmission type RxPDO 5 – 256	Unsigned8	rw	-	No

Subindex 0 specifies the number of the last valid subindex. The individual bits of the 32-bit value of **subindex 1** have the following meaning:

Subindex 1, bit	Value/Meaning
31 (MSB)	0: PDO exists and is valid 1: PDO does not exist or is invalid Only the first four RxPDOs of the CANopen master are activated. All other RxPDOs are deactivated. If more than four RxPDOs are used, the COB-IDs for the additional RxPDOs themselves must be issued (bits 10 - 0 in this subindex). For this reason, bit 31 of the RxPDOs 5 - 256 has a value of 1 and is to be set to 0 by the user as soon as the user issues a COB-ID.
30	0: The RxPDO can be requested via a remote request 1: The RxPDO cannot be requested via a remote request
29	0: 11-bit CAN ID 1: 29-bit CAN ID (not supported by CANopen master option module)
28 - 11	Bits 28 - 11 with 29-bit CAN ID (not supported by CANopen master option module)
10 - 0 (LSB)	COB-ID of the RxPDO. The first four RxPDOs have a default setting.

The transmission type of the RxPDO is specified with subindex 2:

Subindex 2, value	Type	Meaning
0	*Not supported*	
1 – 240	Synchronous	RxPDO with appropriate COB-ID received before the last SYNC telegram is accepted
254	Asynchronous	Each RxPDO with an appropriate COB-ID is accepted upon reception
255	Asynchronous	Each RxPDO with an appropriate COB-ID is accepted upon reception

1600h to 16FFh - Reception PDO (RxPDO) mapping parameters

The objects 1600h bis 16FFh determine the mapped objects of RxPDO 1 to RxPDO 256 of the CANopen master. Object 1600h describes RxPDO 1, object 1601h describes RxPDO 2, object 16FFh describes RxPDO 256 etc..

Index	Subindex	Name	Data type	Access	Default value	Map-able
1600h - 16FFh	-	RxPDO mapping	Record	-	-	-
	0h	Number of mapped objects	Unsigned8	rw	0	No
	1h - 8h	Object to be mapped	Unsigned32	rw	0	No

Subindex 0 specifies the number of mapped objects in the RxPDO. As soon as the mapping of a RxPDO is changed, the number of mapped objects must first be set to zero. The mapping can then be set, and the number of mapped objects in subindex 0 must finally be entered. The mapping information of the objects in the RxPDO are specified in **subindices 1h to 8h**. Subindex 1h contains the information for the first mapped object, subindex 2h for the second mapped object etc.. A maximum of 8 (8h) objects can be mapped, whereby the maximum data range of 8 bytes of the PDO may not be exceeded. The contents of the 32-bit value of **subindices 1h to 8h** have the following meaning:

Byte:	MSB		LSB
	Index of the mapped object (16-bit)	Subindex of the mapped object (8-bit)	Number of bits of the mapped object (8-bit)

The objects are mapped in an RxPDO one after another.

1800h to 18FFh - Transmission PDO (TxPDO) communication parameters

The objects 1800h to 18FFh determine the properties for transmission of TxPDO 1 to TxPDO 256. Object 1800h describes TxPDO 1, object 1801h describes TxPDO 2, object 18FFh describes TxPDO 256 etc..

Index	Subindex	Name	Data type	Access	Default value	Map-able
1800h	-	TxPDO 1 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	5	No
	1h	COB-ID Receive TxPDO 1	Unsigned32	rw	NODEID+0x180	No
	2h	Transmission type TxPDO 1	Unsigned8	rw	255	No
	5h	Event Timer TxPDO 1	Unsigned16	rw	0	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1801h	-	TxPDO 2 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	5	No
	1h	COB-ID Receive TxPDO 2	Unsigned32	rw	NODEID+0x280	No
	2h	Transmission type TxPDO 2	Unsigned8	rw	255	No
	5h	Event Timer TxPDO 2	Unsigned16	rw	0	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1802h	-	TxPDO 3 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	5	No
	1h	COB-ID Receive TxPDO 3	Unsigned32	rw	NODEID+0x380	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
	2h	Transmission type TxPDO 3	Unsigned8	rw	255	No
	5h	Event Timer TxPDO 3	Unsigned16	rw	0	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1803h	-	TxPDO 4 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	5	No
	1h	COB-ID Receive TxPDO 4	Unsigned32	rw	NODEID+0x480	No
	2h	Transmission type TxPDO 4	Unsigned8	rw	255	No
	5h	Event Timer TxPDO 4	Unsigned16	rw	0	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
1804h - 18FFh	-	TxPDO 5 - 256 parameters	Record	-	-	-
	0h	Largest subindex supported	Unsigned8	ro	5	No
	1h	COB-ID Receive TxPDO 5 – 256	Unsigned32	rw	0x80000000	No
	2h	Transmission type TxPDO 5 – 256	Unsigned8	rw	-	No
	5h	Event Timer TxPDO 5 – 256	Unsigned16	rw	0	No

Subindex 0 specifies the number of the last valid subindex. The individual bits of the 32-bit value of **subindex 1** have the following meaning:

Subindex 1, bit	Value/Meaning
31 (MSB)	0: PDO exists and is valid 1: PDO does not exist or is invalid Only the first four TxPDOs of the CANopen master are activated. All other TxPDOs are deactivated. If more than four TxPDOs are used, the COB-IDs for the additional TxPDOs themselves must be issued (bits 10 - 0 in this subindex). For this reason, bit 31 of the TxPDOs 5 - 256 has a value of 1 and is to be set to 0 by the user as soon as the user issues a COB-ID.
30	0: The PDO can be requested via a remote request 1: The PDO cannot be requested via a remote request
29	0: 11-bit CAN ID 1: 29-bit CAN ID (not supported by CANopen master option module)

Subindex 1, bit	Value/Meaning
28 - 11	Bits 28 - 11 with 29-bit CAN ID (not supported by CANopen master option module)
10 - 0 (LSB)	COB-ID of the TxPDO. The first four TxPDOs have a default setting.

The transmission type of the TxPDO is specified with **subindex 2**:

Subindex 2, value	Type	Meaning
0	*Not supported*	
1 – 240	Synchronous	Transmission occurs after transmission of the set number (1 - 240) of SYNC telegrams
254	Asynchronous	The TxPDO is sent as soon as the value of a mapped object has changed or a time event (expiration of an event time, subindex 5) has occurred.
255	Asynchronous	The TxPDO is sent as soon as the value of a mapped object has changed or a time event (expiration of an event time, subindex 5) has occurred.

The time in ms for the event-based PDO transfer of the time event is set in **subindex 5**. If the value is zero, no time event is triggered.

1A00h to 1AFFh - Transmission PDO (TxPDO) mapping parameters

The objects 1A00h to 1AFFh determine the mapped objects of TxPDO 1 to TxPDO 256 of the CANopen master. Object 1A00h describes TxPDO 1, object 1A01h describes TxPDO 2, object 1AFFh describes TxPDO 256 etc..

Index	Subindex	Name	Data type	Access	Default value	Map-able
1A00h - 1AFFh	-	TxPDO mapping	Record	-	-	-
	0h	Number of mapped objects	Unsigned8	rw	0	No
	1h - 8h	Object to be mapped	Unsigned32	rw	0	No

Subindex 0 specifies the number of mapped objects in the TxPDO. As soon as the mapping of a TxPDO is changed, the number of mapped objects must first be set to zero. The mapping can then be set, and the number of mapped objects in subindex 0 must finally be entered. The mapping information of the objects in the PDO are specified in **subindices 1h to 8h**. Subindex 1h contains the information for the first mapped object, subindex 2h for the second mapped object etc.. A maximum of 8 (8h) objects can be mapped, whereby the maximum data range of 8 bytes of the TxPDO may not be exceeded. The contents of the 32-bit value of **subindices 1h to 8h** have the following meaning:

Byte: MSB LSB

Index of the mapped object (16-bit)	Subindex of the mapped object (8-bit)	Number of bits of the mapped object (8-bit)

The objects are mapped in a TxPDO one after another.

1F22h - Concise DCF of network nodes

Contains a list with the "concise DCF" of the individual network nodes loaded on the individual nodes by the CANopen master during network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F22h	-	Concise DCF	Domain	-	-	-
	0h	number entries	Unsigned8 (1 – 127)	ro	127	No
	1h - 7Fh	Concise DCF nodes	Domain	rw	no	No

The object 1F22h contains the "concise DCF" of the individual network nodes in subindices 1 - 127. A subindex is assigned to the number of the network node, i.e. the "concise DCF" of node 18 is in subindex 18, for example. A "concise DCF" (Device Configuration File) is a configuration loaded on the individual nodes by the CANopen master during network bootup. The configuration is described via a succession of objects with accompanying values. This configuration/the "concise DCF" has the following structure:

Entry	Data type
Number of objects	Unsigned32
Index 1	Unsigned16
Subindex 1	Unsigned8
Data length of the object in bytes	Unsigned32
Data	Domain
Index 2	Unsigned16
Subindex 2	Unsigned8
Data length of the object in bytes	Unsigned32
Data	Domain
Index 2	Unsigned16
Subindex 2	Unsigned8
Data length of the object in bytes	Unsigned32
Data	Domain

CANopen objects of the CANopen masters (software version ≥ 01.20)

Entry	Data type
:	:
Index n	Unsigned16
Subindex n	Unsigned8
Data length of the object in bytes	Unsigned32
Data	Domain

The number of following objects is at the beginning of the configuration. This is followed by the individual objects with their index, subindex, data length and data. A maximum of 65,535 bytes can be entered in the subindices. This value applies for all subindices, and not for an individual one!

Example:

A sync interval of 10 ms (object 1006h), a guarding time of 250 ms (object 100Ch) and a life time factor of 4 (object 100Dh) are to be set on the node with number 18 during the network bootup. The data lengths of 32 bits, 16 bits and 8 bits result in accordance with CiA DS 301 (or the node manual). The following data in object 1F22h, subindex 18 of the CANopen master is then entered via an SDO transfer:

Byte	Value	Meaning
0	0x03	Number of objects
1	0x00	
2	0x00	
3	0x00	
4	0x06	Index object 1
5	0x10	
6	0x00	Subindex object 1
7	0x04	Data length object 1
8	0x00	
9	0x00	
10	0x00	
11	0x10	Data object 1
12	0x27	
13	0x00	
14	0x00	
15	0x0C	Index object 2
16	0x10	
17	0x00	Subindex object 2
18	0x02	Data length object 2
19	0x00	
20	0x00	
21	0x00	
22	0xFA	Data object 2
23	0x00	

Byte	Value	Meaning
24	0x0D	Index object 3
25	0x10	
26	0x00	Subindex object 3
27	0x01	Data length object 3
28	0x00	
29	0x00	
30	0x00	
31	0x04	Data object 3

The data length to be specified for SDO transfer is 32 bytes.

1F25h - Network node configuration request

Enables the requesting of configuration of network nodes by the CANopen master during operation.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F25h	-	ConfigureSlave	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	128	No
	1h - 7Fh	Request re-configuration node	Unsigned32	rw	no	No
	80h	Request re-configuration all nodes	Unsigned32	rw	no	No

Configuration/reconfiguration of a network node by the CANopen master during operation can be requested via object 1F25h without the need to perform a complete network boot-up. This may be necessary, for example, if new nodes are added to the network. Reconfiguration is triggered by the writing of the "conf" command to the respective subindex. Subindices 1 to 127 correspond to the number of node to be configured. If the "conf" command is written to subindex 128, all nodes are reconfigured. The "conf" command is represented by the following 32-bit value:

Byte: MSB LSB

f	n	o	c
66h	6Eh	6Fh	63h

The node to be reconfigured may not be found in the OPERATIONAL state. If bit 2 for the respective network node (of corresponding subindex) in object 1F81h is set, the CANopen master carries out a configuration automatically. Details are found in the description on object 1F81h.

1F80h - NMT Startup

Configures the startup behavior of the CANopen master.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F80h	0h	NMT Startup	Unsigned32	rw	0000000Dh	No

The individual bits of the 32-bit value have the following meaning:

Bit	Value/Meaning
0	0: *Not possible* 1: CANopen master option module is NMT master
1	0: After network bootup, switch only those nodes configured as slaves (object 1F81h, bit 0) to the “OPERATIONAL” state 1: After network bootup, switch all nodes to the “OPERATIONAL” state.
2	0: CANopen master switches to the “OPERATIONAL” state automatically after network bootup 1: CANopen master must be switched to the “OPERATIONAL” state by the application (e.g. via CIA405_NMT function block)
3	0: The slaves are switched to the “OPERATIONAL” state automatically after network bootup 1: The slaves must be switched to the “OPERATIONAL” state by the application (e.g. via CIA405_NMT function block)
4	0: When an error control event of a mandatory slave occurs (e.g. EMCY message, node guarding or heartbeat error) (see object 1F81h, bit 3), the slave is handled separately (see object 1F81h) 1: When an error control event of a mandatory slave occurs (e.g. EMCY message, node guarding or heartbeat error) (see object 1F81h, bit 3), all slaves are reset
5 - 31 (MSB)	*Reserved*

1F81h - Network node assignment

Contains a list of the network nodes assigned to the CANopen master and defines the behavior of the CANopen master in case of an error or similar event.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F81h	-	SlaveAssignment	Array	-	-	-
	0h	number entries	Unsigned8 (127)	ro	no	No
	1h - 7Fh	SlaveAssignment	Unsigned32	rw	no	No

The object 1F81h contains the assignment of individual network nodes for the CANopen master in list form in the subindices 1 - 127. The index in the list corresponds to the number of network nodes on the CANopen network.

In greater detail, the contents of a subindex are as follows:

Bit	Value/Meaning
0	0: The node with this number is not a node on the CANopen network, i.e. with nonexistent nodes, bit 0 must be 0 in the respective subindex. 1: The node with this number is a node on the CANopen network.
1	0: The application is informed of an “error control event” or other recognition of a booting slave 1: The application is informed of an “error control event” or other recognition of a booting slave, and the “error control service” is started automatically
2	0: In case of an “error control event” or other recognition of a booting slave, the slave is **not** automatically configured and started 1: In case of an “error control event” or other recognition of a booting slave, the slave is configured and started.
3	0: Optional slave: network is started even if this node does not report in 1: Mandatory slave: network is not started if this node does not report in
4,5,6	*Not supported*
7	*Reserved*
8 - 15 (byte 1)	Retry factor for node guarding. If a value is entered here, it is used for node guarding of the network node by the CANopen master
16 - 31 (bytes 2 + 3)	Guard time for node guarding. If a value is entered here, it is used for node guarding of the network node by the CANopen master. The guard time is specified in ms, whereby only multiples of 10 ms are possible, i.e. 10 ms, 20 ms, 30 ms etc.. Intermediate values are rounded up internally.

1F82h - NMT command request

Enables the requesting of the execution of NMT commands by the CANopen master during operation.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F82h	-	Request NMT	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	128	No
	1h - 7Fh	Request NMT	Unsigned8	rw	no	No
	80h	Request NMT all nodes	Unsigned8	wo	no	No

CANopen master can request the execution of an NMT command via object 1F82h to change a node state. The command is triggered by the writing of the requested command to the respective subindex. Subindices 1 to 127 correspond to the number of the node for which the command is valid. If the command is written to subindex 128, the command is valid for all nodes. The following commands are possible:

Node state	Command/Value
Stopped	4
Operational	5
Reset Node	6
Reset Communication	7
PreOperational	127

1F84h - Network node identification, device type

Enables the specification of the expected device type of a network node upon network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F84h	-	Device Type Identification	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	127	No
	1h - 7Fh	Device Type Identification	Unsigned32	rw	no	No

When the network boots up, the CANopen master can check the device types, i.e. the contents of the object 1000h. The entries in object 1F84h are used for this purpose. Subindices 1 to 127 correspond to the number of the node for which the device type is checked. If a value is not entered in a subindex, checking does not occur. If a value is entered, it is compared to the contents of object 1000h on the network node. If they are different, an error message is sent (COM405_NETWORK_CONTROL function block).

1F85h - Network node identification, manufacturer ID

Enables the specification of the expected vendor ID of a network node upon network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F85h	-	Vendor Identification	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	127	No
	1h - 7Fh	Vendor Identification	Unsigned32	rw	no	No

When the network boots up, the CANopen master can check the vendor IDs, i.e. the contents of the object 1018h (subindex 1). The entries in object 1F85h are used for this purpose. Subindices 1 to 127 correspond to the number of the node for which the vendor ID is checked. If a value is not entered in a subindex, checking does not occur. If a value is entered, it is compared to the contents of object 1018h, subindex 1, on the network node. If they are different, an error message is sent (COM405_NETWORK_CONTROL function block). You must take into account that not all CANopen network nodes support object 1018h, which means that checking would not be possible.

1F86h - Network node identification, product code

Enables the specification of the expected product code of a network node upon network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F86h	-	Product Code	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	127	No
	1h - 7Fh	Product Code	Unsigned32	rw	no	No

When the network boots up, the CANopen master can check the product code, i.e. the contents of the object 1018h (subindex 2). The entries in object 1F86h are used for this purpose. Subindices 1 to 127 correspond to the number of the node for which the product code is checked. If a value is not entered in a subindex, checking does not occur. If a value is entered, it is compared to the contents of object 1018h, subindex 2, on the network node. If they are different, an error message is sent (COM405_NETWORK_CONTROL function block). You must take into account that not all CANopen network nodes support object 1018h, which means that checking would not be possible.

1F87h - Network node identification, revision number

Enables the specification of the expected revision number of a network node upon network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F87h	-	Revision Number	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	127	No
	1h - 7Fh	Revision Number	Unsigned32	rw	no	No

When the network boots up, the CANopen master can check the revision number, i.e. the contents of the object 1018h (subindex 3). The entries in object 1F87h are used for this purpose. Subindices 1 to 127 correspond to the number of the node for which the revision number is checked. If a value is not entered in a subindex, checking does not occur. If a value is entered, it is compared to the contents of object 1018h, subindex 3, on the network node. If they are different, an error message is sent (COM405_NETWORK_CONTROL function block). You must take into account that not all CANopen network nodes support object 1018h, which means that checking would not be possible.

1F88h - Network node identification, serial number

Enables the specification of the expected serial number of a network node upon network bootup.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F88h	-	Serial Number	ARRAY	-	-	-
	0h	number entries	Unsigned8	ro	127	No
	1h - 7Fh	Serial Number	Unsigned32	rw	no	No

When the network boots up, the CANopen master can check the serial number, i.e. the contents of the object 1018h (subindex 4). The entries in object 1F88h are used for this purpose. Subindices 1 to 127 correspond to the number of the node for which the serial number is checked. If a value is not entered in a subindex, checking does not occur. If a value is entered, it is compared to the contents of object 1018h, subindex 2, on the network node. If they are different, an error message is sent (COM405_NETWORK_CONTROL function block). You must take into account that not all CANopen network nodes support object 1018h, which means that checking would not be possible.

1F89h - Wait time for network node boot procedure

Length of the wait time for the boot procedure of "mandatory" network nodes in ms.

Index	Subindex	Name	Data type	Access	Default value	Map-able
1F89h	0h	Boot Time	Unsigned32	rw	0	No

The object specifies the time in ms that a bootup message of "mandatory" slaves (see object 1F81h) is awaited. If the CANopen master does not receive all bootup messages from "mandatory" slaves within this time after the network bootup is started, an error occurs (COM405_NETWORK_CONTROL function block). A time of 0 ms indicates that the CANopen master waits an infinite amount of time for the bootup message from a "mandatory" slave. The wait state is indicated at the COM405_NETWORK_CONTROL function block.

B.3.3 Manufacturer-specific objects of the CANopen master

2001h - Baud rate and node ID of the CANopen master

This object is not yet supported!

Determines the baud rate and node ID of the CANopen master. The values are used based on the settings of object 2002h. If object 2002h has a value of 0, the baud rate and node ID are set via an IEC 61131-3 function block and the values set in object 2001h are not taken into account.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2001h	0h	Baud rate and Node ID	Unsigned32	rw	00 7F h	No

The contents of the 32-bit value of subindex 0h have the following meaning:

Byte:	MSB		LSB
Contents	Reserved (16-bit)	Baud rate (8-bit)	Node ID (8-bit)
Range of values	-	0 - 8	1 - 127

The following baud rate values are permissible:

Baud rate value	Meaning
0	1 mbps
1	800 kbps
2	500 kbps
3	250 kbps
4	125 kbps
5	100 kbps
6	50 kbps
7	20 kbps
8	10 kbps

Default value for the baud rate is 0 (1 mpbs). Default value for node ID is 127.

2002h - Startup behaviour of CANopen master

This object is not yet supported!

Specifies the point to which the CANopen master starts up and then passes control on to IEC 61131-3 function blocks.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2002h	0h	Startup Control Master	Unsigned8	rw	0	No

The individual values have the following meaning:

Value	Meaning
0	Initialization and starting of the network bootup procedure are initiated by IEC 61131-3 function blocks.
1	Initialization is carried out automatically. The contents of object 2001h are used for the baud rate and node ID of the CANopen master. Starting of the network bootup procedure is initiated by IEC 61131-3 function blocks.
2	Initialization is carried out automatically. The contents of object 2001h are used for the baud rate and node ID of the CANopen master. The bootup procedure of the network is started off automatically.
3 - 255	*Reserved*

The set values are not valid until the device is switched off and on again. Manufacturer-specific objects must be saved via object 1010h whenever changes are made.

2003h - User device name

The user can issue his/her own device name with this object.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2003h	0h	User Device Name	Visible String	rw	0	No

Any device name with max. 128 characters can be entered in **subindex 0h**.

2004h - Synchronization with PLC

Specifies the type of synchronization used with the PLC.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2004h	0h	Synchronization PLC	Unsigned8	rw	0	No

The individual values have the following meaning:

Value	Meaning
0	Synchronization with the PLC does not occur.
1	The optional module generates an internal, synchronous IRP based on the “SNYC signal 1” signal, event 11 for event tasks under PROPROG wt II. The SYNC telegram is sent and the synchronous TxPDOs transferred with the generated, synchronous IRP. Ethernet is not deactivated.
2	The optional module generates an internal, synchronous IRP based on the “SNYC signal 2” signal, event 12 for event tasks under PROPROG wt II. The SYNC telegram is sent and the synchronous TxPDOs transferred with the generated, synchronous IRP. Ethernet is not deactivated.
3 - 255	*Reserved*
129	As with 1, except Ethernet is deactivated
130	As with 2, except Ethernet is deactivated
3 - 255	*Reserved*

If Ethernet is not deactivated (1, 2), precise synchronization is not possible!

2005h - Reaction to PLC status

Specifies how the CANopen master reacts to changes to the PLC state.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2005h	0h	Reaction on PLC status	Unsigned8	rw	0	No

The individual values have the following meaning:

Value	Meaning
0	There is no reaction when the PLC exits the “Operation” (RUN) operating mode.
1	The CANopen master option module switches to the “FATAL ERROR” state as soon as the PLC is not in the “Operation” (RUN) operating mode. The network is switched to the “STOPPED” state.
2 - 255	*Reserved*

The set values are not valid until the device is switched off and on again. Manufacturer-specific objects must be saved via object 1010h whenever changes are made.

2020h - Network bootup setting

The boot mechanism of the network can be influenced with this object.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2020h	0h	Network Bootup Adjust	Unsigned32	rw	00000000h	No

The individual bits of the 32-bit value have the following meaning:

Bit	Value/Meaning
0	0: Presence of all possible nodes from 1 to 127 is scanned upon bootup of the network. 1: Only presence of nodes configured in object 1F81h is scanned upon bootup of the network. This setting is not recommended, as non-configured nodes can lead to malfunctions during operation.
1 - 31 (MSB)	*Reserved*

2022h - SDO wait time during the boot procedure of a node

Length of the wait time in ms that the CANopen master waits for an SDO answer during the bootup procedure of a node.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2022h	0h	Node Boot Wait SDO	Unsigned16	rw	50	No

Repeated SDO access to a node is necessary during its bootup procedure. All possible node numbers 1 - 127 are scanned, i.e. object 1000h is read. If a timeout occurs during the reply to this access, the node is viewed as not being present. This timeout is also determined by object 2022h.

If only b maXX 4400 slaves are found on the network, a time of 10 ms can be set.

2023h - Wait time for restoring the default settings

Length of the wait time in ms that the CANopen master waits for a bootup message after a node reset before the DCF is downloaded.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2023h	0h	Node Wait Reset	Unsigned16	rw	5000	No

Before the CANopen master loads the DCF to a node, it must load its default values when command to do so. The node must then be reset. Since the nodes require some time until they have performed a reset, the CANopen master must wait until it can begin downloading the DCF. This wait time is set via this object.

If only b maXX 4400 slaves are found on the network, a time of 50 ms can be set.

2024h - Wait time after network reset before network scanning begins

Length of the wait time in ms that the CANopen master waits until it begins scanning the network after a network reset.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2024h	0h	Network Wait Start Scan	Unsigned16	rw	5000	No

The network is reset (reset communication) before it is scanned. Since the nodes require some time until they have performed a reset, the CANopen master must wait until it can begin scanning. This wait time is set via this object.

If only b maXX 4400 slaves are found on the network, a time of 50 ms can be set.

2025h - Wait time until the next node scan begins

If the CANopen master is missing a node, it cyclically checks whether the module is found on the network.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2025h	0h	Node Wait Next Scan	Unsigned16	rw	2000	No

2100h - Process map configuration

Informs you of the configuration of the process map

Index	Subindex	Name	Data type	Access	Default value	Map-able
2100h	-	PI General Configuration	Array	-	-	-
	0	No. of entries supported	Unsigned8	ro	4	No
	1	Mode	Unsigned8	ro	1	No
	2	Version	Unsigned16	ro	100	No
	3	Size Outputs	Unsigned32	ro	2048	No
	4	Size Inputs	Unsigned32	ro	2048	No

Subindex 1 specifies the mode of the process map. A value of 1 means that the segmented process is used to map the network variables. The segmented process requires configuration of the objects to be mapped (see object 2101h). **Subindex 2** reflects the version number of the process used. **Subindex 3** contains the sizes, in bytes, of the process map for outputs, i.e. data sent to the network by the application (network objects A000h - A279h). **Subindex 4** contains the sizes, in bytes, of the process map for inputs, i.e. data sent from the network to the application (network objects A480h - A6FFh).

2101h - Asynchronous and synchronous areas in process map

Determines the asynchronous and synchronous areas of the process map

Index	Subindex	Name	Data type	Access	Default value	Map-able
2101h	-	PI Asynchronous and Synchronous Area	Array	-	-	-
	0	No. of entries supported	Unsigned8	ro	4	No
	1	Start address and size asynchronous PI outputs	Unsigned32	rw	0000 0100h	No
	2	Start address and size asynchronous PI inputs	Unsigned32	rw	0000 0100h	No
	3	Start address and size synchronous PI outputs	Unsigned32	rw	0400 0100h	No
	4	Start address and size synchronous PI inputs	Unsigned32	rw	0400 0100h	No

The asynchronous and synchronous areas of the process map are determined in subindices 1 - 4 for both outputs and inputs. These areas contain objects transferred to asynchronous or synchronous PDOs. The objects of the process map themselves are determined via objects 2102h (asynchronous objects) and 2103h (synchronous objects).

The 32-bit value of each subindex is divided into two 16-bit values:

Byte:	MSB	LSB
Contents	Start address (byte address) of the area (16-bit)	Size of the area in bytes (16-bit)

The start address specified is relative to the start of the outputs/inputs of the process map. Byte addresses are specified. The size of an area is also specified in bytes. The required size of an area is yielded from the actual created objects in the process map (objects 2102h and 2103h), but multiples of 256 bytes are always to be issued.

You must comply with the following:

- The asynchronous area must be located ahead of the synchronous area
- The asynchronous area must begin at address 0. If there is no asynchronous area, the synchronous area can begin at address 0.
- The areas may not overlap.
- The permissible size of an area is a multiple of 256 bytes
- The size of the process map may not be exceeded on ether side (outputs and inputs) (see object 2101h, subindices 3 and 4, for this purpose).

2102h - Configured asynchronous objects in process map

Enables the specification of asynchronous objects mapped in the process map.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2102h	-	PI asynchronous Object Configuration	Array	-	Optional	-
	0	No. of entries supported	Unsigned8	ro	64	No
	1	1. configured object	Unsigned48	rw	A000 0000 0008h	No
	2	2. configured object	Unsigned48	rw	A040 0008 0018h	No
	3	3. configured object	Unsigned48	rw	A0C0 0020 0010h	No
	4	4. configured object	Unsigned48	rw	A100 0040 0010h	No
	5	5. configured object	Unsigned48	rw	A1C0 0060 0010h	No
	6	6. configured object	Unsigned48	rw	A200 00A0 0010h	No
	7	7. configured object	Unsigned48	rw	A480 0000 0008h	No
	8	8. configured object	Unsigned48	rw	A4C0 0008 0018h	No
	9	9. configured object	Unsigned48	rw	A540 0020 0010h	No
	10	10. configured object	Unsigned48	rw	A580 0040 0010h	No
	11	11. configured object	Unsigned48	rw	A640 0060 0010h	No
	12	12. configured object	Unsigned48	rw	A680 00A0 0010h	No
	13	13. configured object	Unsigned48	rw	0	No
	:	:	:	:	:	:
	64	64. configured object	Unsigned48	rw	0	No

Subindex 0 contains the maximum possible number of configurable objects. The following **subindices 1 - 64** contain the configuration of the objects mapped in the process map. The objects are mapped in the asynchronous area of the process map, i.e. these objects are used for asynchronous PDOs.

The 48-bit value is divided into three 16-bit values:

Byte: MSB LSB

Object index	Start address in DPRAM	Number of subindices used (max. 254)

The object index specifies the index of the configured object. The following areas are permissible for the object index, based on the data type:

- Asynchronous objects written by the application:

Area	CANopen data type
A000h - A01Fh	Integer8
A040h - A05Fh	Unsigned8
A0C0h - A0DFh	Integer16
A100h - A11Fh	Unsigned16

Area	CANopen data type
A1C0h - A1DFh	Integer32
A200h - A21Fh	Unsigned32

- Asynchronous objects read by the application:

Area	CANopen data type
A480h - A49Fh	Integer8
A4C0h - A4DFh	Unsigned8
A540h - A55Fh	Integer16
A580h - A59Fh	Unsigned16
A640h - A65Fh	Integer32
A680h - A69Fh	Unsigned32

The start address in DPRAM is a byte address specified relative to the start of the process map outputs (A000h etc.) or inputs (A480h). The number of subindices used specifies the greatest subindex up to which objects of the specified object index are used.

NOTE

You must ensure that the address area of the asynchronous objects does not overlap the address area of the synchronous objects (object 2103h). To prevent this, the asynchronous objects should be created ahead of the synchronous objects.

2103h - Configured synchronous objects in process map

Enables the specification of synchronous objects mapped in the process map.

Index	Subindex	Name	Data type	Access	Default value	Map-able
2103h	-	PI synchronous Object Configuration	Array	-	Optional	-
	0	No. of entries supported	Unsigned8	ro	64	No
	1	1st configured object	Unsigned48	rw	A020 0400 0008h	No
	2	2nd configured object	Unsigned48	rw	A060 0408 0018h	No
	3	3rd configured object	Unsigned48	rw	A0E0 0420 0010h	No
	4	4th configured object	Unsigned48	rw	A120 0440 0010h	No
	5	5th configured object	Unsigned48	rw	A1E0 0460 0010h	No
	6	6th configured object	Unsigned48	rw	A220 04A0 0010h	No
	7	7th configured object	Unsigned48	rw	A4A0 0400 0008h	No
	8	8th configured object	Unsigned48	rw	A4E0 0408 0018h	No

Index	Subindex	Name	Data type	Access	Default value	Map-able
	9	9th configured object	Unsigned48	rw	A560 0420 0010h	No
	10	10th configured object	Unsigned48	rw	A5A0 0440 0010h	No
	11	11th configured object	Unsigned48	rw	A660 0460 0010h	No
	12	12th configured object	Unsigned48	rw	A6A0 04A0 0010h	No
	13	13th configured object	Unsigned48	rw	0	No
	:	:	:	:	:	:
	64	64th configured object	Unsigned48	rw	0	No

Subindex 0 contains the maximum possible number of configurable objects. The following **subindices 1 - 64** contain the configuration of the objects mapped in the process map. The objects are mapped in the synchronous area of the process map, i.e. these objects are used for synchronous PDOs.

The 48-bit value is divided into three 16-bit values:

Byte: MSB		LSB
Object index	Start address in DPRAM	Number of subindices used (max. 254)

The object index specifies the index of the configured object. The following areas are permissible for the object index, based on the data type:

- Synchronous objects written by the application:

Area	CANopen data type
A020h - A03Fh	Integer8
A060h - A07Fh	Unsigned8
A0E0h - A0FFh	Integer16
A120h - A13Fh	Unsigned16
A1E0h - A1FFh	Integer32
A220h - A23Fh	Unsigned32

- Synchronous objects read by the application:

Area	CANopen data type
A4A0h - A4BFh	Integer8
A4E0h - A4FFh	Unsigned8
A560h - A57Fh	Integer16
A5A0h - A5BFh	Unsigned16
A660h - A67Fh	Integer32
A6A0h - A6BFh	Unsigned32

The start address in DPRAM is a byte address specified relative to the start of the asynchronous process map outputs (A000h etc.) or inputs (A480h). The number of subindices used specifies the greatest subindex up to which objects of the specified object index are used.

NOTE

You must ensure that the address area of the asynchronous objects (object 2101h) does not overlap the address area of the synchronous objects. The synchronous objects should always be created after the asynchronous objects. When issuing the start address of the first synchronous object, sufficient space for any additional necessary asynchronous objects should be available ahead of the address.

Revision index

Revision level	State	Modifications
5.03002.02	10.01.2007	Application Manual extended with ProMaster and ProProg wt III new: Chapter 4.3 ProMaster, ProCANopen, ProProg wt III and Motion Control Chapter 4.3 (old) now chapter 4.4 (new)

Revision index

Index

be in motion