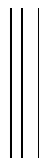
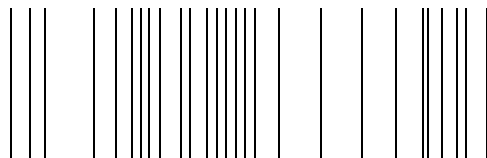
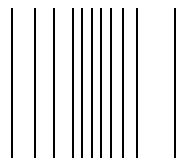


**be in motion** **be in motion**



**BM4-O-PLC-01**

**b maXX drive PLC**

**Application Manual**

**E**

5.02004.04



Title	Application Manual
Product	<b>b maXX drive PLC</b> BM4-O-PLC-01
Last Revision:	01 March 2007
Copyright	<p>Owners may make as many copies as they like of this Application Manual exclusively for their own internal use. You are not allowed to copy or duplicate even extracts from this Application Manual for any other purposes.</p> <p>You are not permitted to exploit or communicate the contents of this Application Manual.</p> <p>Any other designations or company logos used in this Application Manual may be trademarks whose use by third parties for their own purposes may affect the rights of the owner of the trademark.</p>
Binding nature	<p>This Application Manual is a part of the unit/machine. This Application Manual must always be available to operators and be legible. If the unit/machine is sold, the owner must pass on this Application Manual together with the unit/machine.</p> <p>After selling the unit/machine you must pass on this original and all the copies that you made to the purchaser. After disposing of the machine in any way, you must destroy this original and all the copies that you made.</p> <p>When you pass on this Application Manual, all earlier revisions of the corresponding Application Manual are invalidated.</p> <p>Note that all the data/numbers/information that are quoted are <b>current values at the time of printing</b>. This information is <b>not legally binding</b> for dimensioning, calculation and costing.</p> <p>Within the scope of further-development of our products, Baumüller Nürnberg GmbH reserve the right to change the technical data and handling.</p> <p>We cannot guarantee this Application Manual is completely error-free unless this is expressly indicated in our General Conditions of Business and Delivery.</p>
Manufacturer	<p>Baumüller Nürnberg GmbH Ostendstr. 80 - 90 90482 Nuremberg Germany Tel. +49 9 11 54 32 - 0 Fax: +49 9 11 54 32 - 1 30 <a href="http://www.baumueller.de">www.baumueller.de</a></p>



# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	First Steps	5
1.2	Terms Used	5
<b>2</b>	<b>Basic Safety Instructions</b>	<b>7</b>
2.1	Hazard information and instructions	7
2.1.1	Structure of hazard information	8
2.1.2	Hazard advisories that are used	9
2.2	Information signs	11
2.3	Legal information	11
2.4	Appropriate Use	11
2.5	Inappropriate Use	12
2.6	Protective equipment	12
2.7	Personnel training	13
2.8	Safety measures in normal operation	13
2.9	Responsibility and liability	13
2.9.1	Observing the hazard information and safety instructions	13
2.9.2	Danger arising from using this module	14
2.9.3	Warranty and Liability	14
<b>3</b>	<b>Commissioning</b>	<b>15</b>
3.1	General safety regulations	15
3.2	Requirements of the personnel carrying out work	15
3.3	Description/inspection of the safety and monitoring systems	16
3.4	Description and inspection of the controls and displays	16
3.4.1	LEDs for displaying operating status of BM4-O-PLC-01	16
3.4.2	S1 switch/pushbutton for changing operating status conditions of the BM4-O-PLC-01	17
3.5	Commissioning sequence	18
3.5.1	Activation	18
3.5.2	Testing the function	19
3.5.3	Sending a boot project	19
<b>4</b>	<b>Operation</b>	<b>25</b>
4.1	Programming systems PROPROG wt II and ProProg wt III	25
4.2	b maXX drive PLC project	27
4.3	b maXX drive PLC configuration	27
4.4	b maXX drive PLC resource	28
4.4.1	Communication and Connection	31
4.4.2	Control Dialog for Resources	34
4.4.3	Data Area	42
4.4.4	b maXX drive PLC event tasks	45
4.5	b maXX drive PLC user libraries	47
4.5.1	b maXX drive PLC firmware	49
4.5.2	b maXX drive PLC Board functions	50
4.5.3	b maXX drive PLC data types	55
4.5.4	The Standard Function Block Libraries	59
4.5.5	b maXX drive PLC technology components	60
4.5.6	Inserting a User Library into a Project	60
4.6	BACI system description for the b maXX drive PLC	62



## TABLE OF CONTENTS

4.6.1	Direct access to the BACI hardware from the b maXX drive PLC . . . . .	62
4.6.2	Process data communication between b maXX controller and b maXX drive PLC, syn- chronization to a common BACI hardware signal source . . . . .	63
4.6.3	b maXX drive PLC interrupt sources that the application can use . . . . .	65
4.6.4	Generating BACI signals via function block TIMER_A_INIT . . . . .	67
4.6.5	"TIMER_A_INIT" function block . . . . .	68
4.6.6	Sample Configurations . . . . .	74
4.6.7	Example A: Implementing BACI process data communication within a BACI event task 76	
4.6.8	Example B: Implementing BACI process data communication via a self-generated SYNC1 signal . . . . .	79
4.6.9	Example C: Implementing BACI process data communication via an external SYNC1 sig- nal . . . . .	82
4.6.10	Example D: Implementing simple serial RS485 communication via a CPU timer . . .	84
4.6.11	Description of the blocks for serial RS485 communication via the X1 interface (9-pin fe- male Sub-D connector) on the b maXX drive PLC . . . . .	86
4.6.12	Function block TER_INIT . . . . .	88
4.6.13	Function block TER_R . . . . .	93
4.6.14	Function block TER_S . . . . .	96
4.6.15	Function block T64_RSYN . . . . .	98
4.6.16	Function block T64_REC . . . . .	100
4.6.17	Function block TER_USS . . . . .	103
4.7	Code runtimes . . . . .	108
4.7.1	Function block TIME_MEASURE_START . . . . .	110
4.7.2	Function block TIME_MEASURE_END . . . . .	111
4.8	Practical information . . . . .	112
4.8.1	PLC error "Watchdog in task 'x' exceeded!" . . . . .	112
<b>5</b>	<b>Internal communications interface to the (BACI) controller . . . . .</b>	<b>115</b>
5.1	BACI (BAumüller Component Interface) general . . . . .	115
5.2	ProMaster, ProProg wt III and Motion Control . . . . .	116
5.2.1	Requirements . . . . .	116
5.2.2	Procedure to be carried out . . . . .	117
5.2.3	Commissioning of the b maXX 4400 with b maXX controller and b maXX drive PLC . . . 117	
5.2.4	Creating of an IEC project with ProProg wt III . . . . .	117
5.2.5	Creation of a machine configuration in ProMaster . . . . .	119
5.2.6	Configuring EtherCAT communication with ProEtherCAT . . . . .	124
5.2.7	Configuring BACI communication . . . . .	125
5.2.8	Configuration of the PLC . . . . .	130
5.2.9	Downloading the data to the EtherCAT master and the b maXX drive PLC . . . . .	139
5.2.10	Programming the IEC project . . . . .	140
5.3	PROPROG wt II . . . . .	144
5.3.1	b maXX PLC and controller . . . . .	144
5.3.2	Programming BACI communication on the b maXX drive PLC . . . . .	144
5.3.3	Function Blocks for BACI Overview . . . . .	146
	<b>Anhang A - Abbreviations . . . . .</b>	<b>165</b>
	<b>Index . . . . .</b>	<b>167</b>

# INTRODUCTION

This b maXX drive PLC 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.

It is assumed that you have understood and used the b maXX drive PLC Operating Instructions.

In this chapter, we will describe the first steps that you should carry out after getting this unit. We will define terms that are used in this documentation on a consistent basis and will inform you about the responsibilities you must consider when using this unit.

## 1.1 First Steps

---

- ◆ Check the shipment.
- ◆ Pass on all the documentation that was supplied with the plug-in module to the appropriate departments in your company.
- ◆ Deploy suitable personnel for assembly and commissioning.
- ◆ Pass on these operating instructions to this personnel and ensure that they have read and understood the safety instructions and that they are following them.

## 1.2 Terms Used

---

In this documentation, we will also refer to Baumüller's "**b maXX drive PLC**" product as "option module", "plug-in module", „b maXX PLC“ or only „PLC“. Furthermore we will use the term BM4-O-PLC-01.

For a list of the abbreviations that are used, refer to [►Appendix A Abbreviations◄](#) from page 165 onward.

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".



# 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

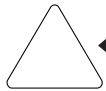
## 2.1 Hazard information and instructions

### CAUTION

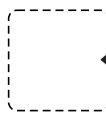
- Damage to property
- Slight to medium personal injury **can** occur

#### 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.

## 2.1 Hazard information and instructions



### 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



Sicherheitsschuhe tragen

## 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, over-current 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 the operating instructions of this module.

- 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 the operating instructions of this module.

### 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 cannot, 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 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

Qualified personnel (specialists) are defined as follows:

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 this plug-in module 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 these operating instructions, we use visually consistent safety instructions that are intended to prevent injury to people or damage to property.



### 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 the Operating Instructions of this module 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 plug-in 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 7 onward and to the Operating Instructions.

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 12 onward or below caused the fault:

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

## COMMISSIONING

In this chapter, we will describe how you commission the b maXX drive PLC plug-in module that you just assembled and installed (see "b maXX PLC Operating Instructions"). Commissioning ensures that the plug-in module functions correctly.

Before starting commissioning, ensure that the following conditions have been met:

- 1 The plug-in module has been assembled correctly.
- 2 The plug-in module has been installed correctly.
- 3 All the safety equipment has been commissioned.
- 4 The b maXX unit is ready for use.

### 3.1 General safety regulations

- Observe the ► [Basic Safety Instructions](#) ◀ from page 7 onward.



#### DANGER

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

- serious personal injury
- death



Danger from: **mechanical effects**. *At commissioning, the drive can rotate.*

Keep far enough away from the rotating parts. Note that when drives are starting up machine parts can be set in motion. In all cases, activate the machine's safety devices.

### 3.2 Requirements of the personnel carrying out work

Commissioning work must only be carried out by trained specialists who have understood the safety regulations and information and can implement them.

### 3.3 Description/inspection of the safety and monitoring systems

Before you commission the b maXX drive PLC, you must eliminate any errors/error messages that may be present on the b maXX 4400 unit. These errors may be due to faulty assembly (e.g. defective cables) or faulty installation (e.g. no power supply). You must not continue with commissioning until you have eliminated the errors.

### 3.4 Description and inspection of the controls and displays

#### 3.4.1 LEDs for displaying operating status of BM4-O-PLC-01

The b maXX drive PLC option module has two red (H2 and H4) and two green (H1 and H3) LEDs as display elements.

These LEDs are used by the operating system of the b maXX drive PLC during initialisation (start-up phase).

During operation (after the start-up phase of the operating system) the LEDs can be used in the application program of the b maXX drive PLC.



Figure 1: LEDs of the BM4-O-PLC-01 plug-in module

#### 3.4.1.1 Switching on and initialisation of BM4-O-PLC-01

- All the option modules in the b maXX 4400 unit must have reached a specific internal operating status after switching in the supply voltage before they may be actuated by the b maXX controller and the b maXX drive PLC.  
This stage, in which the system waits for a global ready message of all the boards is displayed by a counterclockwise-rotating LED bit pattern. (This means that an LED lights up every 500 ms in the sequence H4 -> H2 -> H1 -> H3 -> H4 etc.)
- After the global ready message of all option modules has been issued, the b maXX drive PLC must wait until the b maXX controller recognizes and preinitializes it. This stage is indicated by a clockwise-rotating bit pattern. This means an LED sequence of H4 -> H3 -> H1 -> H2 -> H4 etc. every 500 ms.

The two sequences that we have just described can be completed very quickly, which means that you do not necessarily have to observe the associated operating displays.



After this, a PC and the b maXX drive PLC can, in principle, carry out PROPROG communication via the serial RS232 port on the b maXX controller.

From now on, PROPROG communication is also possible by means of TCP/IP if an option module with Ethernet functionality is present that has been configured for communication with the b maXX drive PLC.

- If a boot project is present, the system now loads it (boot project will be read from flash, translated and stored in SDRAM as an executable program code). The top two LEDs (H2 and H1) flash rapidly to show that the boot project is being loaded.

At the end of the start-up phase, the LEDs show the following PLC-specific operating status conditions:

- No project available, status "POWER ON":  
→ bottom green LED and red one (H3 and H4) light up.
- Project available, status "STOP":  
→ only the bottom right-hand red LED (H4) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:  
→ Only the bottom left-hand green LED (H3) lights up.
- Project available, status "RUN":  
→ both green LEDs (at the bottom and at the top left, H3 and H1) light up.

In the "RUN" status, users can freely program the four LEDs. For information on programming, see [►Function block LED ◄](#) from page 55 onward.

#### 3.4.1.2 Operation of BM4-O-PLC-01

At the end of the start-up phase, the LEDs show the following PLC-specific operating status conditions:

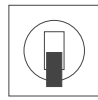
- No project available, status "POWER ON":  
→ bottom green LED and red one (H3 and H4) light up.
- Project available, status "STOP":  
→ only the bottom right-hand red LED (H4) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:  
→ Only the bottom left-hand green LED (H3) lights up.
- Project available, status "RUN":  
→ both green LEDs (at the bottom and at the top left, H3 and H1) light up.

In the "RUN" status, users can freely program the four LEDs. For information on programming, see [►Function block LED ◄](#) from page 55 onward.

#### 3.4.2 S1 switch/pushbutton for changing operating status conditions of the BM4-O-PLC-01

For changing the operating status' there is a switch/pushbutton S1 on the b maXX drive PLC option module.

## 3.5 Commissioning sequence



Pushbutton upward:    RESET  
Switch in middle:       STOP  
Switch at bottom:       RUN

Figure 2:        Switch S1 on the BM4-O-PLC-01 front panel



### NOTE

The user project is only able to start, if the switch/pushbutton S1 is at the bottom „RUN“.

The pushbutton upward resets **only** the b maXX drive PLC option module and not the controller. After resetting, the PLC may no longer be able to communicate with the option modules. In this case, you must reboot the system mandatory.



### DANGER

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

- serious personal injury
- death

Danger from: **mechanical effects**. *You can program the boot project such that the drive and the machine parts respectively rotate or move!*

Keep far enough away from the rotating parts. Note that when drives are starting up machine parts can be set in motion. In all cases, activate the machine's safety devices.

## 3.5 Commissioning sequence

Commissioning is divided into the following procedures:

- 1 Activation.
- 2 Testing the function.

### 3.5.1 Activation

- Read and observe the [►General safety regulations ◀](#) from page 15 onward.
- You must have carried out correctly section "Assembly and Installation" (see ►Operating Instructions b maXX drive PLC◀).
- Set switch/pushbutton S1 on the b maXX drive PLC to „STOP“ (centre position).
- Switch on the b maXX 4400 basic unit.



### NOTE

You must not remove or plug in the b maXX drive PLC plug-in module while the b maXX 4400 basic unit is switched on. Switch the unit off first.

### 3.5.2 Testing the function

- After you switch on the b maXX unit, two status conditions can apply:
  - No boot project (= no user project on the PLC):  
LED H2 (at the top right) lights up briefly after a short time, the bottom LEDs H3 and H4 (red and green) are permanently lit up. This means that no project is present. In this "POWER ON" status, the PLC is waiting for communication.
  - Boot project present:  
When you switch on, the system loads the boot project. When you do this, the top LEDs flash. After a short time, LED H4 (at the bottom right) lights up red. The PLC is in the "STOP" status.
- While switch S1 on the b maXX drive PLC option module is set to "STOP" (centre position), an existing boot project cannot start up.  
If you want to start an existing boot project by setting switch S1 on the b maXX drive PLC option module to the "RUN" (bottom) position, **first** ensure that you have loaded the **correct** boot project for your application for **this** plant in **this** unit on the PLC!

Refer to the b maXX PLC Application Manual for more information on how to ensure that this is the case or how you can send a boot project to the PLC, for example, see [►Operation ◀](#) from page 25 onward.

How you can send a boot project to the b maXX drive PLC, for example, see [►Sending a boot project ◀](#) from page 19 onward.

#### DANGER



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

- serious personal injury    • death



Danger from: **mechanical effects**. A boot project on the PLC can start up if you set switch S1 from "STOP" (center position) to "RUN" (bottom position) or if switch S1 is set to "RUN" when you switch on the b maXX unit. You can program the boot project such that the drive and the machine parts respectively rotate or move!

Keep far enough away from the rotating parts. Note that when drives are starting up machine parts can be set in motion. In all cases, activate the machine's safety devices.

### 3.5.3 Sending a boot project

- 1 Connect your PC's serial port to the female Sub-D connector on the **b maXX controller** (slot F).

#### NOTE



Do **not** connect to the female Sub-D connector on the b maXX drive PLC! This functions as a serial RS485 terminal interface and you cannot use it for communication with PROPROG wt II / ProProg wt III.

- 2 Start the PROPROG wt II (or ProProg wt III) operation software on your PC and choose a project for b maXX drive PLC. This project must have already been successfully compiled. You can also recompile the project using menu item "Build -> Rebuild Project".
- 3 In the operator control program, click on resource control in menu "Online".

## 3.5 Commissioning sequence

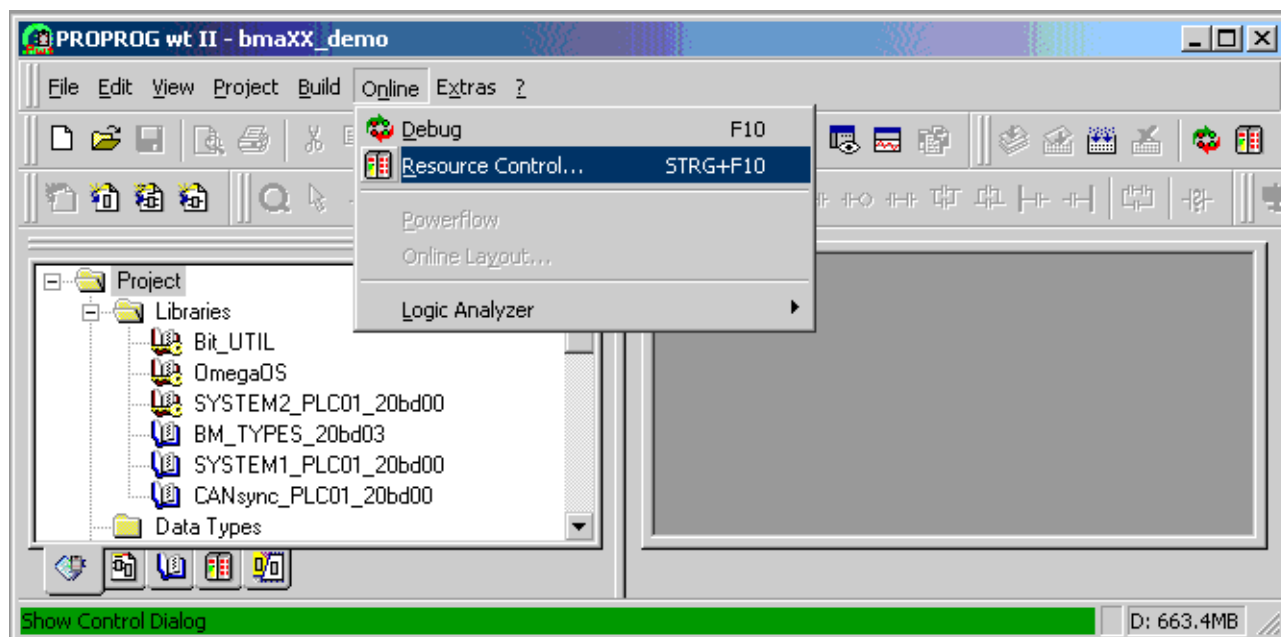


Figure 3: PROPROP wt II user interface

The system opens the following window:



Figure 4: Resource control "On"

4 Click on the "Download" pushbutton. The system displays the following image:

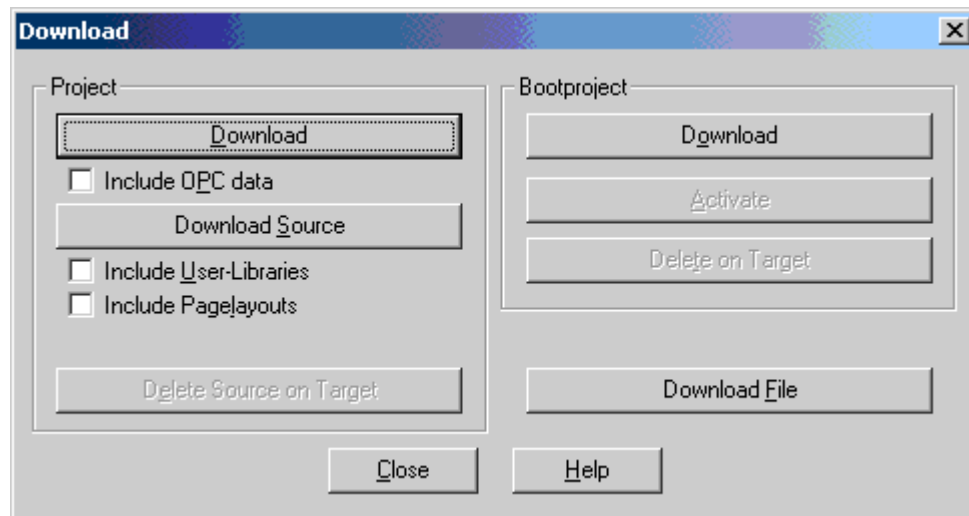


Figure 5: Send

**5** Click on "Download boot project".

- Stage 1: Clear flash memory:  
A boot project that may be present is deleted. In the operator control program, the system now displays a white bar at the bottom of the screen. On the b maXX drive PLC all four LEDs flash at the same time.
- Stage 2: Send project:  
The system sends the boot project. The bar at the bottom of the screen turns blue and shows the progress of the transfer. On the b maXX drive PLC the LEDs (H1 (green), H2 (red), H3 (green), H4 (red)) flash diagonally during transfer.

**NOTE**

After sending is completed, you should switch the b maXX 4400 basic unit off and back on so that the b maXX controller can correctly initialize all the option modules in the correct time order.

Alternatively, it may be enough to start the project on the b maXX drive PLC either by means of menu item "Download" / "Activate boot project" and "Cold boot" or by resetting on the b maXX drive PLC. However, this procedure is highly application-dependent, which means that the commissioning engineer bears the responsibility for using it.

You can stop or restart execution of a project on the b maXX drive PLC using resource control. As an alternative, you can achieve the same effect using switch S1 on the b maXX drive PLC.

**NOTE**

Commissioning engineers should always be aware that in the "Stop" status no application projects can be executed, which means that there is no more communication between the b maXX controller and other option modules.

## 3.5 Commissioning sequence



Figure 6: "Operation" resource control

When you click on the "Info" pushbutton, the system displays the following image containing information:

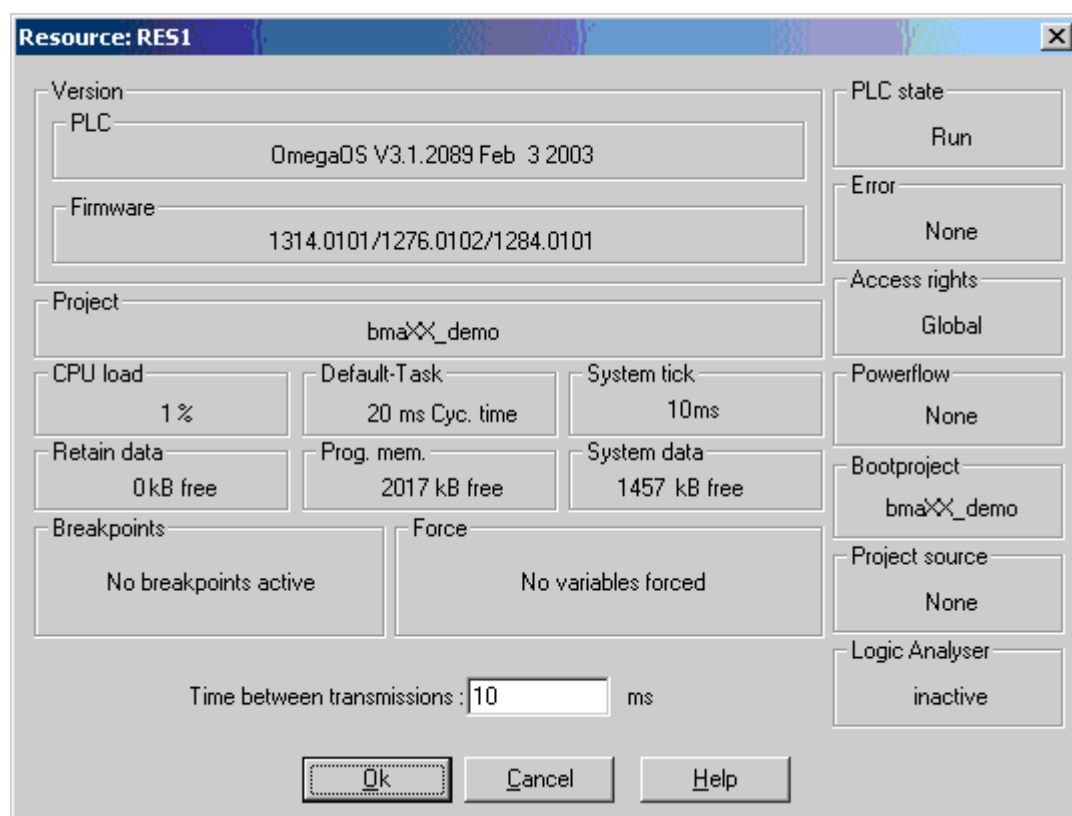


Figure 7: Resource information window

**NOTE**

Meanings of the displayed versions strings in the information window:

PLC string: "OmegaOS V3.1.2078 Feb 3 2003"  
-> "OmegaOS": Product designation  
-> "V3.1.2078": OmegaOS compiler information  
-> "Feb 3 2003": Date on which the PLC version was created.

Firmware string: "1290.0101/1276.0102/1284.0101"  
-> "1314.0101": Version OmegaOS software 6.1314.0101  
-> "1276.0102": Version FPGA software 6.1276.0102  
-> "1284.0101": Version BOOT software 6.1284.0101"

Within the scope of the further technical development of the products higher versions and newer dates can be installed here.

Click on "OK" to acknowledge and go back to resource control (see [▶Figure4](#) on page 20).





## OPERATION

### 4.1 Programming systems PROPROG wt II and ProProg wt III

---

PROPROG wt is a standard programming system that is based on the IEC 61131-3 standard.

The programming system provides powerful functions for the various development stages of PLC applications, like:

- editing
- compiling
- debugging
- printing

The PROPROG wt programming system is based on modern 32-bit Windows technology and allows users to operate the system easily using tools like zoom, scroll, special toolbars, drag & drop, a shortcut manager and dockable windows.

In particular, the system makes possible processing of several configurations and resources within a project as well as integration of project libraries. Apart from this, it has a powerful system for debugging. Using the easy-to-use project tree editor, you can display and edit projects. This makes it possible to represent easily and transparently the complex structure of the IEC 61131-3 standard. This feature allows users to easily paste and edit in the project tree POEs, data types, libraries and configuration elements.

The PROPROG wt programming system consists of a PLC-independent core for programming in the various IEC programming languages: these include the text languages Structured Text (ST) and Statement List (STL) as well as the graphic languages Function Block Diagram (FBD), Ladder Diagram (LAD) and Sequential Function Chart (SFC).

An editor wizard is available for programming in each of the languages, which allows you to quickly and easily paste prepared keywords, statements, operators, functions and function blocks into the individual work sheets. You can also use the editor wizard for declaring variables and data types. The independent core of the system is complemented by special sections that are matched to various PLCs.

The new easy online handling and the 32-bit simulation allow you options for debugging the address status and a real-time multitasking test environment.

An easy-to-use tool for project documentation allows you to print the project in a time-saving optimized form (using less paper) with a page layout that can be specified by individual users.



### DANGER

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

- serious personal injury
- death

**Danger from: mechanical effects.** A boot project on the b maXX drive PLC starts up, if the switch/pushbutton S1 is switched from „STOP“ (center position) to „RUN“ (bottom position) or if the switch is on „RUN“ at start up of the b maXX system. The boot project can be programmed in such a way that the machine/system or parts of the machine/system can be set in motion.

Keep far enough the moving parts of the machine/system. Note that from the other modules which are connected (to the b maXX drive PLC), machine parts can be set in motion. In all cases, activate the machine's safety devices.



### DANGER

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

- serious personal injury
- death

**Danger from: mechanical effects.** Online changes are sent automatically to the PLC and operate immediately without stopping the program execution at the PLC before.

Keep far enough the moving parts of the machine/system. Note that from the other modules which are connected (to the b maXX drive PLC), machine parts can be set in motion. In all cases, activate the machine's safety devices.



### DANGER

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

- serious personal injury
- death

**Danger from: mechanical effects.** If you force or overwrite variables during PLC runtime, the PLC program is running with the forced or overwritten variables.

Keep far enough the moving parts of the machine/system. Note that from the other modules which are connected (to the b maXX drive PLC), machine parts can be set in motion. In all cases, activate the machine's safety devices.

**DANGER**

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

- serious personal injury
- death



*Danger from: **mechanical effects**.* If you use breakpoints during PLC runtime, the PLC program is stopped when reaching a breakpoint.

Keep far enough the moving parts of the machine/system. Note that from the other modules which are connected (to the b maXX drive PLC), machine parts can be set in motion. In all cases, activate the machine's safety devices.

## 4.2 b maXX drive PLC project

You start a new maXX drive PLC project under PROPROGRAM by means of menu item "File/New project". Using the "Template for BM4\_O\_PLCC01", you open a b maXX drive PLC project. Processor type BM4\_O\_PLCC01 is preset in this project in the configuration.

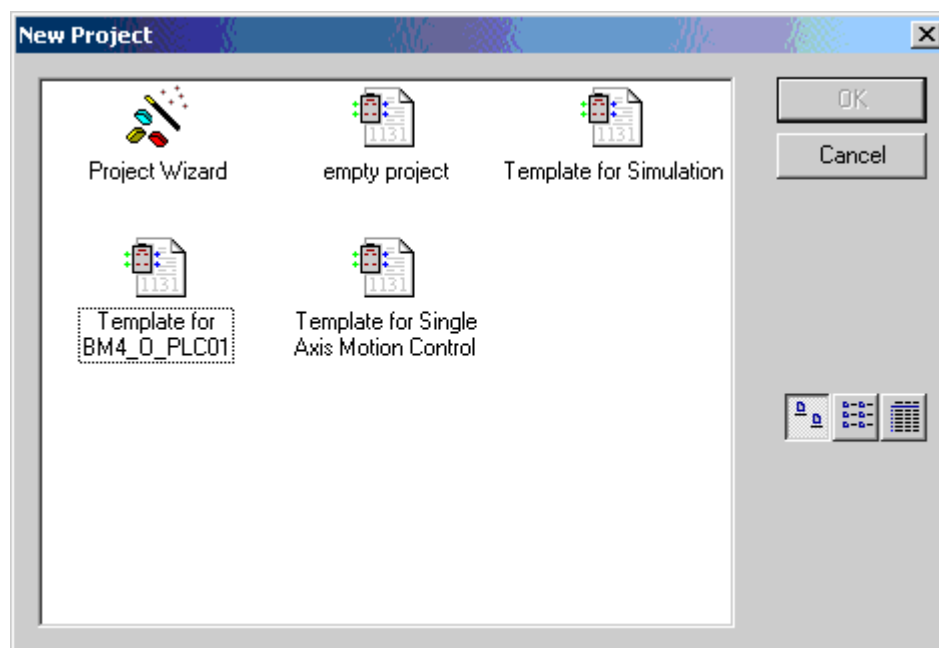


Figure 8: BM4\_O\_PLCC01 project template under "New Project".

## 4.3 b maXX drive PLC configuration

As an IEC 61131-3 programming environment, you can use PROPROGRAM to program different target systems (CPUs). It is also possible to program different target systems in one project. You create a program for the BM4\_O\_PLCC01 target system under "Physical Hardware" using resource BM4\_O\_PLCC01.

You open the template for BM4\_O\_PLCC01 by means of the "New project" menu. Under properties, configuration CONF1 has PLC type SH03\_30.

## 4.4 b maXX drive PLC resource

A configuration consists of at least one resource. The resource contains the specific data range for the b maXX drive PLC, the communication source, global variable worksheets and the tasks containing the program parts.

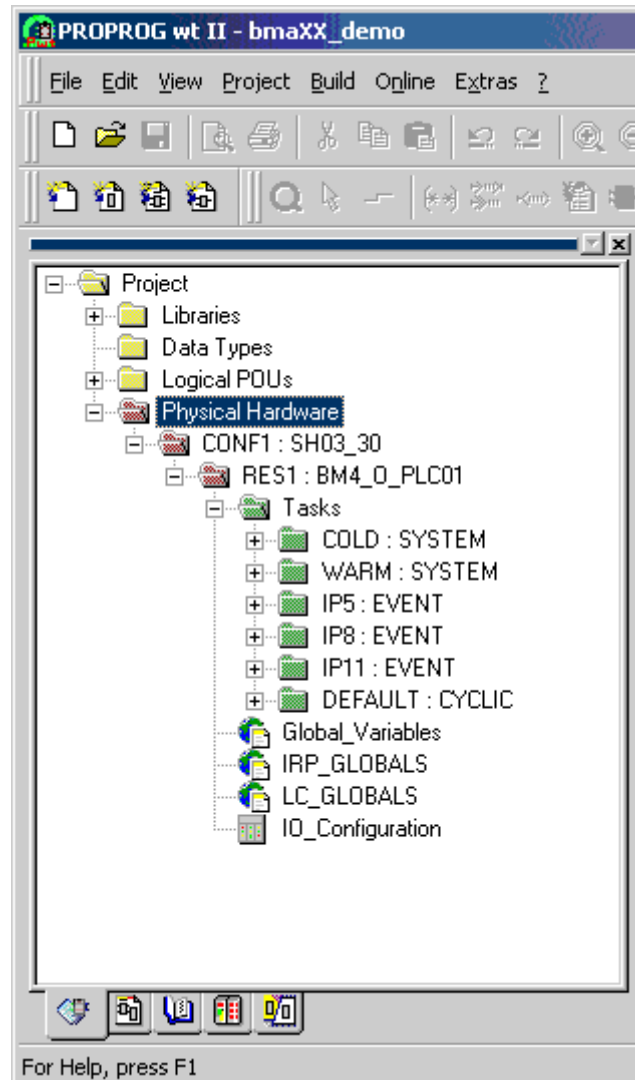


Figure 9: Setting the property of a b maXX drive PLC configuration under "Physical Hardware".

Overview of the settings within the physical hardware:

Project template	Configuration	Resource
BM4_O_PLC01	SH03_30	BM4_O_PLC01

A project can include several configurations each with several resources.

## 4.4 b maXX drive PLC resource

The resource contains the b maXX drive PLC-specific settings for one program:

- Data Area
- Communication source

- Global variable worksheets
- Various tasks using the program
- Documentation worksheets and I/O configuration  
(The I/O configuration is already set and you do not need to change it.)

You can assign several resources with different ports to one configuration. This makes it possible to implement a complete application with several drives in one project.

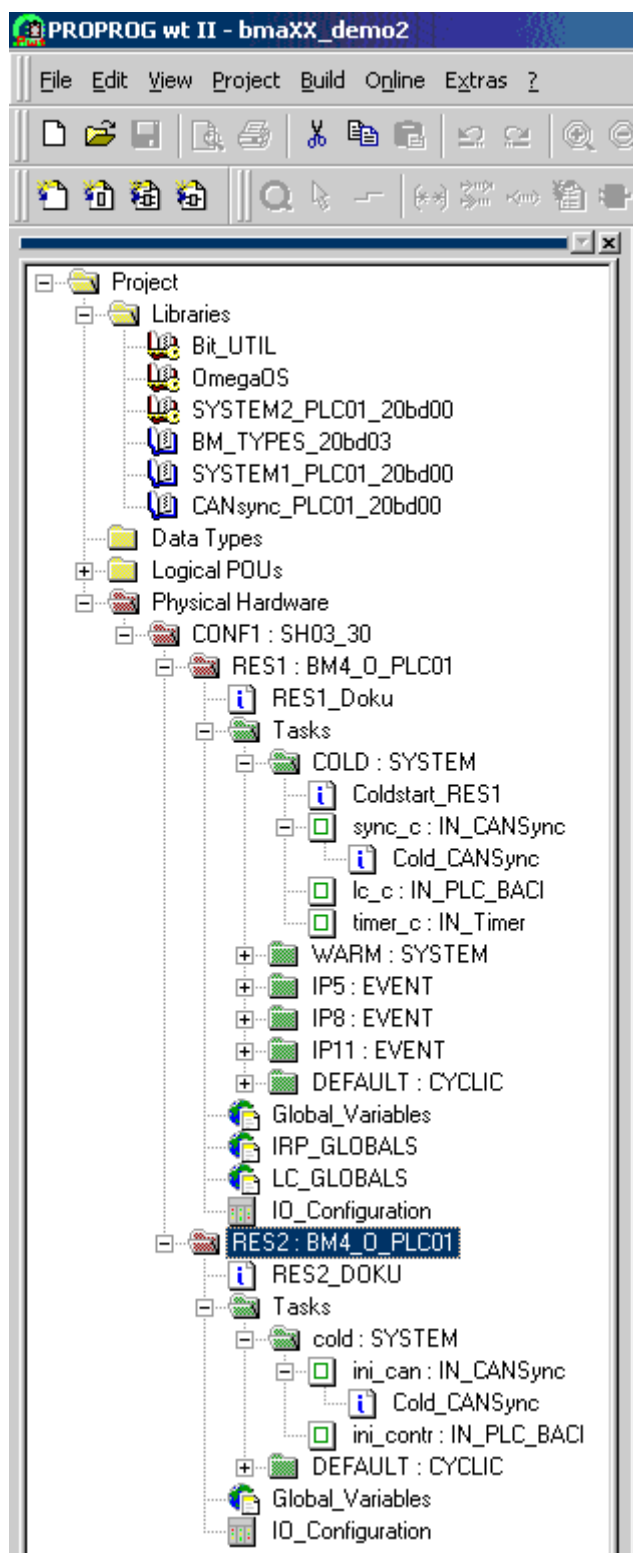


Figure 10: Example of a b maXX drive PLC configuration with several resources.

#### 4.4.1 Communication and Connection

You configure communication with data transfer under "Setting" in the resource's context menu.

You set communication via the selected RS232 port as follows:

- Baud: 38400
- Stop bits: 1
- Data bits: 8
- Parity: None
- Timeout: Default is 2000 ms; communications monitoring during online representation.

The connection is established via the COM1 serial port on the PC:

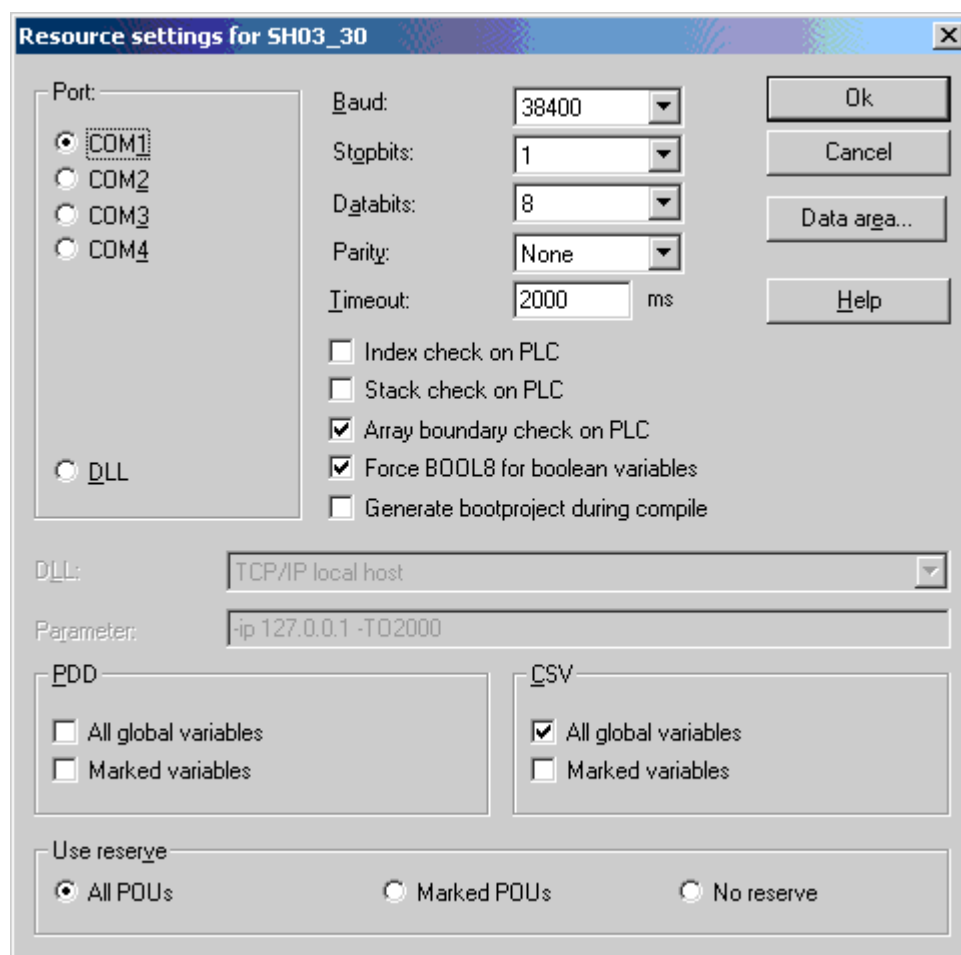


Figure 11: Resource setting within a b maXX drive PLC configuration.

- "Index check on PLC": The system checks the declared field size (index) of an ARRAY at runtime. Important: this increases the code execution time!
- "Stack check on PLC": The system checks for a stack overrun at runtime. The stack memory is reserved with the program task. There is an increase in the data on the stack if you program nested FB instances, for example. Important: this increases the code execution time!

- "Array boundary check on PLC": With absolute addressing, the system checks whether the field exceeds the parameterized data area limits. This check is carried out during compilation on the PC. This does not increase the code execution time.
- "Force BOOL8 for Boolean variables": Activate 8-bit access to Boolean variables. For the b maXX drive PLC, the setting must be activated.
- "Generate boot project during compile": With this function activated, the system generates for each resource a bootfile.pro at compiling of the project and not just at activation of the Send boot project function via the resource control.
- "PDD": Settings for the process data directory. (See also the PROPROG wt II programming manual and the Online Help System of ProProg wt III respectively).
- "CSV": Settings for providing variables for the OPC server. (See also the PROPROG wt II programming manual and the Online Help System of ProProg wt III respectively).
- "Use reserve": Use spare memory to be able to make changes in the function blocks and functions using the "Patch POU" function. (See also the PROPROG wt II programming manual and the Online Help System of ProProg wt III respectively).

The resource setting can be made separately for each b maXX drive PLC resource. Serial or Ethernet communications source

- COM1
- COM2
- COM3
- COM4
- DLL

is used

- for resource control.
- for online representation of variables and structures in the Watch window
- for sending the compiled project.
- for debugging.
- for connecting to the OPC server.



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

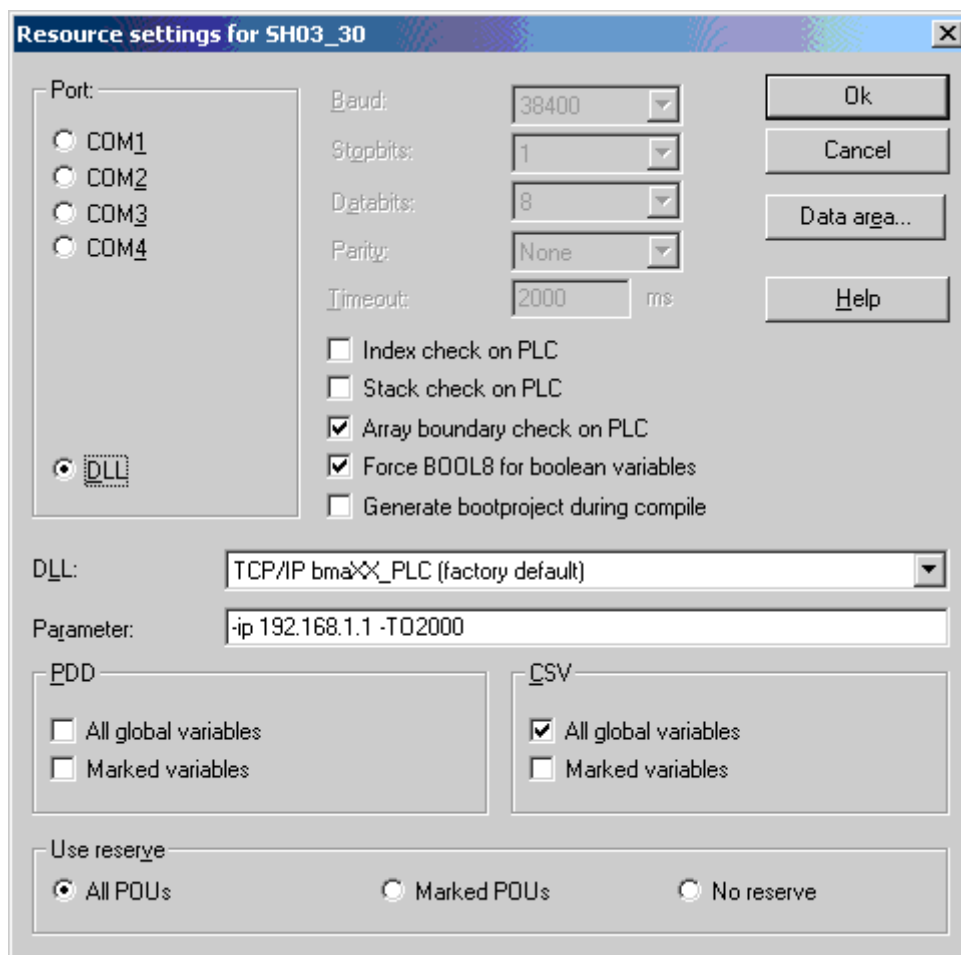


Figure 12: Setting the TCP/IP address

After changing the port to "DLL", users 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, users 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.
- **b maXX drive PLC:**  
To be able to access the b maXX drive PLC via Ethernet, an Ethernet-capable option module (e.g. a BM4-O-ETH-02) must be fitted in the b maXX 4400 basic unit in addition to the b maXX drive PLC. TCP/IP address "192.168.1.1", which is preset in the parameter field, corresponds to the IP address that is preinstalled when the Ethernet card leaves the factory. This means that users can make an initial connection to the module for basic initialization in which they must then enter the TCP/IP address of their TCP/IP network on the machine.

## 4.4 b maXX drive PLC resource

Refer to the respective operating instructions and application manuals of the Ethernet option modules you use for an exact description of the basic initialization procedure.

### 4.4.2 Control Dialog for Resources

Using the control dialog for resources, you set program transfer to the b maXX drive PLC and the operating status of the b maXX drive PLC.

If several resources are active in a project, you must choose the desired resource.

Select a resource in PROPROG wt II:

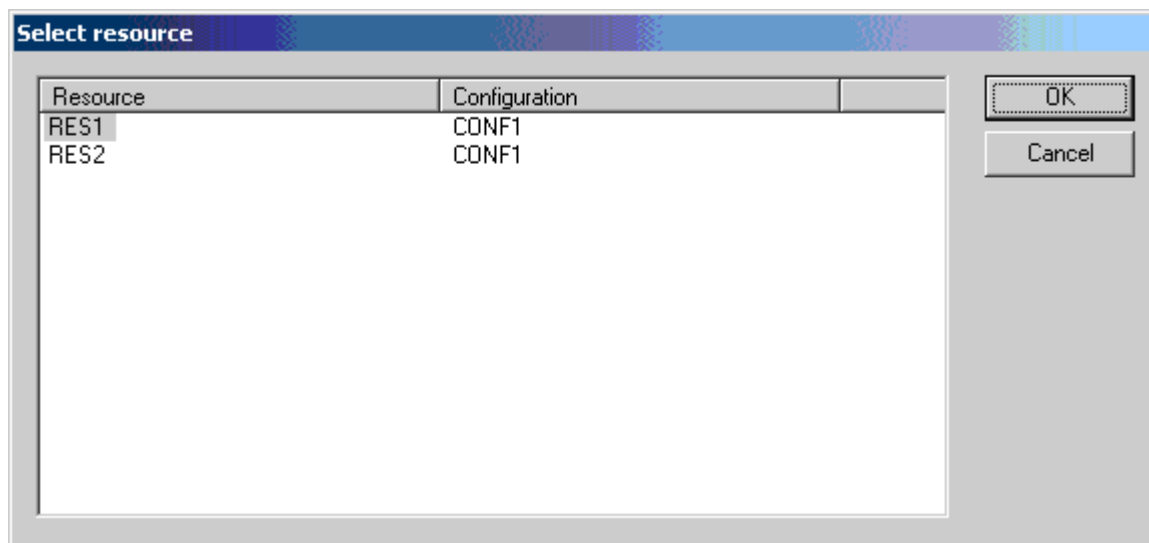


Figure 13: After you choose the resource, the system displays the control dialog for the resource of the assigned b maXX PLC.

The control dialog will be opened.

Select a resource in ProProg wt III:

Click on the desired resource and then press the button „Connect“.

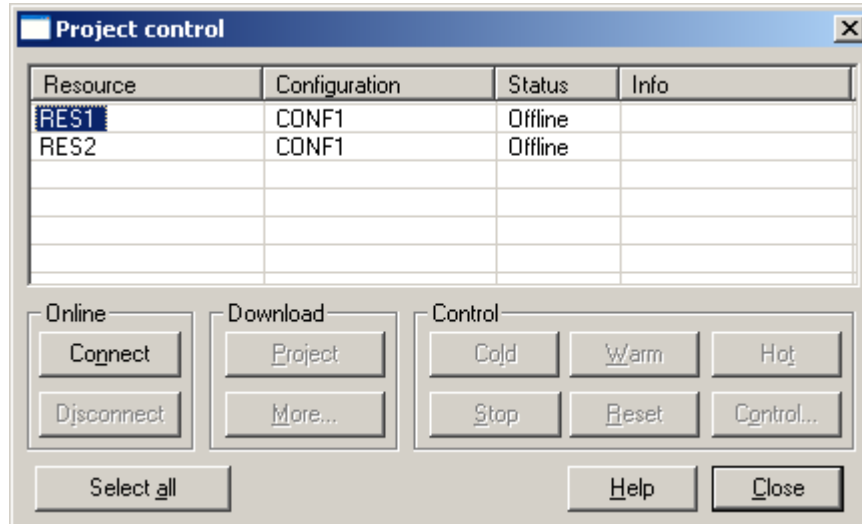


Figure 14: Select the resource in ProProg wt III - connect

Press the button „Control...“ now.

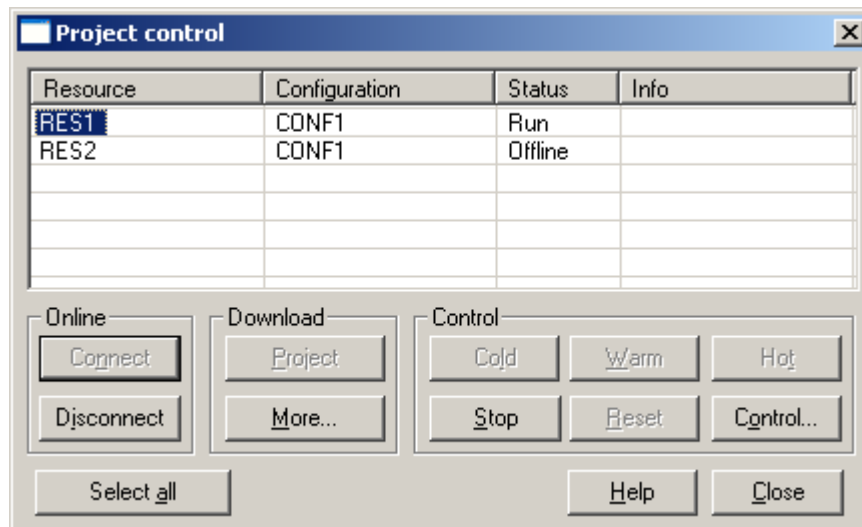


Figure 15: Select the resource in ProProg wt III - open control dialog

The control dialog will be opened.



Figure 16: Functions of the control dialog of the selected resource in the "RUN" status.

Clicking on **Download** transfers the compiled project to the target system.

Sending the project to the target system.

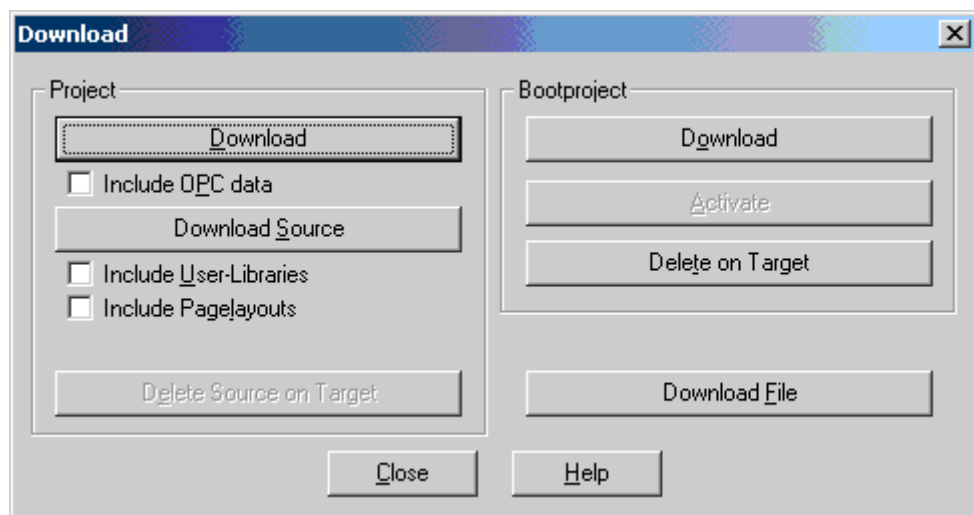


Figure 17: b maXX drive PLC resource transfer to flash memory or to RAM.

Clicking on **Download boot project**, deletes the resource's current boot project, sends the compiled project as a boot project and saves it in the b maXX drive PLC's flash memory. Clicking on **Activate**, loads the project from the b maXX drive PLC's flash memory to RAM.

Clicking on **Delete on Target**, deletes the boot project in flash memory.

Clicking on "**Project / Download**", transmits the compiled project of the resource directly into RAM and you can then start it by means of the control dialog ("Cold" pushbutton). The boot project is retained unchanged in the b maXX drive PLC's flash memory. After the next hardware reset or the next time you switch the controller off and on again, the boot project is loaded again!

**NOTE**

Menu items "Download source", "Download file" and attribute "Include OPC data" are not currently supported and you must not select them.

If you want to use menu item **Boot project / Download** to activate the project that you stored in flash memory, proceed as follows

- Directly: by switching the voltage off and on again

or

- Via PROPROG wt:  
In the program that is loaded in RAM use menu item "**Stop**" to halt it and delete it using menu item "**Reset**". The operating status changes to "**On**".



Figure 18: Resource control "ON" status

In the "Download" menu of the control dialog, click on "Boot project / Activate". This installs the flash RAM project in the b maXX drive PLC's RAM.

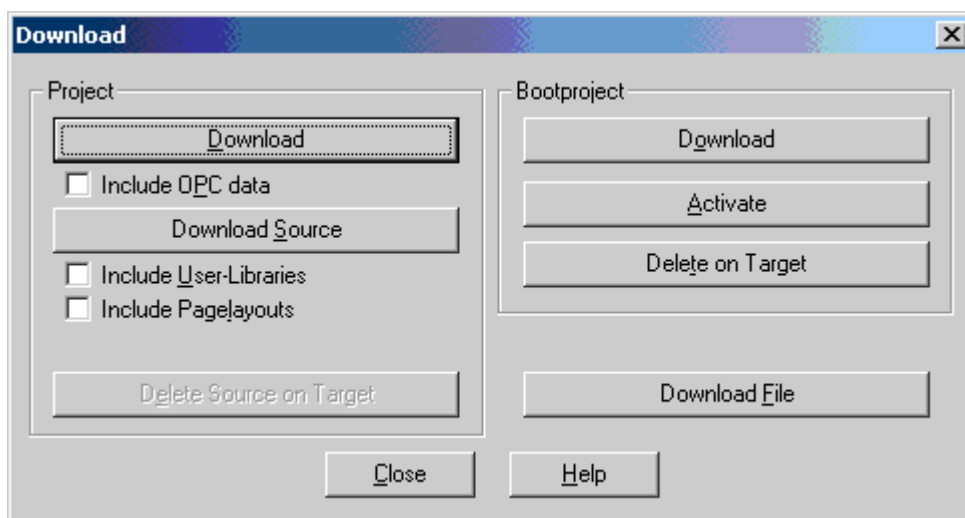


Figure 19: "Download" control dialog

Then, you can start the program by means of **"Cold"** in the control dialog:

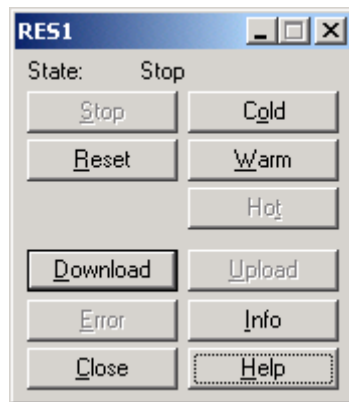


Figure 20: b maXX drive PLC resource control in status "STOP".

### Description of the pushbuttons in maXX drive PLC resource control

- **"Stop"**: This stops execution of the program; the PLC operating status changes from "Run" to "Stop" (The system displays the status at the top of the screen in resource control).
- The **"Reset"** pushbutton: Deletes the RAM project that is stored on the b maXX drive PLC (not the boot project that is stored in flash memory!) The status changes from "Stop" to "On".
- The **"Download"** pushbutton: Calls the page for program transfer (transfer of b maXX drive PLC resource to RAM or flash memory).
- The **"Error"** pushbutton: Here, you can read out error and warning messages that are pending on the b maXX drive PLC if the pushbutton is activated. Clicking on the active error pushbutton inquires the controller's error entries and displays them in the error or warning message window.
- The **"Close"** pushbutton: Closes maXX resource control again.
- Pushbuttons **"Cold"/"Warm"/"Hot"**: Using these PROPROG commands, you can manually start the b maXX drive PLC. The status changes from "Stop" to "Run". You can, however, only start the PLC using the pushbuttons if you set switch S1 on option module BM4-O-PLC-01 to "RUN". Otherwise, the controller stays in the "Stop" status.

In this connection, the three start buttons are differentiated as follows:

- **"Cold"** pushbutton: The PLC cycles once through PROPROG system task "cold boot (SPG1)" in which user program initialization takes place and then switches to cyclical program processing.  
Cold booting is characterized by all the variables being initialized with their default values. If a user did not state a default value in the project, the system sets a default value of "0" (or "FALSE" for Boolean variables).
- **"Warm"** pushbutton: The PLC cycles once through PROPROG system task "warm restart (SPG0)" in which user program initialization takes place and then switches to cyclical program processing.  
By contrast with cold boot, the retain flags retain their values, which means that they are not reset to default values.  
If there are no retain flags on the b maXX drive PLC and the same user code is integrated in the cold boot and the warm restart task, there is no difference between the cold boot and warm restart.

- **"Hot"** pushbutton: The system does not cycle through an initialization task; rather, it switches directly to cyclical program processing.



#### NOTE

You can only start the PLC using one of the cold boot/warm restart/hot restart pushbuttons or by switching the power off and on again if switch S1 on option module BM4-O-PLC-01 is set to "RUN". Otherwise, the controller stays in the "Stop" status and no user code is executed.

If you carry out a hardware reset on switch S1, this switch must also be set to "RUN" afterward.

In the case of a b maXX drive PLC with retain flags, the system cold boots once – whereby the retain flags are also set to the specified default values – after sending of a new boot project and subsequent automatic starting of the program (by turning the power off and on again or a hardware reset at switch S1 of the BM4-O-PLC-01 option module). The system stores "Kalt-start wurde durchlaufen" (Cold boot executed) to the retain area.

At each further automatic change to the "Run" status, the system evaluates this stored piece of information and always carries out a warm restart where the retain flags retain their contents.

If you activate the "Kalt" manual start pushbutton in the b maXX drive PLC resource control, this forces explicit setting of the default values for the retain flags too.

When you save boot projects, automatic restart response depends on whether there have been any changes in the retain flag area: If the system detects on the controller that by comparison with the previous project download retain flags have been created or deleted, or the name, data type or sequence of retain flags has been changed, then it also cold boots once on the change to the "Run" status. This is to guarantee consistency of the retain flags.

With projects, the aim is, if possible, to retain retain data on the controller even after a project download. If there is just a change in the default values, this means that the system does not automatically cold boot and users themselves may possibly have to explicitly carry out a cold boot.

In the case of a b maXX drive PLC without retain flags, the system always cold boots at automatic program starts.

You can query whether a PLC supports retain flags using the "Info" pushbutton (see below).

- **"Upload"** pushbutton: Upload functions in PROPROG wt are not currently supported.
- **"Info"** pushbutton: General information on the b maXX drive PLC itself and the project status on the b maXX drive PLC (for a description, see below).
- **"Help"** pushbutton: This calls general and specific PLC help for the b maXX drive PLC.

### Description of the "resource information" "Info" page.

You call this page by clicking on the "Info" pushbutton in b maXX drive PLC resource control:

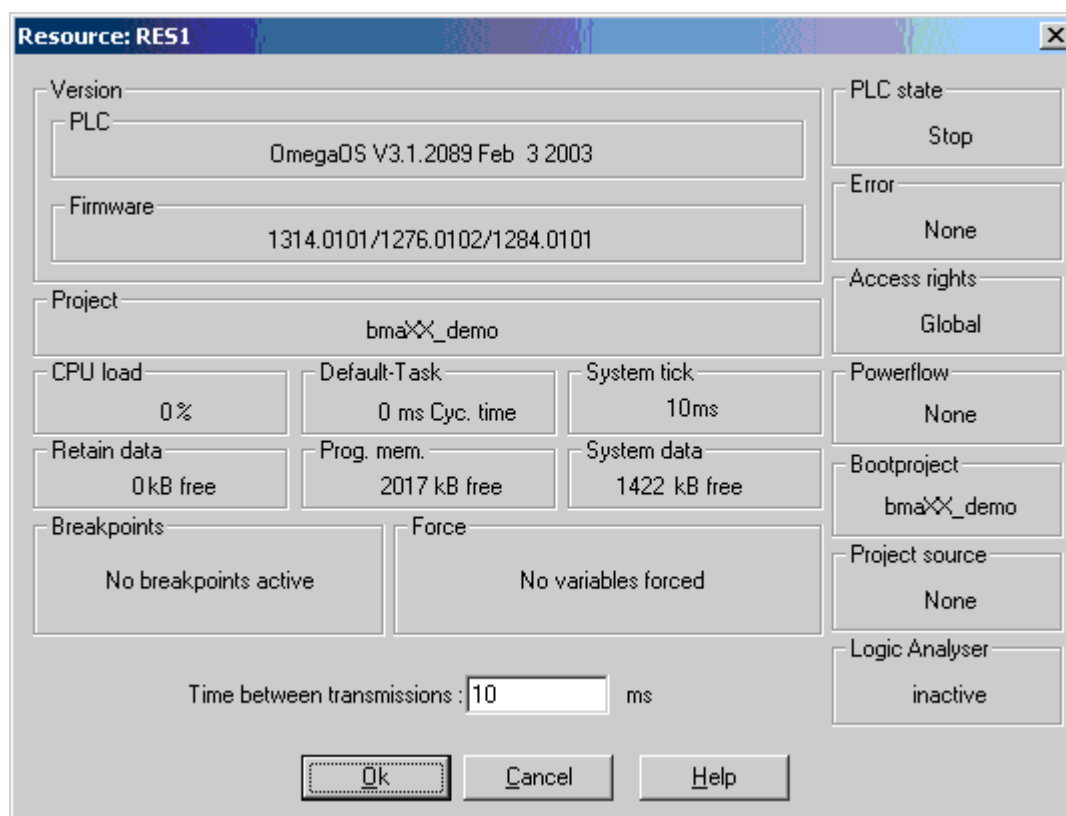


Figure 21: Resource information window

- Version: PLC "OmegaOS V3.1.2089 Feb 3 2003"
  - "OmegaOS": Product designation.
  - „V3.1.2089": OmegaOS Compiler Info.
  - „Feb 3 2003": Date on which the PLC version was created.
- Firmware: "1314.0101/1276.0102/1284.0101"
  - „1314.0101": Version OmegaOS software 6.1314.0101.
  - „1276.0102": Version FPGA software 6.1276.0102.
  - „1284.0101": Version BOOT software 6.1284.0101.
- Project: Name of the active project (the project name may be a maximum of eleven letters long).
- CPU load: The system automatically determines loading in real time and displays it here as a percentage.  
 Amongst other things, this includes the runtimes of the individual cyclical tasks as well as the bypass tasks that are not monitored by means of watchdogs. To guarantee important operating system functions, like online communication, for example, you should aim to achieve total loading of less than 80%. At 100% loading, a system error is generated that stops the b maXX drive PLC.  
 One hundred percent loading is achieved, for example, if model of the default task could not be completed after approximately ten seconds and the user changed the set-



ting of the watchdog for the default task to more than ten seconds. (In the case of watchdog settings of less than ten seconds in the default task, watchdog monitoring would trigger a system error first and check back). You can read the current duration of the default task in the window to the right of the CPU loading display.

You can avoid overloading by dividing the corresponding tasks into several shorter individual tasks with longer call intervals or by skipping individual program sections (e.g. in the bypass tasks) to reduce task runtimes.

- **Default task:** Here, you can view the current cycle time of the default task in ms.
- **System tick:** This time indicates the lowest resolution of the cycle or monitoring times that the PLC runtime system can manage.

Example: It doesn't matter whether you state the interval in a cyclic task as "400 ms" or "409 ms": With a 10-ms system tick, "400 ms" (i.e. rounded down to the duration of one system tick) is always activated.

- **Retain data:** As was explained above, there are b maXX drive PLC versions with and without retain flags.

In the case of a b maXX drive PLC without retain flags, the system displays the value 0 here. In this case, when you click on the "Project/Download" pushbuttons in the "PLC resource transfer" window, the system issues an appropriate PLC error if you are mistakenly using retain flags in the user project or if they were set in the user data range (see below).

In systems containing retain flags, the "Retain data" display shows the available total retain memory area on the PLC. Since retain system information is also stored there in addition to the retain flags (e.g. whether cold booting has already taken place with a project), the area that users can use as retain flags is smaller and can be taken from the data range settings ("Retain / Start user" and "End system", see below).

- **Prog. mem. (Program memory):** Display in kB of the amount of PLC program memory in the active project that is still free.

With 2046 kB of free program memory, you can, for example:

- Program a maximum of 400,000 STL lines (LD/ST statements to global variables)
- Typically program 120,000 STL lines (typical STL statements to structures and instance variables)

- **System data:** Dynamic memory (e.g. 1460 kB) for debug and logic analyzer functions.
- **Breakpoints:** Here, the system displays whether breakpoints are set on the PLC. If this is the case, the system shows the 'Reset breakpoints' checkbox. Select the checkbox if you want to reset all the set breakpoints at the same time. (You can manage breakpoints in the "Online/Debug" menu).
- **Force:** Indicates whether variables were forced. If this is the case, the system shows the 'Reset force list' checkbox. Select the checkbox if you want to reset all the set forced variables at the same time.
- **Free memory for system data runtime system** (for loading, oscilloscope function).
- **Time between transmissions:** Here, users can state a time after which, at the earliest, the programming system starts new communication to the PLC after the previous communication procedure was completed. By reducing the time, users can lower the general PLC loading due to communications tasks. The value is stated in ms.
- **PLC state:** This shows the current b maXX drive PLC operating status (e.g. "Run", "Stop", etc.)

## 4.4 b maXX drive PLC resource

- Error: Indicates whether a PLC error message is pending in the error catalog and if you can read it out using the "Error" pushbutton in b maXX drive PLC resource control.
- Access rights: Indicates the access rights for sending and debuggen on the PLC.
- Flow control: Indicates whether the user activated Address status with flow control ("Online/Address status" menu item).
- Boot project: Name of the (inactive) boot project on the b maXX drive PLC (the project name may be a maximum of eleven letters long).  
Using the page for program transfer (b maXX drive PLC resource transfer) you can choose menu item "Activate" to transfer the boot project to the active RAM project. (This always happens automatically in the case of a hardware RESET or switching the power off and on again).
- Project source: This indicates whether a source project (zwt) is present in the PLC (For memory reasons, this is not currently supported).
- Logic Analyser: This indicates whether logic analysis recording is currently active.

### 4.4.3 Data Area

The data area contains the b maXX drive PLC-specific setting of physical address ranges in the form of flags that users and the compiler PROPROG wt can access.

The settings are stored with the project.

You get to the "data area" form in the project tree in the "Settings/Data area..." context menu in the "Physical Hardware/RES1" resource:

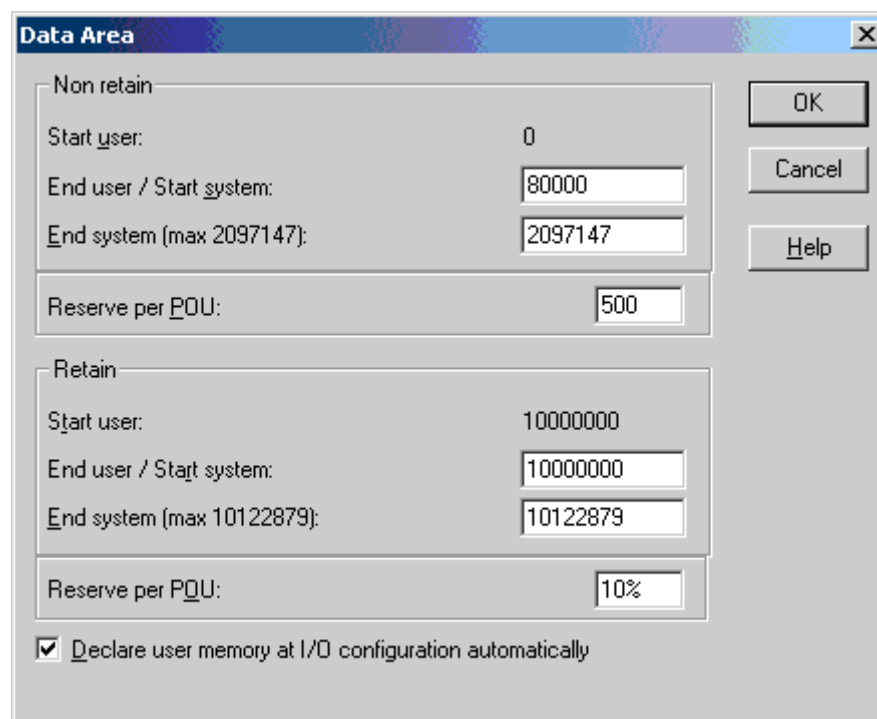


Figure 22: b maXX drive PLC-specific data area resource without retain flags user area.

In this connection, the data area is subdivided according to two criteria:

Firstly, the data area is divided into "non retain flags" and "retain flags".

Secondly, a differentiation is made in both areas between whether the user or the compiler in PROPROG wt are allowed to access these flags.

The two "Reserve per POE" settings that you can see in the picture are for the "Send on-line changes" compiler functionality and they reserve program memory.

Explanation of "non retain" and "retain" flags:

By contrast with non retain flags, retain ones keep their values even when the power is switched off, which means that the PLC can continue running with these values the next time you reboot it. By preference, plant-specific data is stored there that is only determined during operation and cannot be recalculated from other values.

An example of this is a winder diameter that changes continuously in terms of speed and over the machine's operating time.



#### NOTE

The standard version of the BM4-O-PLC-01 option module contains no retain flags. In this case, you should not change the existing standard settings for retain flags in the data area.

In the case of a b maXX drive PLC without retain flags, the system displays in the online „Info“ window a value of 0 at the "retain data" entry. In this case, the system issues an appropriate PLC error in the "PLC resource transfer" window when you download the project by clicking on the "Project/Send" pushbutton if you mistakenly set retain flags in the user data area.

In systems containing retain flags, the "Retain Data" display shows the available total retain memory area on the PLC.

Explanation of **user flag** ("Start user/End user"):

If users want to directly access flags in their project, it is possible to reserve in advance areas for the non retain and retain flags.

You should however not set the user areas too large, since with relatively large projects this may lead in some cases to too little system memory being available for the PRO-PROG compiler.

Explanation of **system flag** ("Start system/End system"):

In PROPROG wt, users can define symbolic variables and use them in their program code without needing to create special flags for this. The system carries this out automatically during compilation in PROPROG wt by mapping these variables to the flags in the system area (i.e. the flags assigned by the system).

Since the system does this every time project code is generated, this means that you can customize separation limits at a later stage in the project too.

The preset standard user area for assigning absolute flag addresses within a b maXX resource in the non retain area is:

- %MB 0 - %MB 79999

The remaining range from %MB 80000 to %MB 2097147, for example, is reserved for the system for automatically converting the non retain symbolic user variables to this flag range.

The standard user area for assigning absolute flag addresses within a b maXX resource in the retain area is deactivated.

In the case of PLC versions with retain flags, users can shift upward the respective project request after the "End user/Start system" partial limit (e.g. from 10000000 to 10020000).

This would correspond to a retain flag area of %MB 10000000 - %MB 10019999 that users can trigger directly in their projects as absolute retain flags.

The remaining range from %MB 10020000 to %MB 10057323, for example, is reserved for the system for automatically converting the non retain symbolic user variables to this flag range.

If users send absolute flag addresses that are in the system area, the system issues an appropriate error message when compiling the project code.

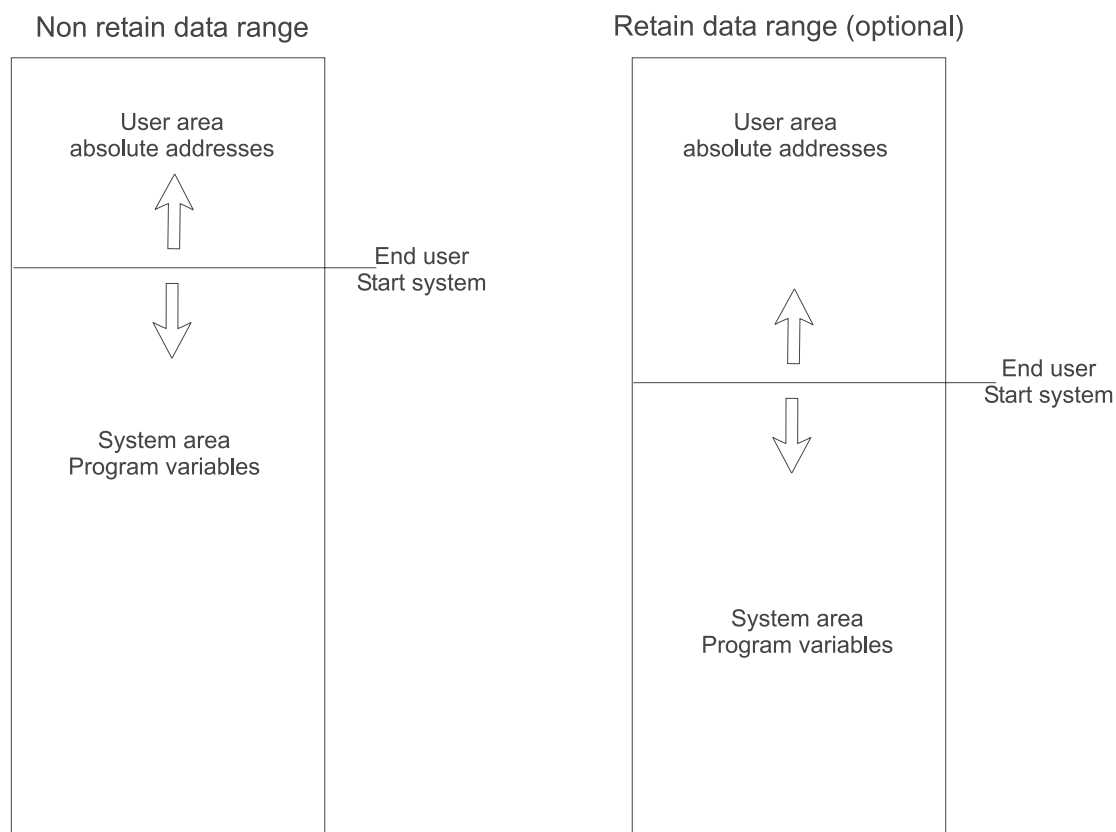


Figure 23: Dividing and setting the b maXX drive PLC data area

### Assigning absolute flag addresses in the program

An absolute b maXX drive PLC address or a variable field with data type

- 16-bit (WORD) can only be assigned for an address that can only be divided without remainder by two and zero.
- 32-bit (DWORD) can only be assigned for an address that can only be divided without remainder by four and zero.

Example:

PROPROG wt II:

You want to declare a variable of data type DWORD in the non retain data range.

```
d_abs_adr AT %MD12 : DWORD;      (* symbolic variable to
                                   absolute address *)
```

Or also:

```
di_abs_adr AT %MD16 : DINT;      (* symbolic variable to
                                   absolute address *)
```

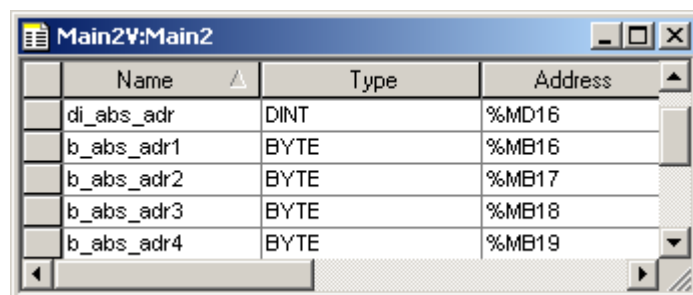
```
b_abs_adr1 AT %MB16 : BYTE;
```

```
b_abs_adr2 AT %MB17 : BYTE;
```

```
b_abs_adr3 AT %MB18 : BYTE;
```

```
b_abs_adr4 AT %MB19 : BYTE;
```

ProProg wt III:



Name	Type	Address
di_abs_adr	DINT	%MD16
b_abs_adr1	BYTE	%MB16
b_abs_adr2	BYTE	%MB17
b_abs_adr3	BYTE	%MB18
b_abs_adr4	BYTE	%MB19

Figure 24: Declaration of variables

One advantage of using absolute flags is that it can give quick and easy access to individual variable sections without needing additional masking and conversion functions:

If, for example, the system writes the value "DINT#16#98765432" to created absolute variable "di\_abs\_adr", the variable value can be further-processed directly as individual bytes (INTEL format: b\_abs\_adr1 = BYTE#16#32, ..., b\_abs\_adr4 = BYTE#16#98).

Caution! This type of programming is no longer instantiated and the sequence could be interrupted by an interrupt and called again from there with the result that after the interrupt ends the system carries out further processing with the wrong values!

#### 4.4.4 b maXX drive PLC event tasks

b maXX drive PLC event tasks are for event-driven program calling (interrupt). Their type and code runtime determine the real-time response.

Implementation of the real-time response depends on the type of reference value setting and the b maXX controller's operating mode. Specified value setting can, for example, be implemented as follows:

- In a standalone drive via a BACI process data event task.
- In a networked drive via a "SYNC Signal 1 option module" event task.

The property of the event task is assigned via its event number:

Event	Designation	Level
0	CPU timer 1	14
1	Reserved	
2	CPU timer 2	13
3	Reserved	
4	BACI process data	13
5	Timer A	14
6	Timer A	13
7	Reserved	
8	BACI process data	14
9, 10	Reserved	
11	Sync-Signal 1 option module	14
12	Sync-Signal 2 option module	14

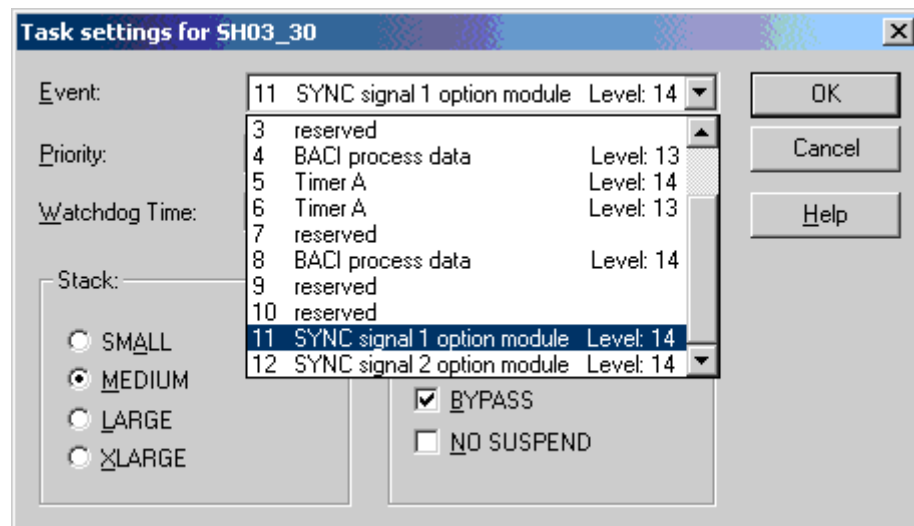


Figure 25: Events of the der b maXX drive PLC event tasks.

**NOTE**

All the b maXX drive PLC tasks with an "EVENT" task type are dependent on the resource and need the **"BYPASS"** attribute. The initialization FBs within the program initialize and call the declared event task and their priorities. This means that with bypass event tasks, the priority, watchdog time, "SAVE FPU" and "NO SUSPEND" are meaningless.

A higher interrupt level means a higher priority.

In the case of several event tasks that can mutually interrupt one another or with multiple-nested function blocks, you should adapt the stack to Large or XLarge.

## 4.5 b maXX drive PLC user libraries

The PROPROG wt user libraries are divided into firmware and user libraries that can be hardware-dependent or hardware-independent. You can only use hardware-dependent libraries in resources of the specified target system. The hardware dependence of b maXX drive PLC libraries is shown by \*\_**PLC01**\_ in the library designation.

**PROPROG wt II:** The version is indicated as follows: 20bd00.

- 20 is the incompatible version.
- 00 is the compatible version.

In the case of compatibility of the input and output variables of the function blocks, the version is incremented by one, e.g. 20bd01, 20bd02, etc. if you add the library to an FB, e.g. an input or output, the version before the "bd" is incremented by one and set to zero after the "bd", e.g. 20bd03 to 21bd00.

**ProProg wt III:** The version is indicated as follows: 30bd00.

- 30 is the incompatible version.
- 00 is the compatible version.

In the case of compatibility of the input and output variables of the function blocks, the version is incremented by one, e.g. 30bd01, 30bd02, etc. if you add the library to an FB, e.g. an input or output, the version before the "bd" is incremented by one and set to zero after the "bd", e.g. 30bd03 to 31bd00.

The user libraries are divided into:

- Firmware: b maXX drive PLC Board functionality  
e.g. starting a PROPROG wt bypass event task.
- Data types: b maXX drive PLC-specifically assembled data types and fields, e.g. register structure of option boards.
- Standard FBs: Elementary FBs for local drive control engineering.
- Technology modules: Drive functionality that can be completely applied.

### Overview of completely hardware independent PLC libraries PROProg wt II:

CD-ROM	Contains library	
Standard libraries	BM_TYPES_20bd06	Baumüller data types
	UNIVERSAL_20bd01	Basic function blocks
	MC_SYS_20bd01	Firmware blocks (used by other libraries)

### ProProg wt III:

CD-ROM	Contains library	
Standard libraries	BM_TYPES_30bd01	Baumüller data types
	UNIVERSAL_30bd00	Basic function blocks
	MC_SYS_30bd00	Firmware blocks (used by other libraries)

### Overview of b maXX group dependent PLC libraries which can used by b maXX drive PLC and b maXX controller PLC (from the stated below version upwards) PROProg wt II:

CD-ROM	Contains library	
TB cam disk	CAM_PLC01_21bd00	FBs for "cam disk" technology component
TB winder	WINDER_PLC01_20bd01	FBs for "winder" technology component
CANopen	CANopen_PLC01_20bd03	FBs for b maXX option module BM4-O-ETH-02 or BM4-O-CAN-04 (CANopen-Master) BM4-O-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_21bd01	FBs für option module BM4-O-ETH-01 or BM4-O-ETH-02 (Ethernet)

### ProProg wt III:

CD-ROM	Contains library	
TB cam disk	CAM_PLC01_30bd00	FBs for "cam disk" technology component
TB winder	WINDER_PLC01_30bd00	FBs for "winder" technology component
CANopen	CANopen_PLC01_30bd00	FBs for option module BM4-O-ETH-02 or BM4-O-CAN-04 (CANopen-Master) BM4-O-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_30bd01	FBs für option module BM4-O-ETH-01 or BM4-O-ETH-02 (Ethernet)



Overview of hardware dependent PLC libraries which can be used only by b maXX drive PLC  
**PROPROG wt II:**

CD-ROM	Contains library	
Standard libraries	SYSTEM1_PLC01_20bd00	Function blocks for BM4-O-PLC-01
	SYSTEM2_PLC01_20bd00	Firmware blocks for BM4-O-PLC-01

**ProProg wt III:**

CD-ROM	Contains library	
Standard libraries	SYSTEM1_PLC01_30bd00	Function blocks for BM4-O-PLC-01
	SYSTEM2_PLC01_30bd00	Firmware blocks for BM4-O-PLC-01

You must state the directory path for libraries under PROPROG wt, Options, Directories. The b maXX drive PLC libraries are inserted in the PROPROG wt project tree under libraries.

You can call HTML help for each FB that gives you a description of the inputs and outputs (see the PROPROG wt II Programming Manual and the Online Help System of ProProg wt III respectively).

#### 4.5.1 b maXX drive PLC firmware

The b maXX drive PLC firmware consists of function blocks (FBs) that use parameter transfers to communicate with functions on the b maXX drive PLC-CPU. You can only use these FBs in a resource-dependent way, i.e. in dependence on the target system.

PROPROG wt II:

You must use library

**SYSTEM2\_PLC01\_20bd00** (or above)

to insert the BM4-O-PLC-01 firmware into a project. The library contains the following range of functions:

- Start bypass event task and freely programmable LEDs on the b maXX drive PLC.
- P, PI controller, 48-bit division via electronic transmission, integration, differentiation.
- Function blocks for interface module to USS® and 3964R® protocols.



### NOTE

The b maXX drive PLC firmware is used by several b maXX drive PLC user libraries. This means that with a b maXX drive PLC user library, it may be necessary to insert the requested firmware library SYSTEM2\_PLC01\_20bd00 or above (See description of the respective user library). The most current version in each case is on the PROPROG program CD and it is automatically installed when you install PROPROG wt II.

At selection of the „Pattern for BM4\_O-PLC01“ under „File\New Project“ the library SYSTEM2\_PLC01\_20bd00 (or above) automatically is set. The standard user library SYSTEM1\_PLC01\_20bd00 (or above) in the normal case also will be necessary in projects and will be delivered on a separate CD (see [b maXX drive PLC user libraries](#) < from page 47 onward).

The same is true for ProProg wt III and its libraries.

ProProg wt III:

You must use library

**SYSTEM2\_PLC01\_30bd00** (or above)

to insert the BM4-O-PLC-01 firmware into a project. The library contains the following range of functions:

- Start bypass event task and freely programmable LEDs on the b maXX drive PLC.
- P, PI controller, 48-bit division via electronic transmission, integration, differentiation.
- Function blocks for interface module to USS<sup>®</sup> and 3964R<sup>®</sup> protocols.

### 4.5.2 b maXX drive PLC Board functions

Firmware library SYSTEM2\_PLC01\_20bd00 (or above) contains function blocks (FBs) for checking event signals for interrupts and board LEDs. Function block INTR\_SET is used to initialize and start b maXX drive PLC event task (bypass). Function block LED is used to allow the use of freely programmable LEDs.

Two FBs for code runtime measurement are provided to optimize code runtimes within b maXX PLC resources.

PROPROG wt II            BM4-O-PLC-01:

The FBs are inserted via user library **SYSTEM1\_PLC01\_20bd00** and above.

ProProg wt III            BM4-O-PLC-01:

The FBs are inserted via user library **SYSTEM1\_PLC01\_30bd00** and above.

You must limit the code block to be measured within a task by placing FBs TIME\_MEASURE\_START and TIME\_MEASURE\_END appropriately. The system outputs the result of the measured code block's runtime at FB TIME\_MEASURE\_END as a time difference in µs (see also the online description of FBs TIME\_MEASURE\_START and TIME\_MEASURE\_END). Code running times up to 10 ms (BM4-O-PLC-01) are able to be measured with these function blocks.

#### 4.5.2.1 Function block INTR\_SET

Function block INTR\_SET starts a bypass event task in a start-up task.

You must set the PROPROG wt event task with the program to the event and to the Bypass attribute.

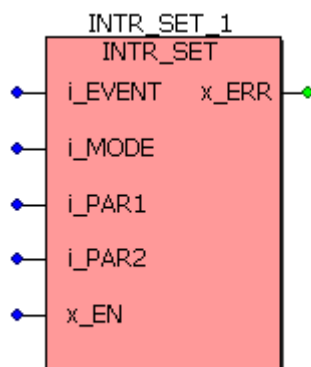


Figure 26: Initializing and enabling a bypass event task via function block INTR\_SET.

Parameter	Input	Value Range
i_EVENT	Interrupt hardware program number	8-bit signed
i_MODE	Reserve	16-bit signed
i_PAR1	CPU timer (1,2) value multiplier to base 50µs	16-bit signed
i_PAR2	Reserve	16-bit signed
x_EN	Block/enable the interrupt	1-bit

Parameter	Output	Value Range
x_ERR	Error bit	1-bit

#### Description

Using FB INTR\_SET, users can configure and activate various system-internal interrupt sources. You can then use these interrupts in the program to activate event tasks.

An event number must be connected at input i\_EVENT. This number specifies the interrupt source of the event task. If the interrupt source in question is a CPU timer interrupt, you must additionally state a factor to the time base 50 µs at input i\_PAR1.

You can use x\_EN = FALSE to disable a previously activated interrupt.

List of event numbers for event tasks:

Event	Designation	Level
0	CPU timer 1	14
2	CPU timer 2	13.
4	BACI process data	13.
5	Timer A	14
6	Timer A	13.
8	BACI process data	14
11	Sync-Signal 1 option module	14
12	Sync-Signal 2 option module	14

The event number at input i\_Event must be identical with the setting of the event task within the resource. The Bypass attribute must be activated.



### NOTE

A higher-priority interrupt interrupts a lower-priority one. For events 0 and 2, CPU timer 1 (or 2) interrupt, you must additionally state at input i\_PAR1 the factor for time base 50 µs.

The higher-priority timer "CPU timer 1" (i\_EVENT = 0) and the lower-priority timer "CPU timer 2" (i\_EVENT = 2) can be used at the same time.

"Board timer A" high priority (i\_EVENT = 5) and "Board timer A" low priority (i\_EVENT = 6) cannot be used at the same time.

Events "SYNC-Signal 1 option module" (i\_EVENT = 11) and "SYNC-Signal 2 option module" (i\_EVENT = 12) cannot be used at the same time. "BACI process data" low priority (i\_EVENT = 4) and "BACI process data" high priority (i\_EVENT = 8) cannot be used at the same time.

In the case of events i\_EVENT = 4, 8, 11 and 12, you do not generally need to use INTR\_SET, since the INTR\_SET block is already permanently integrated in FB "BACI\_INIT" (see standard library "SYSTEM1\_PLC01\_20bd00" (PROPROP wt II) or "SYSTEM1\_PLC01\_30bd00" (ProProg wt III) or above).

### Example 1:

User POE (e.g. called "timer") is to be linked with the higher-priority CPU timer 1 (i\_EVENT = 0) and to be called cyclically at 5-ms intervals.

### Implementation:

In the cold boot and warm start task, the system calls an INI-POE that contains the configured INTR\_SET.

The following formula applies to the timer interval:

$$\text{Timer interval} = i\_PAR1 \cdot 50 \mu\text{s}$$

For the desired 5-ms interval, this yields:  $i\_PAR1 := (5000 \mu s / 50 \mu s) = INT\#100$ .

Representation of the configured and released INTR\_SET:

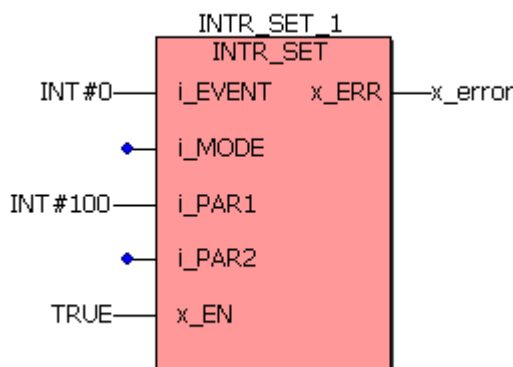


Figure 27: Function block INTR\_SET: Starting a CPU timer 1 interrupt with a call interval of 5 ms.

This sets up and activates the interrupt during the PLC start-up procedure.

You must link user POE "timer" in group task within the BM4\_O\_PLC01 resource as a task type "EVENT" to event "CPU\_Timer 1" with setting option "BYPASS":

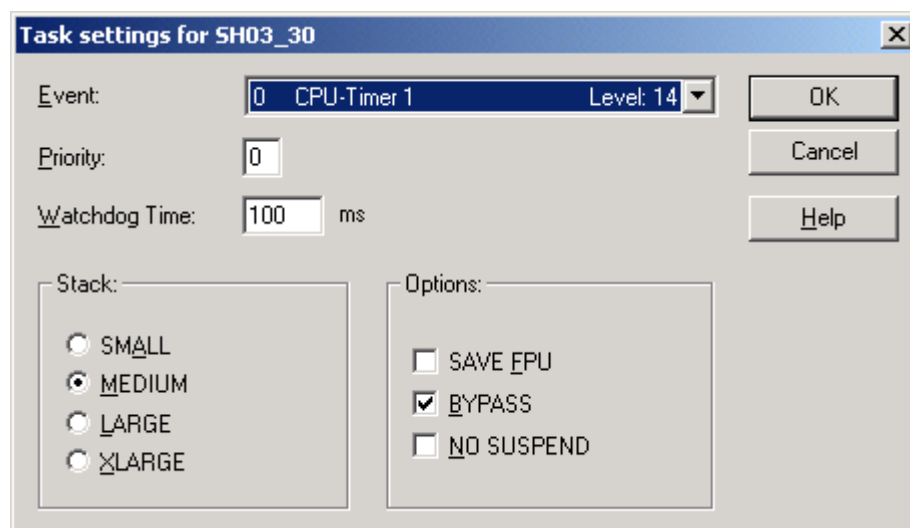


Figure 28: Linking to event "CPU timer 1" as a bypass task.

### Example 2:

Start the "Timer A" interrupt at 5 ms and generate the trigger signal at BACI signal output "TRIGGER1":

By contrast with CPU timer 1 (or 2), you can also use board timer A as a trigger signal for option modules that need the trigger signals. This is because you can only use "Timer A" to both trigger a cyclical interrupt and carry out triggering.

First of all, you set up the event task within the BM4\_O\_PLC01 resource via event "Timer A" with event number 5 (or 6).

Within the cold boot/warm start task, the system initially calls FB "TIMER\_A\_INIT" twice (see standard library "SYSTEM1\_PLC01\_20bd00" (PROPROP wt II) or "SYSTEM1\_PLC01\_30bd00" (ProProg wt III) or above) and then calls FB "INTR\_SET".

- The first call of "TIMER\_A\_INIT" sets up the timer for 5 ms.
- The second call connects the generated 5-ms signal to BACI output „TRIGGER1" which can then be processed by the option modules.
- The "INTR\_SET" call then sets up and activates the timer A interrupt so that users can run their POE synchronized with the generated 5-ms trigger signal.

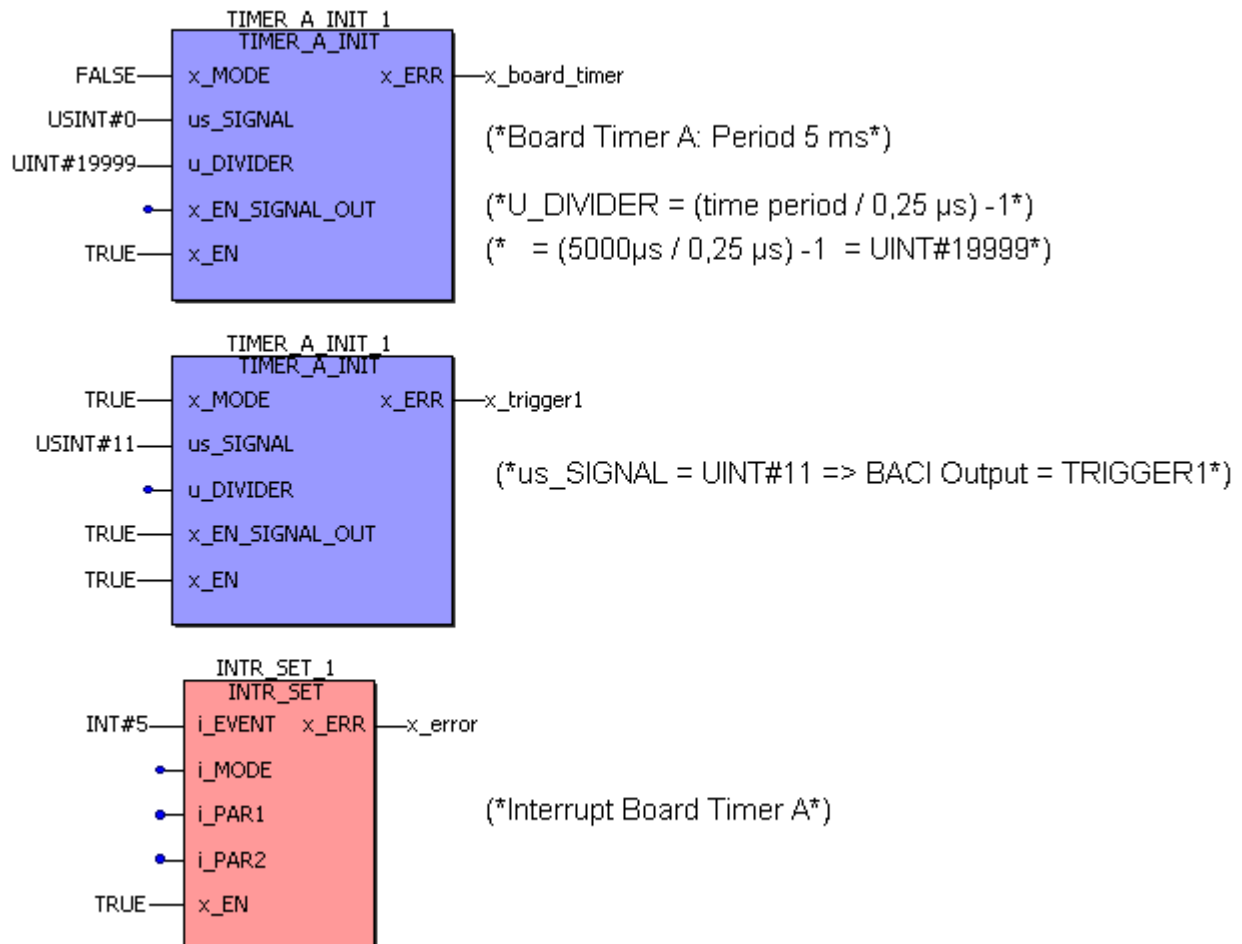


Figure 29: Start of the "Timer A" bypass event task with a period of 5 ms and simultaneous use of "Timer A" as a trigger signal for BACI option modules.

#### 4.5.2.2 Function block LED

You can use the LED function block from firmware library **SYSTEM2\_PLC01\_20bd00** (PROPROP wt II) or **SYSTEM2\_PLC01\_30bd00** (ProProg wt III) (or above) to program the b maXX drive PLC LEDs on the board.

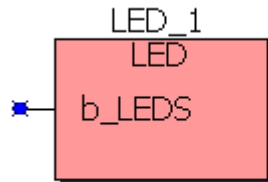


Figure 30: LED function block

Parameter	Input	Value Range
b_LEDS	LED set screen form	8-bit

#### Description

The LEDs on the left of the b maXX drive PLC board light up green; the ones on the right light up red. Bits 0-3 of the 8-bit pattern at the LED input are output to the four LEDs as follows:

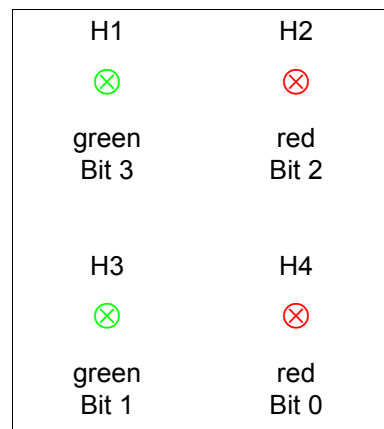


Figure 31: LEDs of the BM4-O-PLC-01 option module

#### 4.5.3 b maXX drive PLC data types

In a PROPROP wt project, data is exchanged between the b maXX drive PLC and the option modules for b maXX drive PLC by means of variables of the appropriate data type. User library **BM\_TYPES\_20bd03** (PROPROP wt II) or **BM\_TYPES\_30bd01** (ProProg wt III) or above makes available a large number of ready-made data types (structures and arrays). In general, these structured data types are used in the ready-made standard and technology libraries, which makes it very easy to communicate with the option modules.

For this, the most important variables for the different functions and slots of the individual option modules are defined in template BM4\_O\_PLC01 ("File/New project.../Template for BM4\_O\_PLC01").

This means that users only need to choose the slot-dependent variable from the template that corresponds to the fitted option module and to connect it to the appropriate inputs and outputs of the function blocks that are being used (that are available in the different libraries for the option module).

In the libraries, reference will be made at the relevant locations to the data types of BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above that were used.

Users can and should of course use the predefined data types in BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above.

You cannot directly call the worksheet of BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above in the project tree; however, you can view the data types contained there by switching the tab in the footer of the project tree editor from "Project" to "Libraries" and then doubleclicking on the BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above worksheet:

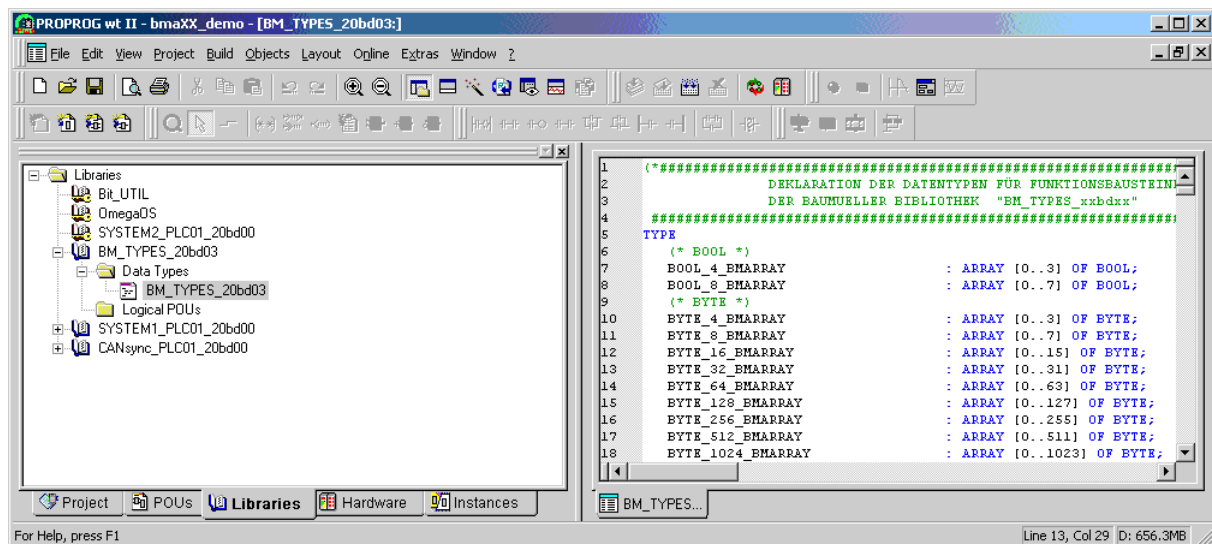


Figure 32: View of Baumüller data types "BM\_TYPES\_20bd03" that are integrated in your project.

You can access these data types when assigning variables via the variable dialog under "Properties/Automatic variable declaration":

PROPROG wt II:

Click on the button „Properties ...“ in PROPROG wt II in variables dialog. The window „Automatic Variables Declaration“ is opened.



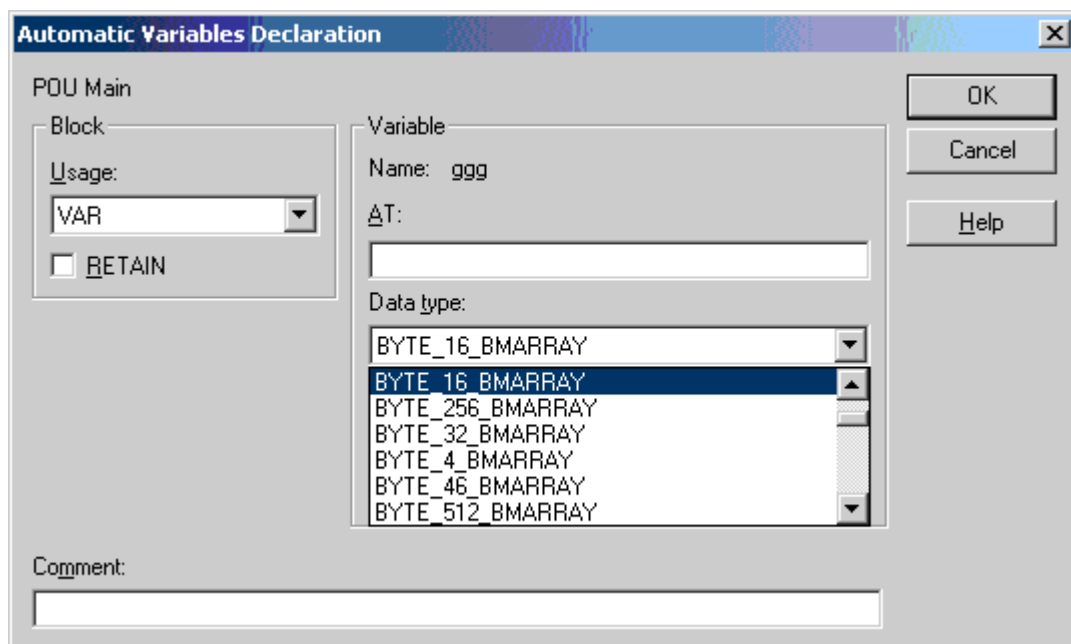


Figure 33: Automatic variable declaration.

ProProg wt III:

The automatic variable declaration is integrated in the variable dialog in ProProg wt III.

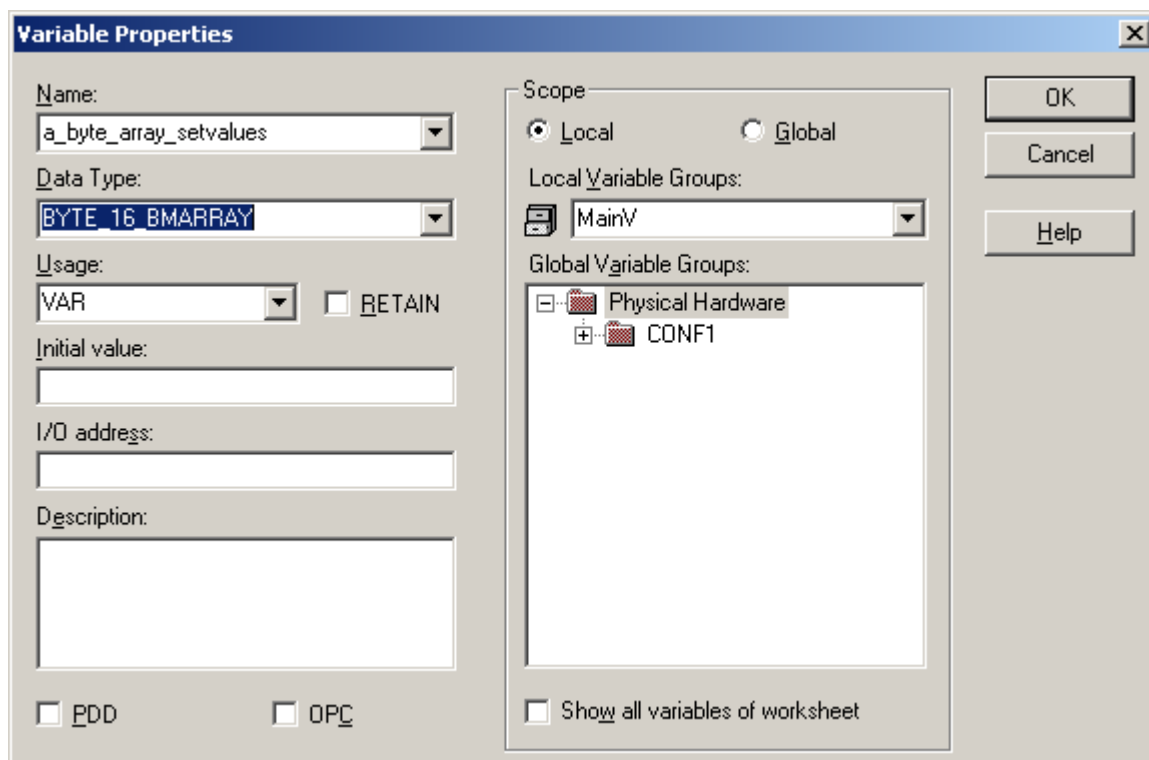


Figure 34: Declaration of variables

When you create a PROPROG wt project using template BM4\_O\_PLCO1, the variables for data exchange are already declared in the global variable worksheet. You will find one (and sometimes two) variables per slot of the option module.

### Example:

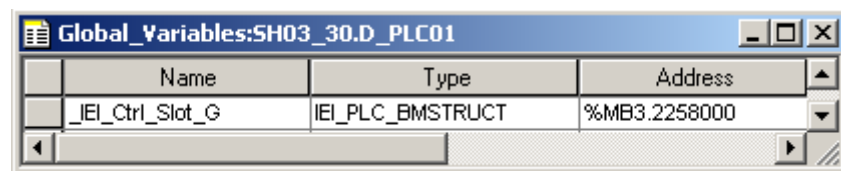
For initialization, option module BM4-O-IEI-01 needs settings in various registers.

With the option module in slot G, this yields the following variable declaration (that is already created in the template):

### PROPROG wt II:

```
_IEI_Ctrl_Slot_G  AT      %MB3.2258000  :  IEI_PLC_BMSTRUCT;
                      (* Option module IEI (BM4-O-IEI-01) *)
```

### ProProg wt III:



Name	Type	Address
_IEI_Ctrl_Slot_G	IEI_PLC_BMSTRUCT	%MB3.2258000

Figure 35: Declaration of variables

The complete register structure of IEI\_PLC\_BMSTRUCT is stored with its elements and the data types used in BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above (extract):

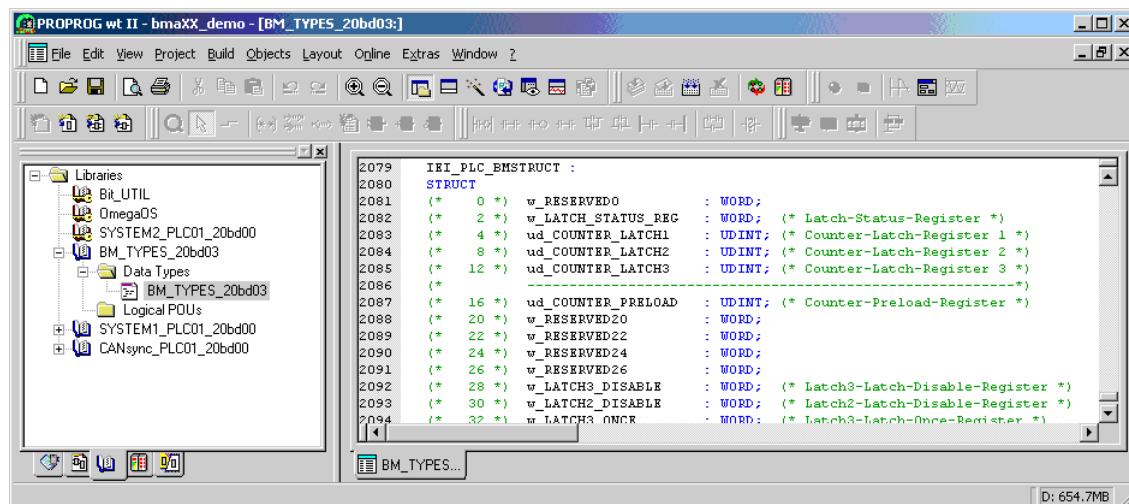


Figure 36: Extract from the register structure of IEI\_PLC\_BMSTRUCT

The elements of variable \_IEI\_Ctrl\_Slot\_G now form the tab of the option module in slot G. For more information on using the option module, see the BM4-O-IEI-01 Operating Instructions.

**NOTE**

In the variable dialog, the system makes available the data types for assignment in a selection dialog. In this connection, the individual data types of library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above are generally indicated by the abbreviation "\_BM" (\_BMARRAY, \_BMSTRUCT, etc.) The library's worksheet is write-protected and you can view it via the library tree (see above).

#### 4.5.4 The Standard Function Block Libraries

The standard libraries contain function blocks (FBs) with the basic functionality for local programming and configuration of the real-time response.

These function blocks are located in:

**PROPROG wt II:**

- **SYSTEM1\_PLC01\_20bd00** (or above)
  - BACI communication (process data, requirements data)
  - Function block for BACI (input, timer, output, trigger)
  - USS® protocol interface module
  - Code runtime measurement
- **SYSTEM2\_PLC01\_20bd00** (or above)
  - b maXX drive PLC firmware
  - Firmware block LED
  - Firmware block INTR\_SET
  - 3964R® protocol interface module
  - TER\_x terminal blocks
- **UNIVERSAL\_20bd01** or above (regardless of the hardware)
  - Drive status and control via FB DRIVE1
  - Extrapolators, ramp generators, position generators, Min-Max and limitation FBs
  - Virtual leading axle FB TRAJECTORY\_GEN1.

**ProProg wt III:**

- **SYSTEM1\_PLC01\_30bd00** (or above; for BM4-O-PLC-01)
  - BACI communication (process data, requirements data)
  - Function block for BACI (input, timer, output, trigger)
  - USS® protocol interface module
  - Code runtime measurement
- **SYSTEM2\_PLC01\_30bd00** (or above; for BM4-O-PLC-01)
  - b maXX drive PLC firmware
  - Firmware block LED
  - Firmware block INTR\_SET
  - 3964R® protocol interface module
  - TER\_x terminal blocks
- **UNIVERSAL\_30bd00** or above (regardless of the hardware)

- Drive status and control via FB DRIVE1
- Extrapolators, ramp generators, position generators, Min-Max and limitation FBs
- Virtual leading axle FB TRAJECTORY\_GEN1.

### 4.5.5 b maXX drive PLC technology components

---

You can extend the standard user libraries by adding complete drive functionality, i.e. the technology components. These are:

- Technology component cam disk:
  - PROPROG wt II: user library  
CAM\_PLC01\_21bd00 (or above)
  - ProProg wt III: user library  
CAM\_PLC01\_30bd00 (or above)
- Technology component register controller:
  - PROPROG wt II: user library  
REGISTER\_PLC01\_20bd00 (or above)
  - ProProg wt III: user library  
REGISTER\_PLC01\_20bd00 (or above)
- Technology component winder:
  - PROPROG wt II: user library  
WINDER\_PLC01\_20bd00 (or above)
  - ProProg wt III: user library  
WINDER\_PLC01\_20bd00 (or above)

The technology components offer drive functionality that provide a large number of application solutions due to interconnection and multiple instantiation.



#### NOTE

For integration, all the user libraries of the technology components need data types BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) and above.

### 4.5.6 Inserting a User Library into a Project

---

In PROPROG wt, you insert user libraries in the project tree under libraries. If the library in question is firmware, you must set .fwl when choosing the file format.

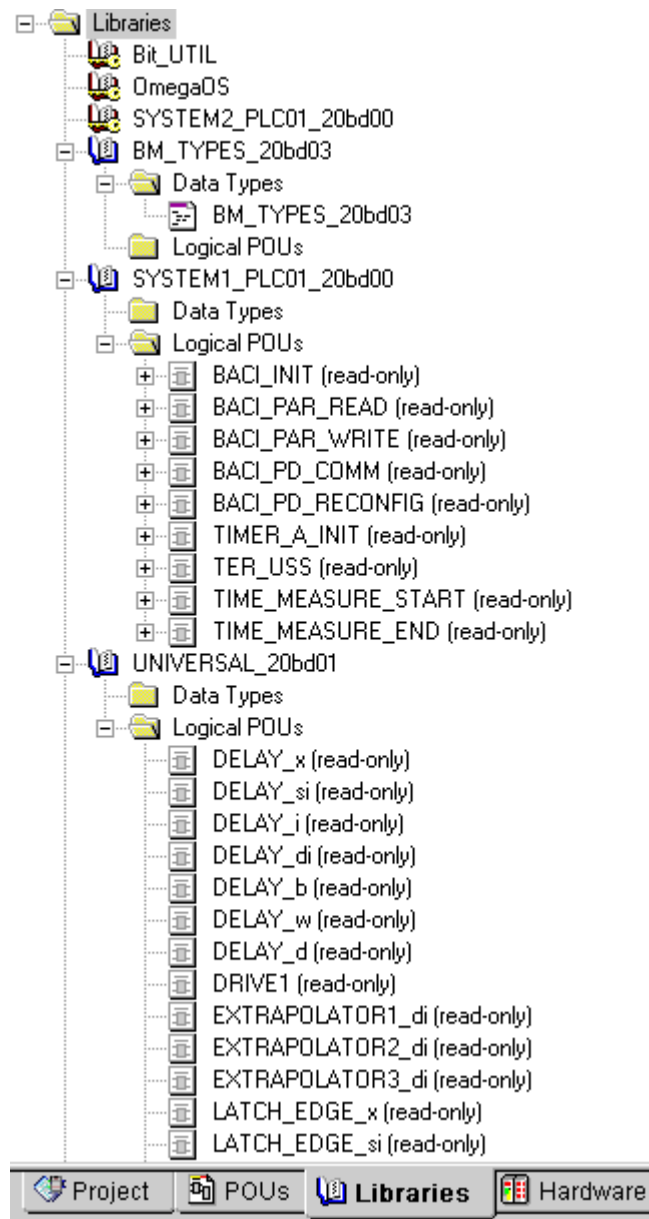


Figure 37: Standard selection of user libraries, firmware and data types using the libraries filter

#### NOTE



Libraries for PROPROG wt are delivered as packed (ZWT format) files). You must unpack the ZWT file under PROPROG wt. When unpacking the file, the system automatically stores the write-protected library in the specified PROPROG wt library path (Tools > options menu). In PROPROG wt II an untitled project displays that you should close without saving it. In ProProg wt III a project will be created with the name of the ZWT file. This project is no longer required.

The system automatically unpacks the firmware libraries to the firmware library directory of PROPROG wt.

### 4.6 BACI system description for the b maXX drive PLC

#### 4.6.1 Direct access to the BACI hardware from the b maXX drive PLC

BACI is a bus system with several slots between which data must be exchanged. In this context, as well as being connected to the common BACI-BUS by the BACI signals, every module is also connected by dual-port RAM.

A differentiation is made between the BACI-Master (b maXX controller and b maXX drive PLC) and the BACI-Slave. Only BACI-Masters can access the option modules that are assigned to them across the BACI-BUS. BACI-Slave option modules cannot access each other let alone the BACI-Master. Each individual option module can use its own dual-port RAM to query and evaluate the data that the master wrote.

The b maXX drive PLC represents itself to the b maXX controller as a BACI-Slave, which means that only the controller can access the PLC via the BACI-BUS. The PLC can use its own dual-port RAM area to evaluate the data that the controller wrote.

Slot overview BACI of the option module in the b maXX 4400 unit:

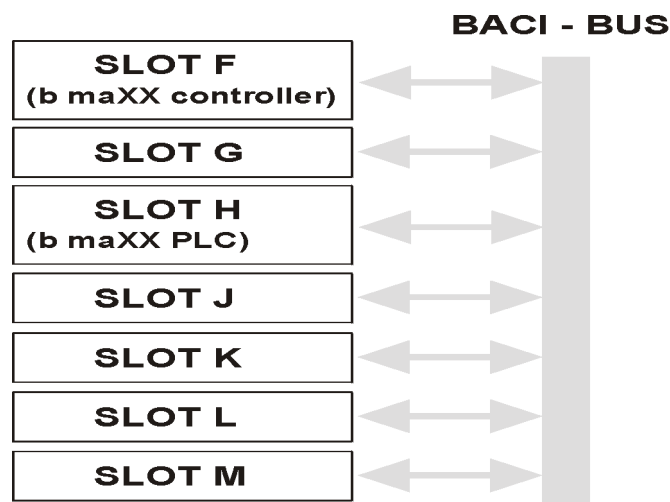


Figure 38: Slot overview

From the PLC's point of view, it is only possible in the application to contact BACI slots G to M to communicate with the option modules that are assigned to the PLC. It is also possible to access SLOT H (the preferred PLC slot) via BACI if the PLC itself is fitted in another slot.

Slot	Assigned IEC 61131-3 range (max.)
SLOT G	%MB 3.20000000 - %MB 3.20262143
SLOT H	%MB 3.30000000 - %MB 3.30262143
SLOT J	%MB 3.40000000 - %MB 3.40262143
SLOT K	%MB 3.50000000 - %MB 3.50262143
SLOT L	%MB 3.60000000 - %MB 3.60262143
SLOT M	%MB 3.70000000 - %MB 3.70262143
b maXX drive PLC's own dual-port RAM range	%MB 3.90000000 - %MB 3.90262143

#### NOTE



The PLC can only ever access SLOTS on a different option module. To evaluate the data from the controller, the PLC must not contact its own slot range in which the PLC itself is located; rather, it must always proceed over its own dual-port RAM range (%MB 3.90000000 - %MB 3.90262143).

In the application, you can then assign a software structure that describes the memory and register structure of an option module and which is used in a ready-made library to a variable that points to the slot that is used.

Global variables of this type are already appropriately predefined in the BM4\_O\_PLC01 template in PROPROG wt in the "Global\_Variables" global variable sheet such that, depending on the option module and the slot that is used, users only need to connect to the library components that are in use. In future, the configurator – which will be integrated at a later expansion stage of PROPROG wt for the b maXX drive PLC – will automatically carry out these tasks. This means that users will no longer have any contact with the SLOT ranges mentioned above.

For further explanations of the structures and functions of the option modules, refer to their respective operating instructions and application manuals.

#### 4.6.2 Process data communication between b maXX controller and b maXX drive PLC, synchronization to a common BACI hardware signal source

In the case of process data communication between the b maXX controller and the b maXX drive PLC a basic differentiation is made between two applications:

1. Process data communication between the b maXX controller and the b maXX drive PLC is on a direct basis with the controller and the PLC not being synchronized to a BACI hardware signal. This means that the controller itself triggers the interrupts on the PLC for communication every time, taking into account the set actual value period after processing of its own controller actual value channel. This prevents the controller and the PLC accessing the BACI at the same time and means that communication is always synchronized between the two devices.

To deal with this case, you must connect at FB "BACI\_INIT" of library SYSTEM1\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher at input i\_EVENT event INT#4 (priority level 13) or INT#8 (priority level 14) and must also create a task with the same event in which process data communication to b maXX controller via FB "BACI\_PD\_COMM" is to run.

### Process data communication directly by b maXX controller (MasterCS\_Actual1):

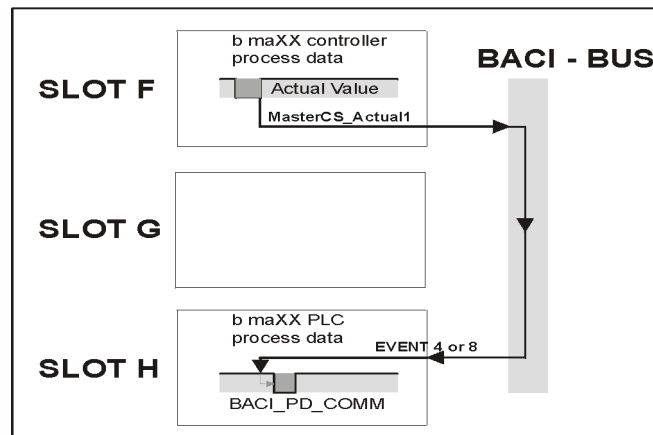


Figure 39: Process data communication

### 2. Direct synchronization to a common source:

In general, the system requests that several controllers – if they are running on an interconnected basis, for example – must run synchronized with one another. You can achieve this by synchronizing all the controllers to a common BUS signal (e.g using a common CANsync ring) that is generated at one location and which the option modules switch through to the respective local BACI buses of the b maXX controllers (preferably to BACI signal SYNC1).

Using case 1 as the basis, this would mean that an option module synchronizes the b maXX controller, which, after processing its assigned actual value channel with its PLC process data, also exchanges data on a synchronized basis.

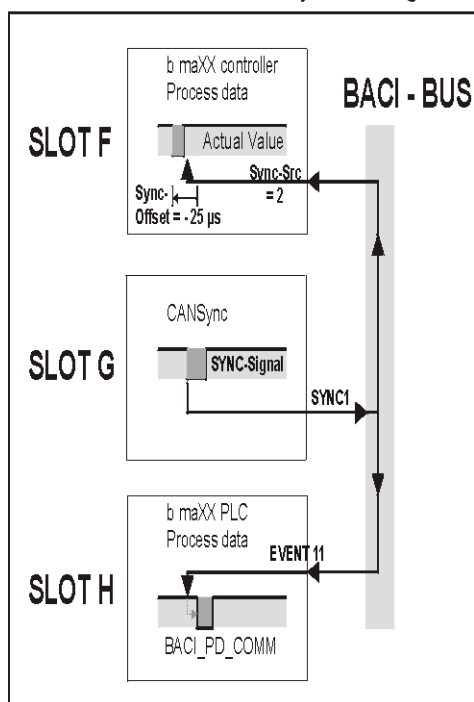
However, since a certain dead time is involved due to the duration of processing in the controller and since it is usually necessary in any case to exchange (field bus) data on a synchronous basis with the option module that passes on or generates the signal to the b maXX controller, in practice a different procedure is chosen: You synchronize both the controller and the PLC directly to the same hardware signal (usually BACI signal SYNC1).

As a result, you can achieve better synchronization of all the controllers and PLCs since it is only hardware-dependent.

For this case 2, you must connect on FB "BACI\_INIT" of library SYSTEM1\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher at input i\_EVENT event INT#11 (use priority level 14, SYNC1 signal from the BACI) or INT#12 (use priority level 14, SYNC2 signal from the BACI) and create an event-like task in which process data communication with the b maXX controller is to run via FB "BACI\_PD\_COMM" and possibly additional field bus communication is to take place with the option module that triggered the event.



Prozess data communication by SYNC1-signal:



Process data communication by SYNC2-signal:

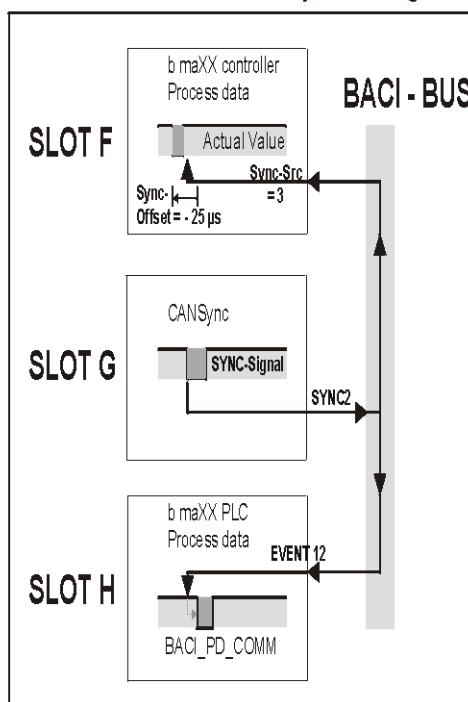


Figure 40: Process data communication via a SYNC1 or SYNC2 signal

You should, however, take into account that in the case of a "Sync-Offset = 0" controller parameter that is preset by default, both the controller and the PLC try to access the BACI data at the same time when exchanging process data and this results in an access conflict. To offset the access instants of the controller and the PLC to the BACI, you must set a negative Sync-Offset of  $-25 \mu s$  in the controller. This ensures that processing of the actual value channel in the controller is always completed and the current actual values from the controller are available to the PLC when it processes the interrupt that the BACI signal triggered.

Refer to the descriptions of FBs "BACI\_INIT" and "BACI\_PD\_COMM" of library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher for information about the necessary settings for synchronization of both cases in the controller and the PLC.

#### 4.6.3 b maXX drive PLC interrupt sources that the application can use

For the b maXX PLC, you can represent hardware signals from the BACI, write accesses from the controller to the PLC, and various PLC-internal sources such as the CPU timer or interrupt sources that can be linked to specific program sections of an application via the event selection of the EVENT-BYPASS task.



### INFORMATION ON EVENT BYPASS EVENT TASKS

If you insert an EVENT task type in PROPROG wt II, you must also set the "BYPASS" attribute.

Since EVENT-BYPASS tasks are called directly by the event and are not monitored by the runtime system, the "interval" and "watchdog time" settings in task management are meaningless and are ignored.

Apart from this, you must in principle not set any breakpoints or activate an address status in the program sections that are linked to an EVENT BYPASS task.

You must set up an interrupt source during the PLC's cold boot/warm restart stage either by means of function block "BACI\_INIT" (for events „4“, „8“, „11“ and „12“, see the standard library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher) or of "INTR\_SET" for events „0“, „2“, „5“, „6“ (see standard library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher). Only then, in PLC status "RUN" when the event occurs, can the system call the program section that is assigned in the EVENT BYPASS task.

Due to the assignment of an event to a program section, this section is always synchronized and the system processes it at the same periods as the event source.

Representation of all the EVENTS on the BM4-O-PLC01 option module that can trigger an interrupt:

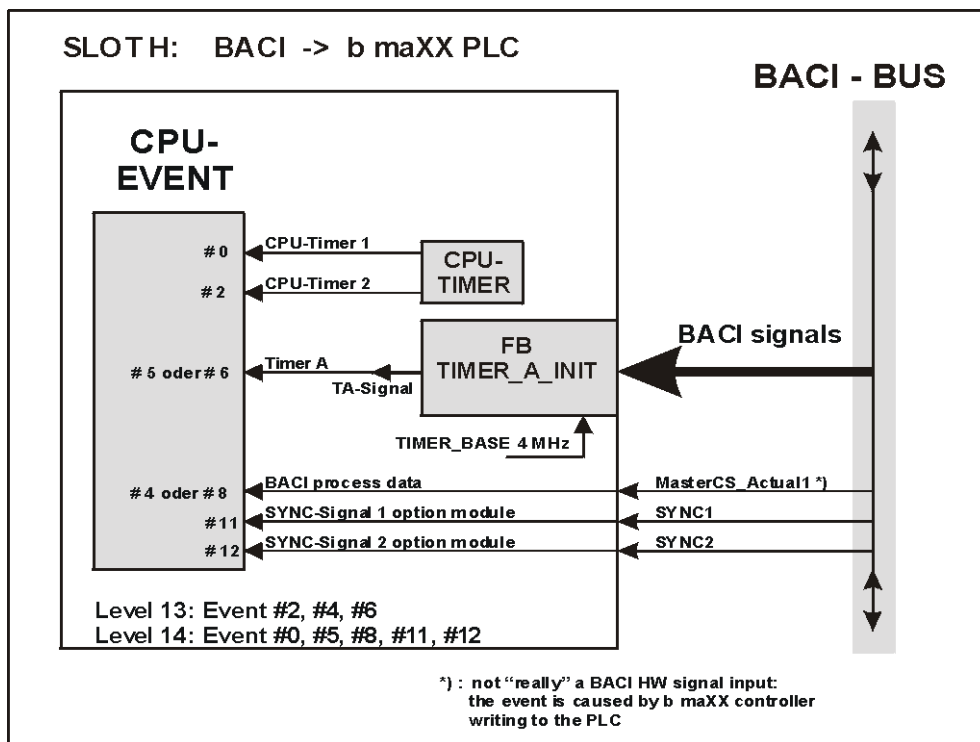


Figure 41: Representation of all EVENTS

Detailed configuration options for the Timer A EVENT by choosing the BACI signal via the INPUT function of block "TIMER\_A\_INIT" and applying to the internal TA signal:

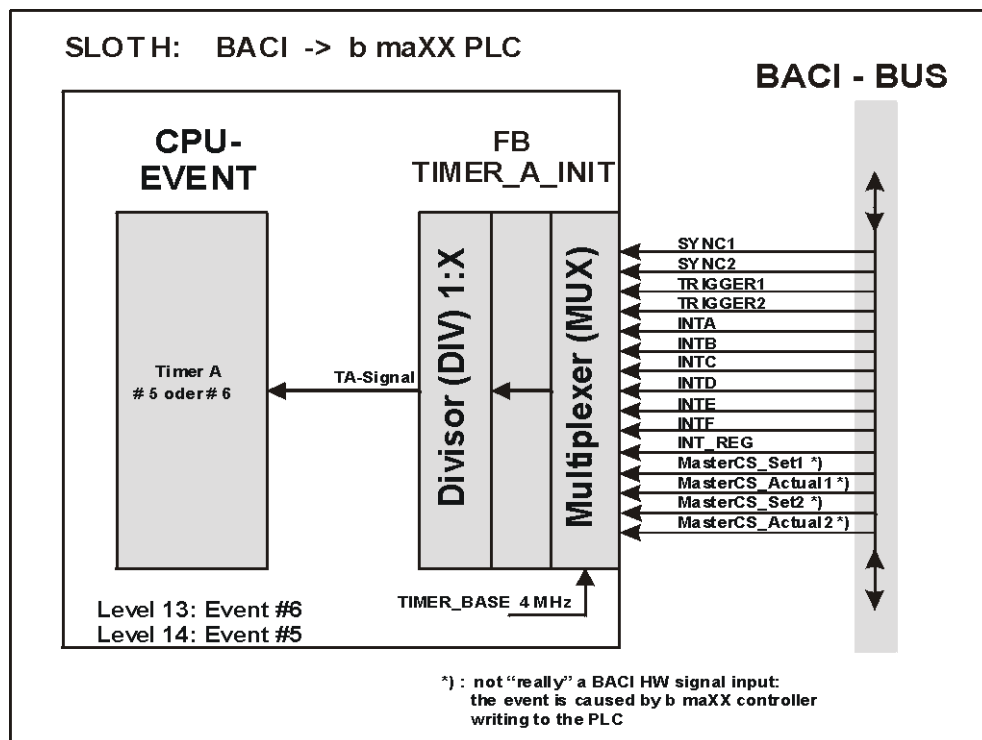


Figure 42: Detailed configuration options

#### 4.6.4 Generating BACI signals via function block TIMER\_A\_INIT

You can also configure the b maXX drive PLC for BACI signal output (b maXX drive PLC -> BACI); in this case, it can function as a trigger source for other option modules, for example.

In this connection, the signal source is generated either on the PLC itself or it comes from another BACI signal that is rerouted via the PLC and may, if necessary be stepped-down (see the description for function block "TIMER\_A\_INIT" of library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher).

Configuration options for generating BACI signals by injecting the internally generated TA signal to the BACI signal line via the OUTPUT function of block "TIMER\_A\_INIT":

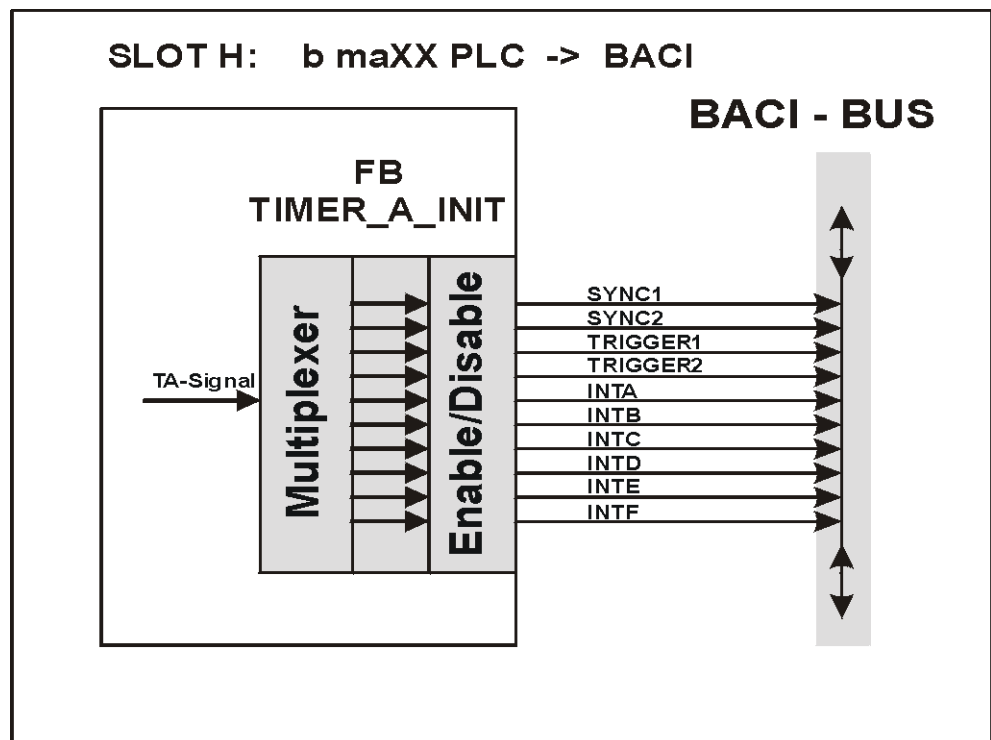


Figure 43: Configuration options for generating BACI signals

By calling the OUTPUT function several times, you can multiplex the internal TA signal to different BACI signals at the same time.

You can reroute BACI signal via the PLC (if desired with signal division) by appropriately combining the INPUT and OUTPUT functions of "TIMER\_A\_INIT" (see the "TIMER\_A\_INIT" description).

### 4.6.5 "TIMER\_A\_INIT" function block

Description of function block "TIMER\_A\_INIT" of library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

The function block for BACI has several functions:

Function	Description
INPUT	You can choose a BACI signal, divide it and multiplex it to the internal TA line and, if desired, use it to trigger a TIMER_A_Interrupt.
TIMER	INPUT function: By selecting the "4MHz" source (= time base of Timer_A) and stating a divider value, you can generate virtually any cycle signal. If desired, you can use it to trigger a TIMER_A_Interrupt.
OUTPUT	The internal TA line is switched to a selectable BACI output.

Function	Description
TRIGGER	If you combine the TIMER function with the OUTPUT function, you can switch the generated cycle signal to one or more BACI outputs.
Enable/disable OUTPUT signal	You can disable BACI outputs and enable them again on selective basis.

**NOTE**

on the TIMER\_A\_INIT block:

Application: It can be integrated at the initialization stage (cold boot/warm restart task) as well as in the DEFAULT task or in any cyclic task you like.

In this connection, you can place the block several times at different locations or several times successively to install the corresponding individual function.

If you use the block in the cyclical section, the system must set input x\_EN back to FALSE after activating the function so that the TIMER can run or the divider can carry out division!

Parameter input	Data type	Description
x_MODE	BOOL	Selection of the basic function (signal direction)
us_SIGNAL	USINT	signal source/destination
u_DIVIDER	UINT	Divider value
x_EN_SIGNAL_OUT	BOOL	Enable/disable the output signal when the OUTPUT function is selected
x_EN	BOOL	Enable the block

Parameter output	Data type	Description
x_ERR	BOOL	Error bit

Input **x\_MODE**:

Using x\_MODE, you can choose the basic function:

x\_MODE = FALSE (us\_SIGNAL = signal source): INPUT, TIMER functions. => The system generates an internal TA signal.

x\_MODE = TRUE (us\_SIGNAL = signal destination): Enable/disable OUTPUT, TRIGGER, OUTPUT signal. If the BACI output has been released, the internal TA signal is switched through to the specified BACI signal destination.

Input **us\_SIGNAL**:

For <b>x_MODE := FALSE:</b> <b>us_SIGNAL :=</b>	Selection of signal source:
USINT#0	4 MHz
USINT#1	MasterCS_Set1
USINT#2	MasterCS_Actual1
USINT#3	MasterCS_Set2
USINT#4	MasterCS_Actual2
USINT#5	INTA
USINT#6	INTB
USINT#7	INTC
USINT#8	INTD
USINT#9	INTE
USINT#10	INTF
USINT#11	TRIGGER1
USINT#12	TRIGGER2
USINT#13	SYNC1
USINT#14	SYNC2
USINT#15	INT_REG
> 15	Not allowed

For <b>x_MODE := TRUE:</b> <b>us_SIGNAL :=</b>	Selection of BACI signal destination
USINT#0 - USINT#4	Not allowed
USINT#5	INTA
USINT#6	INTB
USINT#7	INTC
USINT#8	INTD
USINT#9	INTE
USINT#10	INTF
USINT#11	TRIGGER1
USINT#12	TRIGGER2
USINT#13	SYNC1
USINT#14	SYNC2
> 14	Not allowed

**Input u\_DIVIDER:**

Is only evaluated with the INPUT and TIMER functions ( $x\_MODE = FALSE$ ).

u\_DIVIDER indicates the ratio with which the input signal is divided. In this connection, the following division formula applies:

$$\text{Internal TA signal} := us\_SIGNAL / (u\_DIVIDER + 1)$$

In the case of  $u\_DIVIDER = 0$ , this means that the external BACI signal is switched through directly on a 1:1 basis to the internal TA signal, i.e. the signal is not divided.

**Input x\_EN\_SIGNAL\_OUT:**

Is only evaluated with the OUTPUT and TIMER functions ( $x\_MODE = TRUE$ ).

The internal TA signal is switched to the selected BACI output ( $x\_EN\_SIGNAL\_OUT := TRUE$ ) or is disabled again ( $x\_EN\_SIGNAL\_OUT := FALSE$ ).

The internal TA signal can be output to several BACI outputs by means of several successive block calls and different BACI destinations.

**Input x\_EN:**

You use  $x\_EN = TRUE$  to install the respective function. In the case of  $x\_EN := FALSE$ , the system does not process the block. If the block is executed cyclically, the system should reset input  $x\_EN$  to  $FALSE$  after activation of an individual function so that the TIMER can run or the BACI input signal can be divided.

**Output x\_ERR := TRUE:**

Selection of the BACI destination is not supported!

**Detailed explanation of the functions:****INPUT special function: TIMER ( $x\_MODE := FALSE$  and  $us\_SIGNAL := USINT\#0$ ):**

A timer basis of 4 MHz is permanently preset. Formula for calculating the time period:

$$\text{Time period} = (u\_DIVIDER + 1) \times 0.25 \mu s.$$

Minimum value for "u\_DIVIDER":  $UINT\#999 \Rightarrow \text{Time} = 250 \mu s$

Maximum value for "u\_DIVIDER":  $UINT\#65535 \Rightarrow \text{Time} = 16384 \mu s$

Formula for calculating u\_DIVIDER with the specified time period:

$$u\_DIVIDER = (\text{time period} / 0.25 \mu s) - 1$$

If you want the timer to trigger an interrupt via the internally generated TA signal, the system must, after calling the timer function via "TIMER\_A\_INIT", execute system function block "INTR\_SET" (see firmware library SYSTEM2\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher) with parameters:  $i\_EVENT := INT\#5$  (bzw.  $i\_EVENT := INT\#6$ ) for TIMER A. (The other parameters  $i\_MODE$ ,  $i\_PAR1$  and  $i\_PAR2$  are ignored).

In this connection, you should choose a time period that is not too low or too high; otherwise, the next cyclical interrupt may already be pending before the currently running one has been completed. Since the interrupts are set up as high-priority bypass interrupts,

only this interrupt runs and the PLC goes offline (-> The other application "no longer responds").

To avoid this, you are advised to set an interrupt interval of at least 250 µs. This corresponds to a "u\_DIVIDER" value of  $\geq$  UINT#999.

### Special function: TRIGGER

If you combine the TIMER function with the OUTPUT function, you can switch the generated cycle signal as a trigger signal to one or more BACI outputs.

#### 1. Call of the TIMER function:

A timer basis of 4 MHz is permanently preset. If you connect a divider value at input "u\_DIVIDER" that is less than UINT#3, the system generates a continuous LOW signal at the internal timer output TA. If after this, you want to output the internal TA signal using the OUTPUT function to the BACI (e.g. a PLC function as a BACI trigger source), you should first enter in the INPUT function at least one value greater than UINT#7 to ensure that you do not violate the BACI-BUS timing.

Formula for calculating the time period:

$$\text{Time period} = (\text{u\_DIVIDER} + 1) \times 0.25 \mu\text{s}.$$

Minimum value for "u\_DIVIDER": UINT#8  $\Rightarrow$  Time = 2.25 µs

Maximum value for "u\_DIVIDER": UINT#65535  $\Rightarrow$  Time = 16384 µs

Formula for calculating u\_DIVIDER:

$$\text{u\_DIVIDER} = (\text{time period} / 0.25 \mu\text{s}) - 1$$

If you want the trigger pulse to also trigger an interrupt, it is advisable to set an interrupt interval of at least 250 µs. This corresponds to a "u\_DIVIDER" value of  $\geq$  UINT#999.

#### 2. Call of the OUTPUT function (BACI destination = USINT#11 (Trigger 1)): see below

### OUTPUT special function: Enable/disable BACI signal:

If you want to output the internally generated TA signal to several BACI outputs at the same time, the system can call the block several times in succession using the respective BACI destination and "x\_MODE := TRUE" as well as "x\_EN\_SIGNAL\_OUT := TRUE" for driver enable. Using "x\_EN\_SIGNAL\_OUT := FALSE", you can reverse the enable on a selective basis.

Internal TA signal is intended to generate an interrupt:

After parameterizing "TIMER\_A\_INIT", system function block "INTR\_SET" (see firmware library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher) must be executed with parameters: i\_EVENT := INT#5 (or i\_EVENT := INT#6) for TIMER A. (The other parameters i\_MODE, i\_PAR1 and i\_PAR2 are ignored).

Example application 1:

The b maXX drive PLC on the SYNC1 line is intended to generate for a BM4-O-IEI-01 option module a trigger signal with a 2-ms period. A PLC program section is to be executed synchronously with the generated trigger signal.



1. Step: generate an internal TA signal via the timer function with a time period of 2 ms:  
Execute FB "TIMER\_A\_INIT" with: x\_MODE: = FALSE; us\_SIGNAL := USINT#0; (for timer function); u\_DIVIDER = (2000  $\mu$ s / 0.25  $\mu$ s) -1 = UINT#7999;
2. Step: Output the TA signal as a trigger signal to SYNC1:  
Execute block "TIMER\_A\_INIT" with: x\_MODE: = TRUE; us\_SIGNAL := USINT#13; (for target selection SYNC1); x\_EN\_SIGNAL\_OUT := TRUE;
3. Step: Execute block "INTR\_SET" with i\_EVENT := INT#5.  
=> By creating an EVENT BYPASS task with event „5“, you can execute a PLC program synchronously with the trigger signal.

#### Example application 2:

You want input BACI signal TRIGGER1 to be divided by a value of 10 and then to be switched back to output BACI line TRIGGER2; apart from this, the divided signal should trigger an interrupt.

1. Step: TRIGGER1 -> 1:10 -> internal TA signal:  
Execute FB "TIMER\_A\_INIT" with: x\_MODE: = FALSE; us\_SIGNAL := USINT#11; (for selection of TRIGGER1) u\_DIVIDER := UINT#9; (For division ratio of 1:10)
2. Step: internal TA signal -> TRIGGER2:  
Execute block "TIMER\_A\_INIT" with: x\_MODE: = TRUE; us\_SIGNAL := USINT#12; (for selection of TRIGGER2); x\_EN\_SIGNAL\_OUT := TRUE;
3. Step: Execute block "INTR\_SET" with i\_EVENT := INT#5.  
=> By creating an EVENT BYPASS task with event „5“, you can execute a PLC program synchronously with the divided signal.

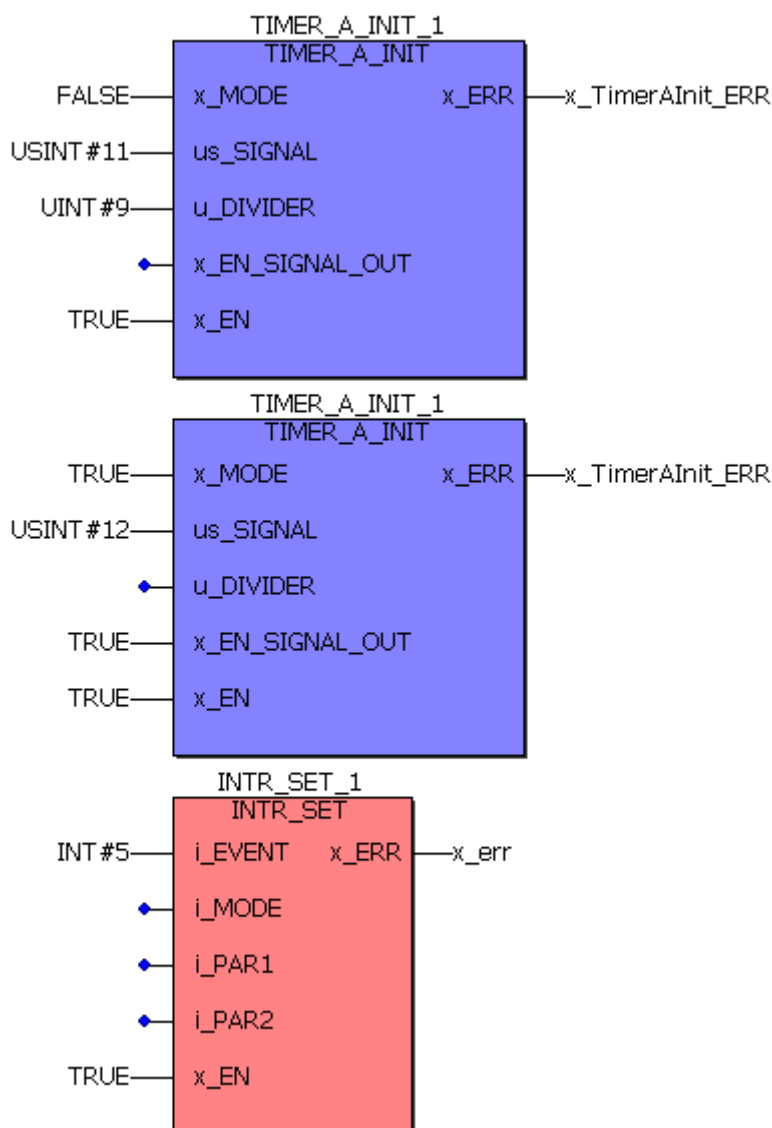


Figure 44: The TRIGGER1 signal triggers every tenth time the TRIGGER2 signal and the TRIGGER1 signal triggers the EVENT BYPASS task „5“

### 4.6.6 Sample Configurations

Depending on the option modules that you use and on the synchronization requirements, you need different configurations for process data communication between the b maXX controller and the PLC, for managing interrupt sources and for providing and processing trigger and sync signals.

The following examples can be used for the majority of applications:

#### Example A:

Implementing process data communication between the b maXX controller and the PLC. The controller functionality "MasterCS\_Actual1" (controller's actual value channel) carries out synchronization.

A SYNC1 signal should be formed on the PLC from the same source; this signal then triggers the BM4-O-IEI-01 option module.

**NOTE**

An option module can be synchronized to event "MasterCS\_Actual1" (actual channel of the controller) via the PLC by relocating this event to a BACI signal (e.g. SYNC1)!

**Example B:**

Implementing high-precision BACI process data communication by generating the SYNC1 signal and triggering a BM4-O-IEI-01 option module at the same time and triggering the b maXX controller via the SYNC1 signal.

**Example C:**

Implementing BACI process data communication within a CANSync event task and triggering a BM4-O-IEI-01 option module as well as the b maXX controller via synchronizing signal SYNC1.

**Example D:**

Setting up a timer event task for a cyclical call in which the system is to process serial RS485 communication (via the 9-pin female Sub-D connector on the b maXX drive PLC BM4-O-PLC-01 interface X1). The function blocks for cyclical communication are placed in a POE that is assigned to this task.

**NOTE**

Code runtime measurement is recommended for optimizing the run times of process data. (For more details on this topic, see [►Code runtimes ◀](#) from page 108 onward)

You can carry out this measurement using FBs TIME\_MEASURE\_START and TIME\_MEASURE\_END from library SYSTEM1\_PLC01\_20bd00 (PROProg wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

## 4.6 BACI system description for the b maXX drive PLC

### 4.6.7 Example A: Implementing BACI process data communication within a BACI event task

Cold boot/warm restart task:

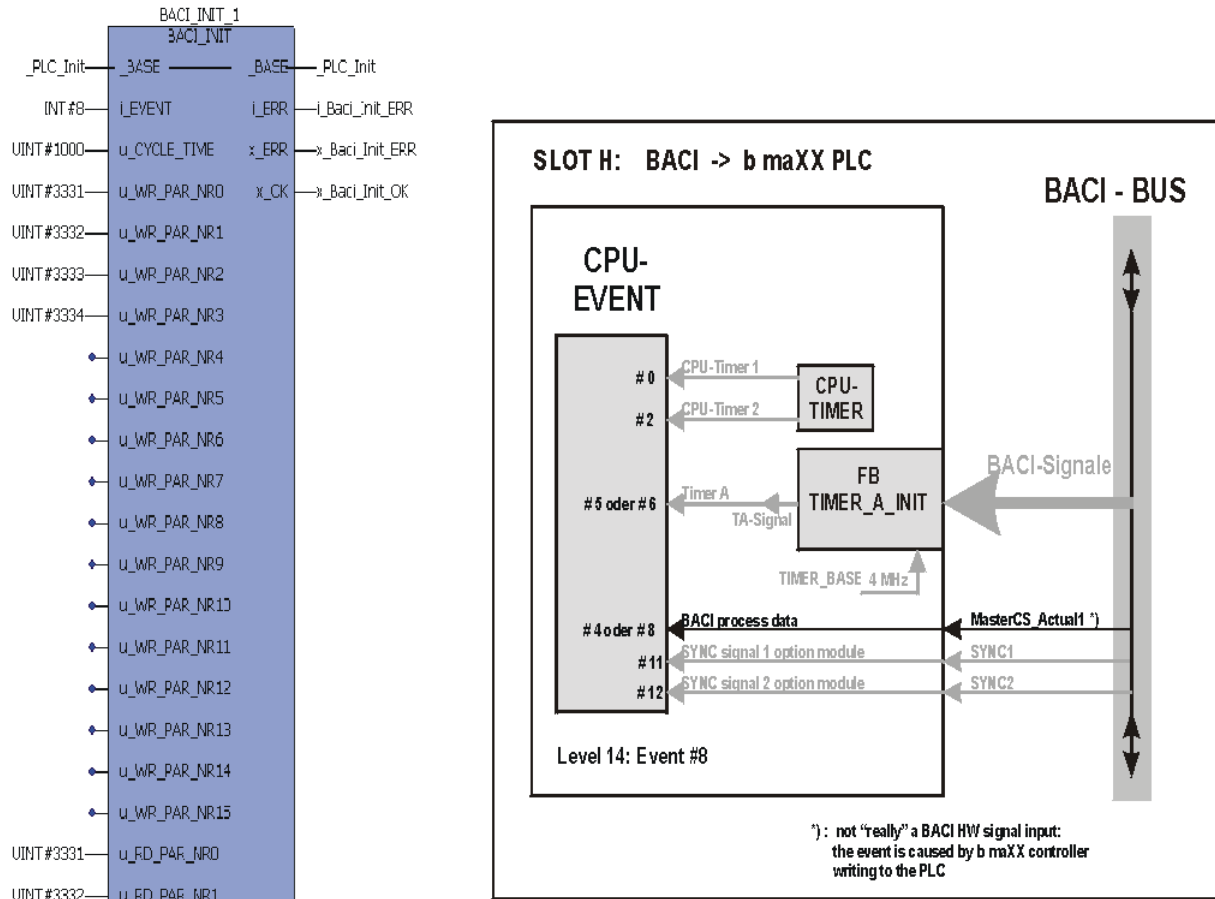


Figure 45: Initialize via the BACI interface process data communication between the b maXX controller and the b maXX drive PLC

The block is called once during the initialization stage (cold boot/warm restart task).

Interconnection:

Global variable "\_PLC\_Init" is already predefined appropriately in the BM4\_O\_PLCO1 template in PROPROGRAM in global variable sheet "Global\_Variables"; this variable is a placeholder for the b maXX drive PLC's own DPRAM range (this is used for controller communication).

**u\_CYCLE\_TIME** = **UINT#1000** sets a cycle time of 1000 µs.

**i\_EVENT** = 8 initializes event "BACI process data" and interrupt level 14 (high priority).

Effect: For process data communication, parameter numbers 3331-3334 are set up as target and actual values. Every time the controller operates event "MasterCS\_Actual1" (= actual value channel of the controller) interrupt event „8" is executed (see the representation to the right of the "BACI\_INIT" call).

PROProg wt II:

FB "BACI\_INIT" is stored in library SYSTEM1\_PLC01\_20bd00 (or higher) and it uses library BM\_TYPES\_20bd03 or higher and firmware library SYSTEM2\_PLC01\_20bd00 or higher. Firmware block "INTR\_SET" is a component of block "BACI\_INIT".

ProProg wt III:

FB "BACI\_INIT" is stored in library SYSTEM1\_PLC01\_30bd00 (or higher) and it uses library BM\_TYPES\_30bd01 or higher and firmware library SYSTEM2\_PLC01\_30bd00 or higher. Firmware block "INTR\_SET" is a component of block "BACI\_INIT".

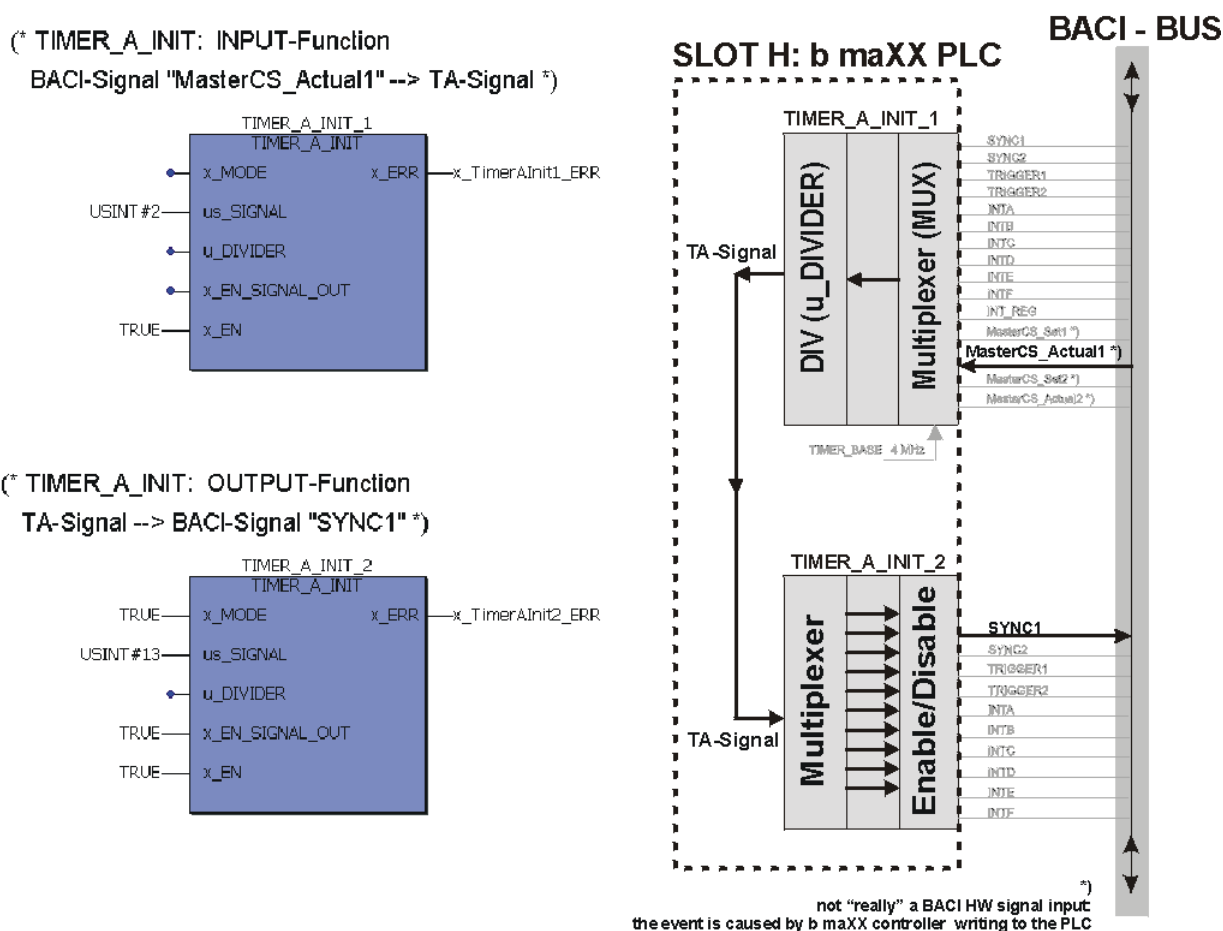


Figure 46: Global function of TIMER\_A\_INIT calls: Every time the controller operates event "MasterCS\_Actual1" (= actual value channel of the controller), the PLC generates a BACI-SYNC1 signal for triggering the BM4-O-IEI-01 option module.

The blocks are called once during the initialization stage (cold boot/warm restart task).

To the right of the calls, you can see how the signal points on the PLC are switched to achieve signal change-over via the PLC hardware.

Step 1 (call "TIMER\_A\_INIT\_1"):

Interconnection: x\_MODE unused (= FALSE) and us\_SIGNAL = USINT#2:

Effect: BACI signal "MasterCS\_Actual1" is switched undivided onto the PLC-internal TA signal:

## 4.6 BACI system description for the b maXX drive PLC

Step 2 (call "TIMER\_A\_INIT\_2"):

Interconnection: x\_MODE = TRUE, x\_EN\_SIGNAL\_OUT = TRUE and us\_SIGNAL = US-INT#13:

Effect: The internal TA signal is interconnected to the BACI-SYNC1 signal and output to the BACI.

FB "TIMER\_A\_INIT" is stored in library SYSTEM1\_PLC01\_20bd00 (PROProg wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

- Triggering a BM4-O-IEI-01 option module via a SYNC1 signal: See the technical description of the BM4-O-IEI-01 module.

Process data communication in the BACI process data BYPASS task:

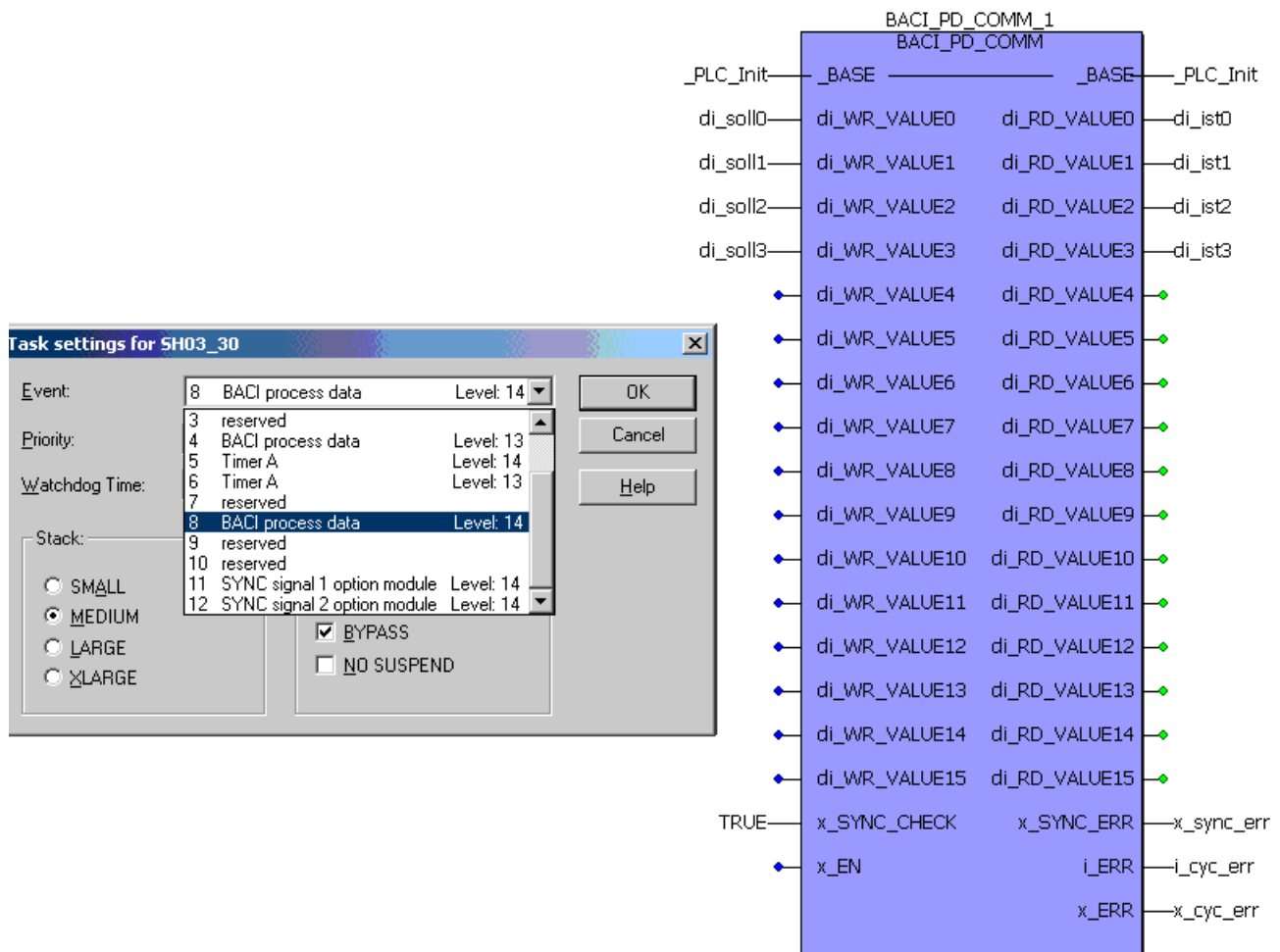


Figure 47: Carrying out process data communication

By means of block "BACI\_PD\_COMM" in a POE that was assigned event „8 process data Level 14", the system now cyclically transfers every 1000  $\mu$ s the target and actual values of the set parameters (in our Example A, these are parameters 3331-3334).

FB "BACI\_PD\_COMM" is stored in library SYSTEM1\_PLC01\_20bd00 (PROProg wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

#### 4.6.8 Example B: Implementing BACI process data communication via a self-generated SYNC1 signal

Implementing high-precision BACI process data communication by generating the SYNC1 signal and triggering a BM4-O-IEI-01 option module at the same time and triggering the b maXX controller via the SYNC1 signal.

The TIMER\_A\_INIT block is used to set timer A to 1 ms and to switch it to the BACI-SYNC1 signal; this is so that the controller and the BM4-O-IEI-01 option module (and also the PLC!) can synchronize this signal:

The SYNC1 signal activates event „11 SYNC signal 1 option module Level 14" and carries out process data communication with the b maXX controller in the associated POE.

Cold boot/warm restart task:

(\* TIMER\_A\_INIT: function timer

Timer A should have a time period of exactly 1 mSec:

=>  $u\_DIVIDER = (time\ period / 0,25\ \mu s) - 1 = 3999$  \*)

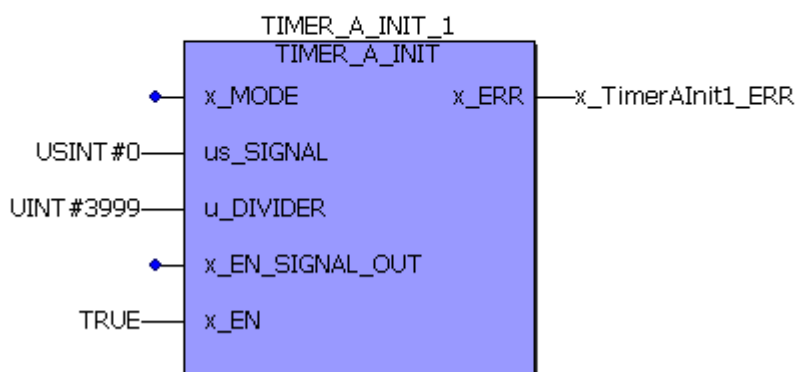


Figure 48: Set up timer A to a 1-ms period

The block is called once during the initialization stage (cold boot/warm restart task).

(\* TIMER\_A\_INIT: function OUTPUT

TA-Signal -> BACI-Signal "SYNC1" \*)

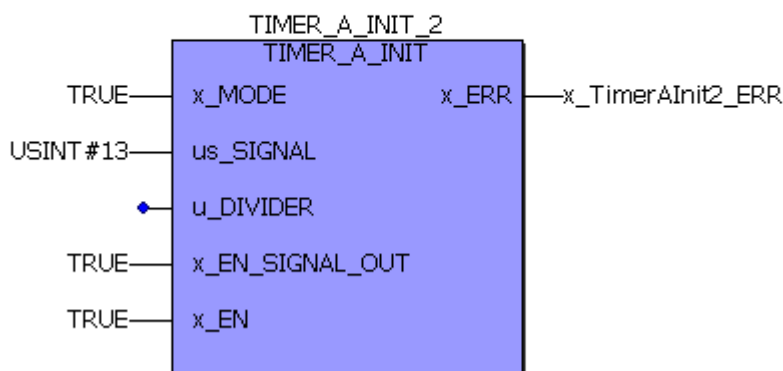


Figure 49: Output to SYNC1-BACI signal:

## 4.6 BACI system description for the b maXX drive PLC

The block is called once during the initialization stage (cold boot/warm restart task).

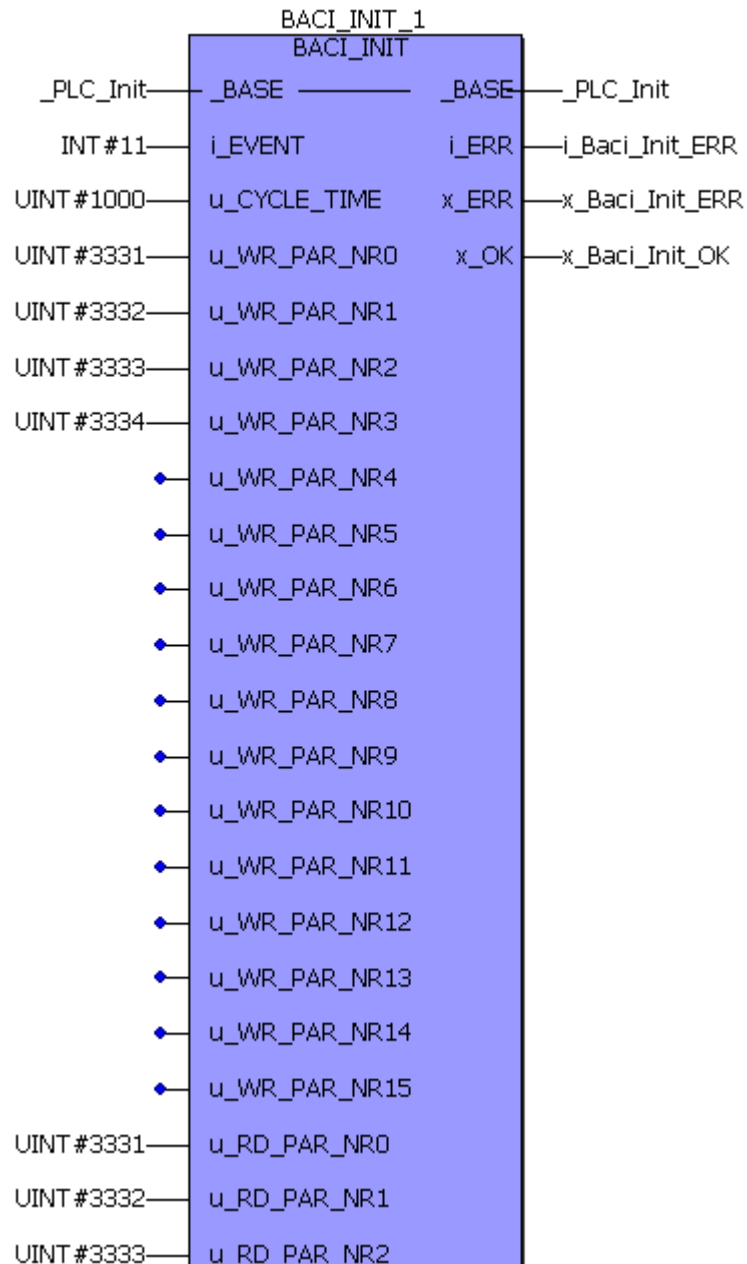


Figure 50: Set up process data communication to the controller via the SYNC1 event. `u_CYCLE_TIME = UINT#1000` sets a cycle time of 1000  $\mu$ s. `i_EVENT = 11` initializes event "SYNC-Signal 1 option module level 14 (high priority)".

The block is called once during the initialization stage (cold boot/warm restart task).



Process data communication in the SYNC signal 1 option module BYPASS task:

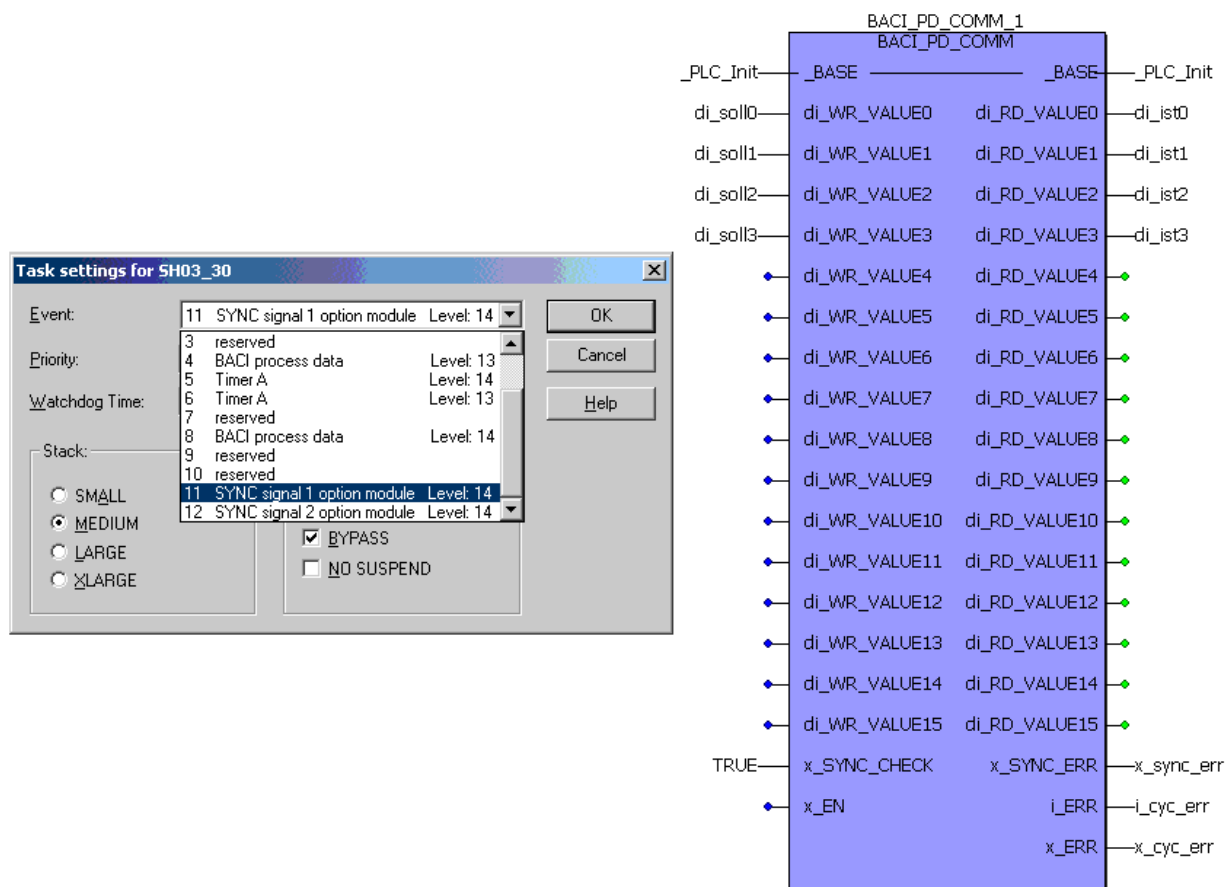


Figure 51: Carrying out process data communication

By means of block "BACI\_PD\_COMM" in a POE that was assigned event „11 SYNC signal 1 option module Level 14“, the system now cyclically transfers every 1000 µs the target and actual values of the set parameters.

The b maXX controller must also be synchronized to the SYNC1 signal source.

For the necessary b maXX controller settings on WinBASS II page "Synchronization":

- Source for sync signal:  
"Use Sync 1 signal from the BACI" (i\_EVENT = INT#11)
- Choose a 1-ms Sync interval (=> the same time period must be connected at Input "u\_CYCLE\_TIME" in µs).
- Sync offset: -25.0 µs.
- Sync tolerance: 2.0 µs.

To offset the access instants of the controller and the PLC to the BACI, you must set a negative Sync-Offset of -25 µs in the controller. This ensures that processing of the actual value channel in the controller is always completed and the current actual values from the controller are available to the PLC when it processes the interrupt that the BACI signal triggered.

Triggering a BM4-O-IEI-01 option module via a SYNC1 signal: See the operating instructions of the BM4-O-IEI-01 option module.

### 4.6.9 Example C: Implementing BACI process data communication via an external SYNC1 signal

Implementing BACI process data communication within a CANsync event task and triggering a BM4-O-IEI-01 option module as well as the b maXX controller via synchronizing signal SYNC1.

Now, you do not generate the SYNC1 signal yourself; rather, the signal comes from a CANsync option module (BM4-O-CAN-05 for CAN-Slave or BM4-O-CAN-06 for CAN-Master). For information on creating the program and initializing the option modules, see the associated operating instructions and application manuals.

Cold boot/warm restart task:

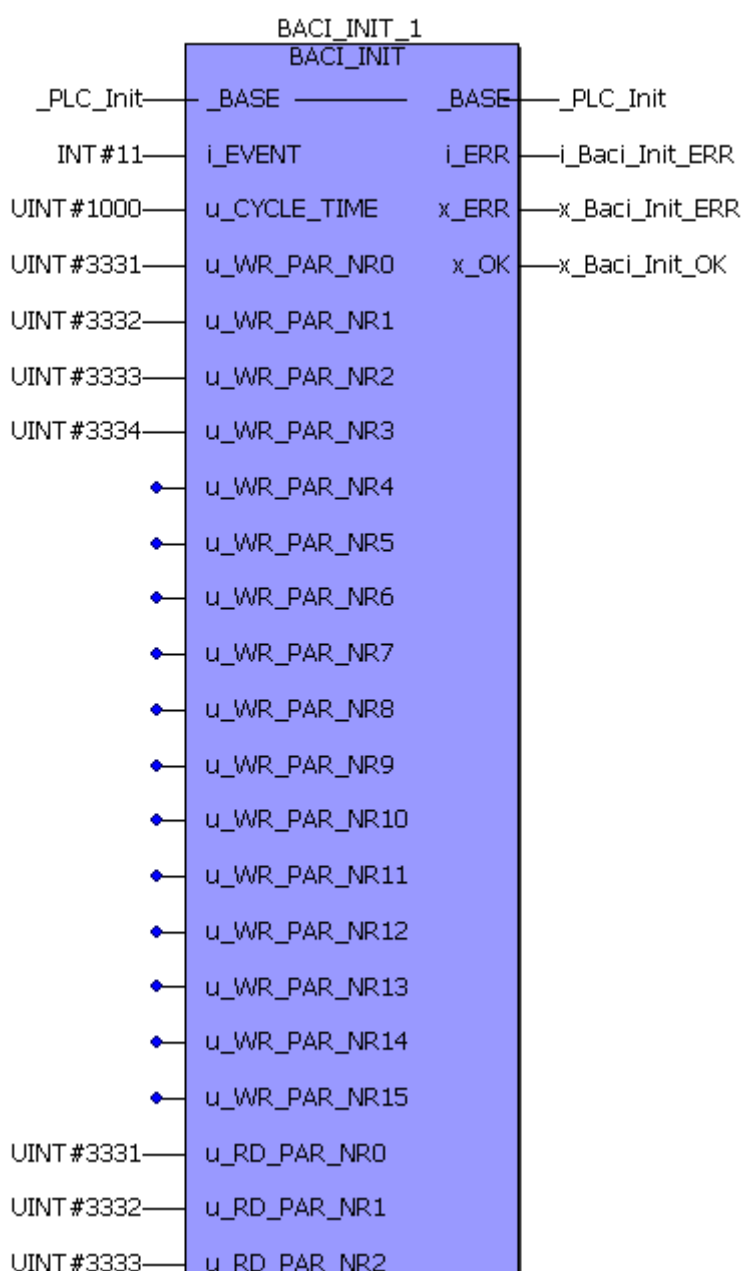


Figure 52: Set up process data communication to the controller via the SYNC1 event. `u_CYCLE_TIME = UINT#1000` sets a cycle time of 1000  $\mu$ s. `i_EVENT = 11` initializes event "SYNC-Signal 1 option module level 14 (high priority)".

The block is called once during the initialization stage (cold boot/warm restart task).

Process data communication in the SYNC signal 1 option module BYPASS task:

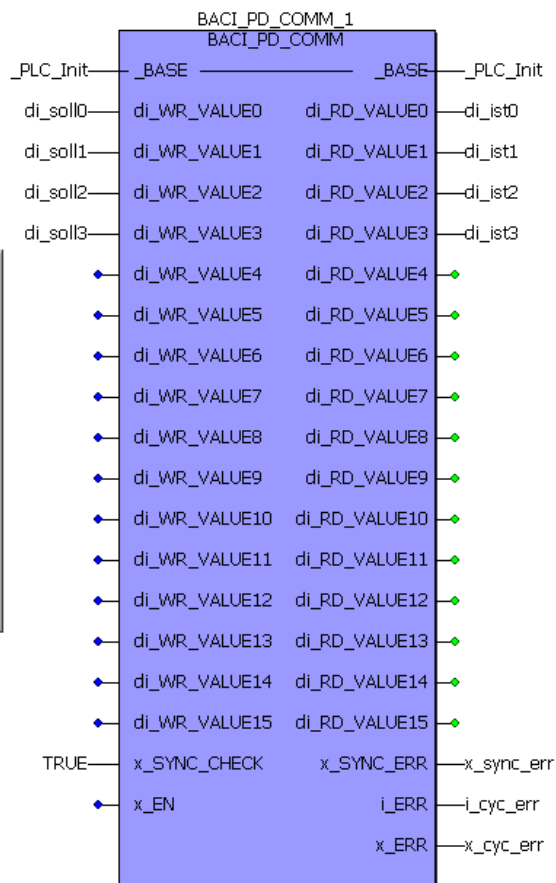
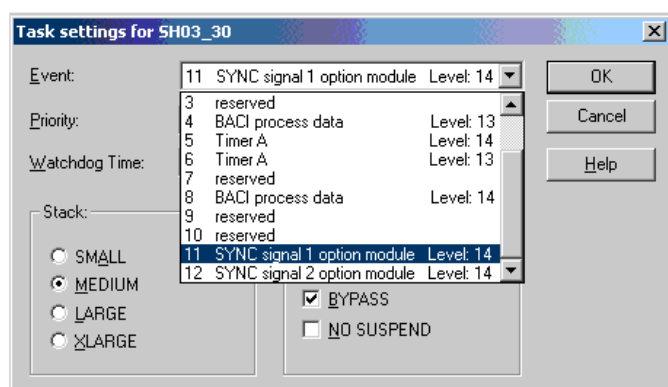


Figure 53: Carrying out process data communication

By means of block "BACI\_PD\_COMM" in a POE that was assigned event „11 SYNC signal 1 option module Level 14“, the system now cyclically transfers every 1000  $\mu$ s the target and actual values of the set parameters.

The b maXX controller must also be synchronized to the SYNC1 signal source.

For the necessary b maXX controller settings on WinBASS II page "Synchronization":

- Source for sync signal:  
"Use Sync 1 signal from the BACI" (i\_EVENT = INT#11)
- Choose a 1-ms Sync interval (=> the same time period must be connected at Input "u\_CYCLE\_TIME" in  $\mu$ s).
- Sync offset: -25.0  $\mu$ s.
- Sync tolerance: 2.0  $\mu$ s.

To offset the access instants of the controller and the PLC to the BACI, you must set a negative Sync-Offset of -25  $\mu$ s in the controller. This ensures that processing of the actual value channel in the controller is always completed and the current actual values from the controller are available to the PLC when it processes the interrupt that the BACI signal triggered.

## 4.6 BACI system description for the b maXX drive PLC

Triggering a BM4-O-IEI-01 option module via a SYNC1 signal: See the operating instructions of the BM4-O-IEI-01 option module.

### 4.6.10 Example D: Implementing simple serial RS485 communication via a CPU timer

Setting up a timer event task for a cyclical call in which the system is to process serial RS485 communication (via the 9-pin female Sub-D connector on the b maXX drive PLC BM4-O-PLC-01 interface X1). The function blocks for cyclical communication are placed in a POE that is assigned to this task. (For details on connection assignments and cables of the X1 interface, see the b maXX drive PLC BM4-O-PLC-01 Operating Instructions).

Cold boot/warm restart task:

UINT #0—terminal\_init[0] (\* protocol type: interface to ASCII-protocol \*)  
UINT #0—terminal\_init[1] (\* protocol mode: RS485-Pull-Up-resistor on (activ)  
RS485-terminator-resistors off (inactiv) \*)  
UINT #0—terminal\_init[2] (\* bidirectional communication activated \*)  
UINT #10—terminal\_init[3] (\* baudrate: 38400 Bd \*)  
UINT #2—terminal\_init[4] (\* parity: even \*)  
UINT #2—terminal\_init[5] (\* 8 databits \*)  
UINT #1—terminal\_init[6] (\* 1 Stopbit \*)

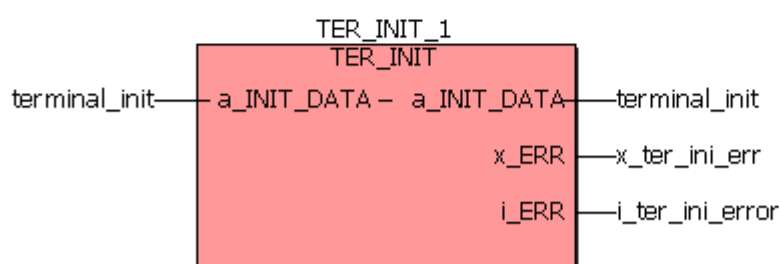


Figure 54: Initializing the serial port

The "TER\_INIT" block is called once during the initialization stage (cold boot/warm restart task). (For detailed information on "TER\_INIT" and the serial port parameters, see [►Description of the blocks for serial RS485 communication via the X1 interface \(9-pin female Sub-D connector\) on the b maXX drive PLC](#) ◀ from page 86 onward or the online description in SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher).

(\* Event #2: CPU timer 2 level 13;  
i\_PAR1 = #200 => interrupt interval time is 10 mSec \*)

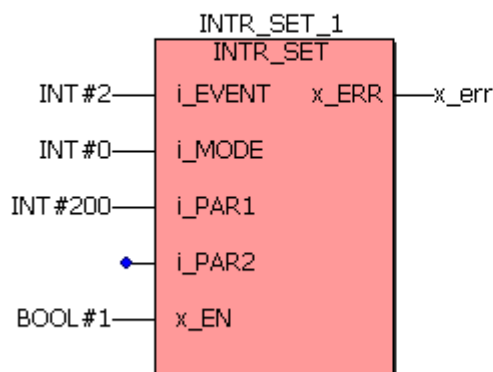


Figure 55: Set up a 10-ms bypass interrupt Level 13 in which the send and receive blocks are to be processed:

The block is called once during the initialization stage (cold boot/warm restart task).  
Serial communication in the CPU timer 2 BYPASS task:



Figure 56: Calling the send and receive blocks in the bypass interrupt (every 10 ms)

## 4.6 BACI system description for the b maXX drive PLC

---

### Brief description

(For detailed information on firmware blocks, see [►Description of the blocks for serial RS485 communication via the X1 interface \(9-pin female Sub-D connector\) on the b maXX drive PLC](#) ◀ from page 86 onward or the online descriptions in SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher):

### Reception block "TER\_R":

The receive job is activated via a rising edge at x\_RECEIVE\_INIT. Until all the characters have been received (number at "ud\_RECEIVE\_SIZE"), x\_BUSY stays TRUE and the system displays at output "ud\_CNT" the characters that have been received and stored in array "a\_BYTE1\_256\_rec".

### Send block "TER\_S":

The send job is activated via a rising edge at x\_SEND\_INIT: All the "ud\_SEND\_SIZE" characters in send buffer array "a\_BYTE1\_256\_send" are sent. As soon as the send job has been completely executed, the system resets x\_BUSY and sets x\_OK.

### 4.6.11 Description of the blocks for serial RS485 communication via the X1 interface (9-pin female Sub-D connector) on the b maXX drive PLC

---

For details on connection assignments and cables of the X1 interface, see the b maXX drive PLC Operating Instructions.



#### CAUTION

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

- property damage.

*Danger from: **short circuit** in the PLC or damage to the node connected to pin 2 of the X1 interface.*

The +5 V at Pin 2 of interface X1 on the BM4-O-PLC-01 Sub-D socket is intended only to supply external Baumüller-specific RS485/RS232 converters; you must not short-circuit or ring connect it with others.

**NOTE****RS485 interface**

RS485 interface X1 is galvanically separated and optically decoupled.

The RS485 is an enhancement of the RS422 for multi-sender operation (this means communication is carried out optionally using several transmitters per transfer path: in this connection, only one sender per transfer path may be active at the same time).

Normally, the differential individual wires are twisted together, which means that parasitic electromagnetic disturbances have an equal effect on both individual wires. Due to difference evaluation at the reception block, these disturbances can be suppressed (common-mode rejection). This means that twisting the wires increases the transmission performance.

Recommendation: In addition to the two-/four-wire cable, you can also use the ground wire; in this connection, both driver blocks (the send driver on one side of the connection and the receive driver on the other side) have the same absolute reference potential. This avoids exceeding of the common-mode input voltage range.

The ground wire must not be connected to the shielding at any point along the entire transmission path.

- Configuration options with the b maXX drive PLC via the software: Using the pull-up/pull-down and terminating resistors that are integrated on the PLC:

When using differential interfaces whose send drivers can be deactivated, it is often necessary to add pull-up/pull-down and/or terminating resistors to the connecting cables.

Using pull-up/pull-down resistors is intended to ensure that a defined potential is always connected to a differential reception unit even if no characters are currently being transferred and the if the associated send unit of the opposite communication partner is switching its drivers on a high-resistance basis (protocol-dependent).

If these types of resistors are not present, the line may float which results in sporadic transmission errors occurring like break errors, framing errors, parity errors, etc.

When using relatively long lines and high transfer rates, you can no longer neglect cable resistors. This means that you must connect terminating resistors to the differential cables at both ends of the bus on the transfer path. If you do not connect terminating resistors, reflections can occur that lead to reception errors. This means that using bus terminators improves the transmission performance.

In the case of network-like structures containing several nodes on one cable, this means that you must ensure that there are neither several pull-up/pull-down resistors in parallel on one cable nor too many terminating resistors (at the most the two at both ends of the bus).

To avoid the time and effort involved in directly integrating pull-up/pull-down and terminating resistors in the male connector or the cable, pull-up/pull-down and terminating resistors are present in the b maXX drive PLC and you can connect and disconnect them separately from one another using the applications software.

For an exact description of this procedure, refer to the description of FB "TER\_INIT" in PROPROG wt firmware library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher (see [Function block TER\\_INIT](#) < from page 88 onward) or their online description.

- Configuration option with the b maXX drive PLC: Two-wire/four-wire cables

Difference between two-wire and four-wire cables:

You can transmit characters on a full-duplex basis across a four-wire cable (it is recommended to use a ground wire too). Full-duplex means that characters can always be transmitted and received across separate channels in both directions at the same time. The 3964R<sup>®</sup> protocol is an example of a full-duplex protocol of this type.

By contrast, there are half-duplex protocols that can handle send and receive jobs via a common two-wire cable (it is recommended to use a ground wire too). In the case of a two-wire cable, the TxD+ wire is short circuited with the RxD+ wire at both ends of the connecting cable; this also applies to the TxD wire with the RxD wire. By contrast with a four-wire cable, this means that you need to lay two less wires, which results in a savings of material.

However, the missing individual wires must be "compensated" by an exact protocol sequence:

The protocol must ensure that characters are never transmitted in both directions at the same time and that only one send unit is ever active at any one time with the other partner send unit being switched off (high-resistance).

In the b maXX drive PLC, the following configuration options are available for this:

You can choose between two-wire and four-wire mode.

The difference is that in two-wire mode the system shuts down the send unit after all the characters have been transmitted. In this connection, you should note that shutting down makes the connecting cable high-resistance unless you programmed pull-up/pull-down resistors.

For an exact description of setting options, refer to the description of FB "TER\_INIT" in PROPROG wt II standard library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher or their online description.

### 4.6.12 Function block TER\_INIT

Firmware FB TER\_INIT initializes the terminal interface (serial port) of the b maXX option module with the necessary interface parameters, e.g. baud rate, parity, etc.

The FB TER\_INIT is contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Parameter input	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface



Parameter output	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
x_ERR	BOOL	Error bit
i_ERR	INT	Error word 0, 1, 2, 3, 4

**NOTE**

All the inputs must be assigned; otherwise, the system does not transfer a defined value!

**Type definition:**

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

**Description:**

Before communication can take place via the terminal interface (serial port) of the b maXX drive PLC option module, it must have been parameterized. You do this via connection a\_INIT\_DATA. The variable that is connected there contains the necessary interface parameters like the baud rate, parity, etc. Actual communication is carried out via the corresponding communications FBs (see below).

Since the terminal interface is initialized only once, TER\_INIT must be called using the cold boot and warm restart tasks.

**Input a\_INIT\_DATA:**

A variable of type UINT\_256\_BMARRAY is connected at input a\_INIT\_DATA. The individual entries of a\_INIT\_DATA have the following meanings:

a_INIT_DATA[Index]	Meaning
a_INIT_DATA[0]	Protocol 0: Any ASCII protocol 1: Interfacing to procedure 3964R <sup>®</sup> to data blocks
a_INIT_DATA[1]	Protocol's operating mode 0: Only RS485 pull-up resistors are active (on) 1: Only the RS485 terminating resistor is active (on) 2: The RS485 pull-up resistors and the RS485 terminating resistor are active (on) 3: The RS485 pull-up resistors and the RS485 terminating resistor are not active (off)
a_INIT_DATA[2]	RS485 operating mode 0: Four-wire RS485 operation 1: Two-wire RS485 operation <sup>1)</sup>

a_INIT_DATA[Index]	Meaning
a_INIT_DATA[3]	Baud rate 0: 50 bps <sup>1)</sup> 1: 110 bps <sup>1)</sup> 2: 150 bps <sup>1)</sup> 3: 300 bps <sup>1)</sup> 4: 600 bps <sup>1)</sup> 5: 1200 bps <sup>1)</sup> 6: 2400 bps <sup>1)</sup> 7: 4800 bps 8: 9600 bps 9: 19200 bps 10: 38400 bps
a_INIT_DATA[4]	Parity 0: No parity bit 1: Odd 2: Even
a_INIT_DATA[5]	Character length 0: Reserved 1: 7 bits per character 2: 8 bits per character
a_INIT_DATA[6]	Stop bit 0: 1 stop bit 1: 1 stop bit 2: 1 stop bit 3: 2 stop bits
a_INIT_DATA[7]	Reserved
a_INIT_DATA[8]	Max. number of communications attempts per message frame <sup>2)</sup>
a_INIT_DATA[9]	Priority <sup>2)</sup> 0: Low priority 1: High priority
a_INIT_DATA[10]	Character delay time (ZVZ) in ms <sup>2)</sup>
a_INIT_DATA[11]	Acknowledgment delay (QVZ) in ms <sup>2)</sup>
a_INIT_DATA[12]	Block delay (BVZ) in ms <sup>2)</sup>
a_INIT_DATA[13]	Reserved
...	...
a_INIT_DATA[255]	Reserved

<sup>1)</sup> Not for interfacing to procedure 3964R<sup>®</sup> to data blocks

<sup>2)</sup> Only for interfacing to procedure 3964R<sup>®</sup> to data blocks

Procedure 3964R<sup>®</sup> is a registered trademark of Siemens AG.

## a) ASCII protocol

Communication using protocol "Any ASCII protocol" is carried out via FBs TER\_S und TER\_R. Users can send n characters with this protocol to a communications partner and receive n characters from a communications partner.

The FBs TER\_S and TER\_R are contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Typical initialization values for the ASCII protocol are:

a_INIT_DATA[Index]	Meaning
a_INIT_DATA[0]	Protocol 0: Any ASCII protocol
a_INIT_DATA[1]	Protocol's operating mode 0: Only RS485 pull-up resistors are active (on) 1: Only the RS485 terminating resistor is active (on) 2: The RS485 pull-up resistors and the RS485 terminating resistor are active (on) 3: The RS485 pull-up resistors and the RS485 terminating resistor are not active (off)
a_INIT_DATA[2]	RS485 operating mode 0: Four-wire RS485 operation 1: Two-wire RS485 operation
a_INIT_DATA[3]	Baud rate 0: 50 bps 1: 110 bps 2: 150 bps 3: 300 bps 4: 600 bps 5: 1200 bps 6: 2400 bps 7: 4800 bps 8: 9600 bps 9: 19200 bps 10: 38400 bps
a_INIT_DATA[4]	Parity 0: No parity bit 1: Odd 2: Even
a_INIT_DATA[5]	Character length 1: 7 bits per character 2: 8 bits per character
a_INIT_DATA[6]	Stop bit 0: 1 stop bit 1: 1 stop bit 2: 1 stop bit 3: 2 stop bits
a_INIT_DATA[7]	Reserved 0

a_INIT_DATA[Index]	Meaning
...	...
a_INIT_DATA[255]	Reserved 0

### b) Interfacing to procedure 3964R®

Interfacing to procedure 3964R® to data blocks is carried out via firmware FBs T64\_RSYN und T64\_REC (are contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Typical initialization values for interfacing to procedure 3964R® to data blocks are:

a_INIT_DATA[Index]	Meaning
a_INIT_DATA[0]	Protocol 1: Interfacing to procedure 3964R® to data blocks
a_INIT_DATA[1]	Protocol's operating mode 0: Only RS485 pull-up resistors are active (on)
a_INIT_DATA[2]	RS485 operating mode 0: Four-wire RS485 operation
a_INIT_DATA[3]	Baud rate (Baud = bps) 7: 4800 bps 8: 9600 bps 9: 19200 bps
a_INIT_DATA[4]	Parity 2: Even
a_INIT_DATA[5]	Character length 2: 8 bits per character
a_INIT_DATA[6]	Stop bit 1: 1 stop bit
a_INIT_DATA[7]	Reserved 0
a_INIT_DATA[8]	Max. number of communications attempts per message frame 2: Max. 2 communications attempts per message frame
a_INIT_DATA[9]	Priority 0: Low priority
a_INIT_DATA[10]	Character delay time (ZVZ) in ms 100
a_INIT_DATA[11]	Acknowledgment delay (QVZ) in ms 500
a_INIT_DATA[12]	Block delay (BVZ) in ms 2000
a_INIT_DATA[13]	Reserved 0

a_INIT_DATA[Index]	Meaning
...	...
a_INIT_DATA[255]	Reserved 0

Error evaluation:

If the system detects faulty parameterization, it sets error bit x\_ERR and specifies the error in more detail by means of error number i\_ERR:

i_ERR	Error on serial port
0	No error
1 to 3	Reserved
4	The set baud rate is not possible

#### 4.6.13 Function block TER\_R

Firmware FB TER\_R receives characters via the terminal interface (serial port) of the b maXX drive PLC option module and stores the characters in an array.

The FB TER\_R is contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Parameter input	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_RECEIVE_BUFFER	ANY <sup>1)</sup>	Array to which characters are to be loaded
ud_RECEIVE_SIZE	UDINT	number of bytes that are to be received
x_RECEIVE_EN	BOOL	Enable/disable interface receive interrupt
x_RECEIVE_INIT	BOOL	Issue receive job

Parameter output	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_RECEIVE_BUFFER	ANY <sup>1)</sup>	Array to which characters are to be loaded
x_BUSY	BOOL	Indicates whether the job is active
ud_CNT	UDINT	Number of characters that have already been received
i_RECEIVE_ERR	INT	Display of a serial receive error
x_OK	BOOL	Indicates a correctly activated job

Parameter output	Data type	Description
x_ERR	BOOL	Error bit
i_ERR	INT	Error number

- <sup>1)</sup> You must connect a 256-byte array. Suitable data types are BYTE\_256\_BMARRAY (: ARRAY [0..255] OF BYTE) or DINT\_64\_BMARRAY (: ARRAY [0..63] OF DINT) from library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) and higher.



### NOTE

All the inputs must be assigned; otherwise, the system does not transfer a defined value!

### Type definition:

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

### Description:

Firmware FB TER\_R must be called cyclically in the user program.

For communication with firmware FBs TER\_S and TER\_R, you must set the hardware-specific interface parameters at firmware FB TER\_INIT (see the description of firmware FB TER\_INIT).

Using protocol "Any ASCII protocol", users can use firmware FB TER\_S to send n characters to a communications partner. Using firmware FB TER\_R, users can receive n characters from a communications partner.

### Input/output a\_INIT\_DATA:

The parameterized initialization array for the terminal interface is connected to a\_INIT\_DATA (see the description of firmware FB TER\_INIT). This means that the same global variable must be connected to a\_INIT\_DATA as to firmware FB TER\_INIT.

### Input/output a\_INIT\_BUFFER:

A (256-byte) array is connected to a\_RECEIVE\_BUFFER (see above). The system fetches the received data/characters (that were entered in the system-internal terminal interface receive buffer) and stores them in this array.

### Input ud\_RECEIVE\_SIZE:

At input ud\_SEND\_SIZE, the system outputs the number of characters that are stored from the system-internal receive buffer of the terminal interface to the array at input/output a\_RECEIVE\_BUFFER.

### Input x\_RECEIVE\_EN:

At activation of a job (rising edge at x\_RECEIVE\_INIT), the system can either enable the terminal interface receive interrupt (x\_RECEIVE\_EN = TRUE) or disable it

(x\_RECEIVE\_EN = FALSE). If x\_RECEIVE\_EN = FALSE, the system deletes all the previously received characters in the system-internal terminal interface receive buffer (regardless of the status of input x\_RECEIVE\_INIT). If the interrupt is disabled, no more characters are received in the system-internal receive buffer.

**Input x\_RECEIVE\_INIT:**

A rising edge at input x\_RECEIVE\_INIT activates a receive job for "ud\_RECEIVE\_SIZE" characters (from the system-internal terminal interface receive buffer).

**Output x\_BUSY:**

Output x\_BUSY is TRUE while a job is being carried out. If the system has fetched all the characters from the system-internal terminal interface receive buffer and written them to the array at input/output a\_RECEIVE\_BUFFER, the job is successful and x\_BUSY changes to FALSE.

**Output ud\_CNT:**

At output ud\_CNT, the system displays the number of characters that have been received up to this point in time (from the terminal interface receive buffer).

**Output x\_OK:**

Output x\_OK displays x\_OK = TRUE for one cycle to indicate whether a new job (rising edge at input x\_RECEIVE\_INIT) has been accepted. The system accepts a job even if a receive job is currently active (x\_BUSY = TRUE).

This means that the system immediately cancels the ongoing job and activates the new one.

**Error evaluation:**

Output i\_RECEIVE\_ERR displays receive errors.

i_RECEIVE_ERR	Error on serial port
0	No error
1	BREAK error
2	FRAME error
3	PARITY error
4	OVERRUN error
5	Internal receive error RxRDY

## 4.6 BACI system description for the b maXX drive PLC

Outputs **x\_ERR**, **i\_ERR**:

If the system detects faulty parameterization, it sets output **x\_ERR** = TRUE and specifies the error in more detail by means of the error number (output **i\_ERR**):

<b>i_ERR</b>	<b>Error on serial port</b>
0	No error
1 to 2	Reserved
3	Terminal interface was not initialized. Check calling of TER_INIT in the cold boot and warm restart task as well as all the a_INIT_DATA connections.

### 4.6.14 Function block TER\_S

Firmware FB TER\_S sends characters from an array across the terminal interface (serial port) of the b maXX drive PLC option module to a communications partner.

The FB TER\_S is contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

<b>Parameter input</b>	<b>Data type</b>	<b>Description</b>
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_SEND_BUFFER	ANY <sup>1)</sup>	Array from which characters are to be sent
ud_SEND_SIZE	UDINT	number of bytes that are to be sent
x_SEND_EN	BOOL	Enable/disable interface send interrupt
x_SEND_INIT	BOOL	Issue send job

<b>Parameter output</b>	<b>Data type</b>	<b>Description</b>
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_SEND_BUFFER	ANY <sup>1)</sup>	Array from which characters are to be sent
x_BUSY	BOOL	Indicates whether the job is active
ud_CNT	UDINT	Number of characters that have already been sent
i_SEND_ERR	INT	Display of serial send error (not currently used, i.e. always 0)
x_OK	BOOL	Indicates a correctly activated job
x_ERR	BOOL	Error bit
i_ERR	INT	Error number

<sup>1)</sup> You must connect a 256-byte array. Suitable data types are BYTE\_256\_BMARRAY (: ARRAY [0..255] OF BYTE) or DINT\_64\_BMARRAY (: ARRAY [0..63] OF DINT) from library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) and higher.



**NOTE**

All the inputs must be assigned; otherwise, the system does not transfer a defined value!

**Type definition:**

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

**Description:**

Firmware FB TER\_S must be called cyclically in the user program.

For communication with firmware FBs TER\_S and TER\_R, you must set the hardware-specific interface parameters at firmware FB TER\_INIT (see the description of firmware FB TER\_INIT).

Using protocol "Any ASCII protocol", users can use firmware FB TER\_S to send n characters to a communications partner. Using firmware FB TER\_R, users can receive n characters from a communications partner.

**Input/output a\_INIT\_DATA:**

The parameterized initialization array for the terminal interface is connected to a\_INIT\_DATA (see the description of firmware FB TER\_INIT). This means that the same global variable must be connected to a\_INIT\_DATA as to firmware FB TER\_INIT.

**Input/output a\_SEND\_BUFFER:**

A (256-byte) array is connected to a\_SEND\_BUFFER (see above). In this array, users store the data/characters that are to be sent.

**Input ud\_SEND\_SIZE:**

You state the number of characters to be sent at ud\_SEND\_SIZE.

**Input x\_SEND\_EN:**

At activation of a job (rising edge at x\_SEND\_INIT), the system can either enable the interface send interrupt (x\_SEND\_EN = TRUE) or disable it (x\_SEND\_EN = FALSE). If the interrupt is disabled, the system no longer processes a send job that is just running or a new one that has just been triggered. If RS485 operating mode is set to "two-wire" (see TER\_INIT), the RS485 send interface is additionally physically disabled (= high-resistance status of the send output for multipoint connections).

**Input x\_SEND\_INIT:**

A rising edge at input x\_SEND\_INIT activates a new job for sending "ud\_SEND\_SIZE" characters. The RS485 send interface is physically enabled.

### Output **x\_BUSY**:

Output **x\_BUSY** is TRUE while a job is being carried out. If the system has sent all the characters from array **a\_SEND\_BUFFER**, the job has been successfully completed and **x\_BUSY** changes to FALSE.

### Output **ud\_CNT**:

At output **ud\_CNT**, the system displays the number of characters that have been set up to this point in time.

### Output **i\_SEND\_ERR**:

This output indicates with **i\_SEND\_ERR** = INT#0 that no send error occurred. Other values of **i\_SEND\_ERR** are reserved.

### Output **x\_OK**:

Output **x\_OK** displays **x\_OK** = TRUE for one cycle to indicate whether a new job (rising edge at input **x\_SEND\_INIT**) has been accepted. The system cannot accept a job until the send procedure that was previously ongoing has been completed (**x\_BUSY** = FALSE) or the user has actively canceled it (**x\_SEND\_EN** = FALSE). While output **x\_BUSY** = TRUE, the system ignores send jobs (**x\_OK** stays FALSE).

### Error evaluation:

If the system detects faulty parameterization, it sets output **x\_ERR** = TRUE and specifies the error in more detail by means of the error number at output **i\_ERR**:

<b>i_ERR</b>	<b>Error on serial port</b>
0	No error
1 to 2	Reserved
3	Terminal interface was not initialized. Check calling of <b>TER_INIT</b> in the cold boot and warm restart task as well as all the <b>a_INIT_DATA</b> connections.

#### 4.6.15 Function block **T64\_RSYN**

Firmware FB **T64\_RSYN** mimics a data block of interfacing to procedure 3964R® to an array. Actual communication is via firmware FB **T64\_REC**.

Procedure 3964R® is a registered trademark of Siemens AG.

The FB **T64\_RSYN** is contained in the firmware library **SYSTEM2\_PLC\_20bd00** (PROPROG wt II) or **SYSTEM2\_PLC01\_30bd00** (ProProg wt III) or higher.

Parameter input	Data type	Description
a_DB_BMARRAY	ANY <sup>1)</sup>	Array that is to be enabled for communication with the interfacing to procedure 3964R® to data blocks.
i_DB_NR	INT	Number of the data block that is to be used to trigger the array of the communications partner.

Parameter output	Data type	Description
a_DB_BMARRAY	ANY <sup>1)</sup>	Array that is to be enabled for communication with the interfacing to procedure 3964R® to data blocks.
i_ERR	INT	Error number

<sup>1)</sup> You must connect a 256-byte array. Suitable data types are SINT\_256\_BMARRAY (: ARRAY [0..255] OF SINT) or DINT\_64\_BMARRAY (: ARRAY [0..63] OF DINT) from library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) and higher.

#### NOTE

All the inputs must be assigned; otherwise, the system does not transfer a defined value!



#### Description:

For interfacing to procedure 3964R® via data blocks, you must make the assignments once in the cold boot and warm restart tasks of the different data block numbers to the different arrays.

Up to 20 assignments are possible, i.e. you can call firmware FB T64\_RSYN a maximum of 20 times. At each call, the system enters the connected number and the associated array in a table that is managed on a system-internal basis. This means that in the cyclical program section the system only needs to call firmware FB T64\_REC that then handles actual communication using this internal table.

Input/output **a\_DB\_BMARRAY**:

At a\_DB\_BMARRAY, you connect a 256-byte variable of data type ARRAY (see above).

Input **i\_DB\_NR**:

Using i\_DB\_NR, you assign the connected array with a corresponding data block number.

## 4.6 BACI system description for the b maXX drive PLC

Error evaluation:

i_ERR	Error description
0	No error
1	You made too many assignments (a maximum of 20 are possible).
2	Missing entry of a_DB_BMARRAY.
3	No array type connected to a_DB_BMARRAY.

### 4.6.16 Function block T64\_REC

Using firmware FB T64\_REC, it is possible to exchange data between the terminal interface (serial port) on the b maXX PLC option module and another node. The FB T64\_REC is contained in the firmware library SYSTEM2\_PLC\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Interfacing to procedure 3964R<sup>®</sup> is used as the transfer protocol and is simplified to the extent that data can only be transferred via data blocks and no successor message frames are supported.

Procedure 3964R<sup>®</sup> is a registered trademark of Siemens AG.

In this connection the terminal interface on b maXX drive PLC option module is always a slave interfacing to procedure 3964R<sup>®</sup> to data blocks.

For interfacing to procedure 3964R<sup>®</sup> to data blocks, you must have made the appropriate assignments between the data block numbers used by communication and the individual array memory areas before using T64\_REC (see firmware FB T64\_RSYN).

Parameter input	Data type	Description
x_STRT	BOOL	Enable for exactly one communication cycle with a rising edge
x_AUTO	BOOL	If TRUE, all subsequent communication cycles are carried out automatically

Parameter output	Data type	Description
i_STAT	INT	Status message (-1, 0, 1, 2, 3, 4)
i_ERR	INT	Error number



#### NOTE

All the inputs must be assigned; otherwise, the system does not transfer a defined value!

**Description:**

Communication is carried out between two nodes (point-to-point connection). In this context, the connected partner is always the master that establishes communication. b maXX drive PLC option module interfacing to procedure 3964R<sup>®</sup> to data blocks represents the responding slave.

Transferring data using RS485 interfaces on the b maXX drive PLC option module is conditional on a four-wire cable (plus ground reference) being used, since the data must be transferred on a bidirectional basis. Apart from this, you must ensure that the differential lines are terminated with appropriate pull-up resistors (see firmware FB TER\_INIT).

Firmware FB T64\_REC should be called at regular intervals (10 ms or more as a rule of thumb). This means that it is sensible to implement the call in a cyclical task or to handle it in a timer event task if the program runtime of the cyclical task is too long.

To initialize the terminal interface, you use firmware FBs TER\_INIT and T64\_RSYN in the cold boot and warm restart task (see firmware FBs TER\_INIT and T64\_RSYN).

**Input x\_STRT:**

A rising edge at Input x\_STRT of firmware FB T64\_REC enables communication on a one off basis. However, the system disables transfer to the partner after the first communication cycle. You can use this function for test purposes to process exactly one communication cycle.

**Input x\_AUTO:**

If TRUE is connected here, the partner processes all the subsequent communications jobs. This also applies if an error is issued at output i\_STAT (= -1).

Normally, users should set this input permanently to TRUE after firmware FB T64\_REC is called for the first time to ensure that the system can process all the communications requests from the partner to the controller.

**Output i\_STAT:**

After starting communication using x\_AUTO = TRUE and with a rising edge at Input x\_STRT, communication is enabled. As soon as output i\_STAT = 2 is displayed, the system can process read and write jobs from the communications partner. The communications partner can then directly access the array contents via the assigned data block numbers (see firmware FB T64\_RSYN).

If a job from the communications partner has been received correctly, output i\_STAT changes from 2 to 3 and the b maXX drive PLC option module processes the job. The next time firmware FB T64\_REC is called, the system sends the answer to the communications partner and output i\_STAT changes from 3 to 4. After the send procedure is completed, output i\_STAT is automatically set = 2 (i.e. no rising edge is necessary at Input x\_STRT). This means that the system can process a new job from the communications partner to the b maXX drive PLC option module. If a receive error occurs during communication, the system displays error status "-1" at output i\_STAT.

## 4.6 BACI system description for the b maXX drive PLC

<b>i_STAT</b>	<b>Internal status of communication of interfacing to procedure 3964R® to data blocks</b>
0	Firmware FB T64_REC is not active or initialization has not been carried out
1	RECEIVE function has been triggered
2	From now on, jobs can be received from the communications partner
3	Message frame received, job will be carried out; initiate "Send response"
4	Wait until response has been sent: Then change from status 1 to status 2
-1	Error occurred! Cause of error specified in more detail in i_ERR

Error evaluation:

If a receive error occurs during communication, the system displays error status "-1" at output i\_STAT with the exact cause of the error being displayed at output i\_ERR. The next time firmware FB T64\_RSYN is called, the system switches automatically to receive mode i\_STAT = 2.

<b>i_ERR</b>	<b>Error number</b>
0	No error
1	Reserved
2	Terminal interface not initialized (see firmware FB TER_INIT)
3	BREAK error terminal interface
4	FRAME error terminal interface
5	PARITY error terminal interface
6	OVERRUN error terminal interface
7 to 15	Reserved
16	Maximum number of communications attempts reached
17	Wrong END identifier received after block transfer
18	Character delay time (ZVZ) exceeded
19	Block delay time (BVZ) exceeded
20	Wrong character as grouping mark (no STX)
21 to 49	Reserved
50	Received message frame of communications partner too short (< 10 characters)
51	Neither "SEND" nor "FETCH" job implemented or source/target area requested by communications partner not supported
52	Access range not released by user (see firmware FB T64_RSYN)
From 53 onward	Reserved

#### 4.6.17 Function block TER\_USS

FB TER\_USS is used to implement communication with interfacing to the USS protocol®. Connection is via the terminal interface (serial RS485 X1 port) of the b maXX drive PLC.

The b maXX drive PLC is the sole master and it can communicate with all the slaves that can process the USS protocol®.



#### NOTE

Refer to the "Baumotronic Communications Software" description for detailed information about the exact way of functioning, the message frame structure and the meanings of the USS protocol® parameters.



#### NOTE

FB TER\_USS is contained in the standard library SYSTEM1\_PLC\_20bd00 (PROPROP wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher and uses firmware FB USS\_SR from firmware library SYSTEM2\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher. This library also contains firmware FB TER\_INIT that is needed for initializing the terminal interface. Apart from this, data types from library BM\_TYPES\_20bd03 (PROPROP wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or higher are used.

The USS protocol® is a registered trademark of Siemens AG.

Parameter input	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Array containing the interface parameters
ud_MSEC_DELAY_MAX	UDINT	Maximum delay time for the characters to be received
us_MSG_MIN	USINT	Minimum number of communications attempts
us_MSG_MAX	USINT	Maximum number of communications attempts
us_ADR	USINT 0 to 31	Address of the USS node
x_BROADCAST	BOOL	Broadcast message frame to all nodes
x_MIRROR	BOOL	Mirror message frame
us_SIZE_PIV	USINT 0, 3, 4	Number of PKWs
us_JOB_IDENT	USINT 0 to 4	Job ID
u_PARA_IDENT_WRITE	UINT	Parameter number
ud_PARA_VALUE_WRITE	UDINT	Write parameter value
u_INDEX_WRITE	UINT	Write index number

## 4.6 BACI system description for the b maXX drive PLC

Parameter input	Data type	Description
us_SIZE_PD	USINT 0 to 3	Number of process data
w_CONTROLWORD	WORD	Control word for process data
ud_PROC_DATA_WRITE	UDINT	Write target value of process data
x_COM_INIT	BOOL	Start bit for message frame (rising edge)
x_COM_CYCL	BOOL	Connection for cyclical data transmission

Parameter output	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Array containing the interface parameters
x_BUSY	BOOL	Display bit for ongoing communication
x_VALID	BOOL	Display bit for valid receive data
us_REPLY_IDENT	USINT	Received job ID
u_PARA_IDENT_READ	UINT	Received parameter number
u_INDEX_READ	UINT	Read index number
ud_PARA_VALUE_READ	UDINT	Received parameter value
w_STATUSWORD	WORD	Received status word of process data
ud_PROC_DATA_READ	UDINT	Received actual value of process data
si_SER_ERR	SINT	Error number of serial port
si_MSG_ERR	SINT	Error number of message frame
si_PARA_ERR	SINT	Error number of job
x_ERR	BOOL	Global error bit

### NOTE



All the inputs must be assigned; otherwise, the system does not transfer a defined value!

### Description:

FB TER\_USS should be called cyclically (as a rule of thumb every 10 ms or more). This means that it is sensible to implement calling of FB TER\_USS in a cyclical task or to handle it in a timer event task if program runtime of the cyclical task is too long or too irregular due to the other program sections that it contains.

Requirements data about the cPKW range can be transferred between the master (b maXX drive PLC) and its connected USS slaves via the link to the USS protocol®.

In the USS protocol®, process data is transferred ("fast target/actual value channel") via the cPZD range.



At a `_INIT_DATA` you must connect the same (globally declared) array as at firmware `FB TER_INIT`, since the system accesses this array during communication.

USS communication is started in the case of a rising edge at input `x_COM_INIT`. If you want new communication cycles to be started automatically, you must set input `x_COM_CYCL = TRUE`.

Output `x_BUSY` stays set until master-slave communication to the node has been completed. The system displays that communication has been completed with `x_VALID = TRUE`. This stays `TRUE` for one program cycle after a falling `x_BUSY` edge.

At input `ud_MSEC_DELAY_MAX`, you connect the maximum timeout in ms by which master-slave communication must have been completed.

If this time is exceeded after communication has been triggered, the system issues a timeout error message. Normally, you should connect a value of approximately 220 ms.

If no error occurs during communication, the system writes with `x_VALID = TRUE` to the outputs with the received values (`w_STATUSWORD`, `ud_PROC_DATA_READ`, `us_REPLY_IDENT`, `u_PARA_IDENT_READ`, `u_INDEX_READ`, `ud_PARA_VALUE_READ`).

Before parameter output, the system checks the sent and received parameter numbers `u_PARA_IDENT_WRITE` and `u_PARA_IDENT_READ`.

In this connection, the system requests the message frame exactly `us_MSG_MIN` times until parameters are output.

If there is an error (e.g. a timeout), the system tries to set up the message frame `us_MSG_MAX` times before `x_VALID` becomes `TRUE` and an error is displayed. Depending on the error that occurred, the system can evaluate the following outputs:

Parameter and receive error, timeout error:

The system outputs the `x_ERR`-Bit and the error numbers (`si_SER_ERR`, `si_MSG_ERR`, `si_PARA_ERR`) the other outputs retain their previous instance values!

Job error:

The system rewrites bits `x_ERR`, `si_SER_ERR`, `si_MSG_ERR` status, job error numbers `si_PARA_ERR`, `us_REPLY_IDENT`, `u_PARA_IDENT_READ`; `u_INDEX_READ` und `ud_PARA_VALUE_READ` retain their previous instance values!

The process data (`w_STATUSWORD` and `ud_PROC_DATA_READ`) is output each time a message frame is set up successfully, even if another parameter job error occurred. This means that process data output is independent of `us_MSG_MIN`!

Inputs `us_MSG_MIN` and `us_MSG_MAX` specify the minimum number of times (`us_MSG_MIN`) and the maximum number of times (`us_MSG_MAX`, e.g. in the case of a timeout message) communication is allowed to take place before there is a response at the output due to the data returned by the slave or to an error message.

Normally, users should use the following initialization values:

```
us_MSG_MIN = 1;  
us_MSG_MAX = 2;
```

Organization of the contents in the message frame structure of the USS protocol<sup>®</sup> is specified by the PKW and PZD quantity. The settings must be identical for all the nodes

## 4.6 BACI system description for the b maXX drive PLC

in the USS ring so that the system can correctly interpret the contents of the exchanged message frame.

<b>us_SIZE_PIV</b>	<b>PKW quantity: Subdivision of requirements data in the USS protocol</b>
4	Doubleword parameter (32-bit types)
3	Word parameter (16-bit types)
0	Message frame contains no requirements data!

<b>us_SIZE_PD</b>	<b>PZD quantity: Subdivision of process data in the USS protocol</b>
3	Control/status word and doubleword target/actual values (32-bit types)
2	Control/status word and word target/actual values (16-bit types)
1	Control/status word only
0	Message frame contains no process data!

<b>us_JOB_IDENT</b>	<b>Job ID (Master -&gt; Slave)</b>
0	No job
1	Read parameter value (answer in ud_PARA_VALUE_READ)
2	Write parameter value ud_PARA_VALUE_WRITE of type word (16-bit)
3	Write parameter value ud_PARA_VALUE_WRITE of type doubleword (32-bit)
4	Read an element from the parameter description (which on is in u_INDEX_WRITE)
5 to ...	(Reserved)

<b>us_REPLY_IDENT</b>	<b>Answer ID (Slave -&gt; Master)</b>
0	No job
1	Parameter word (16-bit) was written (us_JOB_IDENT = 2) or read (us_JOB_IDENT = 1) (ud_PARA_VALUE_READ contains the transferred value)
2	Parameter doubleword (32-bit) was written (us_JOB_IDENT = 3) or read (us_JOB_IDENT = 1) (ud_PARA_VALUE_READ contains the transferred value)
3	Element from parameter description was read (answer is in ud_PARA_VALUE_READ)
4 to 6	(Reserved)
7	Parameter error: Job not executable! (For error identification, see si_PARA_ERR)
8 to ...	(Reserved)

**Error evaluation:**

<b>si_SER_ERR</b>	<b>Error on serial port</b>
0	No error
1	(Reserved)
2	Wrong serial initialization array a_INIT_DATA connected
3	Interface not initialized
4 to 9	(Reserved)
10	Break on receiving
11	Frame error on receiving
12	Parity error on receiving
13.	Overrun error on receiving

<b>si_MSG_ERR</b>	<b>Message frame error on sending/receiving</b>
0	No error
1	The us_SIZE_PIV range that is connected to the FB is not supported
2	The us_SIZE_PD range that is connected to the FB is not supported
3	us_MSG_MIN > us_MSG_MAX or us_MSG_MIN or us_MSG_MAX = 0
4	Invalid address connected to us_ADR (only 0..31 are permitted)
5	Sent and received addresses are different
6	Sent and received parameter numbers u_PARA_IDENT_WRITE / u_PARA_IDENT_READ are different!
7	us_SIZE_PIV range wrong
8	us_SIZE_PD range wrong
9	STX character not received!
10	Received and expected length indication (us_SIZE_PIV/us_SIZE_PD) are different!
11	BCC checksum error
12	Timeout error on receiving

<b>si_PARA_ERR</b>	<b>Parameter error</b>
0	No error
-1	Parameter number is not supported (impermissible u_PARA_IDENT_WRITE)
1	Parameter cannot be changed
2	MIN/MAX limitation
3	Faulty index u_INDEX_WRITE

## 4.7 Code runtimes

si_PARA_ERR	Parameter error
4	No array type
5	Wrong data type
6	Setting not allowed
7	Description element cannot be changed
8 to 100	(Reserved)
101	Undetermined error
102	Service not implemented
103	Parameter format too large for PKW range
104	PBE element not present

## 4.7 Code runtimes

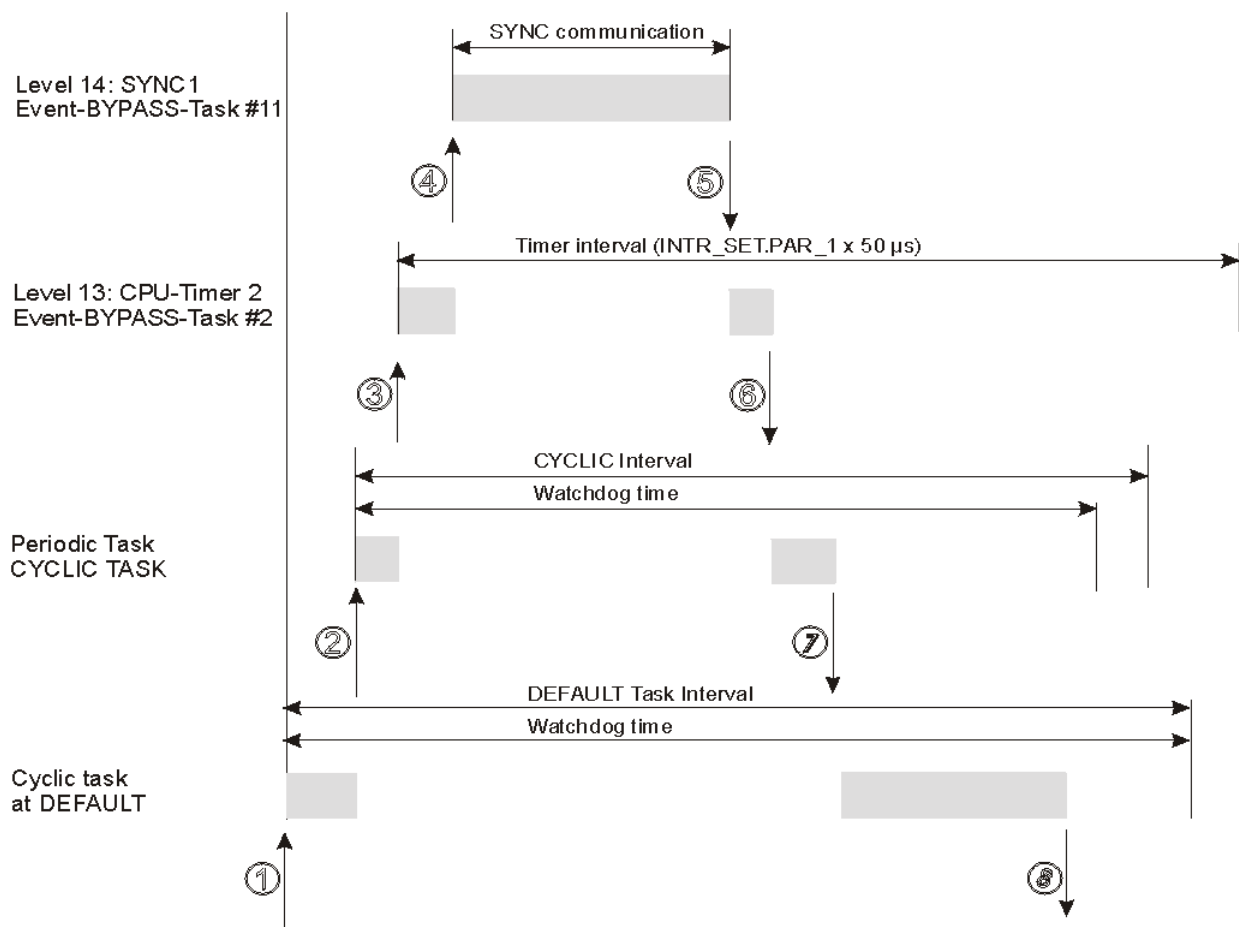


Figure 57: Distribution of code runtimes to resource BM4\_O\_PLCO1

**NOTE**

All the events must be created as tasks with the "BYPASS" property.

A higher-priority event task interrupts a lower-priority one; this means that you should always use timer event tasks that could run at the same time as synchronous bus systems with the lower level 13 so that they can be interrupted.

In the case of cyclical tasks that are not BYPASS task, users must enter in the task's "Settings" the cyclical sampling time (the "Interval" setting) and a monitoring time (the "Watchdog" setting). The runtime system monitors these cyclical tasks and the default task for the watchdog during operation.

In this connection, you should note that the set watchdog does not take into account the pure runtime of this task; rather, it also includes the total of the maximum individual runtimes that can interrupt this task.

If the monitoring time is set too low, the system issues PLC error "Watchdog in Task 'x' exceeded!" to inform users of the system overload that has occurred and the controller enters the safe "STOP" status at the same time.

The watchdogs in BYPASS task are ignored.

**Further factors influencing the watchdogs in the cyclical non-BYPASS tasks and the default task:**

You should allow certain time reserves in the monitoring times to allow for other system functions that are running in the background, e.g task call shells and for functions that are called for online representations of variables in the watch list and/or in the global variable worksheets and which increase the system load.

Apart from this, users can activate specific monitoring functions that are active during runtime:

These include the settings in the resource for "Index checking for PLC" and "Stack inspection for PLC". You can possibly reset these settings after commissioning is completed to save runtime resources.

For the BYPASS tasks, you can exactly determine the actual system loading during operation from a worst case point of view using the Time\_Measure blocks from library SYSTEM1\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher. Using the FBs, you can acquire runtimes of a maximum of 10 ms at very high resolutions.

If you have to measure relatively large time intervals in the lower-priority cyclical tasks, it is possible to form a continuous global counter value in a BYPASS interrupt that runs with a fixed time base. This timer value can then be evaluated directly in the cyclical tasks by subtraction of the timer value. (For information on setting up a cyclical CPU timer as a BYPASS task, see the description of FB "INTR\_SET" in firmware library SYSTEM2\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

It is possible to deactivate a BYPASS event at runtime using FB "INTR\_SET" of firmware library SYSTEM2\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher using x\_EN = FALSE.

### Avoiding system overloading:

In the case of watchdog overshoots, you must set the (interval or watchdog) settings upward and/or minimize the runtimes in the BYPASS interrupts. Do this either by dividing the program in the interrupt itself (i.e. alternate program section executions per interrupt) or by relocating to lower-priority non-BYPASS tasks with longer call intervals or by locating program sections directly in the default task.

If you do not want to carry out runtime monitoring in cyclical non-BYPASS task, you can also set up a watchdog that is higher than the interval that is set for this task.

#### 4.7.1 Function block TIME\_MEASURE\_START

You can use this function block for TIME\_MEASURE together with function block TIME\_MEASURE\_END to determine code runtimes and call times for tasks.

The FB TIME\_MEASURE\_START is contained in the standard library SYSTEM1\_PLC\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

Output parameter	Data type	Description
ud_CNT_START	UDINT	Current system timer value in system units

### Description

FB TIME\_MEASURE\_START is used together with FB TIME\_MEASURE\_END for time measurements on the b maXX drive PLC. The system determines the runtime of the program code between calling of FB TIME\_MEASURE\_START (start of time measurement) and calling FB TIME\_MEASURE\_END (end of time measurement).

FB TIME\_MEASURE\_START reads the current value in system units from a system timer and outputs it at ud\_CNT\_START. For its part, FB TIME\_MEASURE\_END reads the current system timer value in system units and generates the difference to the value read by FB TIME\_MEASURE\_START. This means that at both FBs the same variable must be connected to ud\_CNT\_START.



### NOTE

Correct time measurement is only possible if both FBs are used in an event task with the highest priority. Runtimes up to a maximum of 10 ms can be determined. In the case of program code with runtimes longer than 10 ms, no correct values will be returned.

#### 4.7.2 Function block TIME\_MEASURE\_END

You can use this function block for TIME\_MEASURE together with function block TIME\_MEASURE\_START to determine code runtimes and call times for tasks.

The FB TIME\_MEASURE\_END is contained in the standard library SYSTEM1\_PLC\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher.

Input parameter	Data type	Description
ud_CNT_START	UDINT	System timer value from FB TIME_MEASURE_START

Output parameter	Data type	Description
ud_CNT_END	UDINT	Current system timer value in system units
di_DIFF_in_counts	DINT	Difference between ud_CNT_END and ud_CNT_START
di_DIFF_in_mysec	DINT	Time difference in ms

#### Description

FB TIME\_MEASURE\_START is used together with FB TIME\_MEASURE\_END for time measurements on the b maXX drive PLC. The system determines the runtime of the program code between calling of FB TIME\_MEASURE\_START (start of time measurement) and calling FB TIME\_MEASURE\_END (end of time measurement).

FB TIME\_MEASURE\_START reads the current value in system units from a system timer and outputs it at ud\_CNT\_START. For its part, FB TIME\_MEASURE\_END reads the current system timer value in system units and generates the difference to the value read by FB TIME\_MEASURE\_START. This means that at both FBs the same variable must be connected to ud\_CNT\_START.

The system outputs the time difference to di\_DIFF\_in\_mysec; the difference of the system timer values in system units is available in di\_DIFF\_in\_counts. The system timer value that FB TIME\_MEASURE\_END reads out can be seen at ud\_CNT\_END.

To determine a task's call time, you place the two FBs at the start of a POE: in this connection, FB TIME\_MEASURE\_END must be placed before FB TIME\_MEASURE\_START. You interconnect to variables in the same way as with code runtime measurement. You can read the call time at di\_DIFF\_in\_mysec.



#### NOTE

Correct time measurement is only possible if both FBs are used in an event task with the highest priority. Runtimes up to a maximum of 10 ms can be determined. In the case of program code with runtimes longer than 10 ms, no correct values will be returned.

### 4.8 Practical information

#### 4.8.1 PLC error "Watchdog in task 'x' exceeded!"

This PLC error message is generated by the runtime system to inform users of the following system overloading:

In the case of cyclical tasks that are not BYPASS task, users must enter in the task's "Settings" the cyclical sampling time (the "Interval" setting) and a monitoring time (the "Watchdog" setting). The runtime system monitors these tasks during operation. If the runtime system detects a violation of the selected settings, the PLC enters the "Stop" status and issues the "Watchdog in Task 'x' exceeded!" error message.

##### **Possible reasons for a violation:**

If users employ BYPASS tasks that are heavily loaded (e.g. a pure interrupt runtime of 1.5 ms with a 2-ms sampling time) and which interrupt each other (e.g. another BYPASS timer task with a 300-ms runtime), this places an additional load on the CPU computing time. In a worst case runtime scenario, this can lead to the PLC errors mentioned above in a low-priority cyclical task (i.e. a non-BYPASS task) that is monitored by the operating system.

##### **What you should count on:**

This means that when setting up the monitoring time, you should not just take into account the pure runtime of the task to be set up; rather, you must also consider time reserves for other system functions that are running, e.g. for the task call shells that are running in the background and for online representations of variables in the watch lists and/or in the global variable worksheets. You should also not forget that users can activate specific monitoring functions that are active at runtime (e.g. the settings in the resource for "Index checking for PLC" and "Stack inspection for PLC") and may be reset after commissioning is completed to save resources.

For the BYPASS tasks, you can exactly determine the actual system loading during operation from a worst case point of view using the Time\_Measure blocks from library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or higher. Using the FBs, you can acquire runtimes of a maximum of 10 ms at very high resolutions.

If you have to measure relatively large time intervals in the lower-priority cyclical tasks, it is possible to form a continuous global counter value in a BYPASS interrupt that runs with a fixed time base. This timer value can then be evaluated directly in the cyclical tasks by subtraction of the timer value. (For information on setting up a cyclical CPU timer as a BYPASS task, see the description of FB "INTR\_SET" in firmware library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher).

##### **Avoiding overloading:**

In the case of watchdog overshoots, you must set the (interval or watchdog) settings upward and/or minimize the runtimes in the BYPASS interrupts. Do this either by dividing the program in the interrupt itself (i.e. alternate program section executions per interrupt)



or by relocating to lower-priority non-BYPASS tasks with longer call intervals or by locating program sections directly in the default task.



# INTERNAL COMMUNICATIONS INTERFACE TO THE (BACI) CONTROLLER

## 5.1 BACI (BAumüller Component Interface) general

---

The BACI is an internal communications interface for exchanging data between the b maXX controller and the b maXX drive PLC.

When carrying out communication, a differentiation is made between process data communication and requirements data communication.

Process data communication comprises reading and writing time-critical set values and actual values and the status and control word in a definable time raster.

Service data communication comprises reading and writing non-time-critical parameters of the controller.

Several ways are available for exchanging data between the b maXX controller and the b maXX drive PLC.

- ProMaster, ProProg wt III and Motion Control

The configuration of the data exchange in ProMaster is simply and fast. The programming of the machine function in ProProg wt III with the Motion Control function blocks is simply and fast.

Example projects:

ProMaster project:                      Example\_1\_3\_2.bmxml

IEC project in ProProg wt III:      Example\_BM4\_O\_ECT02\_MA\_2.mwt

See [▶ProMaster, ProProg wt III and Motion Control ◀](#) from page 116 onward.

- PROPROP wt II

Using the IEC 61131-3 programming system PROPROP wt II to program the data exchange (BACI communication) and the machine function PROPROP wt II.

See [▶PROPROP wt II ◀](#) from page 144 onward.

## 5.2 ProMaster, ProProg wt III and Motion Control

---

Network variables are created per default for Motion Control at creating a project in ProMaster.

In an IEC project the Motion Control function blocks read or write these default network variables. The complete machine function in an IEC project can be programmed via the Motion Control function blocks.

Additional network variables can be created during configuration of the data exchange in ProMaster project.

In a ProProg wt III IEC project these additional network variables can be read or written.

The interval in which the (process) data will be exchanged between b maXX drive PLC and b maXX controller will be configured in ProMaster.

If writing or reading of service data is required, the respective Motion Control function blocks are used.

### 5.2.1 Requirements

---

The requirements are

- a PC with
  - the ProMaster engineering framework
  - the IEC 61131-3 programming system ProProg wt III with the firmware libraries
    - Bit\_UTIL\_30bd00
    - OmegaOS\_30bd00
    - SYSTEM2\_PLC01\_30bd00
    - MC\_SYS\_30bd00
  - and the libraries
    - BM\_TYPES\_30bd00
    - SYSTEM1\_PLC01\_30bd00
    - MOTION\_TYPES\_30bd02
    - MOTION\_CONTROL\_30bd03
    - MOTION\_MULTI\_AXIS\_30bd03 (if multi-axis motion control is used)
- Physical commissioning of the components.  
 In the following example an EtherCAT network will be created. In our example, the components are the EtherCAT master
  - b maXX 4400 with
    - b maXX drive PLC (BM4-O-PLC-01)
    - EtherCAT master option module (BM4-O-ECT-02-...)
  - and the EtherCAT slave
    - b maXX 4400 with
    - EtherCAT slave option module (BM4-O-ECT-01-...)

For this purpose, see the respective operating instructions.

### 5.2.2 Procedure to be carried out

To use the b maXX drive PLC and the b maXX controller for Motion Control, the following steps must be carried out.

- 1 Commissioning of the b maXX 4400 with b maXX controller and b maXX drive PLC
- 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  
(a small EtherCAT network will be created in the following example)
- 4 Configuration of EtherCAT communication and subsequent downloading to the EtherCAT master  
(with testing of the EtherCAT network with this configuration if appropriate)
- 5 Configuration of the BACI communication (local axis)
- 6 Configuration of the PLC with linking of the IEC project and ProMaster, configuration of the cam data
- 7 Possible programming of the IEC 61131-3 project for ProProg wt III (application) and subsequent downloading to the b maXX drive PLC
- 8 Operation of the application

Naturally, you can also begin with Step 2 and not perform physical commissioning (Step 1) until just prior to Step 6.

The ProMaster project **Example\_1\_3\_2.bmxml** and the IEC 61131-3 project (ProProg wt III project) **Example\_BM4\_ECT02\_MA\_2.mwt/.zwt** are created here.

### 5.2.3 Commissioning of the b maXX 4400 with b maXX controller and b maXX drive PLC

For details on commissioning the b maXX 4400 with b maXX controller and the b maXX drive PLC option module, please refer to the respective operating instructions.

For details on commissioning the EtherCAT network, please refer to the respective operating instructions for the EtherCAT master for b maXX drive PLC option module and the operating instructions of the other EtherCAT network nodes.

### 5.2.4 Creating of an IEC project with ProProg wt III

The following explains how a motion control IEC project is created.

Multi-axis motion control is set in the ProMaster default settings for motion control. For this reason, it makes sense 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 project via the "File\Open Project / Unpack Project..." menu.

In our example, we use the IEC template project "Tmpl\_PLC01\_MA\_2\_0104.zwt" (for BM4\_O\_PLC01).

This project is found in the directory <ProMaster installation directory>\projects\IEC templates, e.g. in the ProMaster standard installation directory C:\Baumuller\ProMasterNET\projects\IEC-Templates.

The IEC template project "Tmpl\_PLC01\_MA\_2\_0104.zwt" is unpacked as "Example\_BM4\_O\_ECT02\_MA\_2.mwt".

The template contains the libraries

- Bit\_UTIL\_30bd00
- BM\_TYPES\_30bd01
- MC\_SYS\_30bd00
- MOTION\_TYPES\_30bd02
- MOTION\_CONTROL\_30bd03
- MOTION\_MULTI\_AXIS\_30bd03

Save the project. In our example, we use the project name "Example\_BM4\_O\_ECT02\_MA\_2.mwt".

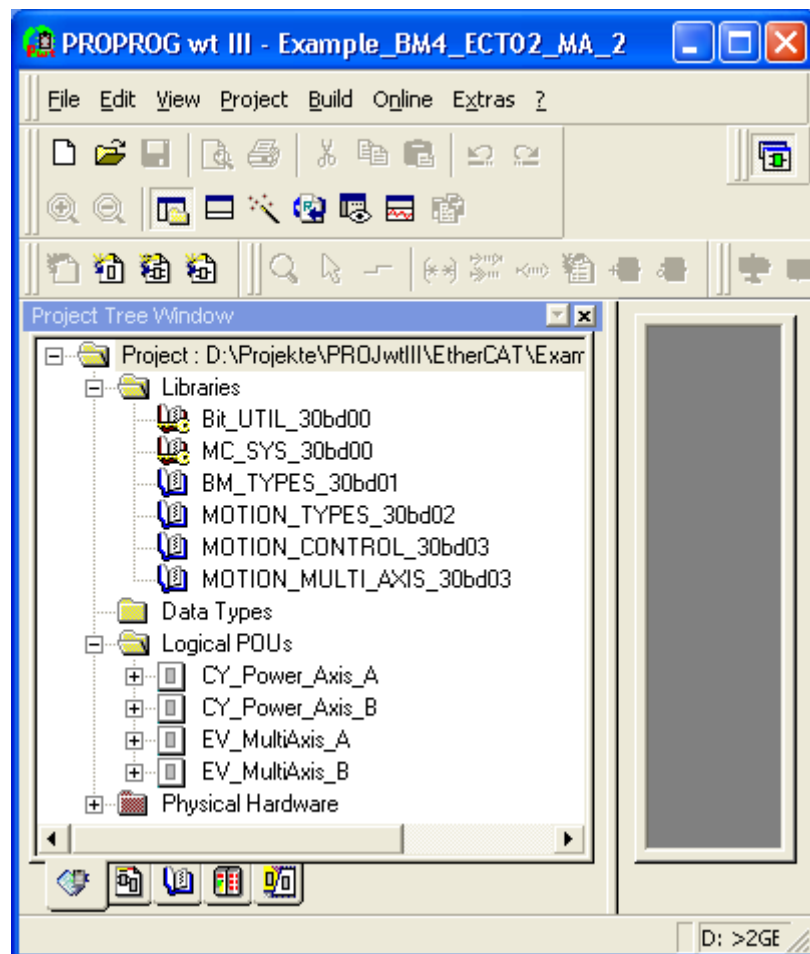


Figure 58: ProProg wt III - Creating the project "Example\_BM4\_O\_ECT02\_MA\_2.mwt"

The communication settings must now be configured. The serial interface COM1 is set by default. If you would like to use Ethernet, see Section [Communication and Connection](#) from page 31 onward.

When using an IEC project in ProMaster, it is absolutely necessary that the "Create boot project when compiling" checkbox in the "Resource Settings" window (select resource, choose context menu „Settings“) is activated for ProProg wt III. This creates the "boot-file.pro" file, which is required for the download to the b maXX drive PLC.

**NOTE**

The "Create boot project when compiling" checkbox under "Resource Settings" must be activated in ProProg wt III.

The project "Example\_BM4\_O\_ECT02\_MA\_2.mwt" is linked to the ProMaster project later on in Section [►Configuration of the PLC ◀](#) from page 130 onward.

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

### 5.2.5 Creation of a machine configuration in ProMaster

This section explains how a project is created in ProMaster and how a machine configuration is created.

Open ProMaster.

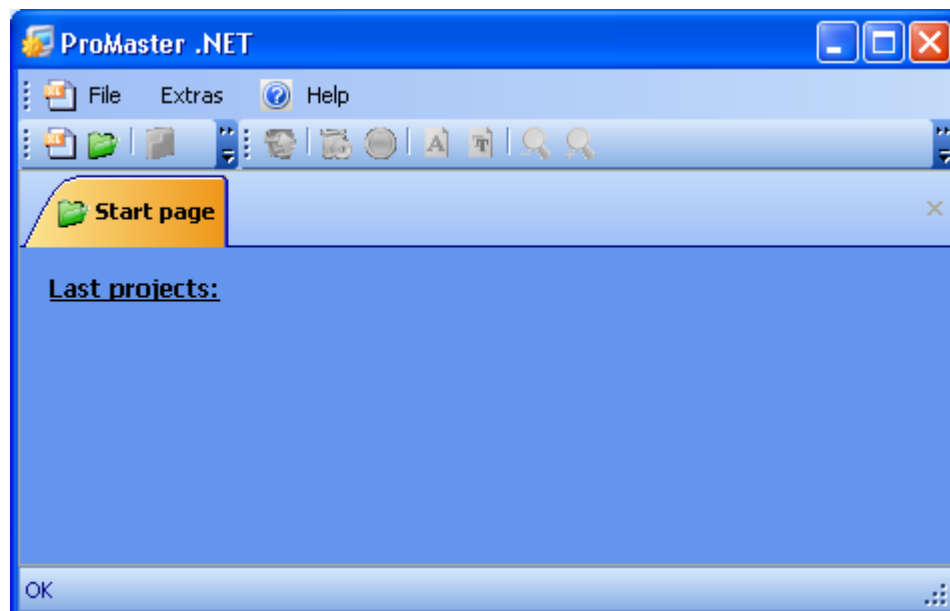


Figure 59: ProMaster with no projects

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

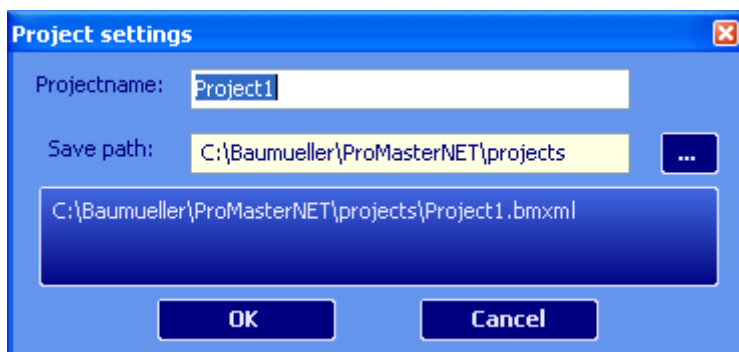


Figure 60: ProMaster - Project name

Enter the name (in the edit box) and the save location (via path selection) of the project. We use the name "Example\_1\_3\_2.bmxml" for our example project.

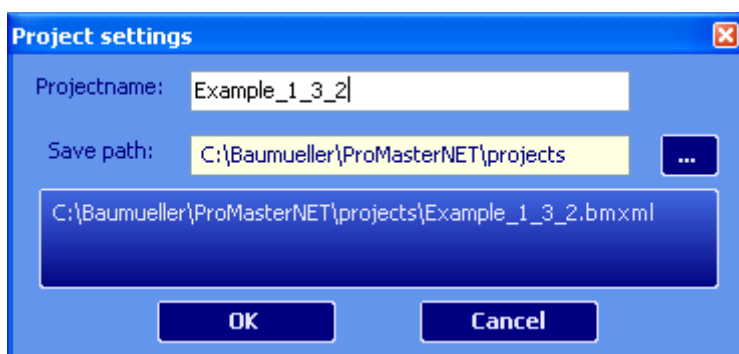


Figure 61: ProMaster - Issuing a project name

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



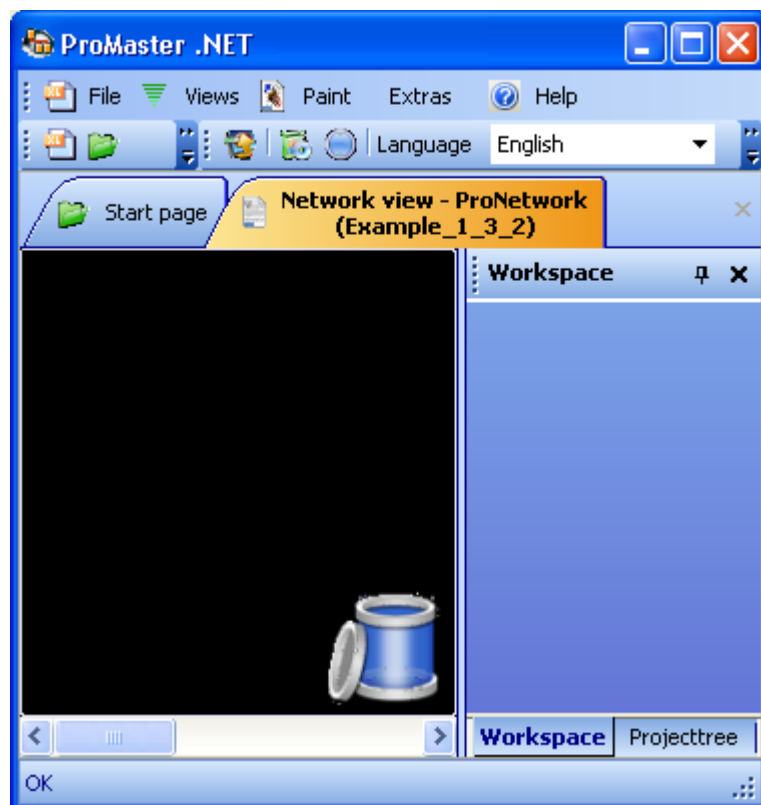


Figure 62: ProMaster - Issuing a project name

Now open the Baumüller catalog via the menu "Views\Catalog". The Baumüller catalog contains the devices, bus system and components provided by Baumüller by default.

In our example, we use

- the devices
  - "b maXX 4000 EtherCAT master (2-row)" (from the group "b maXX 4000 drive");  
b maXX 4000 with two slots,  
Ethernet with EtherCAT master option module in slot G,  
b maXX drive PLC in slot H
  - "b maXX 4000 EtherCAT slave (2-row)" (from the group "b maXX 4000 drive");  
b maXX 4000 with two slots,  
Ethernet with EtherCAT slave option module in slot G
- the bus system
  - „EtherCAT bus“ is created automatically

Drag the "b maXX 4000 EtherCAT master (2-row)" device from the Baumüller catalog and drop it in the ProMaster network view. The bus system is connected automatically. Now drag the "b maXX 4000 EtherCAT slave (2-row)" device and drop it on the EtherCAT bus (only possible when the bus changes color).

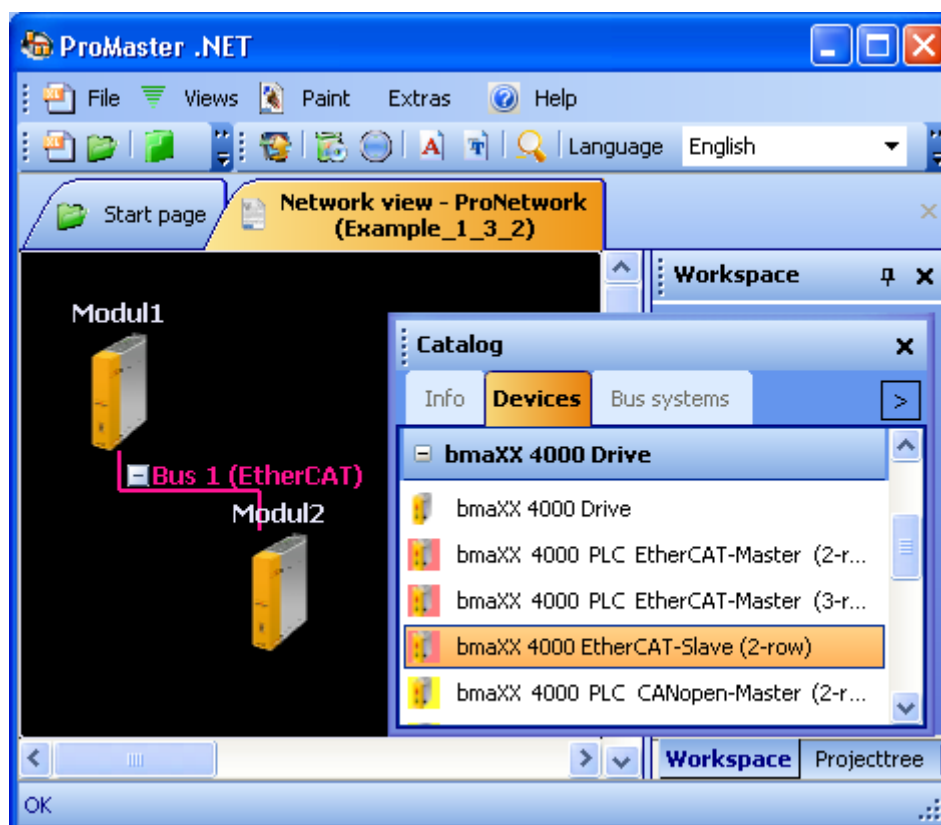


Figure 63: ProMaster - Machine configuration created

In the picture, you can now see the EtherCAT master (Modul1) and the EtherCAT slave (Modul2). Additional information on the modules can be obtained via the respective tool tip or the "Project Tree" window. The "Catalog" window is now closed to make things clearer.

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

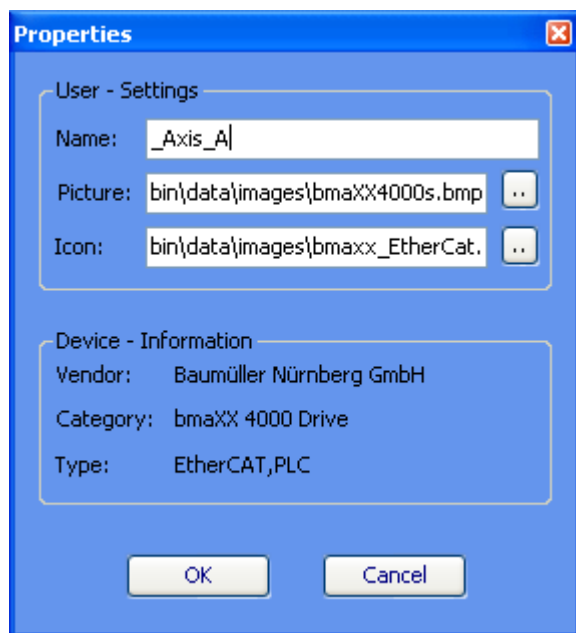


Figure 64: ProMaster - Module properties

In our example, we issue the name `_Axis_A` to the EtherCAT master and the name `_Axis_B` to the EtherCAT slave (`_Axis_A` and `_Axis_B` are the default axis variable names of the motion control axis in our IEC project). We retain the respective images (\*.bmp). The new names now appear in the ProMaster network view and in the project tree.

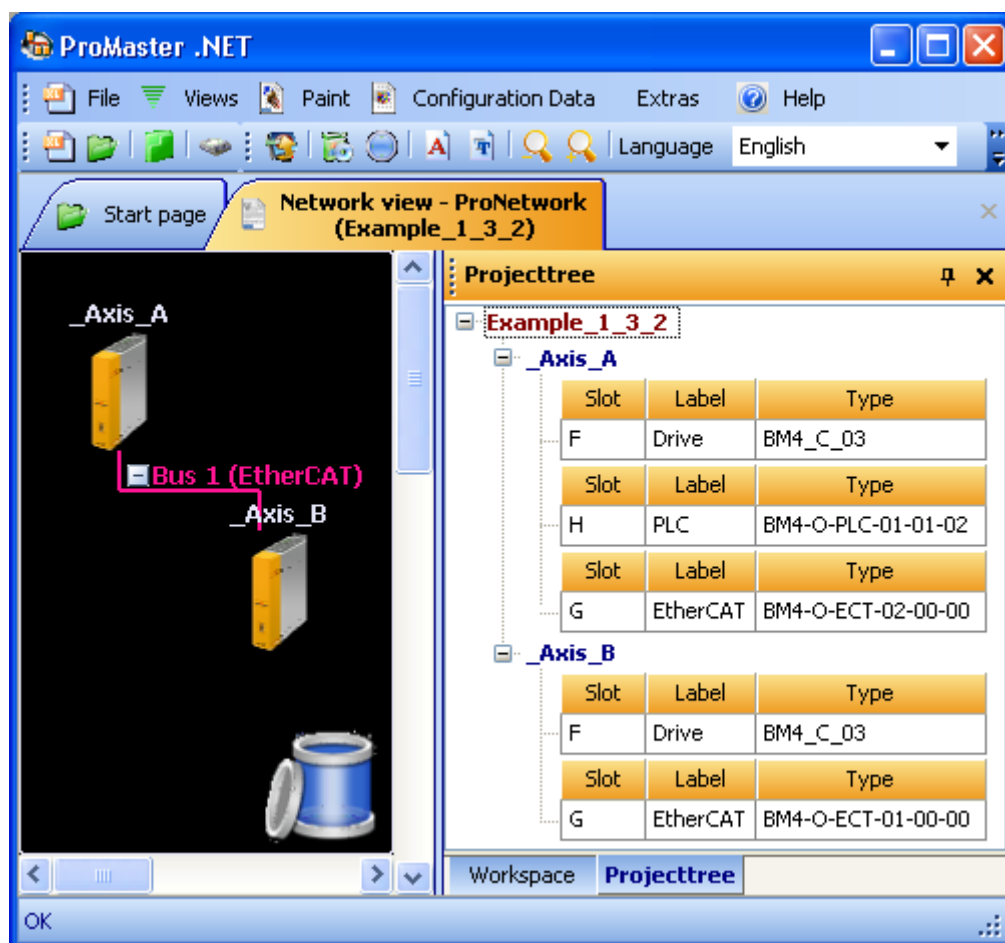


Figure 65: ProMaster - Machine configuration - Renamed module

The communication settings for the communication from the ProMaster to the b maXX drive PLC can be set by clicking the “\_Axis\_A” device and selecting “Communication settings” via the context menu. Then save the project via the menu “File\Save project”.

Motion control is activated by default in ProMaster for the local axis and the components with EtherCAT (EtherCAT master drive, EtherCAT slave drive). This reduces the settings to be configured to a minimum.

If you would like to use the default settings, you do not need to configure BACI communication with „Motion Control (local axis)” and the EtherCAT communication with ProEtherCAT.

In this case, continue with [►Configuration of the PLC ◄](#) from page 130 onward.

### 5.2.6 Configuring EtherCAT communication with ProEtherCAT

Motion control is activated by default in ProMaster for the components with EtherCAT (EtherCAT master drive, EtherCAT slave drive). This reduces the settings to be configured to a minimum.

If you would like to use the default settings, you do not need to configure EtherCAT communication with ProEtherCAT.

In this case, continue with [►Configuration of the PLC ◄](#) from page 130 onward.

Changing the EtherCAT communication settings is described in the respective application manuals. In our example see the „Ethernet with EtherCAT master for b maXX drive PLC“ application manual.

## 5.2.7 Configuring BACI communication

Open the „Motion Control (local axis)“ window for our b maXX controller (local axis) by clicking the „\_Axis\_A“ device and then selecting "Configuration Data (Components)\Controller[...]\Motion Control (local axis)" via the context menu.

When the machine configuration was created, the local axis ( i.e. the BACI communication between b maXX controller and b maXX drive PLC) was set by default so, that the BACI settings are configured for motion control (dragging and dropping of a b maXX with PLC).

The window „Motion Control (local axis) is opened.

### 5.2.7.1 Ident tab

On the „Ident“ tab b maXX controller specific informations (controller table version, controller type and the controller firmware version) are displayed.

If your b maXX controller has another than the set (controller table) version, you can select in the combo box „Version“ the respective (controller table) version.

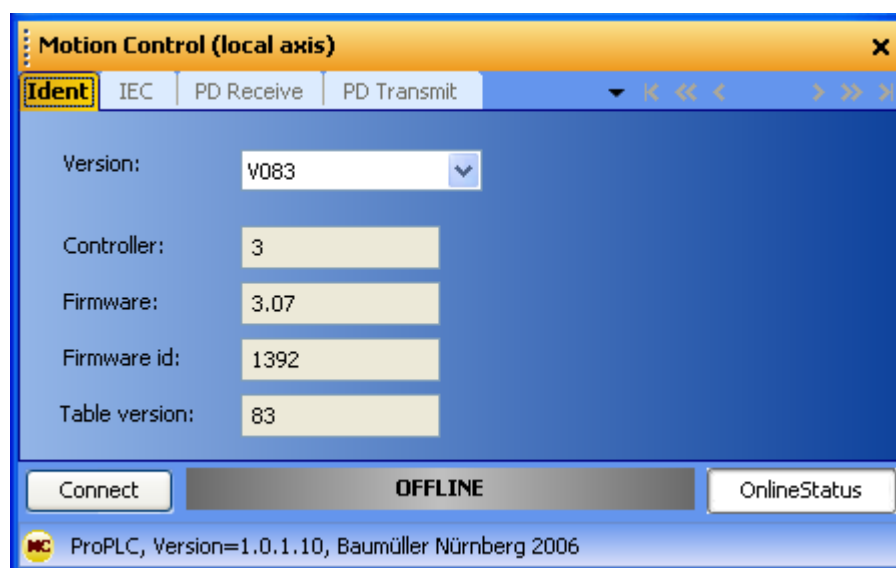


Figure 66: Motion Control (local axis) - „Ident“ tab

### 5.2.7.2 IEC tab

On the „IEC“ tab the network variables are displayed which are used for the communication between b maXX drive PLC and b maXX controller. The network variables differ in standard network variables and additional network variables for the BACI communication, if additional parameters are created on the „PD Receive“ and „PD Transmit“ tab.

The following must be heeded:

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

These are the standard motion control network variables for the setpoint values:

"w_controlword"	for the control word of the axis
"ud_PosIpSetAngle1"	for the synchronization position setpoint angle of the axis

These are the standard motion control network variables for the actual values:

"w_statusword"	for the status word of the axis
"ud_Enc1ActAngle1"	for the synchronization position actual angle of the axis
"i_OperationModeAct"	for the operating mode of the axis

### NOTE



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

The number after the network variable name is an internal number used to differentiate between identically sounding, automatically generated network variable names.

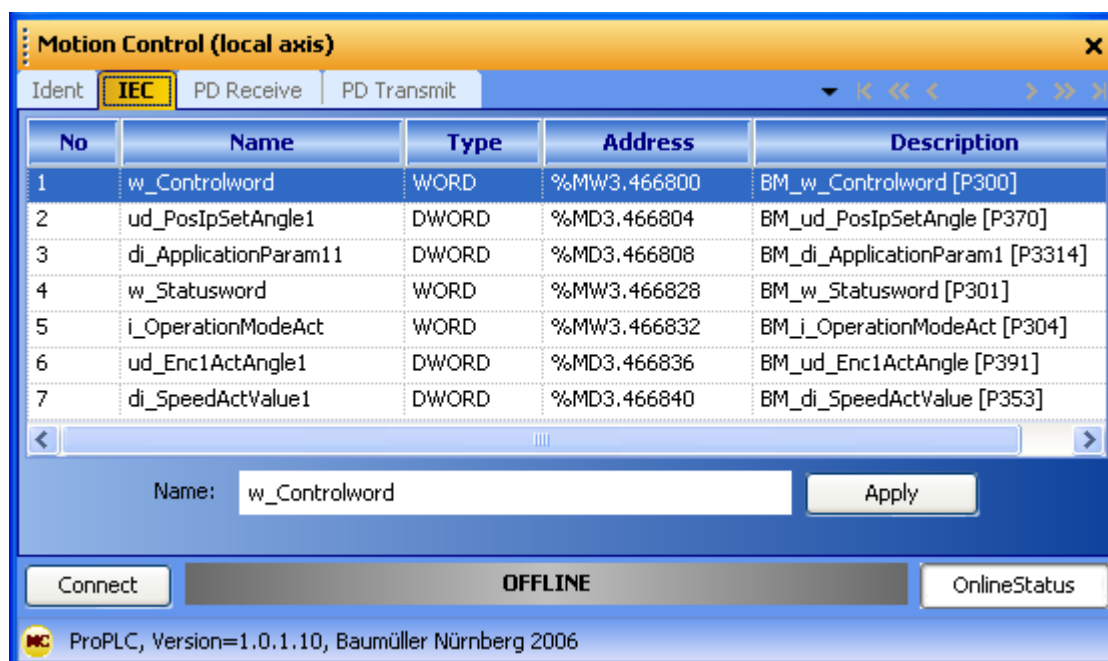


Figure 67: Motion Control (local axis) - „IEC“ tab

If you created the additional network variables (and their link to the parameters of the b maXX controller) that were suggested when configuring BACI communication in Sections [PD Receive tab](#) from page 127 onward and [PD Transmit tab](#) from page 128 onward, you will see the additional network variables.

The additional network variable for the setpoint value is:

"di\_ApplicationParam1.." for application parameter 1 of the axis

The additional network variable for the actual value is:

"di\_SpeedActValue.." for the actual speed value of the axis

### 5.2.7.3 PD Receive tab

Communication from the b maXX drive PLC to the b maXX controller is configured on the "PD Receive" tab.

The default configuration is already set for motion control. You can expand on the configuration. Note that the motion-control configuration has a higher priority than the configuration by the user.

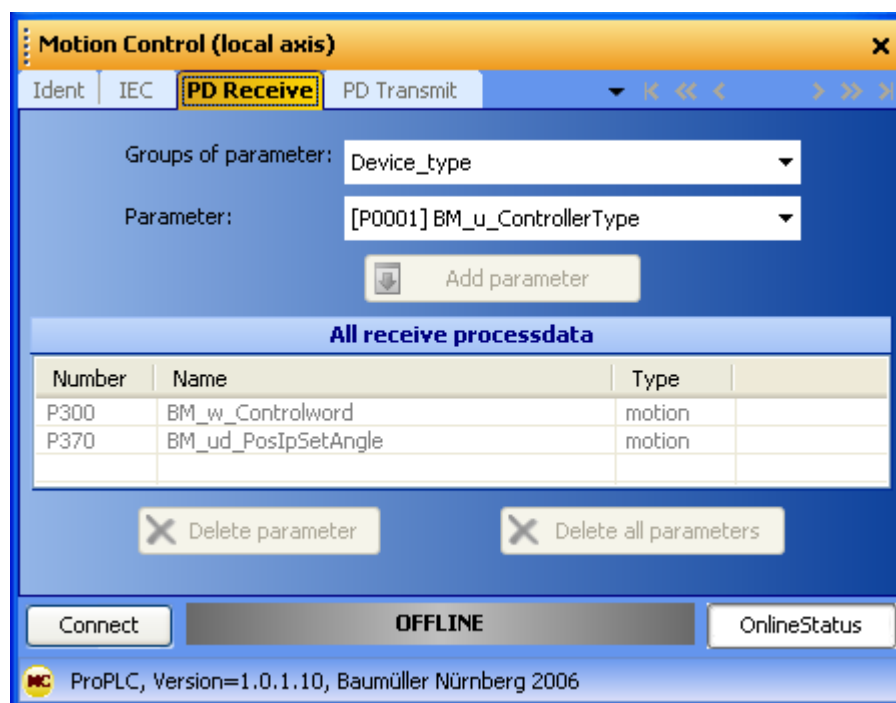


Figure 68: Motion Control (local axis) - „PD Receive“ tab with default motion control configuration

Linking of a parameter in the b maXX controller with a network variable in the IEC project occurs here.

Procedure:

- Select a parameter group of the b maXX controller.  
Example: We select the group „Configuration\_Application“ in the „Groups of parameter“ combo box.
- Select a parameter from this group.  
Example: We select the parameter 3314 „[3314]BM\_di\_ApplicationParam1“ in the „Parameter“ combo box.
- Click the button „Add Parameter“. The just prepared linking is checked for conformity to the settings and is entered in the list.

The b maXX controller parameter 3314 (application parameter 1) is now linked to a network variable in the IEC project. In our case the name of the network variable in the IEC project is „di\_Application\_Param11“. To this network variable will be written in the IEC project. The value written to the parameter 3314 (Application parameter 1) is written via the BACI communication in the controller.

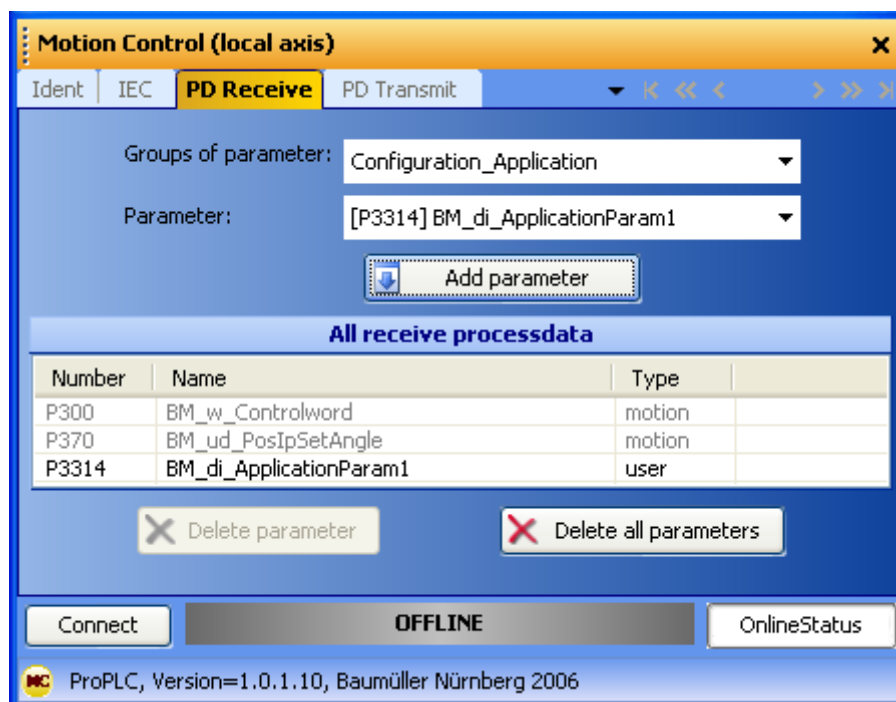


Figure 69: Motion Control (local axis) - „PD Receive“ tab with default motion control configuration and additional parameter

### NOTE



ProMaster automatically takes the 30-character limit for variable names in the IEC project in ProProg wt III into account.

The variable names from the additional configuration can be changed in Section [►IEC tab ◀](#) from page 125 onward.

#### 5.2.7.4 PD Transmit tab

Communication from the b maXX controller to the b maXX drive PLC is configured on the “PD Transmit” tab.

The default configuration is already set for motion control. You can expand on the configuration. Note that the motion-control configuration has a higher priority than the configuration by the user.



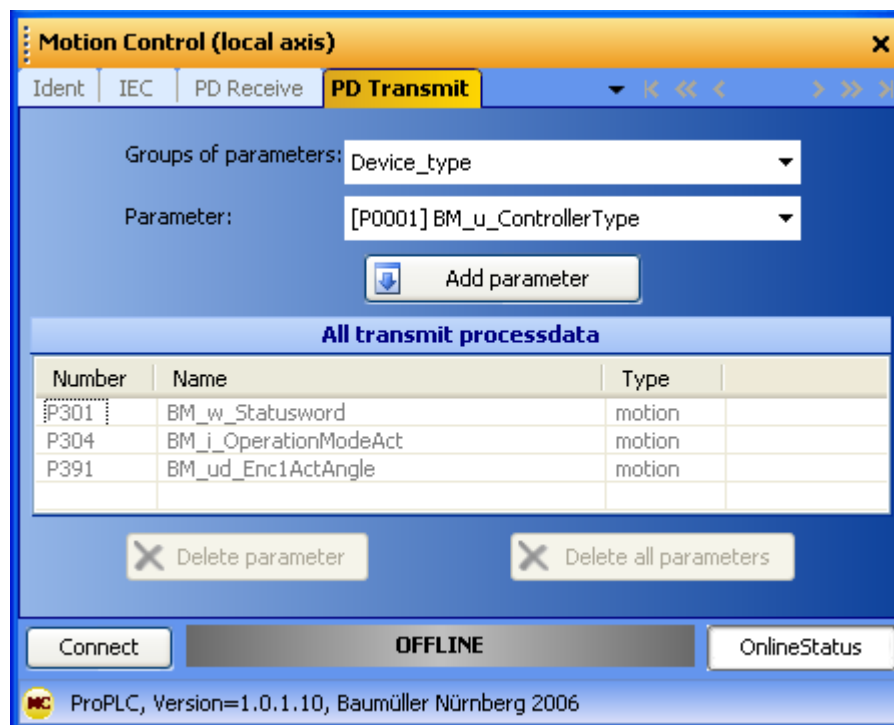


Figure 70: Motion Control (local axis) - „PD Transmit“ tab with default motion control configuration

Linking of a parameter in the b maXX controller with a network variable in the IEC project occurs here.

Procedure:

- Select a parameter group of the b maXX controller.  
Example: We select the group „Speed\_controller“ in the „Groups of parameters“ combo box.
- Select a parameter from this group.  
Example: We select the parameter 353 „[0353]BM\_di\_SpeedActValue“ in the „Parameter“ combo box.
- Click the button „Add Parameter“. The just prepared linking is checked for conformity to the settings and is entered in the list.

The b maXX controller parameter 353 (speed actual value) is now linked to a network variable in the IEC project. In our case the name of the network variable in the IEC project is „di\_SpeedActValue1“. This network variable will be read in the IEC project. The value will be read from the Controller parameter 353 (speed actual value) via the BACI communication.

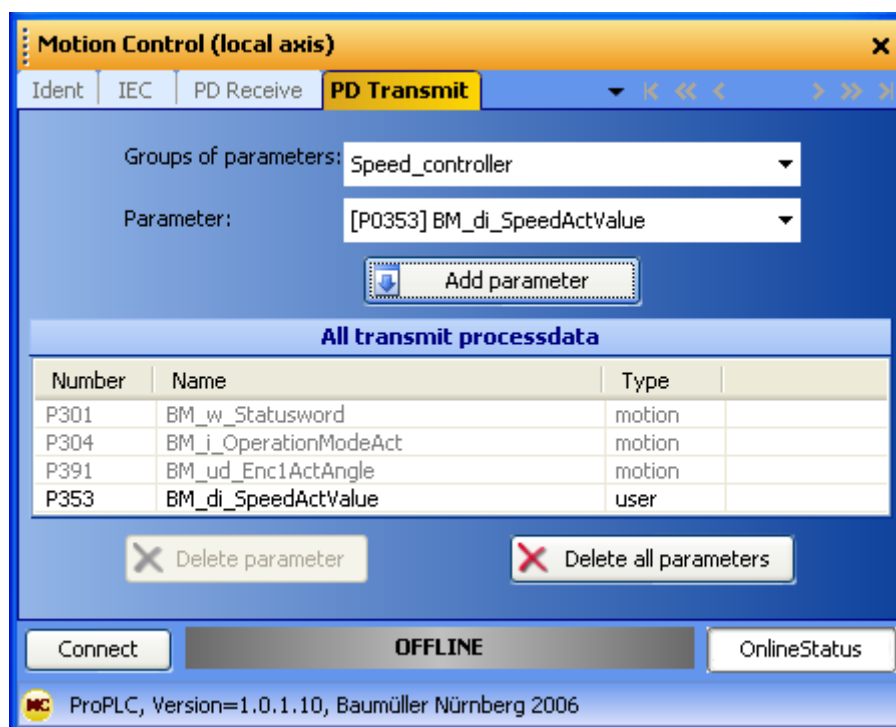


Figure 71: Motion Control (local axis) - „PD Transmit“ tab with default motion control configuration and additional parameter



### Note

ProMaster automatically takes the 30-character limit for variable names in the IEC project in ProProg wt III into account.

The variable names from the additional configuration can be changed in Section [►IEC tab ◄](#) from page 125 onward.

### 5.2.8 Configuration of the PLC

Once the configuration of BACI communication, the communication of EtherCAT network and the creation of the IEC project have occurred, the data for the b maXX drive PLC are to be configured.

For this purpose,

- the IEC project is included in the ProMaster project
- the cam data sets are selected

In addition,

- cam data can be generated with the cam editor ProCAM

Then click the "Update total IEC project" button in the "ProPLC" window. The data for the machine configuration, both for the EtherCAT network and the b maXX drive PLC, (including local axis) are then generated.

**NOTE**

The network variables from the configuration of the BACI communication (local axis) are not displayed in ProPLC.

With the "Update total IEC project" button these network variables are written in the global work sheet of the IEC project in ProProg wt III (in the section LocalAxisVariables) too.

**5.2.8.1 IEC**

This section explains how an IEC project is linked with ProMaster.

Open the "ProPLC" window in the network view for our local axis (and EtherCAT master) in the ProMaster project. Click the "\_Axis\_A" device and then select "Configuration Data (Components)\PLC (slot H)\PLC - Configuration (ProPLC)" via the context menu and then the "IEC" tab.

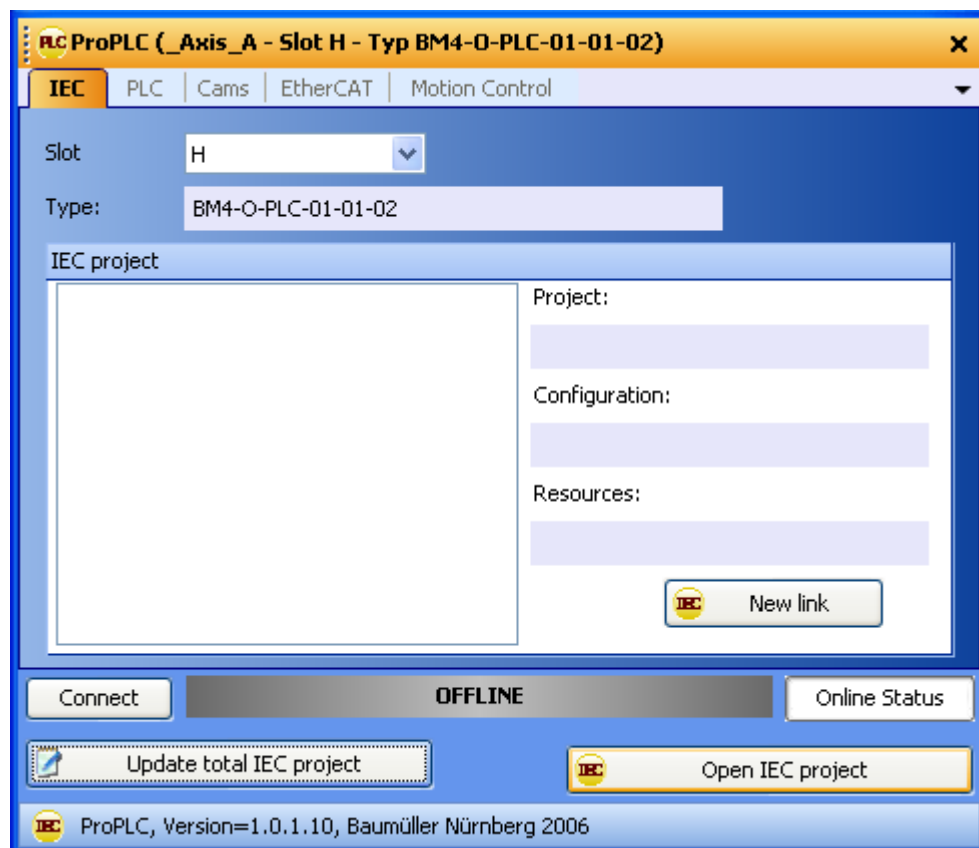


Figure 72: ProPLC - Connecting with example IEC project



### Note

Only ProProg wt III projects can be opened with ProMaster. If you would like to use an existing PROPROG wt II project, you must first open it with ProProg wt III (thus converting it) and can then open and use it in ProMaster.

Note that your PROPROG wt II libraries will also be converted here!

Click the "New link" button and select our example IEC project "Example\_BM4\_O\_ECT02\_MA\_2.mwt" created in Section [▶Creating of an IEC project with ProProg wt III](#) ◀ from page 117 onward.

The IEC project just selected is displayed on the "IEC" tab.

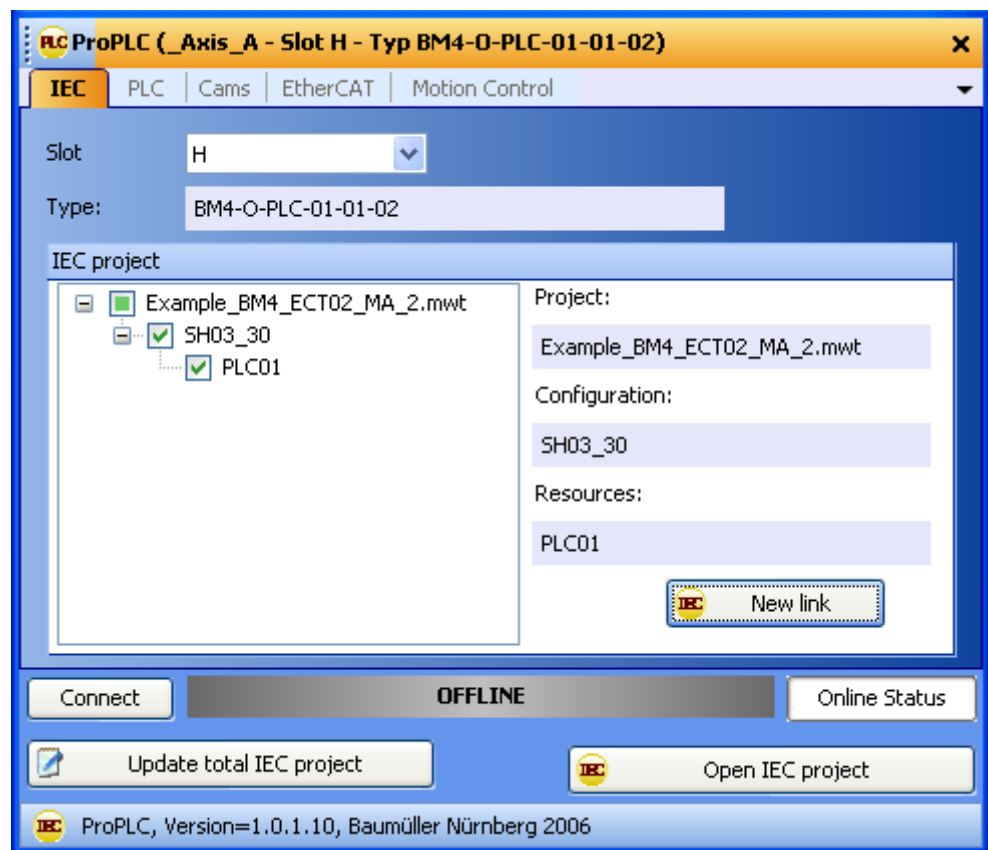


Figure 73: ProPLC - IEC tab

If you link an existing application (or template) from ProProg wt III with the ProMaster project, the names of the devices in the ProMaster may not match those in the IEC project. In this case, see Section [▶Programming the IEC project](#) ◀ from page 140 onward.

### 5.2.8.2 PLC

After downloading the data to the EtherCAT master and b maXX drive PLC (see [▶ Downloading the data to the EtherCAT master and the b maXX drive PLC ◀](#) from page 139 onward), the IEC project on the b maXX drive PLC can be activated and started on the PLC tab ("PLC Status" section).

In addition, data of the b maXX drive PLC, such as the operating system version, firmware version, IEC project and boot project, is also displayed in the "PLC Info" section. Information on the files in the flash memory of the b maXX drive PLC, e.g. on the cam records (see [▶ 5.2.8.3 Cams ◀](#)) is displayed in the "Flash Directory" section (after "Connect" and "Update")

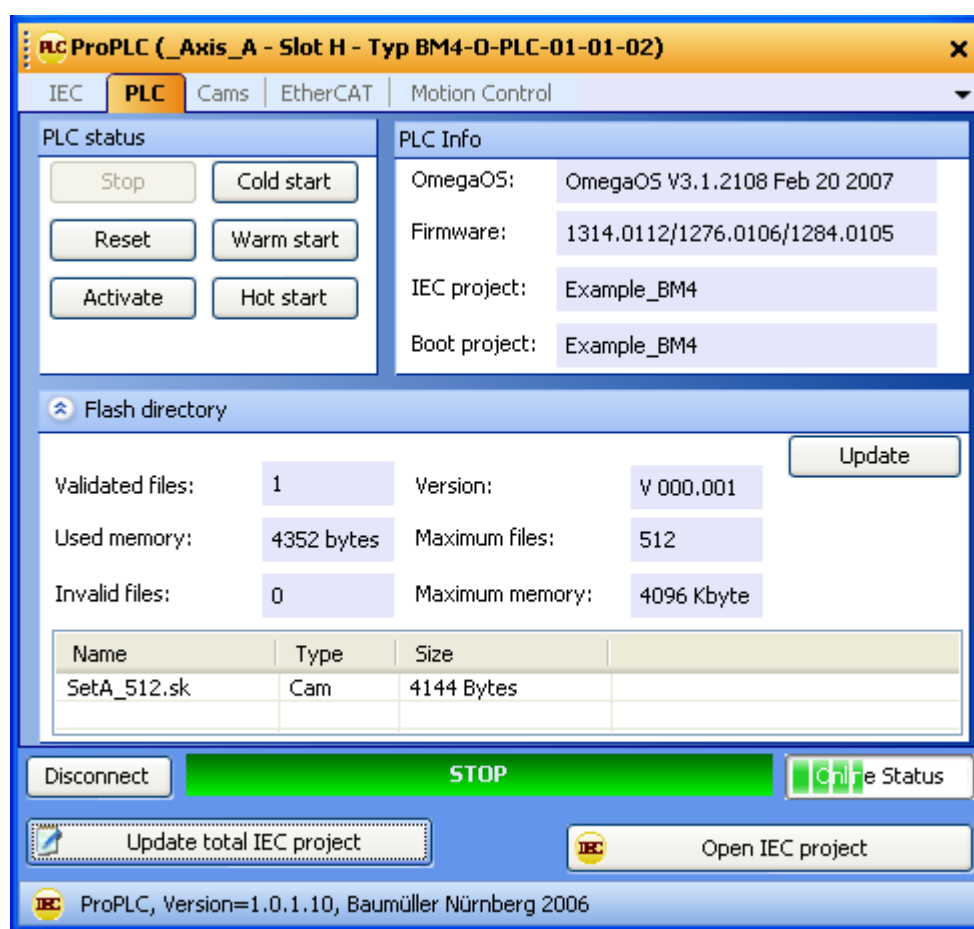


Figure 74: ProPLC - PLC tab after download of the cam (cam tab)

### 5.2.8.3 Cams

This section explains how a cam data set is generated and linked with ProMaster.

Open the "PLC Configuration" window in the network view for our local axis (and EtherCAT master) in the ProMaster project by clicking the "\_Axis\_A" device and then select "Configuration Data (Components)\PLC (slot H)\PLC - Configuration (ProPLC)" via the context menu and then the "Cams" tab.

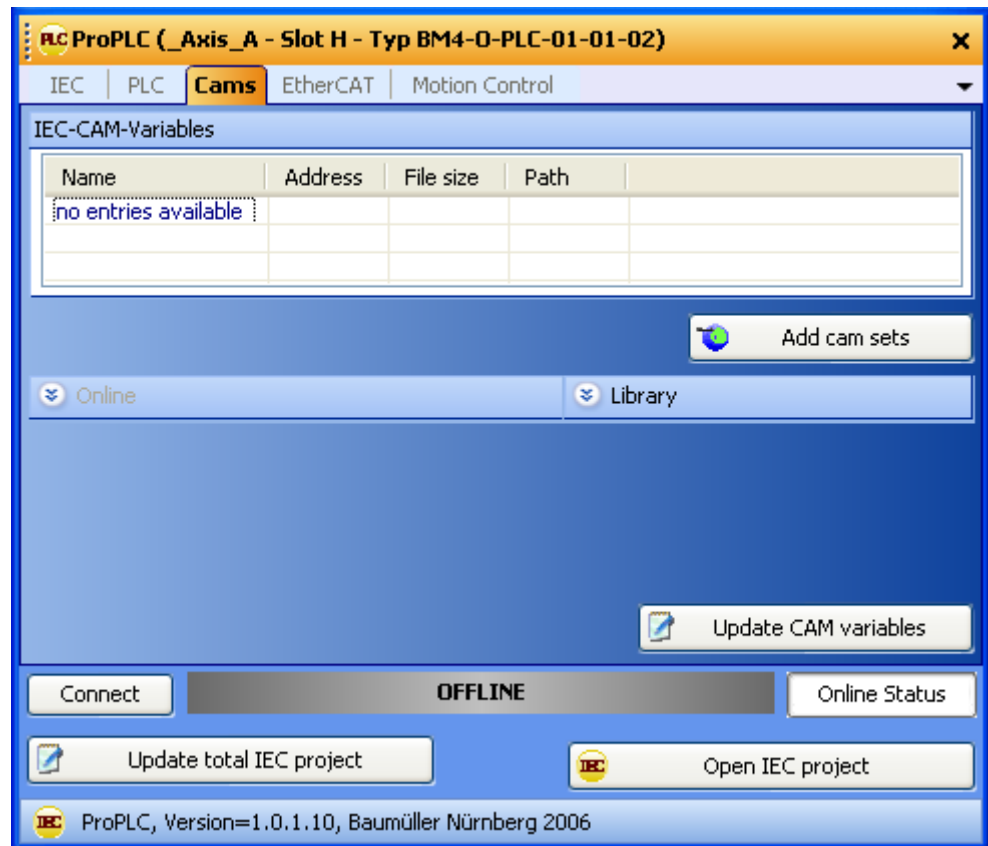


Figure 75: ProPLC - "Cams" tab with no cam data set

Press the "Add Data Sets" button. The "ProMaster.NET - Cams" window is opened.  
Press the "Add" button in the "Hard Disk" section and select the path on your hard disk that contains your cam data.



### NOTE

Example cam data is found in the installation directory of ProCAM in the "examples" subfolder (e.g. C:\...\Baumueller\ProCam 2\examples).

If you would like to create a cam, click the "ProCAM" button, which opens the cam editor. You can edit your cams there.

Now create your cam data set (in the "\*.sk Files" section) by dragging and dropping different cam data (in the "Hard Disk" section).

In our example, we select the cam data "ExampleGerade\_MC.kbin" and "ExamplePendelP5\_MC.kbin", which are combined in the "SetA\_512" cam data set.

**NOTE**

You can display the cam data graphically (in the respective section) via the "Preview" button. This simplifies cam selection.

Then click the "Save" button. This creates the cam data set and makes ProMaster available.

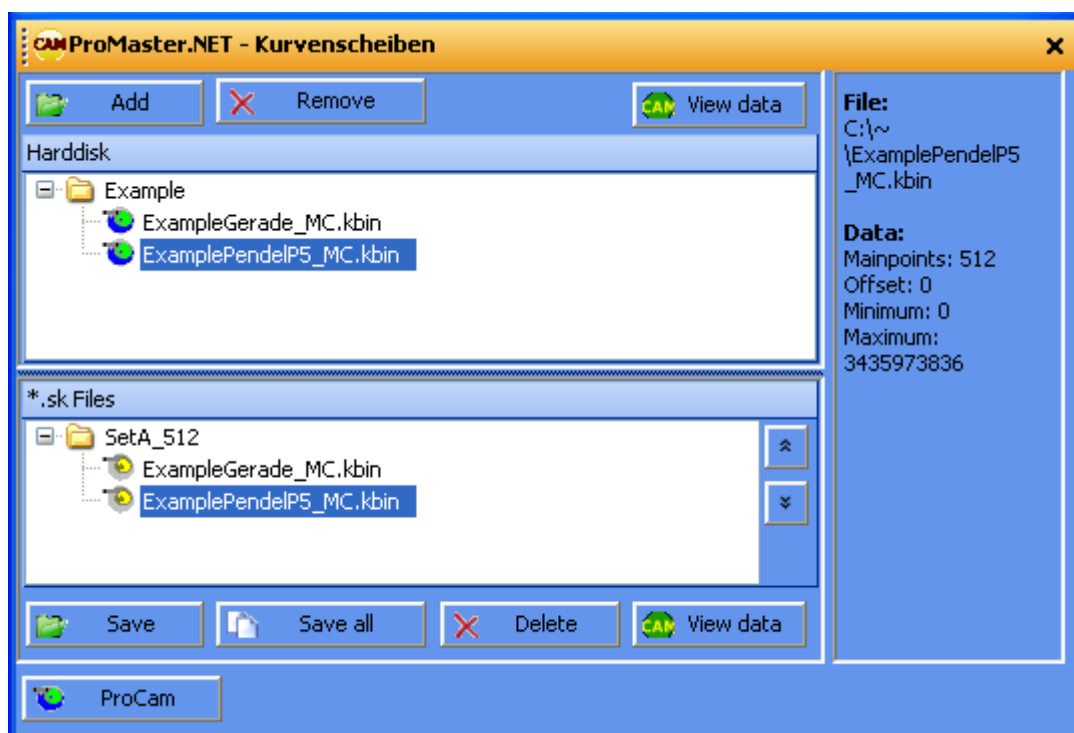


Figure 76: ProPLC (- Cams tab -) Generating cam data set

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

You can now see the name of the variables of the cam data set in the IEC project, its address in the IEC project and the file name of the cam data set on the hard disk via the "Cams" tab.

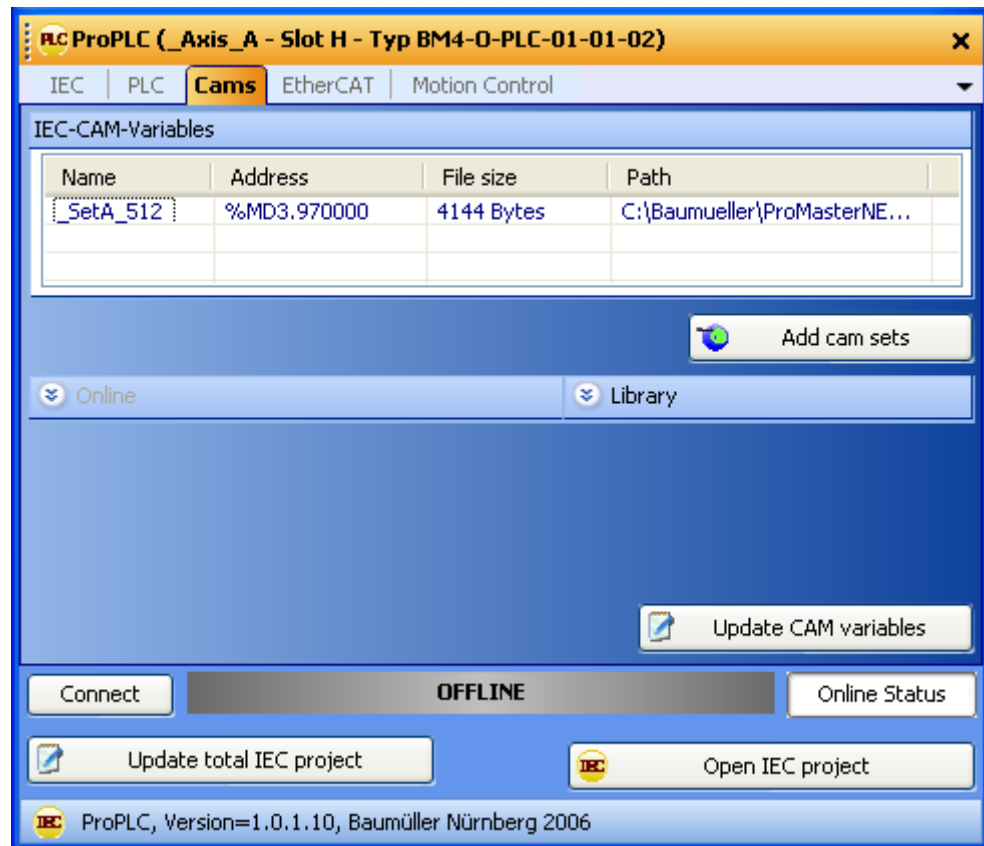


Figure 77: ProPLC - "Cams" tab with cam data set

If you link an existing application (or template) from ProProg wt III with the ProMaster project, the cam data set name in ProMaster may not match that in the IEC project. In this case, see section [►Programming the IEC project ◄](#) from page 140 onward.

From the "Cam" tab, you can clear the flash memory on the b maXX drive PLC via the "Clear Flash" button ("↓ Online" section). This may be necessary if the flash memory is "full" because of various downloads (both cam data sets and IEC projects).

In addition, you can download cam data sets to the b maXX drive PLC individually via the "Download Data Sets" button on the "Cams" tab.

Now load the cam data sets onto the b maXX drive PLC via the „Download Data Sets“ button ("↓ Online" section).

### 5.2.8.4 EtherCAT

The name, data type, address and comment of the network variables of the EtherCAT slave in the IEC project are displayed on the "EtherCAT" tab.

If motion control is used, communication occurs via the axis variable. ProMaster recreates the axis variable with the device name of the EtherCAT slave after the "Update total IEC project" button is pressed (see section [►Downloading the data to the EtherCAT master and the b maXX drive PLC ◄](#) from page 139 onward). ProMaster also creates network variables in the IEC project. These are divided into standard network variables for motion control and, if additional objects were created during configuration of the EtherCAT slave, additional network variables for EtherCAT.



The "EtherCAT" tab displays these network variables, both the standard network variables for motion control and the additional network variables for EtherCAT.

The following must be heeded:

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

These are the standard motion control network variables for the setpoint values:

"u_controlword.."	for the control word of the axis
"ud_PoslpSetAngel.."	for the synchronization position setpoint angle of the axis

These are the standard motion control network variables for the actual values:

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

---

**NOTE**

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

---

---

**NOTE**

The standard network variables must be read in the motion control event task.

---

The number after the network variable name is an internal number used to differentiate between identically sounding, automatically generated network variable names.

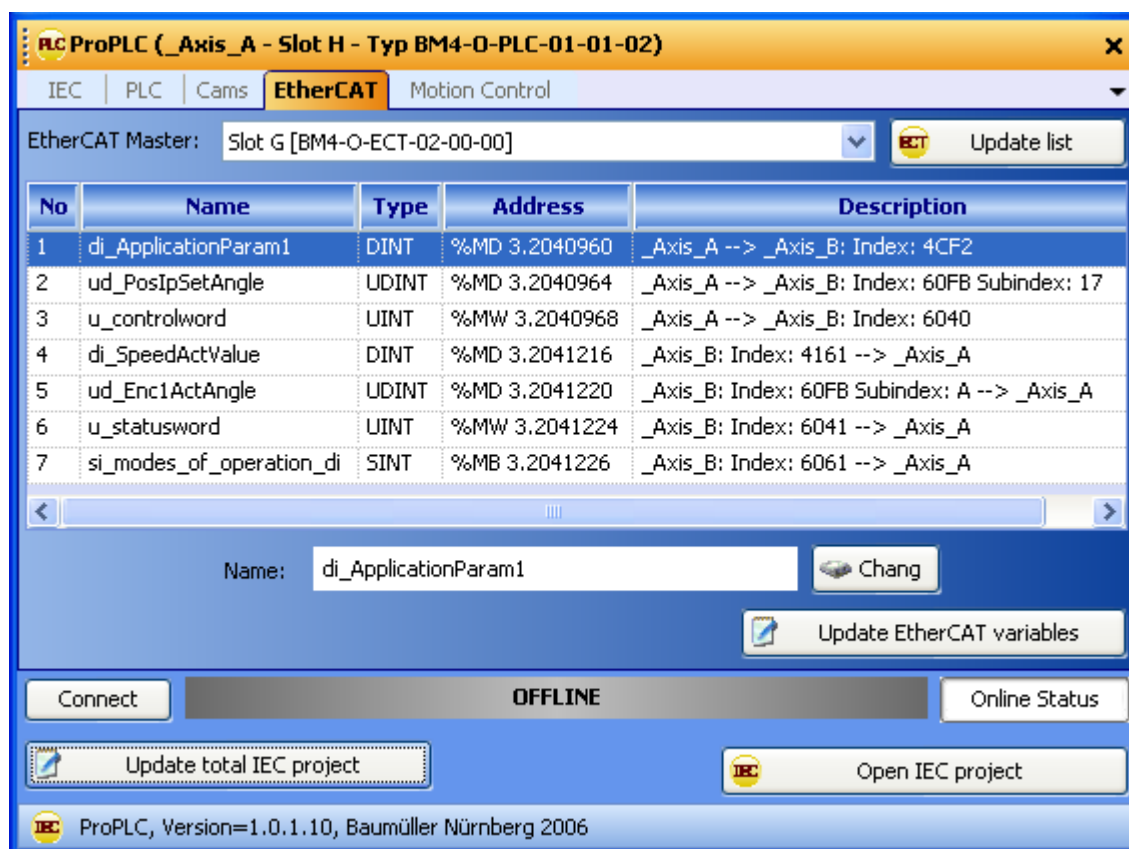


Figure 78: ProPLC - EtherCAT tab

If you created additional network variables (and their link to EtherCAT slaves) when configuring EtherCAT slave communication in Section [Configuring EtherCAT communication with ProEtherCAT](#) from page 124 onward, you will see the additional network variables.

E.g. an additional network variable for the set value is:

"di\_ApplicationParam1.." for application parameter 1 of the axis,

e.g. an additional network variable for the actual value is:

"di\_SpeedActValue.." for the actual speed value of the axis

When changing the network variable names, note that each variable name can be issued only once in the IEC project. This is especially important to note when using several EtherCAT master modules on the b maXX drive PLC.



### NOTE

Each variable name may be issued only once in the IEC project.

**NOTE**

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

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

For more informations please refer to Ethernet with EtherCAT master for b maXX drive PLC application manual.

**5.2.8.5 Motion Control**

All general settings for motion control will be made on the "Motion Control" tab in the future.

If you would like to use the default motion control settings, you do not need to make any settings here.

**5.2.9 Downloading the data to the EtherCAT master and the b maXX drive PLC**

At present, downloading of the data for the EtherCAT master occurs see Ethernet with EtherCAT master for b maXX drive PLC application manual.

Downloading of the cam data sets for the b maXX drive PLC currently occurs via the "Cams" tab of the PLC configuration. For this purpose, see Section [►Cams◄](#) from page 133 onward.

Downloading the IEC project for the bmaXX drive PLC currently occurs via ProProg wt III, as usual.

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

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

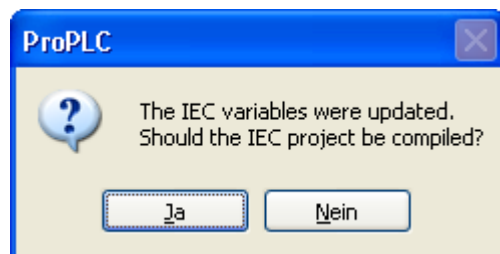


Figure 79: ProMaster - compile the updated IEC project

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

Now download the IEC project to the b maXX drive PLC (ProProg wt III menu „Online\ Ressource Control...“ → „Send“ → „Send“ Boot Project).

Now perform a reset of the b maXX drive PLC and then switch the PLC to the „RUN“ state (alternatively, you can switch the entire EtherCAT network off and then on again).

You can now control the local axis `_Axis_A` and the EtherCAT slave axis `_Axis_B` via the IEC project in ProProg wt III.

### 5.2.10 Programming the IEC project

---

#### 5.2.10.1 General information

---

For information on how to program a motion control application in the IEC project in ProProg wt III, please refer to the motion control application manual and the Online Help System of ProProg wt III.

ProProg wt III is opened with our IEC project "Example\_BM4\_O\_ECT02\_MA\_2.mwt" via the context menu on `_Axis_A` "Configuration Data (Components)\PLC (Slot H)\IEC Programming (ProProg wt III)". You can edit the IEC project as usual.

ProMaster has written

- the motion-control axis variables (in the MC\_Axis\_Variables section)
- the cam data set (in the MC\_CamDataSet section)
- the network variables for EtherCAT (in the EtherCATVariables section)
- the network variables of the local axis (in the LocalAxisVariables section)  
(standard network variables for motion control and additional network variables for the local axis, see [►Exchanging data ◀](#) from page 142 onward)

in the global variable worksheet "Global\_Variables".

Name	Type	Usage	Description	Address
<b>Global_Variables</b>				
<b>MyApplication</b>				
_MyMaster	AXI...	VAR_GLO...	Not from ProMaster	
<b>MC_AxisVariables</b>				
_Axis_A	AXI...	VAR_GLO...	1. Achse	%MD3.451500
_Axis_B	AXI...	VAR_GLO...	2. Achse	%MD3.457300
<b>MC_MasterRef</b>				
<b>MC_SystemReserved</b>				
<b>MC_AxisRef</b>				
<b>MC_BacilnitRef</b>				
<b>MC_AxisPDORef</b>				
<b>MC_TriggerRef</b>				
<b>MC_IRPRef</b>				
<b>MC_CamDataSet</b>				
s_File_SetA_512	FileS...	VAR_GLO...		%MD3.970000
_SetA_512	MC_...	VAR_GLO...		%MD3.970000
<b>EtherCAT Variables</b>				
<b>LocalAxisVariables</b>				
w_Controlword	WORD	VAR_GLO...	BM_w_Controlword [P300]	%MW3.466800
ud_PoslpSetAngle1	UDINT	VAR_GLO...	BM_ud_PoslpSetAngle [P370]	%MD3.466804
di_ApplicationParam11	DINT	VAR_GLO...	BM_di_ApplicationParam1 [P3314]	%MD3.466808
w_Statusword	WORD	VAR_GLO...	BM_w_Statusword [P301]	%MW3.466828
i_OperationModeAct	INT	VAR_GLO...	BM_i_OperationModeAct [P304]	%MW3.466832
ud_Enc1ActAngle1	UDINT	VAR_GLO...	BM_ud_Enc1ActAngle [P391]	%MD3.466836
di_SpeedActValue1	DINT	VAR_GLO...	BM_di_SpeedActValue [P353]	%MD3.466840

Figure 80: ProProg wt III - Global variable worksheet with the ProMaster data

The following, in particular, must be heeded:

The device name of the device on the EtherCAT bus (local axis/EtherCAT master, EtherCAT slave(s)) in ProMaster is also the name of the axes in the IEC project in ProProg wt III. This means that if you have linked an existing application (or template) from ProProg wt III with the ProMaster project, the names of the devices in ProMaster may not match those in the IEC project. In this case, there are two possible solutions:

- 1 ProMaster changes the axis names to the device names in ProMaster in the "MC\_AxisVariables" section of the "Global\_Variables" variable worksheet in ProProg wt III, and the user changes the axis names in the POEs to the device names in ProMaster (via the ProProg wt III function "Replace Globally").
- 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 EtherCAT slave the device name `_Axis_B` in ProMaster. This is also the axis name from our motion control template used for our IEC project.

This also applies for the cam data set name in the IEC project in ProProg wt III. This means that if you link an existing application (or template) from ProProg wt III with the ProMaster project, the name of the cam data set in ProMaster may not match that in the IEC project. 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 "Replace Globally").

### 5.2.10.2 Exchanging data

A global variable is required for exchanging data between

- the function modules in the IEC project on the b maXX drive PLC (BM4-O-PLC-0x) and the local axis and
- the function modules in the IEC project on the b maXX drive PLC (BM4-O-PLC-0x), the Ether-CAT master option module (BM4-O-ETH-02) and (via the EtherCAT field bus) the EtherCAT slave.

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

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

Our axis variable for the EtherCAT slave axis has the axis name `_Axis_B`.

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

The same applies for the variable for the cam data, which has the cam data set name `_SetA_512` in the IEC project and ProMaster.

The axis variable and the cam data variable are connected to the motion control function modules in the IEC project. The machine functions are programmed with the motion control function modules.

Information on using the motion control function modules is found in the motion control application manual.

If motion control is used, communication occurs via the axis variable. ProMaster creates network variables in the IEC project. These are divided into standard network variables for motion control and, if additional objects were created during configuration of the BACI communication, additional network variables for the local axis.

The following must be heeded:

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

#### `_Axis_A` (local axis)

These are the standard motion control network variables for the set values:

<code>"w_controlword"</code>	for the control word of the axis
<code>"ud_PoslpSetAngle1"</code>	for the synchronization position setpoint angle of the axis

These are the standard motion control network variables for the actual values:

<code>"w_statusword"</code>	for the status word of the axis
<code>"ud_Enc1ActAngle1"</code>	for the synchronization position actual angle of the axis
<code>"i_OperationModeAct"</code>	for the operating mode of the axis

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

If you created the additional network variables (and their link to the parameters of the b maXX controller) that were suggested when configuring BACI communication in Sections [►PD Receive tab ◀](#) from page 127 onward and [►PD Transmit tab ◀](#) from page 128 onward, the additional network variable for the set values are:

"di\_ApplicationParam1.." for application parameter 1 of the axis

The additional network variable for the actual value is:

"di\_SpeedActValue.." for the actual speed value of the axis

### \_Axis\_B (EtherCAT slave axis)

These are the standard motion control network variables for the set values:

"u\_controlword.." for the control word of the axis

"ud\_PosIpSetAngel.." for the synchronization position setpoint angle of the axis

These are the standard motion control network variables for the actual values:

"u\_statusword.." for the status word of the axis

"ud\_Enc1ActAngle.." for the synchronization position actual angle of the axis

"si\_modes\_of\_operation\_display.." for the operating mode of the axis

If you created the additional network variables (and their link to EtherCAT slaves) that were suggested when configuring EtherCAT slave communication in sections [►PD Receive tab ◀](#) from page 127 onward and [►PD Transmit tab ◀](#) from page 128 onward, you see the additional network variable for the set values.

E.g. an additional network variable for the set value is:

"di\_ApplicationParam1.." for application parameter 1 of the axis,

e.g. an additional network variable for the actual value is:

"di\_SpeedActValue.." for the actual speed value of the axis

The number after the network variable name is an internal number used to differentiate between identically sounding, automatically generated network variable names.



#### NOTE

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

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



#### NOTE

If motion control is used, the network is started automatically via motion control.



### NOTE

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

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

---

## 5.3 PROPROG wt II

---

### 5.3.1 b maXX PLC and controller

---

#### Process data:

The b maXX drive PLC takes the controller's actual values and the status word in a definable time raster from BACI and transfers the reference values and the control word via BACI to the controller.

At transfer of the actual values and the status word, the system triggers the "BACI process data" hardware event in the b maXX drive PLC. This event can trigger an event task in the b maXX drive PLC in which BACI process data communication can be carried out.

The system sets the instant or the timing code for communication at initialization of the process data communication in the user program using FB BACI\_INIT.

#### Service data:

The b maXX drive PLC transfers a read parameter or write parameter job to BACI. The controller reads the job from BACI, processes the appropriate parameters in the controller (see the respective controller description) and returns the result to BACI. The b maXX drive PLC then reads the result of communication from BACI.

### 5.3.2 Programming BACI communication on the b maXX drive PLC

---

Due to compatibility the function blocks of the libraries for PROPROG wt II are available in the libraries for ProProg wt III. The function blocks are written in the following.



### NOTE

The following function blocks, also written for ProProg wt III, are not used for new projects with ProMaster, ProProg wt III and Motion Control

---



You program the b maXX drive PLC using the PROPROG wt II or ProProg wt III programming system (see the PROPROG wt II programming manual or the Online Help System of ProProg wt III).

The Baumüller user libraries SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) and SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or above are available for programming BACI communication.

In library SYSTEM1\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM1\_PLC01\_30bd00 (ProProg wt III) or above, function blocks are available for initializing process data communication, for process data communication and for service data communication.

## You need the following FBs for process data communication:

BACI_INIT	Initializing process data communication.
BACI_PD_COMM	Process data communication

## You need the following FBs for requirements data communication:

BACI_PAR_READ	Service data communication (read parameter job)
BACI_PAR_WRITE	Service data communication (write parameter job)

**Library SYSTEM2\_PLC01\_20bd00 (PROPROG wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or above contains amongst others, the following function block:**

INTR_SET	is used by FB BACI_INIT (FB for linking and activating a hardware signal with the BACI process data event.)
----------	--

### 5.3.3 Function Blocks for BACI Overview

In addition to the standard function blocks, you can use manufacturer-defined function blocks if you have logged on manufacturer-defined libraries in a project.

Note: Logging on of libraries is described in the general help.

The following function blocks are available for BACI:

Function block	Brief description
BACI_INIT	For initializing BACI ( <b>BA</b> umüller <b>C</b> omponent <b>I</b> nterface) in the b maXX drive PLC
BACI_PAR_READ	Read BACI parameter
BACI_PAR_WRITE	Write BACI parameter
BACI_PD_COMM	Process data communication via BACI in the b maXX drive PLC for a maximum of eight target and actual values
BACI_PD_RECONFIG	Reconfigure BACI process data communication during ongoing operation

## 5.3.3.1 BACI\_INIT

You can use this function block for BACI to initialize process data communication between the b maXX controller and b maXX drive PLC via the BACI interface.

**Note**

FB BACI\_INIT uses library BM\_TYPES\_20bd03 (PROPROP wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or higher and firmware library SYSTEM2\_PLC01\_20bd00 (PROPROP wt II) or SYSTEM2\_PLC01\_30bd00 (ProProg wt III) or higher.

Application: During the initialization stage (cold boot/warm restart task).

**Note**

Necessary b maXX controller settings for i\_EVENT = INT#11 (=SYNC1 signal source) or i\_EVENT = INT#12 (=SYNC2 signal source) on the WinBASS II "Synchronization" side:

- Source for sync signal:
  - "Use Sync 1 signal from BACI" (i\_EVENT = INT#11) or
  - "Use Sync 2 signal from BACI" (i\_EVENT = INT#12).
- Choose Sync interval (=> the same time period must be connected at input "u\_CYCLE\_TIME" in µs).
  - Sync offset: -25.0 µs.
  - Sync tolerance: 2.0 µs.

Setting the Sync offset guarantees that the controller has already finished the actual value cycle before the PLC starts its communication cycle.

Otherwise, BACI access conflicts can result when transferring target and actual values in block BACI\_PD\_COMM.

Parameter input	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
i_EVENT	INT 0, 4, 8, 11, 12	Event
u_CYCLE_TIME	UINT	BACI cycle time in µs
u_WR_PAR_NR0	UINT	Specified value parameter number 0
u_WR_PAR_NR1	UINT	Specified value parameter number 1
u_WR_PAR_NR2	UINT	Specified value parameter number 2
u_WR_PAR_NR3	UINT	Specified value parameter number 3
u_WR_PAR_NR4	UINT	Specified value parameter number 4
u_WR_PAR_NR5	UINT	Specified value parameter number 5
u_WR_PAR_NR6	UINT	Specified value parameter number 6

Parameter input	Data type	Description
u_WR_PAR_NR7	UINT	Specified value parameter number 7
u_WR_PAR_NR8	UINT	Specified value parameter number 8
u_WR_PAR_NR9	UINT	Specified value parameter number 9
u_WR_PAR_NR10	UINT	Specified value parameter number 10
u_WR_PAR_NR11	UINT	Specified value parameter number 11
u_WR_PAR_NR12	UINT	Specified value parameter number 12
u_WR_PAR_NR13	UINT	Specified value parameter number 13
u_WR_PAR_NR14	UINT	Specified value parameter number 14
u_WR_PAR_NR15	UINT	Specified value parameter number 15
u_RD_PAR_NR0	UINT	Actual value parameter number 0
u_RD_PAR_NR1	UINT	Actual value parameter number 1
u_RD_PAR_NR2	UINT	Actual value parameter number 2
u_RD_PAR_NR3	UINT	Actual value parameter number 3
u_RD_PAR_NR4	UINT	Actual value parameter number 4
u_RD_PAR_NR5	UINT	Actual value parameter number 5
u_RD_PAR_NR6	UINT	Actual value parameter number 6
u_RD_PAR_NR7	UINT	Actual value parameter number 7
u_RD_PAR_NR8	UINT	Actual value parameter number 8
u_RD_PAR_NR9	UINT	Actual value parameter number 9
u_RD_PAR_NR10	UINT	Actual value parameter number 10
u_RD_PAR_NR11	UINT	Actual value parameter number 11
u_RD_PAR_NR12	UINT	Actual value parameter number 12
u_RD_PAR_NR13	UINT	Actual value parameter number 13
u_RD_PAR_NR14	UINT	Actual value parameter number 14
u_RD_PAR_NR15	UINT	Actual value parameter number 15
t_TIME	TIME	Monitoring time in seconds

Parameter output	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
i_ERR	INT	Error number
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

### Description

FB BACI\_INIT make it possible to initialize process data communication between the b maXX controller and the b maXX drive PLC.

FB BACI\_PD\_COMM is available for process data communication for applications in the associated bypass interrupt (see Input i\_EVENT).

Input/output \_BASE:

At \_BASE, you must connect a global variable of data type BACI\_CTRL\_BMSTRUCT.

This global variable is predefined as \_PLC\_Init in the BM4\_O\_PLCO1 template in PROPROG wt in global variable sheet Global\_Variables.

Alternatively, users can create this variable themselves.

In this case, you must locate it by means of declaration of global variables to the base address of the b maXX drive PLC's dual-port RAM, i.e. %MB3.9000000.

Example:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

Where:

\_PLC\_Init is the variable name with the data type short designation "\_" for Struct

BACI\_CTRL\_BMSTRUCT is the data type

%MB3.9000000 is the base address of the b maXX drive PLC's dual-port RAM

Input i\_EVENT:

i\_EVENT = 4 initializes event "BACI process data" with interrupt level 13 (low priority).

i\_EVENT = 8 initializes event "BACI process data" with interrupt level 14 (high priority).

i\_EVENT = 11 initializes event "SYNC-Signal 1 option module" with interrupt level 14 (high priority).

i\_EVENT = 12 initializes event "SYNC-Signal 2 option module" with interrupt level 14 (high priority).

If input i\_EVENT is not assigned or i\_EVENT = 0, the system does not initialize any events. In this case, new parameter numbers are reconfigured but no interrupt source is set up.

If you want to reconfigure parameter numbers during operation, you should use block BACI\_PD\_RECONFIG.

If i\_EVENT is not 0, 4, 8, 11 or 12, the system sets i\_ERR to the value INT#3.

Input u\_CYCLE\_TIME:

You specify the cycle time of process data communication at input u\_CYCLE\_TIME. The cycle time of process data communication corresponds to the interval at which the "BACI process data" event triggers an interrupt on the b maXX drive PLC.

The u\_CYCLE\_TIME value must be a multiple of 125 µs and be at least 250 µs.

If  $u\_CYCLE\_TIME < 250$ , 250  $\mu s$  is set as the cycle time of process data communication.

If  $u\_CYCLE\_TIME < 125 * x$ , the system rounds up and sets  $125 \mu s * x$  as the cycle time of process data communication.

Example:

$u\_CYCLE\_TIME = 100$ : 250  $\mu s$  is set.

$u\_CYCLE\_TIME = 1900$ : 2000  $\mu s$  (2 ms) is set.

Inputs  $u\_WR\_PAR\_NR0$  to  $u\_WR\_PAR\_NR15$ :

You state the parameter numbers at inputs  $u\_WR\_PAR\_NR0$  to  $u\_WR\_PAR\_NR15$  of the specified values that are to be transferred cyclically. The system then cyclically transfers the specified values themselves in the assigned BYPASS task via block `BACI_PD_COMM`.

Inputs  $u\_RD\_PAR\_NR0$  to  $u\_RD\_PAR\_NR15$ :

You state the parameter numbers at inputs  $u\_RD\_PAR\_NR0$  to  $u\_RD\_PAR\_NR15$  of the actual values that are to be transferred cyclically. The system then cyclically transfers the actual values themselves in the assigned BYPASS task via block `BACI_PD_COMM`.

Input  $t\_TIME$ :

You set the monitoring time in seconds at input  $t\_TIME$ . If Input  $t\_TIME$  is not assigned, this yields a presetting of `TIME#3s`. If it is not possible to carry out initialization during the monitoring time, the system issues a corresponding timeout error message.

Output  $x\_OK$ :

Output  $x\_OK$  is set to `TRUE` if it was possible to correctly initialize process data communication between the `b maXX` controller and `b maXX` drive PLC via the BACI interface.

Outputs  $x\_ERR$ ,  $i\_ERR$ :

If an error occurs, the system sets error bit  $x\_ERR$  to `TRUE` and outputs error number  $i\_ERR$ .

Error number i\_ERR:

i_ERR	Error
-32768 to -1	Reserved
0	Error i_EVENT initialization
1	Timeout basic init b maXX controller: Wait for ConfDone bit or config flag bit.
2	Timeout Reconfig b maXX controller: Wait for ConfDone bit.
3	Timeout Reconfig b maXX drive PLC: Error in the PLC settings.
4	Input i_EVENT is not 0, 4, 8, 11 or 12.
5 to 32767	Reserved

### 5.3.3.2 BACI\_PAR\_READ

You can use this function block for BACI to read a service data value (a parameter value) from the b maXX controller via the BACI interface.



#### NOTE

FB BACI\_PAR\_READ uses library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above.

Application: Integrated preferably in the DEFAULT task or in any cyclic task.

Parameter input	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
u_PAR_NR	UINT	Parameter number
t_TIME	TIME	Monitoring time
x_EN	BOOL	Release
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
di_PAR_VALUE	DINT	Read parameter value
x_PAR_FORMAT	BOOL	Format of the parameter (16-/32-bit)
x_BUSY	BOOL	Communication is active
i_ERR	INT	Error number
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

#### Description

FB BAPS\_PAR\_READ sends a parameter read job to the b maXX controller via parameter number input u\_PAR\_NR. The system displays the parameter value that the b maXX controller requested at output ud\_PAR\_VALUE; the format of the parameter is displayed at output x\_PAR\_FORMAT. If an error occurs when executing the read parameter job, the system sets error bit x\_ERR and outputs error number i\_ERR.

FB BACI\_PAR\_READ can be set several times (instantiated) in the same way as FB BACI\_PAR\_WRITE. However, only one parameter job may be active at the time (i.e. only one FB BACI\_PAR\_READ or BACI\_PAR\_WRITE may be active at the time).

Input/output \_BASE:

At \_BASE, you must connect a global variable of data type BACI\_CTRL\_BMSTRUCT.



This global variable is predefined as `_PLC_Init` in the `BM4_O_PLC01` template in `PROPROG` wt in global variable sheet `Global_Variables`.

Alternatively, users can create this variable themselves.

In this case, you must locate it by means of declaration of global variables to the base address of the b maXX drive PLC's dual-port RAM, i.e. `%MB3.9000000`.

Example:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

Where:

`_PLC_Init` is the variable name with the data type short designation "\_" for Struct

`BACI_CTRL_BMSTRUCT` is the data type

`%MB3.9000000` is the base address of the b maXX drive PLC's dual-port RAM

Input `u_PAR_NR`:

At input `u_PAR_NR`, you state the parameter number of the parameter whose value you want to read.

Input `t_TIME`:

You set the monitoring time in seconds at input `t_TIME`. If Input `t_TIME` is not assigned, this yields a presetting of 3 s. If it is not possible to complete the job within the monitoring time, the system issues a timeout message.

Input `x_EN`:

Communication is started by means of `x_EN = TRUE`. If `x_EN` is set to `FALSE` before `x_BUSY=FALSE`, it is assumed that communication was cancelled deliberately. You must then reset FB `BACI_PAR_READ` by means of `x_RESET = TRUE`.

Input `x_RESET`:

You use `x_RESET = TRUE` to reset the FB.

Output `di_PAR_VALUE`:

The read parameter value is output at output `di_PAR_VALUE`.

Output `x_PAR_FORMAT`:

At output `x_PAR_FORMAT`, the system outputs the format of the parameter.

With a 16-bit parameter, `x_PAR_FORMAT = FALSE`; with a 32-bit parameter, `x_PAR_FORMAT = TRUE`.

In the case of other parameter formats (e.g. STRING parameter, not currently supported), `x_PAR_FORMAT = FALSE` stays the same and the system issues a number at `i_ERR`.

Output x\_BUSY:

Output x\_BUSY indicates by TRUE that communication is active.

Output x\_OK:

The system sets output x\_OK to TRUE when communication has been completed successfully.

Outputs x\_ERR, i\_ERR:

If an error occurs, the system sets error bit x\_ERR to TRUE and specifies the error at output i\_ERR.

Error number i\_ERR:

<b>i_ERR</b>	<b>Error</b>
-32768 to -3	Reserved
-2	Timeout
-1	Read parameter job on the BACI interface was canceled (possibly due to another instance of FB BACI_PAR_READ or FB BACI_PAR_WRITE)
0	No error
1 to 4	Internal message from b maXX controllers 1 - 4
5 to 127	Reserved
128 to 141	Internal message from b maXX controllers 128 - 141
142 to 255	Reserved
256 to 273	Internal message from b maXX controllers 256 - 273
274 to 511	Reserved
512	No data type can be determined
513	Data type is not supported
514	Data type is STRING (is not currently supported).
515 to 32767	Reserved

## 5.3.3.3 BACI\_PAR\_WRITE

You can use this function block for BACI to write a service data value (a parameter value) to the b maXX controller via the BACI interface.

**NOTE**

FB BACI\_PAR\_WRITE uses library BM\_TYPES\_20bd03 (PROPROP wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above.

Application: Integrated preferably in the DEFAULT task or in any cyclic task.

Parameter input	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
u_PAR_NR	UINT	Parameter number
di_PAR_VALUE	DINT	Parameter value
t_TIME	TIME	Monitoring time
x_EN	BOOL	Release
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
x_BUSY	BOOL	Communication is active
i_ERR	INT	Error number
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

**Description**

FB BACI\_PAR\_WRITE sends a write parameter job to the b maXX controller via parameter number input u\_PAR\_NR and parameter value di\_PAR\_VALUE. If an error occurs when executing the write parameter job, the system sets error bit x\_ERR and outputs error number i\_ERR.

FB BACI\_PAR\_WRITE can be set several times (instantiated) in the same way as FB BACI\_PAR\_READ. However, only one parameter job may be active at the time (i.e. only one FB BACI\_PAR\_READ or BACI\_PAR\_WRITE may be active at the time).

Input/output \_BASE:

At \_BASE, you must connect a global variable of data type BACI\_CTRL\_BMSTRUCT.

This global variable is predefined as \_PLC\_Init in the BM4\_O\_PLCO1 template in PROPROP wt in global variable sheet Global\_Variables.

Alternatively, users can create this variable themselves.

In this case, you must locate it by means of declaration of global variables to the base address of the b maXX drive PLC's dual-port RAM, i.e. %MB3.9000000.

Example:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

Where:

`_PLC_Init` is the variable name with the data type short designation "\_" for Struct

`BACI_CTRL_BMSTRUCT` is the data type

`%MB3.9000000` is the base address of the b maXX drive PLC's dual-port RAM

Input `u_PAR_NR`:

At input `u_PAR_NR`, you state the parameter number of the parameter whose value you want to write.

Input `di_PAR_VALUE`:

You state the parameter value to be written at input `di_PAR_VALUE`.

Input `t_TIME`:

You set the monitoring time in seconds at input `t_TIME`. If Input `t_TIME` is not assigned, this yields a presetting of 3 s. If it is not possible to complete the job within the monitoring time, the system issues a timeout message.

Input `x_EN`:

Communication is started by means of `x_EN = TRUE`. If `x_EN` is set to FALSE before `x_BUSY=FALSE`, it is assumed that communication was cancelled deliberately. You must then reset FB `BACI_PAR_WRITE` in each case by means of `x_RESET = TRUE`.

Input `x_RESET`:

You use `x_RESET = TRUE` to reset the FB.

Output `x_BUSY`:

Output `x_BUSY` indicates by TRUE that communication is active.

Output `x_OK`:

The system sets output `x_OK` to TRUE when communication has been completed successfully.

Outputs `x_ERR`, `i_ERR`:

If an error occurs, the system sets error bit `x_ERR` to TRUE and specifies the error at output `i_ERR`.

Error number i\_ERR:

i_ERR	Error
-32768 to -3	Reserved
-2	Timeout
-1	Write parameter job on the BACI interface was canceled (possibly due to another instance of FB BACI_PAR_READ or FB BACI_PAR_WRITE)
0	No error
1 to 4	Internal message b maXX controllers 1 to 4
5 to 127	Reserved
128 to 141	Internal message b maXX controllers 128 to 141
142 to 255	Reserved
256	Invalid parameter number
257	Invalid data type
258	Value less than minimum value
259	Value greater than maximum value
260	Parameter is read-only, i.e. you are not allowed to write parameters
261	Internal message b maXX controller 261 -
262	Parameter cannot be changed due to the operating status
263	Parameter value is invalid
264 to 273	Internal message b maXX controllers 264 to 511
274 to 281	Reserved
282	Write parameter impossible, because „BACI write access service data“ not enabled in drive manager (WinBASS II)
283 to 32767	Reserved

### 5.3.3.4 BACI\_PD\_COMM

You can use this function block for BACI to carry out process data communication between the b maXX controller and b maXX drive PLC via the BACI interface.



#### NOTE

FB BACI\_PAR\_COMM uses library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above.

Application: In a BYPASS task in which process data communication is to be carried out (i.e. event 4, 8, 11 or 12).

Parameter input	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
di_WR_VALUE0	DINT	Specified value 0
di_WR_VALUE1	DINT	Specified value 1
di_WR_VALUE2	DINT	Specified value 2
di_WR_VALUE3	DINT	Specified value 3
di_WR_VALUE4	DINT	Specified value 4
di_WR_VALUE5	DINT	Specified value 5
di_WR_VALUE6	DINT	Specified value 6
di_WR_VALUE7	DINT	Specified value 7
di_WR_VALUE8	DINT	Specified value 8
di_WR_VALUE9	DINT	Specified value 9
di_WR_VALUE10	DINT	Specified value 10
di_WR_VALUE11	DINT	Specified value 11
di_WR_VALUE12	DINT	Specified value 12
di_WR_VALUE13	DINT	Specified value 13
di_WR_VALUE14	DINT	Specified value 14
di_WR_VALUE16	DINT	Specified value 15
x_SYNC_CHECK	BOOL	Enable synchronization monitoring
x_EN	BOOL	Enable

Parameter output	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
di_RD_VALUE0	DINT	Actual value 0
di_RD_VALUE1	DINT	Actual value 1

Parameter output	Data type	Description
di_RD_VALUE2	DINT	Actual value 2
di_RD_VALUE3	DINT	Actual value 3
di_RD_VALUE4	DINT	Actual value 4
di_RD_VALUE5	DINT	Actual value 5
di_RD_VALUE6	DINT	Actual value 6
di_RD_VALUE7	DINT	Actual value 7
di_RD_VALUE8	DINT	Actual value 8
di_RD_VALUE9	DINT	Actual value 9
di_RD_VALUE10	DINT	Actual value 10
di_RD_VALUE11	DINT	Actual value 11
di_RD_VALUE12	DINT	Actual value 12
di_RD_VALUE13	DINT	Actual value 13
di_RD_VALUE14	DINT	Actual value 14
di_RD_VALUE15	DINT	Actual value 15
x_SYNC_ERR	BOOL	Synchronization error bit
i_ERR	INT	Error number
x_ERR	BOOL	Error bit

### Description

The specified values are sent to the b maXX controller; the b maXX controller receives the actual values and outputs them.

Input/output \_BASE:

At \_BASE, you must connect a global variable of data type BACI\_CTRL\_BMSTRUCT.

This global variable is predefined as \_PLC\_Init in the BM4\_O\_PLCO1 template in PROPROG wt in global variable sheet Global\_Variables.

Alternatively, users can create this variable themselves.

In this case, you must locate it by means of declaration of global variables to the base address of the b maXX drive PLC's dual-port RAM, i.e. %MB3.9000000.

Example:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

Where:

\_PLC\_Init

is the variable name with the data type short designation "\_" for Struct

BACI\_CTRL\_BMSTRUCT

is the data type

%MB3.9000000

is the base address of the b maXX drive PLC's dual-port RAM

Inputs di\_WR\_VALUE0 to di\_WR\_VALUE15:

At inputs di\_WR\_VALUE0 to di\_WR\_VALUE15, you connect the specified values whose parameter numbers were stated at FB BACI\_INIT (inputs u\_WR\_PAR\_NR0 to u\_WR\_PAR\_NR15) at initialization of process data communication between the b maXX controller and b maXX drive PLC via the BACI interface (or the parameter numbers that were changed via BACI\_PD\_RECONFIG).

Input x\_SYNC\_CHECK:

Using x\_SYNC\_CHECK = TRUE, you enable synchronization monitoring between the b maXX controller and b maXX drive PLC via the BACI interface. The default setting for x\_SYNC\_CHECK = TRUE, i.e. synchronization monitoring is enabled.

Input x\_EN:

Communication is enabled by means of x\_EN = TRUE. The default setting is x\_EN = TRUE, i.e. communication is enabled.

Outputs di\_RD\_VALUE0 to di\_RD\_VALUE15:

At outputs di\_RD\_VALUE0 to di\_RD\_VALUE15, the system outputs the actual values whose parameter numbers were stated at FB BACI\_INIT (inputs u\_RD\_PAR\_NR0 to u\_RD\_PAR\_NR15) at initialization of process data communication between the b maXX controller and b maXX drive PLC via the BACI interface (or the parameter numbers that were changed via BACI\_PD\_RECONFIG).

Output x\_SYNC\_ERR:

If you activated synchronization monitoring, the system monitors the Alive Counter. If a synchronization error occurs (i.e. one or more communications cycles are missing), the system sets synchronization error bit x\_SYNC\_ERR to TRUE.

Outputs x\_ERR, i\_ERR:

If an error occurs, the system sets error bit x\_ERR to TRUE and outputs error number i\_ERR.



Error number i\_ERR:

If errors occur, check the setting in WinBASS II "Synchronization" page

(Example: SyncOffset = -25.0 µs).

i_ERR	Error
-32768 to -6	Reserved
-5	BACI access conflict at actual value transfer: The b maXX controller has set its "data not valid flag" and "data access flag". => b maXX drive PLC is not allowed to read any actual values from the b maXX controller.
-4	BACI access conflict at actual value transfer: The b maXX controller has set its "data not valid flag" after b maXX drive PLC set its own "data access flag". => b maXX drive PLC is not allowed to read any actual values from the b maXX controller.
-3	BACI access conflict at actual value transfer: The b maXX controller has set its "data access flag". => b maXX drive PLC is not allowed to read any actual values from the b maXX controller.
-2	BACI access conflict at specified value transfer: The b maXX controller has set its "data not valid flag" after b maXX drive PLC set its own "data access flag". => b maXX drive PLC is not allowed to read any specified values from the b maXX controller.
-1	BACI access conflict at specified value transfer: The b maXX controller has set its "data access flag". => b maXX drive PLC is not allowed to read any specified values from the b maXX controller.
0	No error
1 to 32767	Reserved

### 5.3.3.5 BACI\_PD\_RECONFIG

You can use this function block for BACI to reconfigure process data communication between the b maXX controller and b maXX drive PLC via the BACI interface during ongoing operation.



#### NOTE

FB BACI\_PAR\_RECONFIG uses library BM\_TYPES\_20bd03 (PROPROG wt II) or BM\_TYPES\_30bd01 (ProProg wt III) or above.

During reconfiguration, FB BACI\_PD\_COMM must be disabled (x\_EN := FALSE).

By contrast with block BACI\_INIT, you can only change the parameter numbers; the cycle time and interrupt source that were set via BACI\_INIT are retained.

Application: For runtime reasons, ntegrated preferably in the DEFAULT task or in any cyclic task. Can also run in a BYPASS task (e.g. event 4, 8, 11 or 12).

Parameter input	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
u_WR_PAR_NR0	UINT	Specified value parameter number 0
u_WR_PAR_NR1	UINT	Specified value parameter number 1
u_WR_PAR_NR2	UINT	Specified value parameter number 2
u_WR_PAR_NR3	UINT	Specified value parameter number 3
u_WR_PAR_NR4	UINT	Specified value parameter number 4
u_WR_PAR_NR5	UINT	Specified value parameter number 5
u_WR_PAR_NR6	UINT	Specified value parameter number 6
u_WR_PAR_NR7	UINT	Specified value parameter number 7
u_WR_PAR_NR8	UINT	Specified value parameter number 8
u_WR_PAR_NR9	UINT	Specified value parameter number 9
u_WR_PAR_NR10	UINT	Specified value parameter number 10
u_WR_PAR_NR11	UINT	Specified value parameter number 11
u_WR_PAR_NR12	UINT	Specified value parameter number 12
u_WR_PAR_NR13	UINT	Specified value parameter number 13
u_WR_PAR_NR14	UINT	Specified value parameter number 14
u_WR_PAR_NR15	UINT	Specified value parameter number 15
u_RD_PAR_NR0	UINT	Actual value parameter number 0
u_RD_PAR_NR1	UINT	Actual value parameter number 1
u_RD_PAR_NR2	UINT	Actual value parameter number 2
u_RD_PAR_NR3	UINT	Actual value parameter number 3

Parameter input	Data type	Description
u_RD_PAR_NR4	UINT	Actual value parameter number 4
u_RD_PAR_NR5	UINT	Actual value parameter number 5
u_RD_PAR_NR6	UINT	Actual value parameter number 6
u_RD_PAR_NR7	UINT	Actual value parameter number 7
u_RD_PAR_NR8	UINT	Actual value parameter number 8
u_RD_PAR_NR9	UINT	Actual value parameter number 9
u_RD_PAR_NR10	UINT	Actual value parameter number 10
u_RD_PAR_NR11	UINT	Actual value parameter number 11
u_RD_PAR_NR12	UINT	Actual value parameter number 12
u_RD_PAR_NR13	UINT	Actual value parameter number 13
u_RD_PAR_NR14	UINT	Actual value parameter number 14
u_RD_PAR_NR15	UINT	Actual value parameter number 15
x_CFG_INIT	BOOL	Release
t_TIME	TIME	Monitoring time in seconds

Parameter output	Data type	Description
_BASE	BACI_CTRL_BMSTRUCT	Operating data for BACI
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

### Description

FB BACI\_PD\_RECONFIG allows you to reconfigure the specified value and actual value parameter numbers for process data communication between the b maXX controller and b maXX drive PLC in ongoing operation.

You can only carry out reconfiguration when FB BACI\_PD\_COMM is disabled (x\_EN := FALSE).

Input/output \_BASE:

At \_BASE, you must connect a global variable of data type BACI\_CTRL\_BMSTRUCT.

This global variable is predefined as \_PLC\_Init in the BM4\_O\_PLCO1 template in PROLOG wt in global variable sheet Global\_Variables.

Alternatively, users can create this variable themselves.

In this case, you must locate it by means of declaration of global variables to the base address of the b maXX drive PLC's dual-port RAM, i.e. %MB3.9000000.

Example:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

Where:

<code>_PLC_Init</code>	is the variable name with the data type short designation "_" for Struct
<code>BACI_CTRL_BMSTRUCT</code>	is the data type
<code>%MB3.9000000</code>	is the base address of the b maXX drive PLC's dual-port RAM

Inputs `u_WR_PAR_NR0` to `u_WR_PAR_NR15`:

You state the parameter numbers at inputs `u_WR_PAR_NR0` to `u_WR_PAR_NR15` of the specified values that are to be transferred cyclically via FB `BACI_PD_COMM`.

Inputs `u_RD_PAR_NR0` to `u_RD_PAR_NR15`:

You state the actual value parameter numbers at inputs `u_WR_PAR_NR0` to `u_WR_PAR_NR15` of the actual values that are to be transferred cyclically to inputs `u_RD_PAR_NR0` to `u_RD_PAR_NR15`.

Input `t_TIME`:

You set the monitoring time in seconds at input `t_TIME`. If Input `t_TIME` is not assigned, this yields a presetting of TIME 3s. If it is not possible to complete reconfiguration during the monitoring time, the system sets error bit `x_ERR`.

Eingang `x_CFG_INIT`:

`x_CFG_INIT = TRUE` starts reconfiguration.

Output `x_ERR`:

A timeout occurred during reconfiguration. Possible cause of the error: Block FB `BACI_PD_COMM` was not disabled at an `x_EN` or the system skips it.

Output `x_OK`:

Output `x_OK` is set to TRUE if the b maXX controller has taken over the new list of specified and actual value parameter numbers.

Now you can reset `x_CFG_INIT` and reenale process data communication via block FB `BACI_PD_COMM` (`x_EN = TRUE`).



## APPENDIX A ABBREVIATIONS

<b>AA</b>	Analog outputs function module	<b>DWort</b>	Doubleword (32-bit)
<b>Para.</b>	Paragraph	<b>EMK</b>	Electromagnetic constant
<b>AC</b>	Alternating Current	<b>EMC</b>	Electromagnetic compatibility
<b>ADR</b>	Address byte	<b>EN</b>	European standard
<b>AE</b>	Analog inputs function module	<b>EOF</b>	End of File
<b>AK</b>	Job/answer identifier	<b>EXT, ext</b>	External
<b>BACI</b>	Baumüller Component Interface	<b>FI</b>	Leakage current
<b>BAPS</b>	Baumüller drives parallel interface	<b>HLG</b>	Ramp function generator function module
<b>BASS</b>	Baumüller drives serial interface	<b>HM</b>	Main menu
<b>BB</b>	Ready for operation	<b>HW</b>	Highword
<b>BBext</b>	Ready for operation (external)	<b>I/O</b>	Input/Output
<b>BBint</b>	Ready for operation (internal)	<b>IKG</b>	Rotary encoder function module
<b>BEDAS</b>	Operating data store	<b>ID-Nr.</b>	Identification number
<b>BOF</b>	Begin of File	<b>Inc</b>	Count unit of the position
<b>BSE</b>	External reference for 24 V control input	<b>IND</b>	Index
<b>BUB</b>	Ballast unit	<b>Ink</b>	Number of scale marks of the rotary encoder
<b>BUC</b>	Baumüller feed/return feed unit	<b>INK.</b>	Incremental
<b>BUG</b>	Baumüller converter basic feed unit	<b>IW</b>	Actual value
<b>BUM</b>	Baumüller individual power unit	<b>IWK</b>	Actual value channel
<b>BUS</b>	Baumüller power module	<b>LED</b>	Light Emitting Diode
<b>CPU</b>	Central Processing Unit	<b>LW</b>	Lowword
<b>DA</b>	Digital/analog	<b>16#</b>	Prefix for hexadecimal numbers
<b>DAC</b>	Digital/analog converter	<b>mtr.</b>	Medium time lag (fuse)
<b>DB</b>	Data byte (8 bits)	<b>n = 0</b>	RPM speed = 0
<b>DC</b>	► direct current ► Drive Control	<b>NN</b>	Height above sea level
<b>DIN</b>	Deutsches Institut für Normung e.V. (German Standards Institute)	<b>PKE</b>	Parameter identifier
<b>DOPPELW</b>	Doubleword (32-bit)	<b>PKW</b>	Parameter identifier value
<b>DW</b>	Data word (16-bit)	<b>PZD</b>	Process data
		<b>R</b>	Reserved

---

<b>PE</b>	Protective earth
<b>SL</b>	PE conductor
<b>SM</b>	Synchronous motor
<b>STX</b>	Start of Text
<b>SW</b>	Specified value
<b>SWG</b>	Specified value generator function module
<b>SWK</b>	Specified value channel
<b>USS</b>	USS protocol function module
<b>USS<sup>®</sup></b>	Trademark of Siemens, universal serial interface
<b>VBG</b>	Management commercial employers' liability insurance association
<b>VDE</b>	Verband deutscher Elektrotechniker (German Association of Electrical Engineers)
<b>ZK</b>	Intermediate circuit



# Index

## B

b maXX drive PLC	
Firmware	49
BACI function blocks	
Overview	146
BACI interface	
Process data communication	158
Read parameter	152
Write parameter	155
Basic functionality	59
Baumüller	7
Board function	50

## C

CAM_PLC_21bd00	60
CAM_PLC_30bd00	60
Communication source	28
Communication via RS232	31
Control Dialog for Resources	34

## D

Data Area	28, 42
Directory path for libraries	49
Documentation worksheets	29

## E

Event Task	45
------------	----

## F

Firmware library	47
Function block	49
INTR_SET	51
TER_USS	103
TIME_MEASURE_END	50, 111
TIME_MEASURE_START	50, 110
TIMER_A_INIT	67

## G

Global variable worksheets	29
----------------------------	----

## I

I/O configuration	29
Initializing the BACI interface	147
INTR_SET	51

## L

LED display	16
LED function block	55
Level	
Interrupt	46, 52

## N

New project	27
-------------	----

## P

Personnel	13
qualified	13
Process data communication	147, 158
Reconfiguring	162
Programming languages	25
Project	
New, open	27
Property	
Event Task	46
PROPROG wt II	25
PROPROG wt II library	61

## Q

Qualified Personnel	13
---------------------	----

## R

Real-time response	45
REGISTER_PLC_20bd00	60
REGISTER_PLC_30bd00	60
Resource	28
Settings	31
Resource BM4_O_PLC01	27
Resources	
control dialog	34
Responsibility and liability	13

## S

Send project to target system	36
Specialist	13
Standard library	59
SYSTEM1_PLC_20bd00	50, 59
SYSTEM2_PLC_20bd00	59

## T

T64_RSYN	98
Technology component	
Cam disk	60
Register controller	60
Winder	60
TER_R	93
TER_S	96
TER_USS	103
Terms	
Definition	5
TIME_MEASURE_END	50, 111
TIME_MEASURE_START	50, 110
TIMER_A_INIT	67

## U

UNIVERSAL_20bd00	59
User library	47
Inserting in a project	60



## Index

---

### W

Warranty and Liability	14
Watchdog	40, 109, 112
monitoring	41
WINDER_PLC_20bd00	60
WINDER_PLC_30bd00	60

### Z

ZWT format file	61
-----------------	----





**be in motion**



All the information in these Operating Instructions 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. Note that all the data/numbers/information that are quoted are current values at the time of printing. This information is not legally binding for dimensioning, calculation and costing. Before using the information listed in these Operating Instructions as the basis for your own calculations and/or applications, make sure that you have the latest most current information. This means that we accept no responsibility for the accuracy of the information.