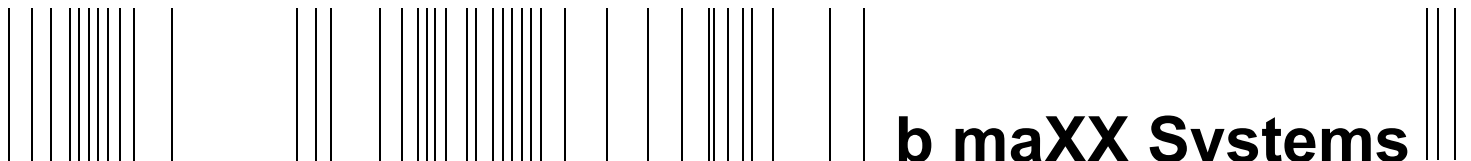


be in motion **be in motion**



b maXX Systems

b maXX controller PLC

Application Manual

E

5.04019.03



Title	Application Manual
Product	b maXX controller PLC (BMC-M-PLC-01/-02)
Last Revision:	15 February 2007
Art. no.	378555
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 current values at the time of printing. This information is not legally binding 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 www.baumueller.de</p>



TABLE OF CONTENTS

1	Introduction	5
1.1	First Steps	5
1.2	Terms Used	5
2	Basic Safety Instructions	7
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
3	Commissioning	15
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 conditions of BMC-M-PLC-01	16
3.4.2	S1 switch/pushbutton for changing operating status conditions of the BMC-M-PLC-01.	18
3.4.3	LEDs for displaying operating status conditions of BMC-M-PLC-02	19
3.4.4	Rotary switch S1 of BMC-M-PLC-02	22
3.4.5	S2 switch/pushbutton for changing operating status conditions of BMC-M-PLC-02	22
3.5	Commissioning sequence	23
3.5.1	Activation of BMC-M-PLC-01	23
3.5.2	Testing the function of BMC-M-PLC-01	23
3.5.3	Activation of BMC-M-PLC-02	24
3.5.4	Testing the function of BMC-M-PLC-02	25
3.5.5	Sending a boot project	25
4	Internal Communication Interface to the System Components	29
4.1	CBPB (Controller Based Parallel Bus)	29
4.2	I/O-Bus	29
4.3	Transmission time	30
5	Operation	31
5.1	Programming systems PROPROG wt II and ProProg wt III	31
5.2	b maXX controller PLC project	33
5.3	b maXX controller PLC configuration	34
5.4	b maXX PLC resource	36
5.4.1	Communication and Connection	37
5.4.2	Control Dialog for Resources	41
5.4.3	Data Area	49



TABLE OF CONTENTS

5.4.4	b maXX controller PLC event tasks	52
5.5	b maXX controller PLC user libraries	54
5.5.1	b maXX controller PLC firmware	56
5.5.2	b maXX controller PLC Board functions	57
5.5.3	b maXX PLC data types	65
5.5.4	The Standard Function Block Libraries	69
5.5.5	b maXX PLC technology components	70
5.5.6	Inserting a User Library into a Project	71
5.6	CBPB system description for the b maXX controller PLC	73
5.6.1	Direct access to the CBPB hardware from the b maXX controller PLC	73
5.6.2	Process data communication between b maXX controller PLC and system components, synchronization to a common CBPB hardware signal source	75
5.6.3	b maXX controller PLC interrupt sources that the application can use	76
5.6.4	Generating CBPB signals via function block TIMER_A_INIT	78
5.6.5	"TIMER_A_INIT" function block	79
5.6.6	Sample Configurations	90
5.6.7	Example A: Setting up a CPU-timer 1 interrupt	90
5.6.8	Example B: Setting up a board timer A interrupt and triggering of the SYNC-signal 1	91
5.6.9	Example C: Implementing simple serial RS485 communication via a CPU timer	95
5.6.10	Description of the blocks for serial RS485 communication via the X2 interface (9-pin female Sub-D connector) on the b maXX controller PLC	97
5.6.11	Function block TER_INIT	99
5.6.12	Function block TER_R	104
5.6.13	Function block TER_S	108
5.6.14	Function block T64_RSYN	111
5.6.15	Function block T64_REC	112
5.6.16	Function block TER_USS	115
5.7	Code runtimes	122
5.7.1	Function block TIME_MEASURE_START	124
5.7.2	Function block TIME_MEASURE_END	125
5.8	Practical information	126
5.8.1	General PLC error	126
5.8.2	PLC error "Watchdog in task 'x' exceeded!"	126
5.8.3	PLC error "Global ready message (RST signal) is missing!"	127
5.8.4	PLC error "No access to the CX-controller in the power supply unit possible!"	128
5.8.5	PLC error "I/O bus error occurs! Error code: ..."	128
6	Utilize modules (to the right of the b maXX controller PLC)	131
6.1	ProMaster	131
6.1.1	Create a ProProg wt III project for the b maXX controller PLC	132
6.1.2	Creating a ProMaster project using b maXX controller PLC	134
6.1.3	Configuring the b maXX System using ProMaster	138
6.1.4	Configuring the I/Os with ProMaster	139
6.2	ProModul module configurator	146
6.2.1	Create a PROPROG wt II project for the b maXX controller PLC	147
6.2.2	Configuration of the b maXX system with the module configurator	148
6.3	Configuration of the system components / modules	153
6.3.1	Digital inputs / digital outputs	153
6.3.2	Analog inputs / analog outputs	155
6.3.3	Alarm output	157
	Anhang A - Abbreviations	159
	Revision index	161
	Index	163

1

INTRODUCTION

This b maXX controller PLC Application Manual is an important component of your b maXX system; 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 controller PLC (BMC-M-PLC-01) Operating Instructions and the power supply unit (BMC-M-PSB-01) 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 module to the appropriate departments in your company.
- ◆ Deploy suitable personnel for assembly and commissioning.
- ◆ Pass on the 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 controller PLC**" product as "PLC", or "Module". Furthermore we use the terms „BMC-M-PLC-01“ and „BMC-M-PLC-02“ respectively.

For a list of the abbreviations that are used, refer to [▶Appendix A Abbreviations◀](#) from page 159 onward.

In this documentation, we will also refer to the product „Power supply unit for b maXX controller PLC“ as „module“ or „power supply unit“.

We will also use the term "b maXX system" for a product consisting of „Power supply unit“, „controller PLC“ and more components.

BASIC SAFETY INSTRUCTIONS

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

In this chapter, we will describe the risks that can occur when working with a Baumüller 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



Hazard information will show you the dangers, that can lead to injuries or even to death. Always follow the hazard information given in this document.

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

DANGER

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

WARNING

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

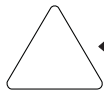
CAUTION

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

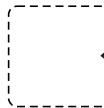
2.1 Hazard information and instructions

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



wear safety shoes

2.2 Information signs



NOTE

This indicates particularly important information.

2.3 Legal information

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

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

Operators are responsible for carrying out servicing and commissioning in accordance with the safety regulations, applicable standards and any and all other relevant national or local regulations with regard to cable rating and protection, grounding, isolators, 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 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 module. The information below has no claim to being complete; always observe all the information that is given in these operating instructions.

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

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

2.5 Inappropriate Use

Below, we will list some examples of inappropriate use. The information below should give you an idea of what is meant by inappropriate use of the 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 added the module on an other unit/module as the power supply unit for b maXX controller PLC.
- You ignored information in these operating instructions.
- You did not use the module as intended.
- You handled the 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 module without written permission from Baumüller Nürnberg GmbH.
- You ignored the maintenance instructions in the component descriptions.
- You incorrectly combined the module with third-party products.
- You combined the b maXX 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 modules are protected by their packaging. Do not remove the module from its packaging until just before you intend to mount it.

The housing of the module provides IP20 protection to the modules from dirt and damage due to static discharges from contact. This means that you never use a module with damaged housing.

2.7 Personnel training



WARNING

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

- serious personal injury
- death

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

Qualified personnel (specialists) are defined as follows:

Qualified Personnel

Electrical engineers and electricians of the customer or of third parties who are authorized by Baumüller Nürnberg GmbH and who have been trained in installing and commissioning Baumüller b maXX 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 b maXX 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 b maXX systems' place of installation, observe the applicable safety regulations for the plant in which this unit is installed.
- ▶ Provide the b maXX system with additional monitoring and protective equipment if the safety regulations demand this.
- ▶ Observe the safety measures for the unit in which the module is installed.

2.9 Responsibility and liability

To be able to work with this 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 these Operating Instructions and must observe the instructions and information they contain – this applies in particular to the safety instructions.

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

2.9.2 Danger arising from using this module

The b maXX controller PLC 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.

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 controller PLC module that you just assembled and installed (see "b maXX PLC Operating Instructions"). Commissioning ensures that the module functions correctly.

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

- 1 The module has been assembled correctly.
- 2 The module has been installed correctly.
- 3 All the safety equipment has been commissioned.
- 4 The b maXX system 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, parts of the machine/system or the complete machine/system can move.*

Keep far enough the moving parts of the machine/system. Note that from the connected (to the b maXX controller PLC) other modules 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 controller PLC module, you must ensure that the 24 V power supply is connected to the power supply unit for b maXX controller PLC and that the power supply is in accordance with the specifications. You must not continue with commissioning until you have ensured this.

Observe during commissioning that the b maXX controller PLC can be commissioned only together with the power supply unit and if necessary other system components.

3.4 Description and inspection of the controls and displays

3.4.1 LEDs for displaying operating status conditions of BMC-M-PLC-01

The b maXX controller PLC module has eight LEDs (four green (H1, H3, H5, H7) and four red (H2, H4, H6, H8)) as display elements.

These LEDs are used by the operating system of the b maXX controller PLC during initialisation.

All system components of the b maXX system must have reached a specific internal operating status (global ready message) after switching in the supply voltage before they may be actuated by the b maXX controller PLC

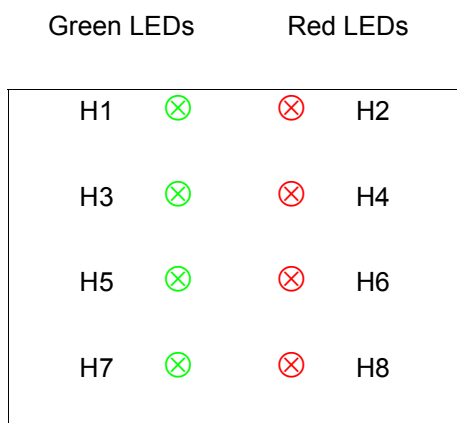


Figure 1: LEDs of the b maXX controller PLC module (BMC-M-PLC-01)

3.4.1.1 Switching on and initialisation of BMC-M-PLC-01

- After switching on the b maXX controller PLC module waits for the global ready message of the following modules:
 - Power supply unit for b maXX controller PLC
 - Modules plugged in left handed of the controller PLC (e.g. Ethernet with CANopen-Master module)

This stage is indicated by a counterclockwise-rotating bit pattern. This means that an LED lights up every 500 ms in the sequence H1 (green) → H3 (green) → H4 (red) → H2 (red) → H1 (green) etc.

If the global ready message is not issued within of max. 8 s, LED H6 (red) is switched on.

To remove the cause of faults see [▶ Troubleshooting and eliminating errors](#) in „Operating Instructions b maXX controller PLC“.

- After the global ready message of the modules the b maXX controller PLC initializes the I/O bus.

This stage is indicated by a clockwise-rotating bit pattern. This means that an LED lights up every 500 ms in the sequence H1 (green) → H2 (red) → H4 (red) → H3 (green) → H1 (green) etc.

If the I/O bus cannot be initialized within 8 s or an error occurs at I/O bus initialization, LED H8 (red) is switched on.

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 controller PLC module can, in principle, carry out PRO-PROG communication via the serial RS232 port X1.

From now on, PROPROG communication is also possible by means of TCP/IP if a module with Ethernet functionality (e.g. Ethernet with CANopen-Master module) is placed on the left hand of the b maXX controller PLC and has been configured for communication with the b maXX controller PLC module.

- If a boot project is present, the system now loads it. The top two LEDs (H2 and H1) flash rapidly to show that the boot project is being loaded.



NOTE

If the global ready message of the modules fails (H6 (red) is switched on) **no** user program can run. A PLC error message is generated. The PLC remains in status „STOP“.

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

- No project available, status "POWER ON":
→ LEDs H3 (green) and H4 (red) light up.
- Project available, status "STOP":
→ Only LED H4 (red) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:
→ Only LED H3 (green) lights up.
- Project available, status "RUN":
→ LEDs H1 (green) and H3 (green) light up.
- Global ready message fails:
→ LED H6 (red) lights up, PLC does **not** go in status „RUN“.
- I/O bus cannot be initialized:
→ LED H8 (red) lights up.

3.4 Description and inspection of the controls and displays

In the "RUN" status, users can freely program the eight LEDs. For information on programming, see the b maXX controller PLC Application Manual in the chapter entitled "b maXX controller PLC Board Functions / Function Block LED8".



NOTE

If the I/O bus initialization was not successful (H8 (red) is switched on), the boot project is loaded and the user program run.

I.e. you could use the b maXX controller PLC and the other modules without the I/O bus modules.

3.4.1.2 Operation of BMC-M-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":
→ LEDs H3 (green) and H4 (red) light up.
- Project available, status "STOP":
→ Only LED H4 (red) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:
→ Only LED H3 (green) lights up.
- Project available, status "RUN":
→ LEDs H1 (green) and H3 (green) light up.
- Global ready message fails:
→ LED H6 (red) lights up, PLC does **not** go in status „RUN“.
- I/O bus cannot be initialized:
→ LED H8 (red) lights up.

In the "RUN" status, users can freely program the eight LEDs. For information on programming, see the b maXX controller PLC Application Manual in the chapter entitled "b maXX controller PLC Board Functions / Function Block LED8".



NOTE

If the I/O bus initialization was not successful (H8 (red) is switched on), the boot project is loaded and the user program run.

I.e. you could use the b maXX controller PLC and the other modules without the I/O bus modules.

3.4.2 S1 switch/pushbutton for changing operating status conditions of the BMC-M-PLC-01

For changing the operating status' there is a switch/pushbotton on the b maXX controller PLC module.

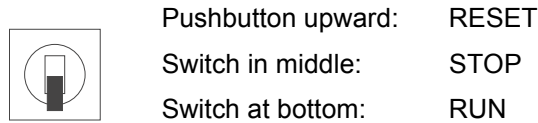


Figure 2: Switch S1 on the b maXX controller PLC module (BMC-M-PLC-01)



NOTE

The user project can only start when switch/pushbutton S1 is in the bottom "RUN" position and all modules have sent their global ready message to the b maXX controller PLC (H6 (red) is switched off).

The pushbutton upward resets the „b maXX controller PLC module“, the „power supply unit for b maXX controller PLC“ and the other system components (i.e. modules on left hand of the b maXX controller PLC and on right hand of the power supply unit).



DANGER

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

- serious personal injury
- death

Danger from: **mechanical effects**. *At commissioning of the b maXX controller PLC and of the connected power supply unit (and other connected system components) with a complete application program can be started the machine/system or parts of the machine/system. Due to a wrong address on the power supply unit (see [▶Operating Instructions Power Supply Unit for b maXX controller PLC](#) [◀](#)) the machine/system or parts of the machine/system can behave unexpectedly.*

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

3.4.3 LEDs for displaying operating status conditions of BMC-M-PLC-02

The b maXX controller PLC module has twelve LEDs (five green ones (H1, H3, H5, H7, H11), five red ones (H2, H4, H6, H8, H10), one blue (H9) and one orange/green (H12)) as display elements.

These LEDs are used by the operating system of the b maXX controller PLC during initialisation.

The operator can use the LEDs in the application program on the b maXX controller PLC module during operation (after the start-up phase of the operating system).

All system components of the b maXX system must have reached a specific internal operating status (global ready message) after switching in the supply voltage before they may be actuated by the b maXX controller PLC.

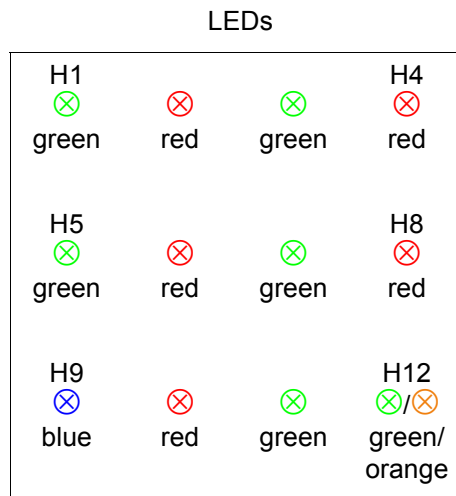


Figure 3: LEDs of the b maXX controller PLC module (BMC-M-PLC-02)

3.4.3.1 Switching on and initialisation of BMC-M-PLC-02

- After switching on a basis initialization of the b maXX controller PLC module is carried out, which is required for further initialization. At the end of the basis initialization H9 (blue) is switched on.
- Afterwards the b maXX controller PLC module waits for the global ready message of the following modules:
 - Power supply unit for b maXX controller PLC
 - Modules plugged in left handed from the controller PLC (e.g. Ethernet with CANopen-Master module)

This stage is indicated by a counterclockwise-rotating bit pattern. This means that an LED lights up every 500 ms in the sequence H1 (green) → H5 (green) → H6 (red) → H2 (red) → H1 (green) etc. H9 (blue) is switched off.

If the global ready message of the modules is not issued within max. 8 s, LED H4 (red) is switched on.

To remove the cause of errors see ▶ Operating instructions b maXX controller PLC, section „Troubleshooting and eliminating errors“.

- After the global ready message of the modules the b maXX controller PLC initializes the I/O-bus.

This stage is indicated by a clockwise-rotating LED pattern. This means that a LED lights up every 500 ms in the sequence H1 (green) → H2 (red) → H6 (red) → H5 (green) → H1 (green) etc.

If the I/O-bus cannot be initialized within 8 s or an error occurs at I/O-bus initialization, LED H8 (red) is switched on.

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 controller PLC module can, in principle, carry out PRO-PROG communication via the serial RS232 port X1.

From now on, PROPROG communication is also possible by means of TCP/IP if a module with Ethernet functionality (e.g. Ethernet with CANopen-Master module) is placed on the left hand of the b maXX controller PLC and has been configured for communication with the b maXX controller PLC module.

- If a boot project is present, the system now loads it. The two LEDs (H1 and H2) flash rapidly to show that the boot project is being loaded.

**NOTE**

If the global ready message of the modules fails (H4 (red) is switched on) **no** user program can run. A PLC error message is generated. The PLC remains in status „STOP“.

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

- No project available, status "POWER ON":
→ LEDs H5 (green) and H6 (red) light up.
- Project available, status "STOP":
→ Only LED H6 (red) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:
→ Only LED H5 (green) lights up.
- Project available, status "RUN":
→ LEDs H1 (green) and H5 (green) light up.
- Global ready message fails:
→ LED H4 (red) lights up, PLC does **not** go in status „RUN“.
- I/O bus cannot be initialised:
→ LED H8 (red) lights up.

**NOTE**

If the I/O-bus initialization was not successful (H8 (red) is switched on), the boot project is loaded and the user program run.

I.e. you could use the b maXX controller PLC and the other modules without the I/O-bus modules.

3.4.3.2 Operation of BMC-M-PLC-02

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

- No project available, status "POWER ON":
→ LEDs H5 (green) and H6 (red) light up.
- Project available, status "STOP":
→ Only LED H6 (red) lights up.
- Project available, status "INIT", the controller is at the cold boot or warm restart stage:
→ Only LED H5 (green) lights up.

3.4 Description and inspection of the controls and displays

- Project available, status "RUN":
→ LEDs H1 (green) and H5 (green) light up.
- Global ready message fails:
→ LED H4 (red) lights up, PLC does **not** go in status „RUN“.
- I/O bus cannot be initialised:
→ LED H8 (red) lights up.

In the "RUN" status, users can freely program the twelve LEDs. For information on programming, see the b maXX controller PLC Application Manual in the chapter entitled "b maXX controller PLC Board Functions / Function Block LED12".



NOTE

If the I/O-bus initialization was not successful (H8 (red) is switched on), the boot project is loaded and the user program run.

I.e. you could use the b maXX controller PLC and the other modules without the I/O-bus modules.

3.4.4 Rotary switch S1 of BMC-M-PLC-02

The rotary switch S1 of BMC-M-PLC-02 is provided for future developments and is not used now.

3.4.5 S2 switch/pushbutton for changing operating status conditions of BMC-M-PLC-02

For changing the operating status' there is a switch/pushbutton S2 on the b maXX controller PLC module.



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

Figure 4: Switch S2 on the b maXX controller PLC module (BMC-M-PLC-02)



NOTE

The user project can only start when switch/pushbutton S2 is in the bottom "RUN" position and all modules have sent their global ready message to the b maXX controller PLC (H4 (red) is switched off).

The pushbutton upward resets the „b maXX controller PLC module“, the „power supply unit for b maXX controller PLC“ and the other system components (i.e. modules on left hand of the b maXX controller PLC and on right hand of the power supply unit).

**DANGER**

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

- serious personal injury
- death

Danger from: **mechanical effects**. *At commissioning of the b maXX controller PLC and of the connected power supply unit (and other connected system components) with a complete application program can be started the machine/system or parts of the machine/system. Due to a wrong address on the power supply unit (see ▶Operating Instructions Power Supply Unit for b maXX controller PLC ◀) the machine/system or parts of the machine/system can behave unexpectedly.*

Keep far enough the moving parts of the machine/system. Note that from the (to the b maXX controller PLC) other modules which are connected, 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 of BMC-M-PLC-01

- 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 controller PLC◀).
- Set switch/pushbutton S1 on the b maXX controller PLC module to "STOP" (center position).
- Switch on the 24 V DC of the power supply unit.

**NOTE**

You must not connect the b maXX controller PLC module with other system components or disconnect them, if the 24 V DC power supply is switched on. First switch the 24 V DC power supply off.

3.5.2 Testing the function of BMC-M-PLC-01

- After you switch on the b maXX system, two status conditions can apply:
 - No boot project (= no user project on the b maXX controller PLC): LED H2 (red) lights up briefly after a short time, the LEDs H3 (green) and H4 (red) are permanently lit up. This means that no project is present on the b maXX controller PLC. In this "POWER ON" status, the b maXX controller PLC is waiting for PRO-PROG 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 (red) lights up. The b maXX controller PLC is in the "STOP" status.
- While switch/pushbutton S1 on the b maXX controller PLC 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 controller PLC module to the "RUN" (bottom) position, **first** ensure that you have loaded the **correct** boot project for your application for **this** plant in **this** b maXX controller PLC and the correct address is preset at the power supply unit!

Refer to the b maXX controller 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 b maXX controller PLC, for example

DANGER



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

- serious personal injury
- death



Danger from: **mechanical effects**. *At commissioning of the b maXX controller PLC and of the connected power supply unit (and other connected system components) with a complete application program can be started the machine/system or parts of the machine/system. Due to a wrong address on the power supply unit (see ►Operating Instructions Power Supply Unit for b maXX controller PLC ◄) the machine/system or parts of the machine/system can behave unexpectedly.*

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

3.5.3 Activation of BMC-M-PLC-02

- Read and observe the ►Basic Safety Instructions ◄ from page 7 onward.
- You must have carried out correctly section "Assembly and Installation" (see ►Operating instructions b maXX controller PLC BMC-M-PLC-01/-02◄).
- Set switch/pushbutton S2 on the b maXX controller PLC module to "STOP" (center position).
- Switch on the 24 V DC of the power supply unit.

NOTE



You must not connect the b maXX controller PLC module with other system components or disconnect them, if the 24 V DC power supply is switched on. Switch the 24 V DC power supply off first.

3.5.4 Testing the function of BMC-M-PLC-02

- After you switch on the b maXX system, two status conditions can apply:
 - No boot project (= no user project on the b maXX controller PLC):
LED H2 (red) lights up briefly after a short time, the LEDs H5 (green) and H6 (red) are permanently lit up.
This means that no project is present on the b maXX controller PLC.
In this "POWER ON" status, the b maXX controller PLC is waiting for PROPROG 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 H6 (red) lights up. The b maXX controller PLC is in the "STOP" status.
- While switch/pushbutton S2 on the b maXX controller PLC 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 S2 on the b maXX controller PLC module to the "RUN" (bottom) position, **first** ensure that you have loaded the **correct** boot project for your application for **this** plant in **this** b maXX controller PLC and the correct address is preset at the power supply unit!

Refer to the b maXX controller 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 b maXX controller PLC, for example

3.5.5 Sending a boot project

- 1 Connect your PC's serial port to the female Sub-D connector on the **b maXX controller PLC (X1)**.
- 2 Start the PROPROG wt II (or ProProg wt III) operation software on your PC and choose a project for b maXX controller PLC. This project must have already been successfully compiled. You can also recompile the project using menu item "Code -> Regenerate project".
- 3 In the operator control program, click on „Resource control“ in directory „Online“.

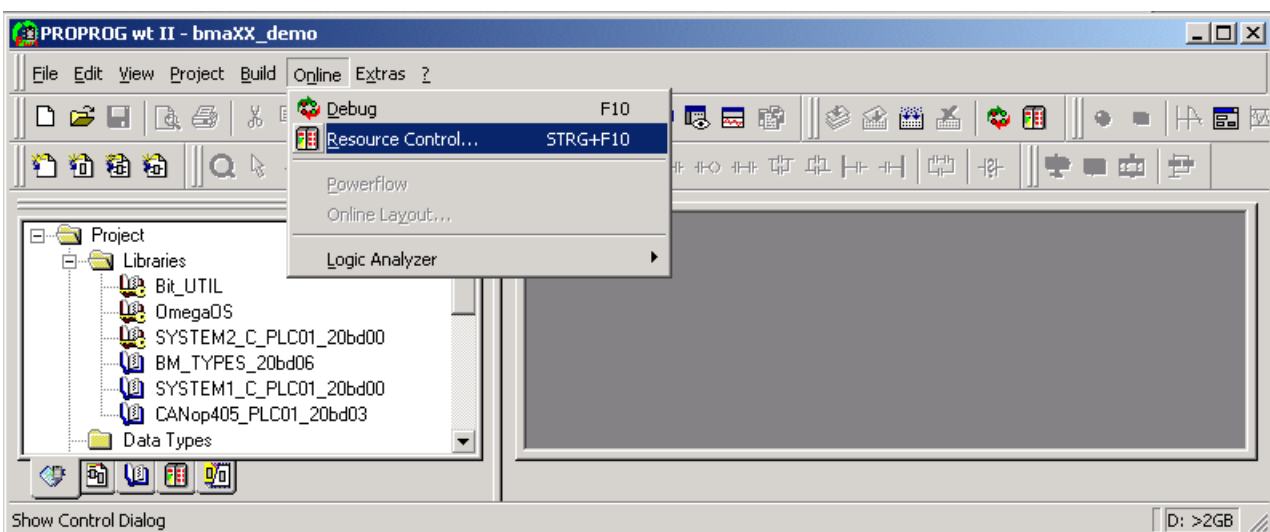


Figure 5: PROPROG wt II user interface

The system opens the following window:



Figure 6: Resource control "On"

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

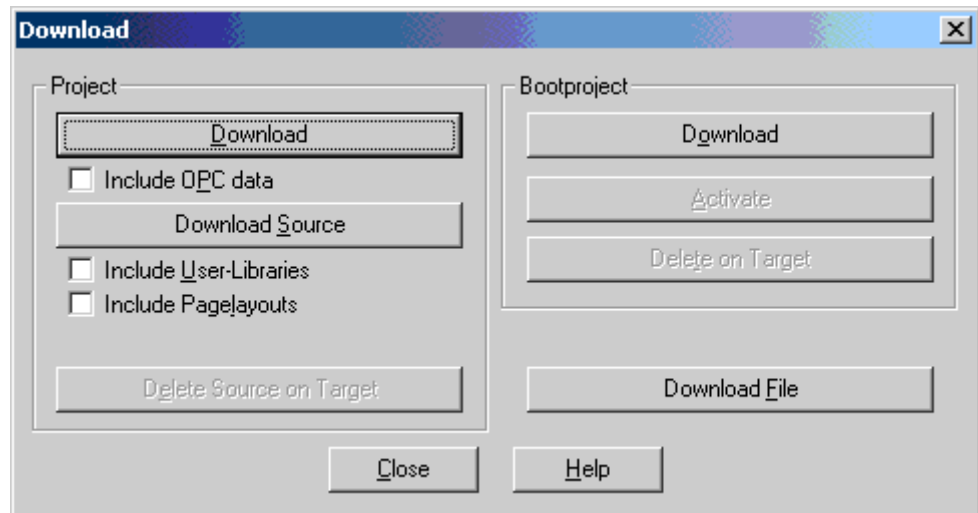


Figure 7: Send

5 Click on "Download boot project".

- o 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 controller PLC-01 the LEDs H1 up to H8 flash at the same time. On the b maXX controller PLC-02 the LEDs H1 up to H12 flash at the same time.

- o 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 controller PLC-01 the LEDs H1 (green), H2 (red), H3 (green) and H4 (red) flash diagonally during transfer. On the b maXX controller PLC-02 the LEDs H1 (green), H2 (red), H5 (green) and H6 (red) flash diagonally during transfer.

**NOTE**

After sending is completed, you should switch the b maXX system off and back on so that the b maXX controller PLC can correctly recognize the global ready message of the other modules (modules placed left hand of the b maXX controller PLC and power supply unit).

Alternatively, it may be enough to start the project on the b maXX controller PLC either by means of menu item "Download" / "Activate boot project" and "Cold boot" or by resetting on the b maXX controller 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 controller PLC using resource control. As an alternative, you can achieve the same effect using switch S1 (BMC-M-PLC-01) and S2 (BMC-M-PLC-02) respectively, on the b maXX controller 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 PLC and other modules.



Figure 8: "Operation" resource control

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

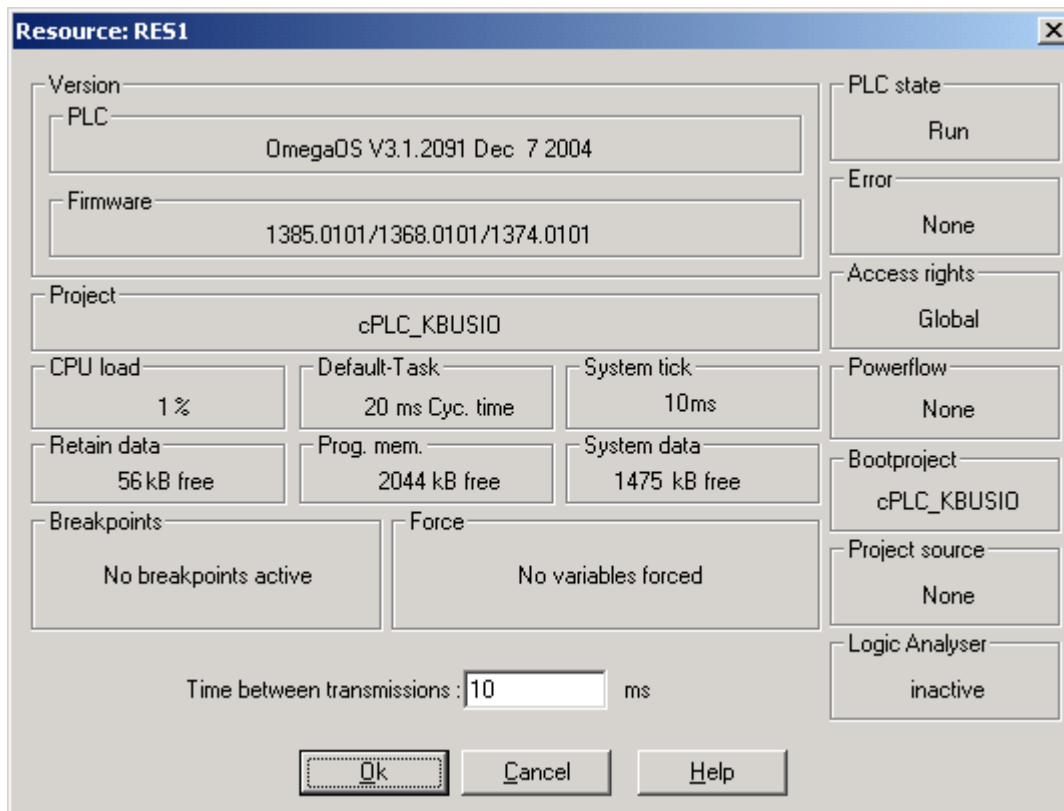


Figure 9: Resource information window

NOTE



Meanings of the displayed versions strings in the information window:

PLC string: "OmegaOS V3.1.2091 Dec 7 2004"
 -> "OmegaOS": Product designation
 -> "V3.1.2091": OmegaOS compiler information
 -> "Dec 7 2004": Date on which the PLC version was created.

Firmware string: "1385.0101/1368.0101/1374.0101"
 -> "1385.0101": Version OmegaOS software 6.1385.0101
 -> "1368.0101": Version FPGA software 6.1368.0101
 -> "1374.0101": Version BOOT software 6.1374.0101"

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 [▶Figure 6](#) on page 26).

INTERNAL COMMUNICATION INTERFACE TO THE SYSTEM COMPONENTS

The communication to the system components (modules) placed left hand of the b maXX controller PLC will be carried out via CBPB (Controller Based Parallel Bus).

The communication to the system components placed right hand of the b maXX controller PLC will be carried out via I/O-bus.

4.1 CBPB (Controller Based Parallel Bus)

The CBPB is an internal communication interface for data exchange between b maXX controller PLC and the system components placed left hand of the b maXX controller PLC.

The access to the system components (placed left hand of the b maXX controller PLC) are displayed as accesses to the DPRAM of the respective system components for the user.

A parametrization of the interface is not necessary.

Instructions for programming and/or parametrization of the respective system component you get in the application manuals of the respective system components and in the online help of the b maXX configurator.

4.2 I/O-Bus

The I/O bus is an internal communication interface for data exchange between b maXX controller PLC and the system components / modules placed right hand of the b maXX controller PLC.

The accesses to the system components (placed right hand of the b maXX controller PLC) are displayed as access to the inputs and outputs of the respective system components for the user.

A parametrization of the interface is not necessary.

4.3 Transmission time

The inputs and outputs are available as a flag image in the PLC. By default the data communication from and to the IO bus is carried out with the system tick cycle (see [► b maXX PLC resource ◀](#) from page 36 onward).

If in the IEC application an input variable is copied to an output variable, then the time until a change of the digital input results in the digital output, is the maximum transmission time t_{\max} .

$$t_{\max} = t_{id} + 2 \cdot t_{st} + t_{cyc} + (t_{IO} + 10\%)$$

whereas	t_{id}	Input delay of the digital input
	t_{st}	System tick
	t_{cyc}	Sampling interval calling the copy command cyclic
	t_{IO}	IO bus cycle time

The data communication from and to the IO bus should be synchronized on a cyclic event by means of the function block IO_BUS_CYCLE. In this case the data communication is started at the moment of the function block call. The maximum transmission time from the digital input to the digital output is as following:

$$t_{\max} = t_{id} + 3 \cdot t_{cyc} + (t_{IO} + 10\%)$$

whereas	t_{id}	Input delay of the digital input
	t_{cyc}	Sampling interval calling the copy command cyclic
	t_{IO}	IO bus cycle time

The IO bus cycle time (in μs) can be read out from the flag (UINT) %MW3.110036 for the existing configuration.

Examples:

IO-Bus with 1*DI8000 and 1*DO8000

The copy command is located in a 4 ms timer interrupt.

1. Standard data communication

t_{id}	3 ms	Input delay of the digital input DI8000
t_{st}	10 ms	System tick (PLC-01)
t_{cyc}	4 ms	Sampling interval for the copy command
t_{IO}	422 μs	IO bus cycle time (from %MW3.110036)

$$t_{\max} = 3 \text{ ms} + 2 \cdot 10 \text{ ms} + 4 \text{ ms} + (0,422 \text{ ms} + 0,042 \text{ ms}) = 27,464 \text{ ms}$$

2. Data communication with the function block IO_BUS_CYCLE in the same 4 ms timer interrupt as the copy command.

t_{id}	3 ms	Input delay of the digital input DI8000
t_{cyc}	4 ms	Sampling interval for the copy command
t_{IO}	422 μs	IO bus cycle time (from %MW3.110036)

$$t_{\max} = 3 \text{ ms} + 3 \cdot 4 \text{ ms} + (0,422 \text{ ms} + 0,042 \text{ ms}) = 15,464 \text{ ms}$$

OPERATION

5.1 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 controller PLC-01 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. A boot project on the b maXX controller PLC-02 starts up, if the switch/pushbutton S2 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. Due to a wrong address on the power supply unit (see ▶Operating Instructions Power Supply Unit for b maXX controller PLC◀) the machine/system or parts of the machine/system can behave unexpectedly.

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

5.2 b maXX controller PLC project

You start a new maXX controller PLC project under PROPROG wt by means of menu item "File/New project". Using the "Template for BMC_M_PLCC01", you open a b maXX controller PLC project. Processor type BMC_M_PLCC01 is preset in this project in the configuration.

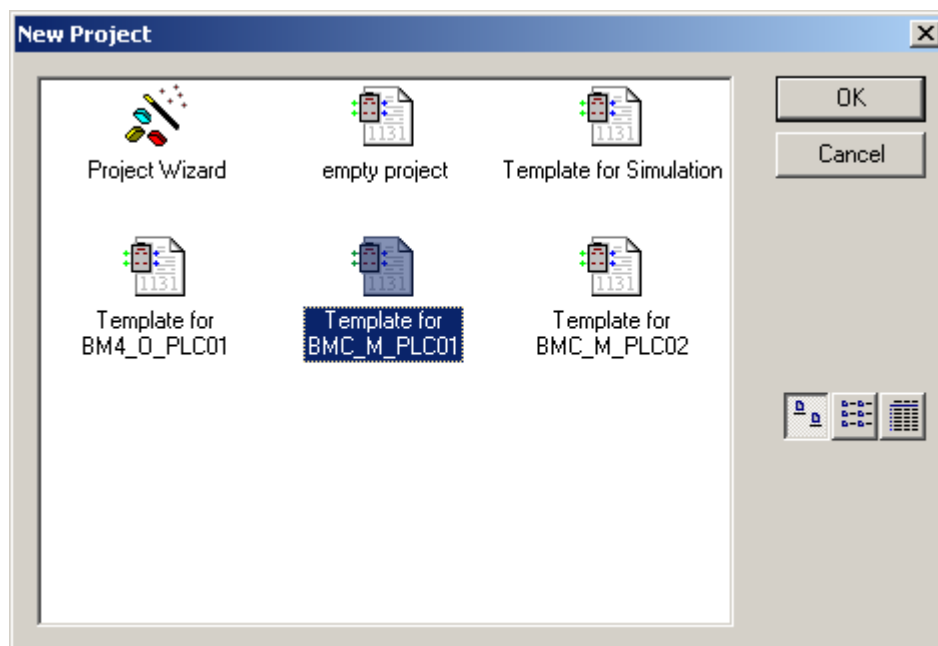


Figure 10: BMC_M_PLCC01 project template under "New Project".

Using the "Template for BMC_M_PLCC02", you open a b maXX controller PLC project. Processor type BMC_M_PLCC02 is preset in this project in the configuration.

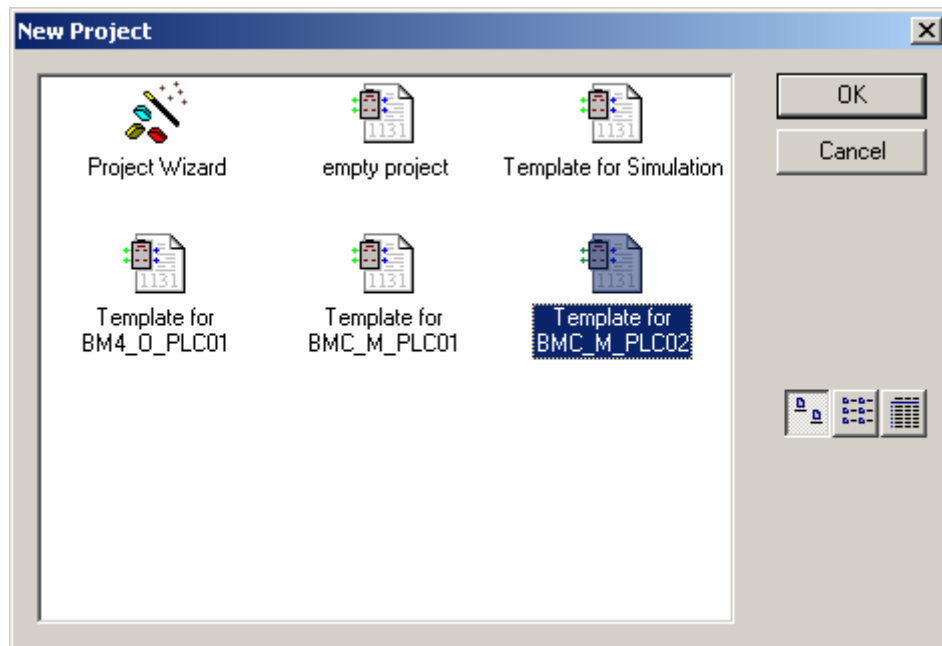


Figure 11: BMC_M_PLCO2 project template under "New Project"

NOTE



The programming system **ProProg wt III** is required for programming the b maXX controller PLC **BMC-M-PLC-02**.

5.3 b maXX controller PLC configuration

As an IEC 61131-3 programming environment, you can use PROPROG wt to program different target systems (CPUs). It is also possible to program different target systems in one project. You create a program for the BMC_M_PLCO1 target system under "Physical Hardware" using resource BMC_M_PLCO1.

You open the template for BMC_M_PLCO1 by means of the "New project" menu. Under properties, configuration CONF1 has PLC type SH03_30.

You create a program for the BMC_M_PLCO2 target system under "Physical Hardware" using resource BMC_M_PLCO2.

You open the template for BMC_M_PLCO2 by means of the "New project" menu. Under properties, configuration CONF1 has PLC type XPLC_40.

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

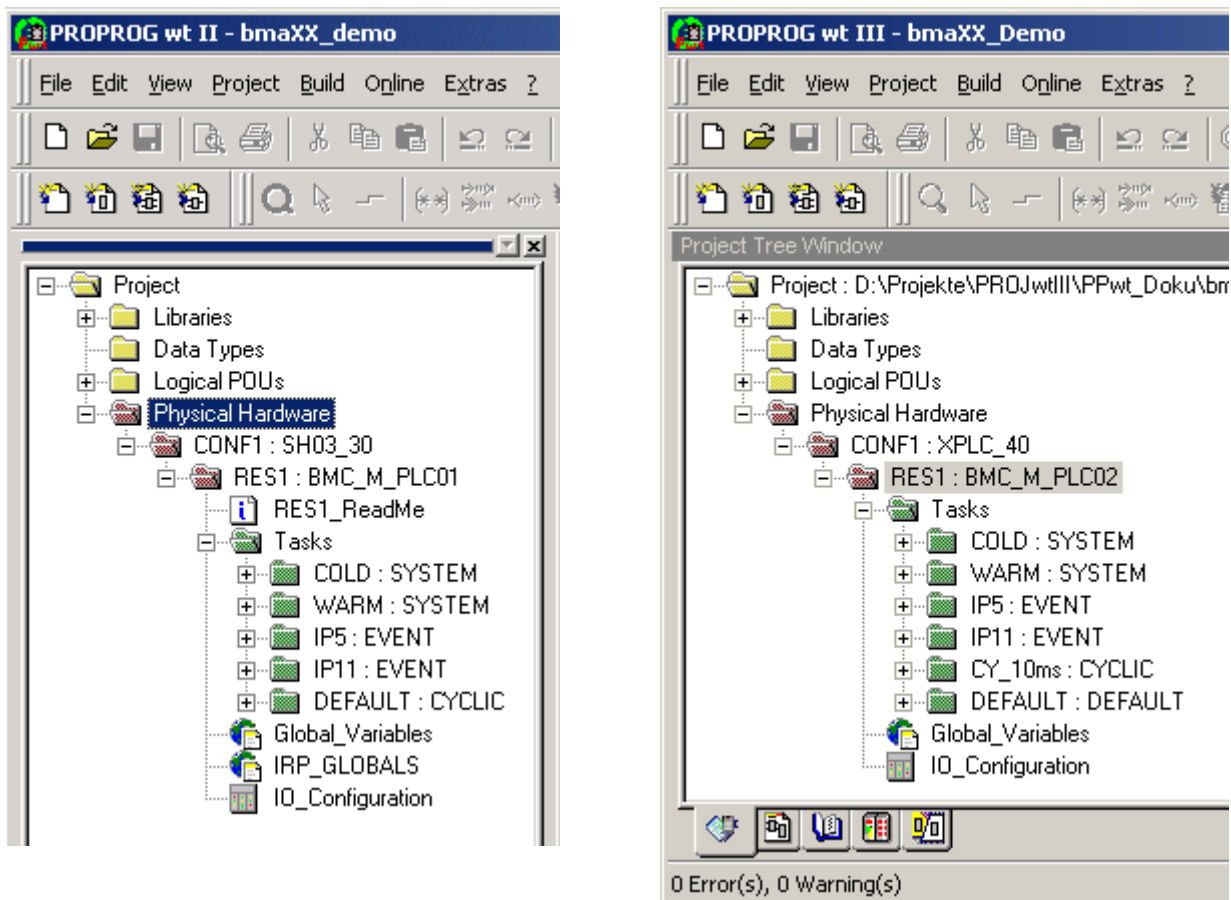


Figure 12: Setting the property of a b maXX PLC configuration under "Physical Hardware".
(left hand BMC-M-PLC-01; right hand BMC-M-PLC-02)

Overview of the settings within the physical hardware:

Project template	Configuration	Resource
BMC_M_PLC01	SH03_30	BMC_M_PLC01
BMC_M_PLC02	XPLC_40	BMC_M_PLC02

A project can include several configuration each with several resources.

NOTE



The programming system **ProProg wt III** is required for programming the b maXX controller PLC **BMC-M-PLC-02**.

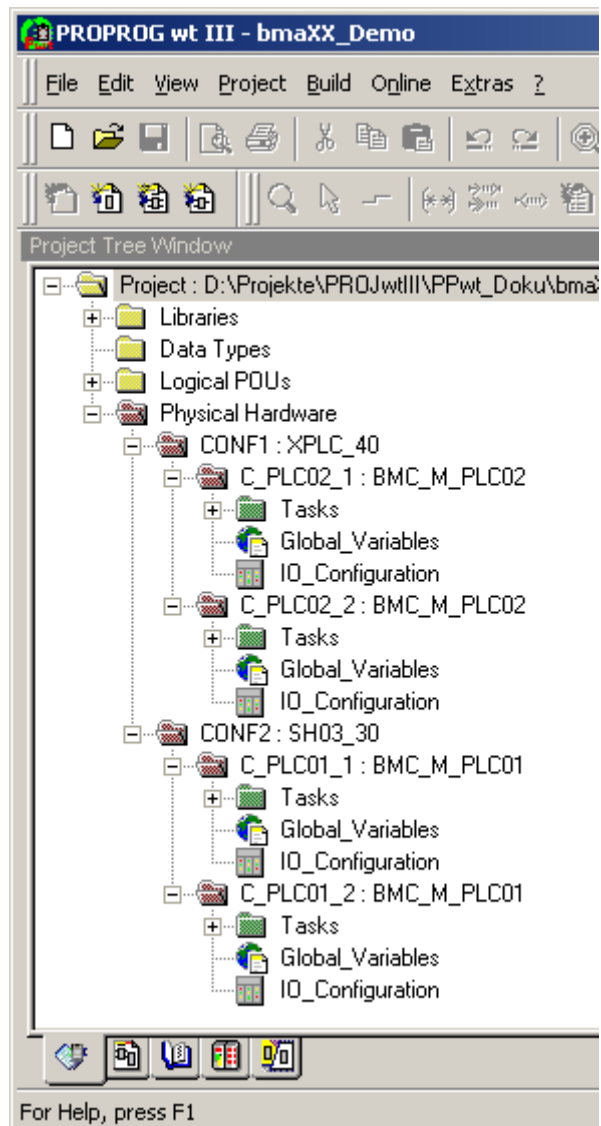


Figure 13: Project with several configurations and resources

5.4 b maXX PLC resource

The resource contains the b maXX 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 PLCs or b maXX systems in one project.

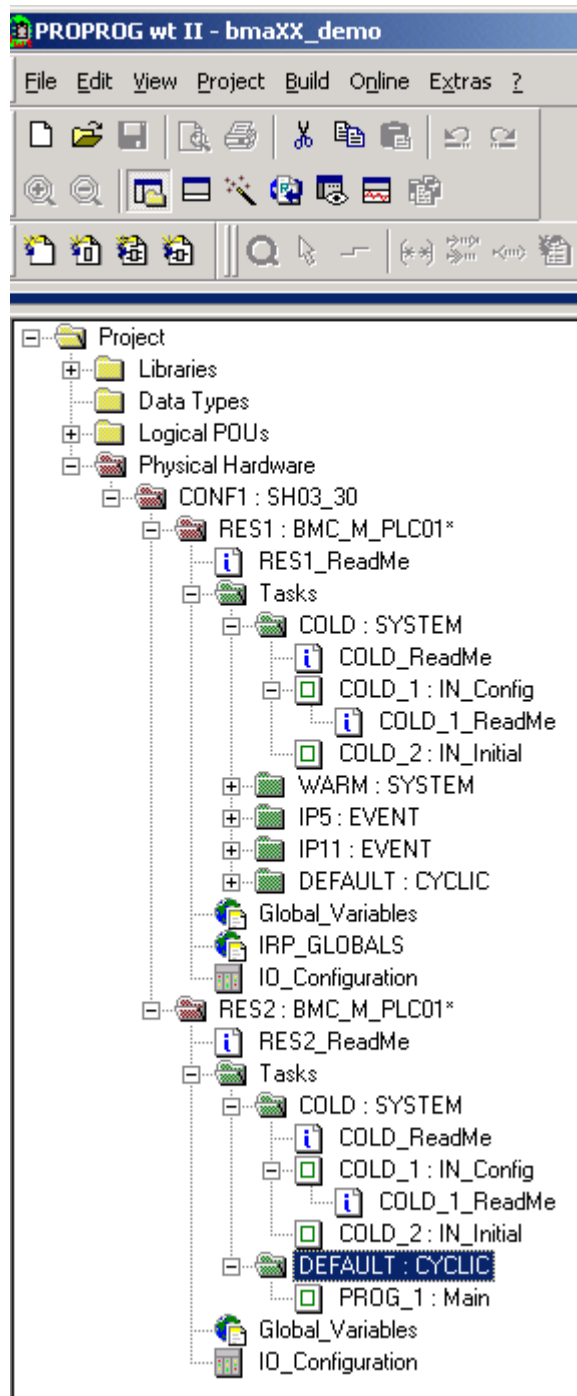


Figure 14: Example of a b maXX PLC configuration with several resources (BMC-M-PLC-01).

5.4.1 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:

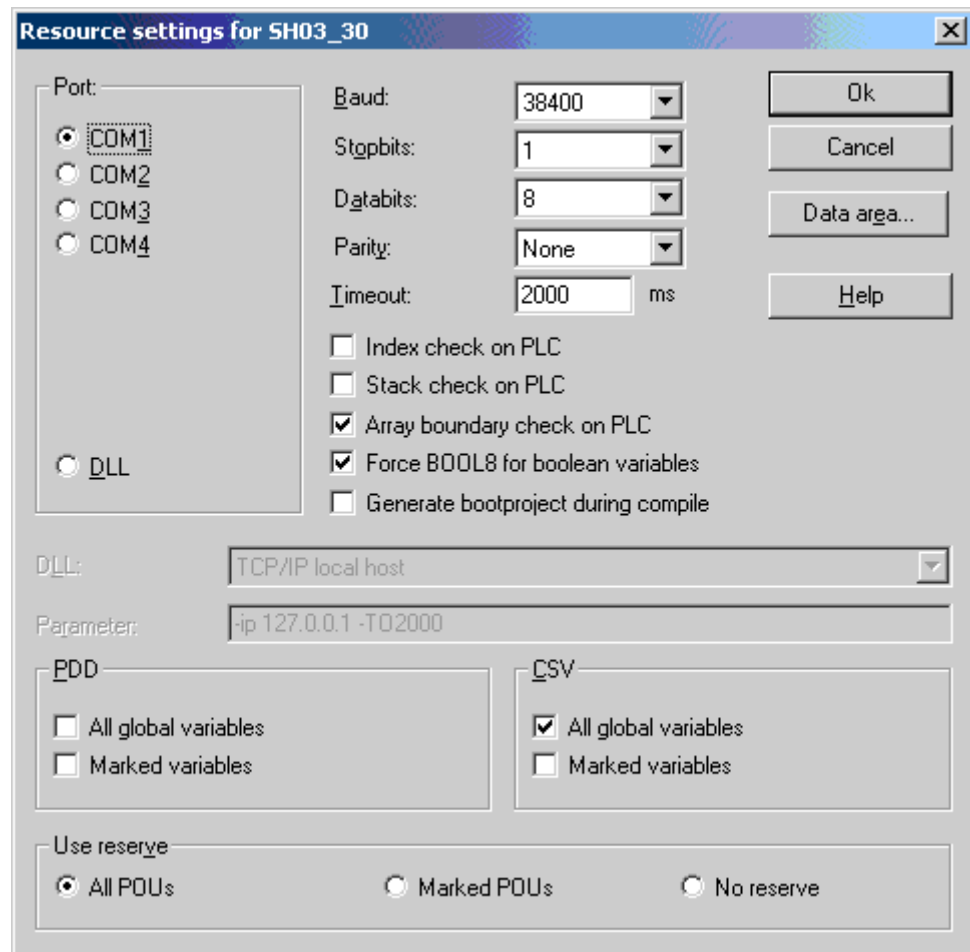


Figure 15: Resource setting within a b maXX PLC configuration.

- "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 controller 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 / Online help system of ProProg wt III).
- "CSV": Settings for providing variables for the OPC server. (See also the PROPROG wt II programming manual / Online help system of ProProg wt III).
- "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 / Online help system of ProProg wt III).

The resource setting can be made separately for each b maXX controller 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

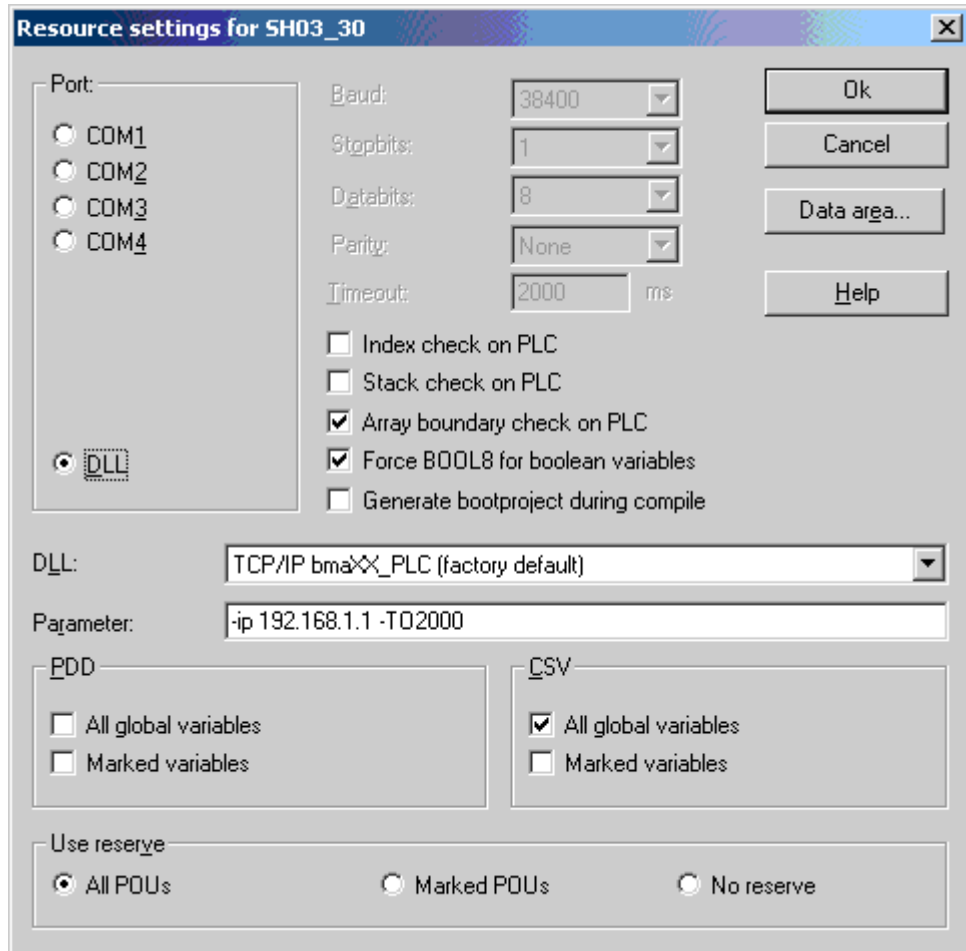


Figure 16: Setting the TCP/IP address

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 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 controller PLC:**
To be able to access the b maXX controller PLC via Ethernet, an Ethernet-capable module (e.g. a BMC-M-ETH-02) must be fitted in the b maXX system in addition to the b maXX controller 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.

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

5.4.2 Control Dialog for Resources

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

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

Select a resource in PROPROG wt II:

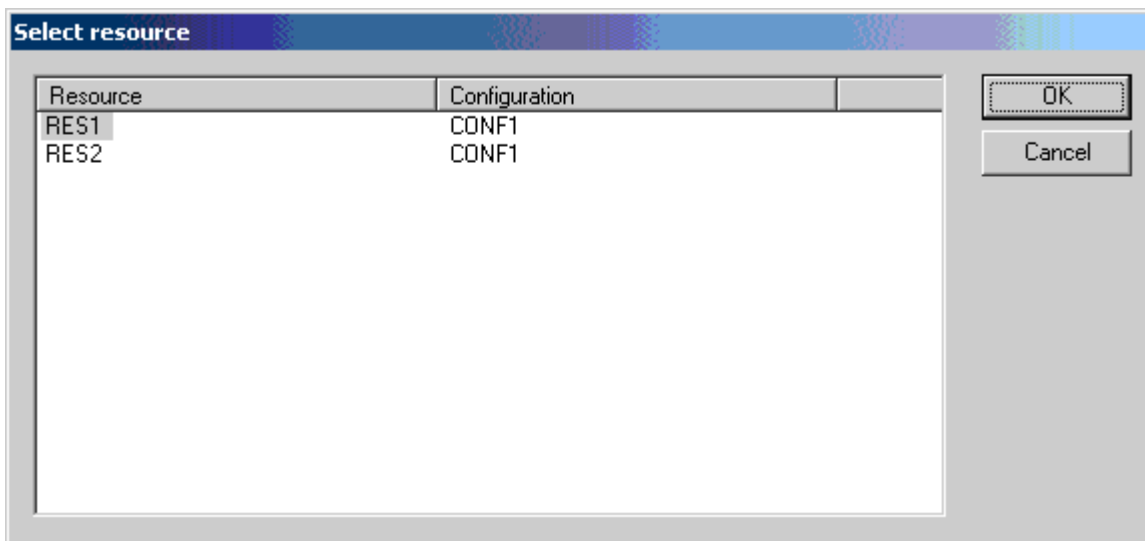


Figure 17: Select the resource in PROPROG wt II - display the control dialog

The control dialog will be opened.

Select a resource in ProProg wt III:

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

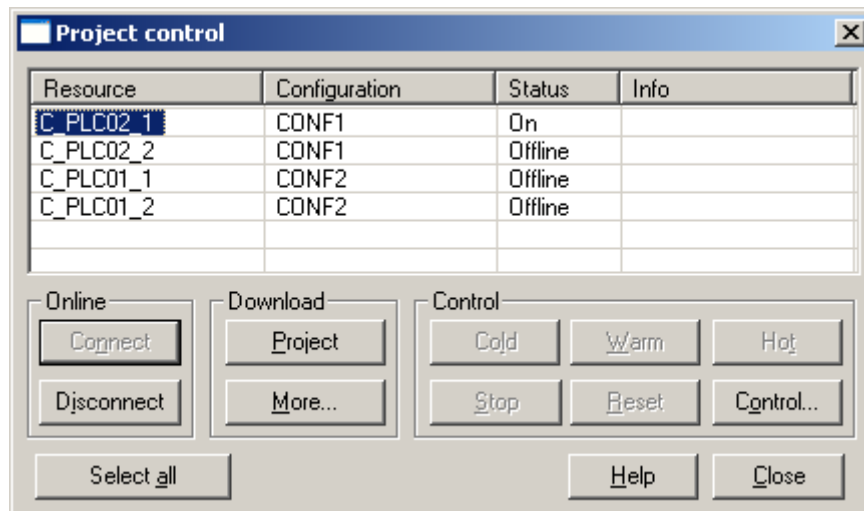


Figure 18: Select the resource in PROPROG wt III - connect

Press the button „Control...“ now.

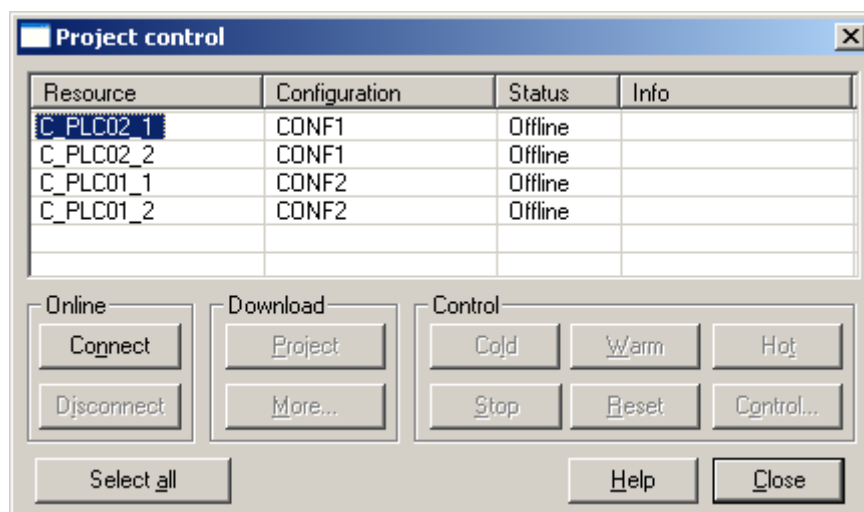


Figure 19: Select the resource in PROPROG wt III - open control dialog

The control dialog will be opened.



Figure 20: 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.

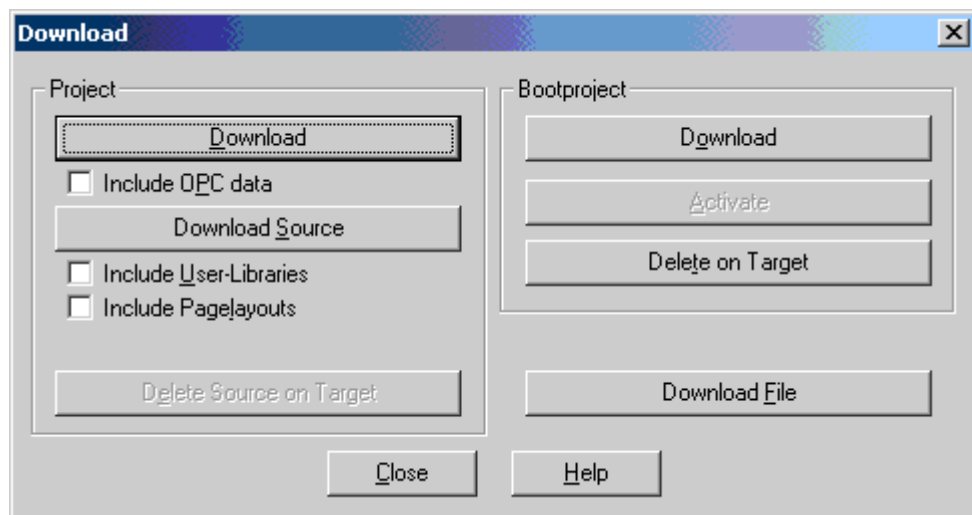


Figure 21: b maXX PLC resource transfer to flash memory or to RAM.

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 controller PLC's flash memory. Clicking on **Activate**, loads the project from the b maXX controller 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 controller 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 22: 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 controller PLC's RAM.

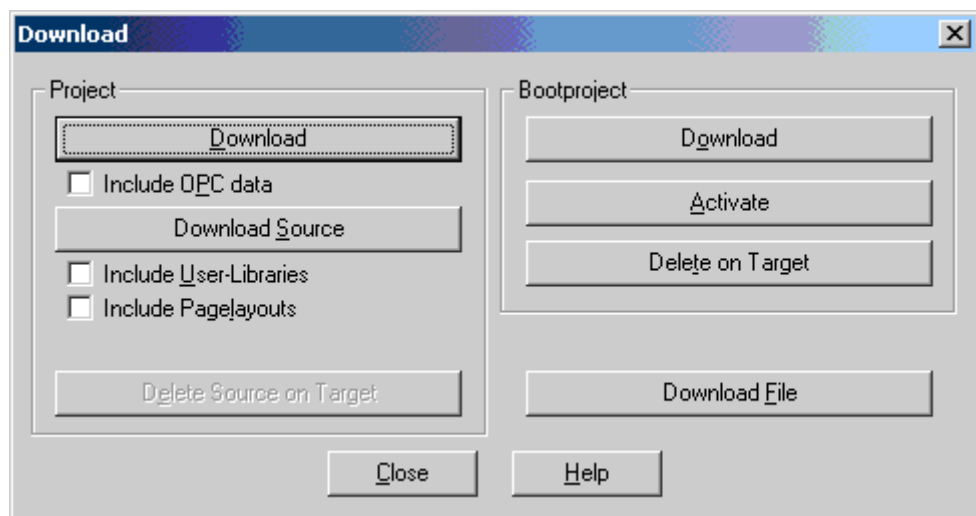


Figure 23: "Download" control dialog

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



Figure 24: b maXX PLC resource control in status "STOP".

Description of the pushbuttons in maXX 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 controller 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 controller 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 controller 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 b maXX resource control again.
- Pushbuttons **"Cold"/"Warm"/"Hot"**: Using these PROPROG commands, you can manually start the b maXX controller 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 BMC-M-PLC-01 to "RUN" and if you set switch S2 on option module BMC-M-PLC-02 to "RUN" respectively. 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 controller 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 BMC-M-PLC-01 is set to "RUN" and if switch S2 on option module BMC-M-PLC-02 is set to "RUN" respectively. Otherwise, the controller stays in the "Stop" status and no user code is executed.

If you carry out a hardware reset on switch S1 (or S2 on the BMC-M-PLC-02 module), this switch must also be set to "RUN" afterward.

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 BMC-M-PLC-01 module and at switch S2 of the BMC-M-PLC-02 module respectively). The system stores „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 "Cold" manual start pushbutton in the b maXX controller 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.



WARNING

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

serious personal injury • death

*Danger from: **mechanical effects**. The PLC operation mode can be changed with the control dialog. This can have a bearing on the function of the machine.*

Keep far enough the moving parts of the machine/system. In all cases, activate the machine's safety devices.

- **"Upload"** pushbutton: Upload functions in PROPROG wt are not currently supported.

- **"Info"** pushbutton: General information on the b maXX controller PLC itself and the project status on the b maXX controller PLC (for a description, see below).
- **"Help"** pushbutton: This calls general and specific PLC help for the b maXX controller PLC.

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

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

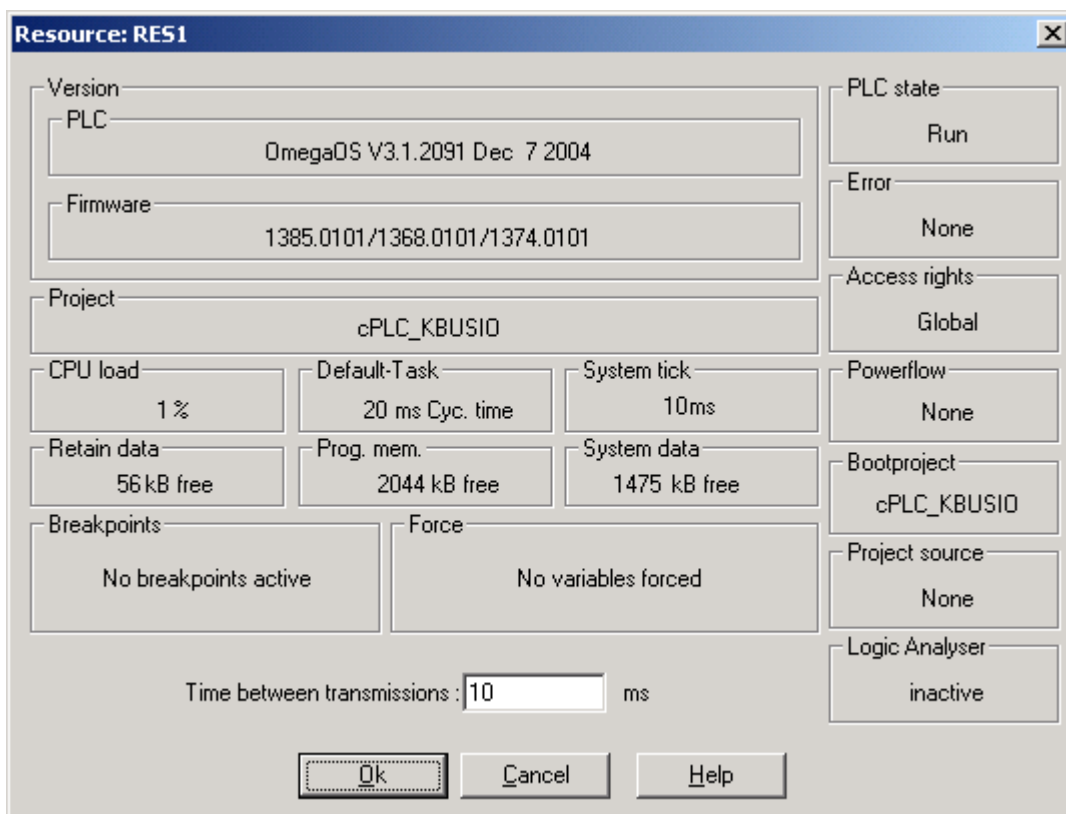


Figure 25: Resource information window

- Version: PLC "OmegaOS V3.1.2091 Dec 7 2004"
 - "OmegaOS": Product designation.
 - „V3.1.2091“: OmegaOS Compiler Info.
 - „Dec 7 2004“: Date on which the PLC version was created.
- Firmware: "1385.0101/1368.0101/1374.0101"
 - „1385.0101“: Version OmegaOS software 6.1385.0101.
 - „1368.0101“: Version FPGA software 6.1368.0101.
 - „1374.0101“: Version BOOT software 6.1374.0101.
- 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 controller 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 setting 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:**
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 controller PLC operating status (e.g. "Run", "Stop", etc.)

- 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 controller PLC resource control.
- Access rights: Indicates the access rights for sending and debugging 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 controller PLC (the project name may be a maximum of eleven letters long).
Using the page for program transfer (b maXX controller 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.

5.4.3 Data Area

The data area contains the b maXX controller 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 26: b maXX controller 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 PROPROGRAM 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.

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 PROPROGRAM compiler.

You see in „info“ in the message window how many bytes of the memory are used for these flags. Before a project code generation must take place at which „no reserve“ in the resource settings must be activated.

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

In PROPROGRAM, 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 PROPROGRAM 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.

You see in „info“ in the message window how many bytes of the memory are used for these flags. Before a project code generation must take place at which „no reserve“ in the resource settings must be activated.

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.

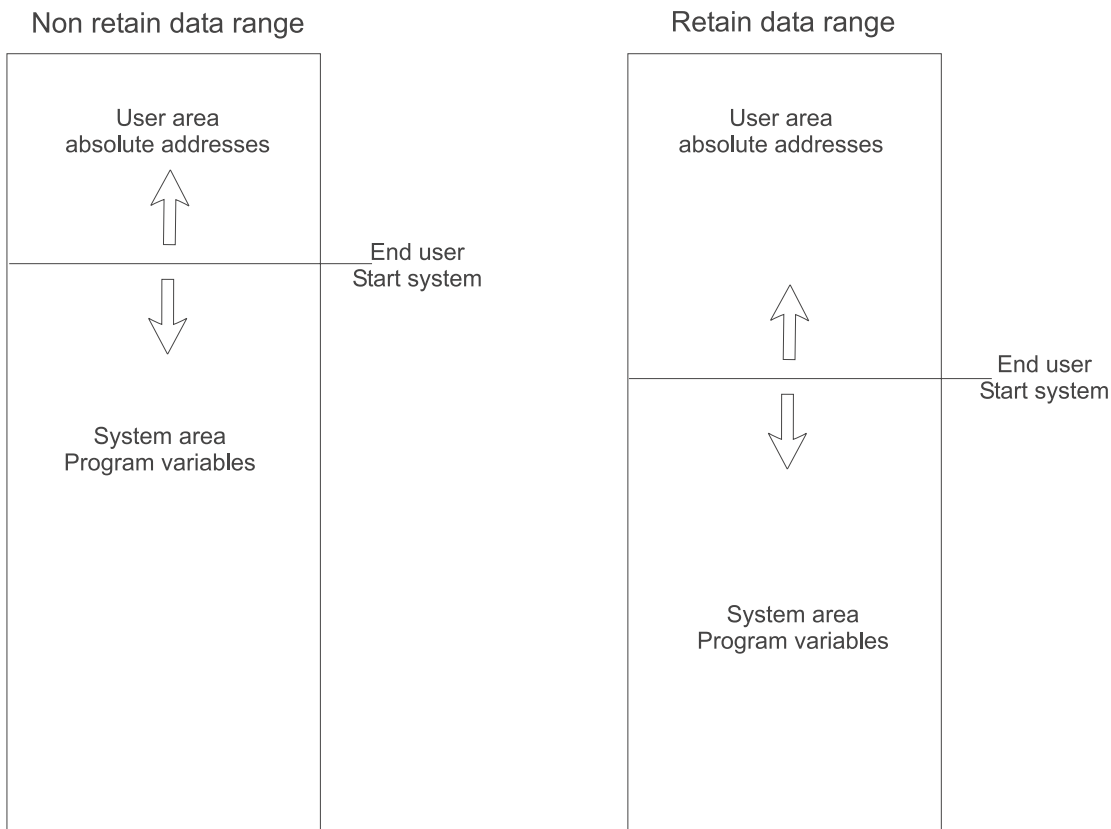


Figure 27: Dividing and setting the b maXX controller PLC data area

Assigning absolute flag addresses in the program

An absolute b maXX controller 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:

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

PROPROGRAM wt II:

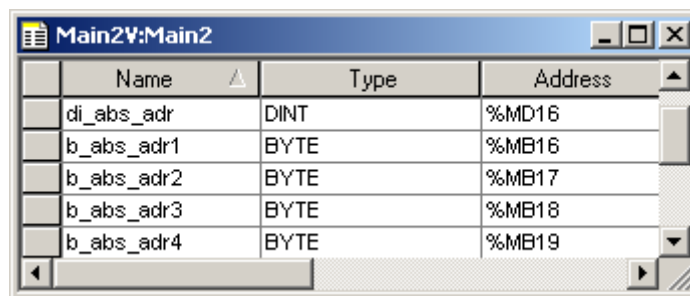
```
d_abs_adr AT %MD12 : DWORD; (* symbolic variable to absolute address *)
```

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 28: 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", 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!

5.4.4 b maXX controller PLC event tasks

b maXX controller PLC event tasks are for event-driven program calling (interrupt).

Their type and code runtime determine the real-time response of the b maXX system. The real-time response depends not only on the b maXX controller PLC, but also on the other system components.



NOTE

The data of the system components / modules placed right hand of the power module for b maXX controller PLC will be updated synchronously to the operating system timer tick (BMC-M-PLC-01 every 10 ms and BMC-M-PLC-02 every 2 ms respectively) of the b maXX controller PLC.

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, 4	Reserved	
5	Timer A	14
6	Timer A	13
7, 8, 9, 10	Reserved	
11	Sync-Signal 1 option module	14
12	Sync-Signal 2 option module	14

NOTE



The reserved events in particular Event 4 and 8 are not implemented in b maXX controller PLC.

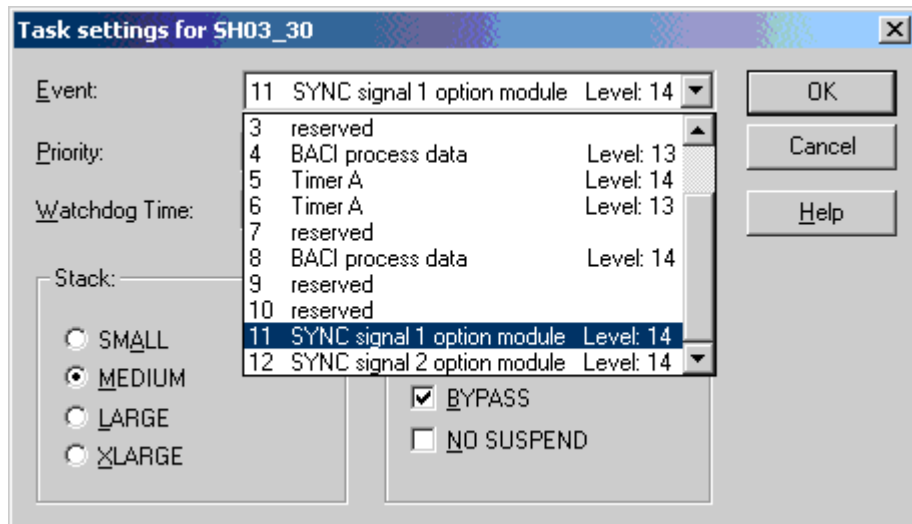


Figure 29: Events of the der b maXX controller PLC event tasks (event 4 and 8 are not implemented in the b maXX controller PLC)



NOTE

All the b maXX controller 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.

5.5 b maXX controller PLC user libraries

The PROPROG wt user libraries are divided into firmware and user libraries that can be hardware-dependent, b maXX group dependent or hardware-independent.

You can use group dependent libraries in several resources of the same b maXX group.

You can only use hardware-dependent libraries in resources of the specified target system. The hardware dependence of b maXX PLC libraries is shown by `*_C_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 controller PLC Board functionality
e.g. starting a PROPROG wt bypass event task.
- Data types: b maXX controller PLC-specifically assembled data types and fields, e.g. register structure of system components (CANopen-Master).
- Standard FBs: Elementary FBs for control engineering for drives connected to the b maXX controller PLC via a field bus system (e.g. CANopen).

- Technology modules: Functionalities for drives connected to the b maXX controller PLC via a field bus system (e.g. CANopen).

**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 controller PLC and b maXX drive 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 module BMC-M-ETH-02 or BMC-M-CAN-04 (CANopen-Master) and BMC-M-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_21bd01	FBs for b maXX module BMC-M-ETH-01 or BMC-M-ETH-02 (Ethernet)

ProProg wt III:

CD-ROM	Contains library	
TB cam disk	CAM_PLC01_30bd00 CAM_PLC02_30bd00	FBs for "cam disk" technology component
TB winder	WINDER_PLC01_30bd00 WINDER_PLC02_30bd00	FBs for "winder" technology component

CD-ROM	Contains library	
CANopen	CANopen_PLC01_30bd00 CANopen_PLC02_30bd00	FBs for b maXX module BMC-M-ETH-02 or BMC-M-CAN-04 (CANopen-Master) and BMC-M-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_30bd01 TCP_PLC02_30bd01	FBs for b maXX module BMC-M-ETH-01 or BMC-M-ETH-02 (Ethernet)

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

CD-ROM	Contains library	
Standard libraries	SYSTEM1_C_PLC01_20bd00	Function blocks for BMC-M-PLC-01
	SYSTEM2_C_PLC01_20bd00	Firmware blocks for BMC-M-PLC-01

ProProg wt III:

CD-ROM	Contains library	
Standard libraries	SYSTEM1_C_PLC01_30bd00	Function blocks for BMC-M-PLC-01
	SYSTEM1_C_PLC02_30bd02	Function blocks for BMC-M-PLC-02
	SYSTEM2_C_PLC01_30bd00	Firmware blocks for BMC-M-PLC-01
	SYSTEM2_C_PLC02_30bd02	Firmware blocks for BMC-M-PLC-02

You must state the directory path for libraries under PROPROG wt, Options, Directories. The 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 / Online help system of ProProg wt III).

5.5.1 b maXX controller PLC firmware

The b maXX controller PLC firmware consists of function blocks (FBs) that use parameter transfers to communicate with functions on the b maXX 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_C_PLC01_20bd00 (or above)

to insert the BMC-M-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 controller 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 controller PLC firmware is used by several b maXX controller PLC user libraries. This means that with a b maXX controller PLC user library, it may be necessary to insert the requested firmware library SYSTEM2_C_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 „Template for BMC_M_PLC01“ under „File\New Project“ the library SYSTEM2_C_PLC01_20bd00 (or above) automatically is set. The standard user library SYSTEM1_C_PLC01_20bd00 (or above) in the normal case also will be necessary in projects and will be delivered on a separate CD (see overview in [b maXX controller PLC user libraries](#) < from page 54 onward).

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

ProProg wt III:

You must use library

SYSTEM2_C_PLC01_30bd00 (or above)

to insert the BMC-M-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 controller PLC.
- P, PI controller, 48-bit division via electronic transmission, integration, differentiation.
- Function blocks for interface module to USS[®] and 3964R[®] protocols.

You must use library

SYSTEM2_C_PLC02_30bd02 (or above)

to insert the BMC-M-PLC-02 firmware into a project. The library contains the following range of functions:

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

5.5.2 b maXX controller PLC Board functions

Firmware libraries SYSTEM2_xxx_yybdzz (or above) contain function blocks (FBs) for checking event signals for interrupts and board LEDs. Function block INTR_SET is used to initialize and start b maXX PLC event task (bypass). Function block LED8

(BMC-M-PLC-01) and LED12 (BMC-M-PLC-02) respectively 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 controller PLC resources.

PROPROG wt II: BMC-M-PLC-01:

- The FBs are inserted via user library **SYSTEM1_C_PLC01_20bd00** and above.

ProProg wt III: BMC-M-PLC-01:

- The FBs are inserted via user library **SYSTEM1_C_PLC01_30bd00** and above.

ProProg wt III: BMC-M-PLC-02:

- The FBs are inserted via user library **SYSTEM1_C_PLC02_30bd02** 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 (BMC-M-PLC-01) and 15 min (BMC-M-PLC-02) respectively are able to be measured with these function blocks.

5.5.2.1 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.

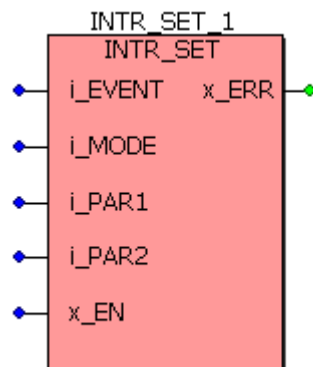


Figure 30: Initializing and enabling a bypass event task via function block INTR_SET.

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 50 μs	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
5	Timer A	14
6	Timer A	13
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.

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: $i_PAR1 := (5000 \mu\text{s} / 50 \mu\text{s}) = \text{INT}\#100$.

Representation of the configured and released INTR_SET:

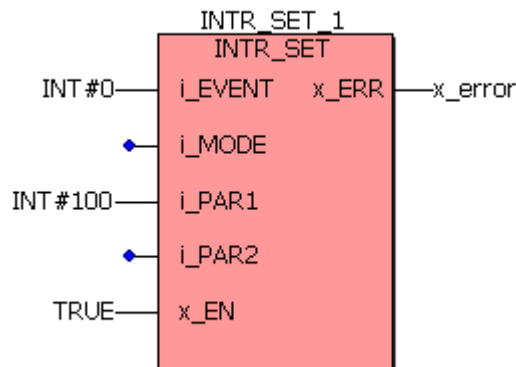


Figure 31: Function block INTR_SET: Starting a CPU timer 1 interrupt with a call interval of 5 ms.

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

You must link user POE "timer" in group task within the BMC_M_PLCOx resource as a task type "EVENT" to event "CPU_Timer 1" with setting option "BYPASS":

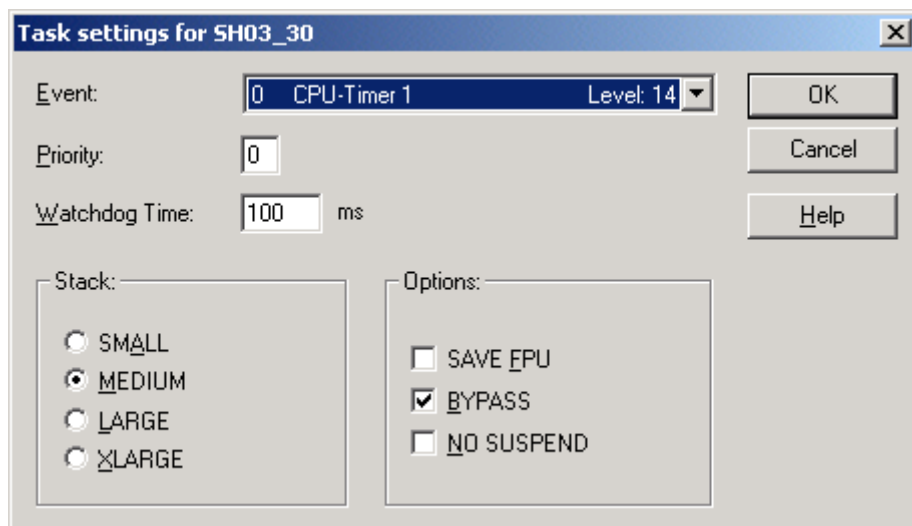


Figure 32: Linking to event "CPU timer 1" as a bypass task.

Example 2:

Start the "Timer A" interrupt at 5 ms and generate the trigger signal at CBPB 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.

BMC-M-PLC-01:

First of all, you set up the event task within the BMC_M_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_C_PLC01_20bd00" or above (PROPROG wt II) and „SYSTEM1_C_PLC01_30bd00" or above (ProProg wt III) respectively) 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 CBPB 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.

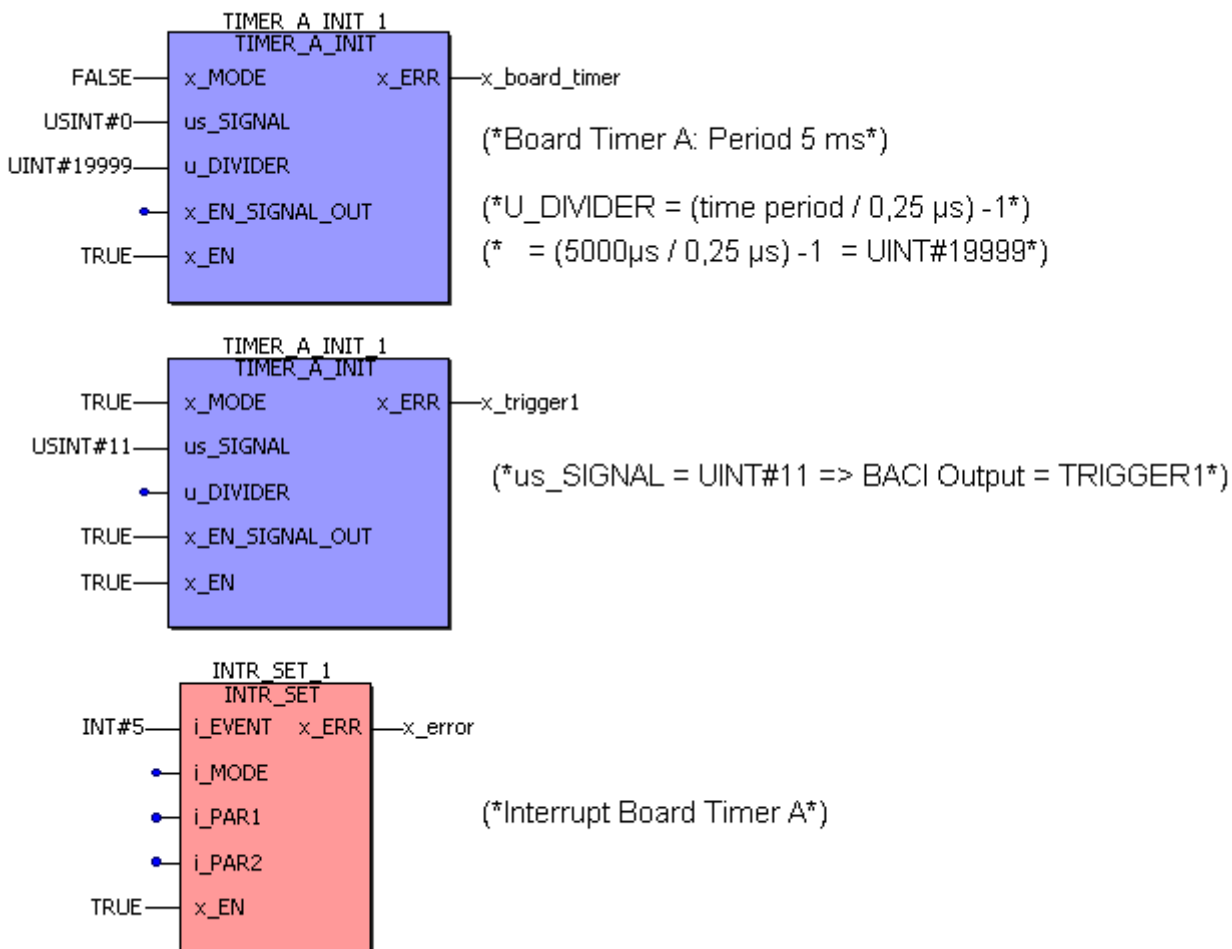


Figure 33: Start of the "Timer A" bypass event task with a period of 5 ms and simultaneous use of "Timer A" as a trigger signal for system components (placed left hand of the b maXX controller PLC) on BMC-M-PLC-01

BMC-M-PLC-02:

First of all, you set up the event task within the BMC_M_PLC02 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_C_PLC02_30bd02" 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 CBPB 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.

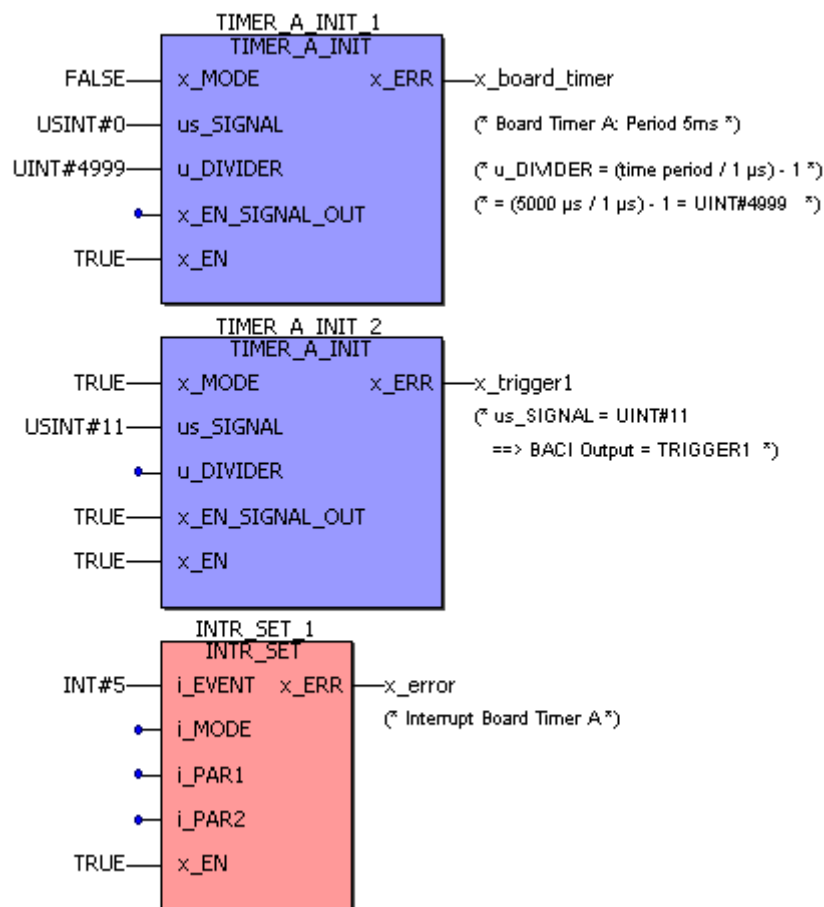


Figure 34: Start of the "Timer A" bypass event task with a period of 5 ms and simultaneous use of "Timer A" as a trigger signal for system components (placed left hand of the b maXX controller PLC) on BMC-M-PLC-02

5.5.2.2 BMC-M-PLC-01 Function block LED8

You can use the LED function block from firmware library **SYSTEM2_C_PLC01_20bd00** or above (PROPROG wt II) and **SYSTEM2_C_PLC01_30bd00** or above (ProProg wt III) respectively to program the b maXX controller PLC LEDs on the board.



Figure 35: LED8 function block for BMC-M-PLC-01

Parameter	Input	Data type
x_H1	LED H1	BOOL
x_H2	LED H2	BOOL
x_H3	LED H3	BOOL
x_H4	LED H4	BOOL
x_H5	LED H5	BOOL
x_H6	LED H6	BOOL
x_H7	LED H7	BOOL
x_H8	LED H8	BOOL

Description

The LEDs on the left of the b maXX controller PLC BMC-M-PLC-01 light up green; the ones on the right light up red.

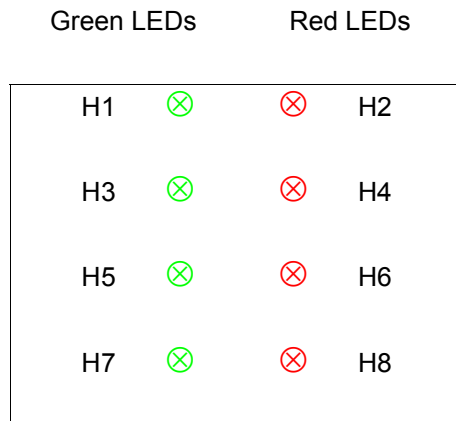


Figure 36: LEDs of the b maXX controller PLC module (BMC-M-PLC-01)

5.5.2.3 BMC-M-PLC-02 Function block LED12

You can use the LED function block from firmware library **SYSTEM2_C_PLC02_30bd02** or above (ProProg wt III) respectively to program the b maXX controller PLC LEDs on the board.

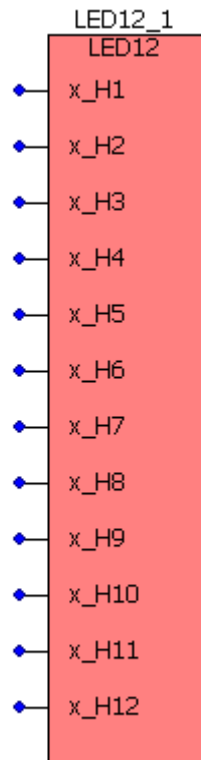


Figure 37: LED12 function block for BMC-M-PLC-02

Parameter	Input	Data type
x_H1	LED H1	BOOL
x_H2	LED H2	BOOL
x_H3	LED H3	BOOL
x_H4	LED H4	BOOL
x_H5	LED H5	BOOL
x_H6	LED H6	BOOL
x_H7	LED H7	BOOL
x_H8	LED H8	BOOL
x_H9	LED H9	BOOL
x_H10	LED H10	BOOL
x_H11	LED H11	BOOL
x_H12	LED H12	BOOL

Description

The b maXX controller PLC module has twelve LEDs (five green ones (H1, H3, H5, H7, H11), five red ones (H2, H4, H6, H8, H10), one blue (H9) and one orange/green (H12)) as display elements .

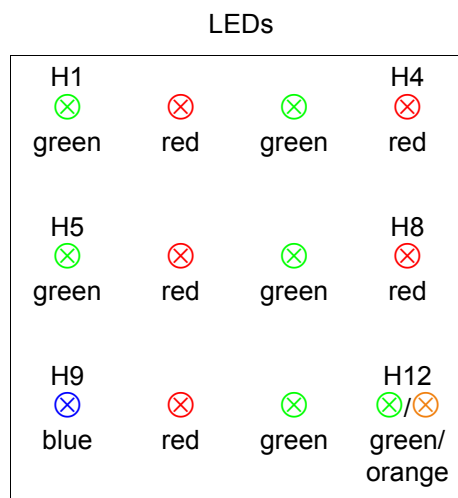


Figure 38: LEDs of the b maXX controller PLC module (BMC-M-PLC-02)

5.5.3 b maXX PLC data types

In a PROPROG wt project, data is exchanged between the b maXX controller PLC and the system components for b maXX controller PLC by means of variables of the appropriate data type. User library BM_TYPES_20bd06 or above (PROPROG wt II) and BM_TYPES_30bd01 or above (ProProg wt III) respectively 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 system components.

For this, the most important variables for the different functions and slots of the individual system components are defined in the templates BMC_M_PLC01 ("File/New project.../Template for BMC_M_PLC01") and BMC_M_PLC02 ("File/New project.../Template for BMC_M_PLC02") respectively.

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_20bd06 or above (PROPROG wt II) and BM_TYPES_30bd01 or above (ProProg wt III) respectively and that were used.

Users can and should of course use the predefined data types in BM_TYPES_20bd06 (PROPROG wt II) and BM_TYPES_30bd01 or above (ProProg wt III) respectively.

You cannot directly call the worksheet of "BM_TYPES_20bd06" (PROPROG wt II) and BM_TYPES_30bd01 (ProProg wt III) respectively 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_20bd06" (PROPROG wt II) and BM_TYPES_30bd01 (ProProg wt III) respectively worksheet:

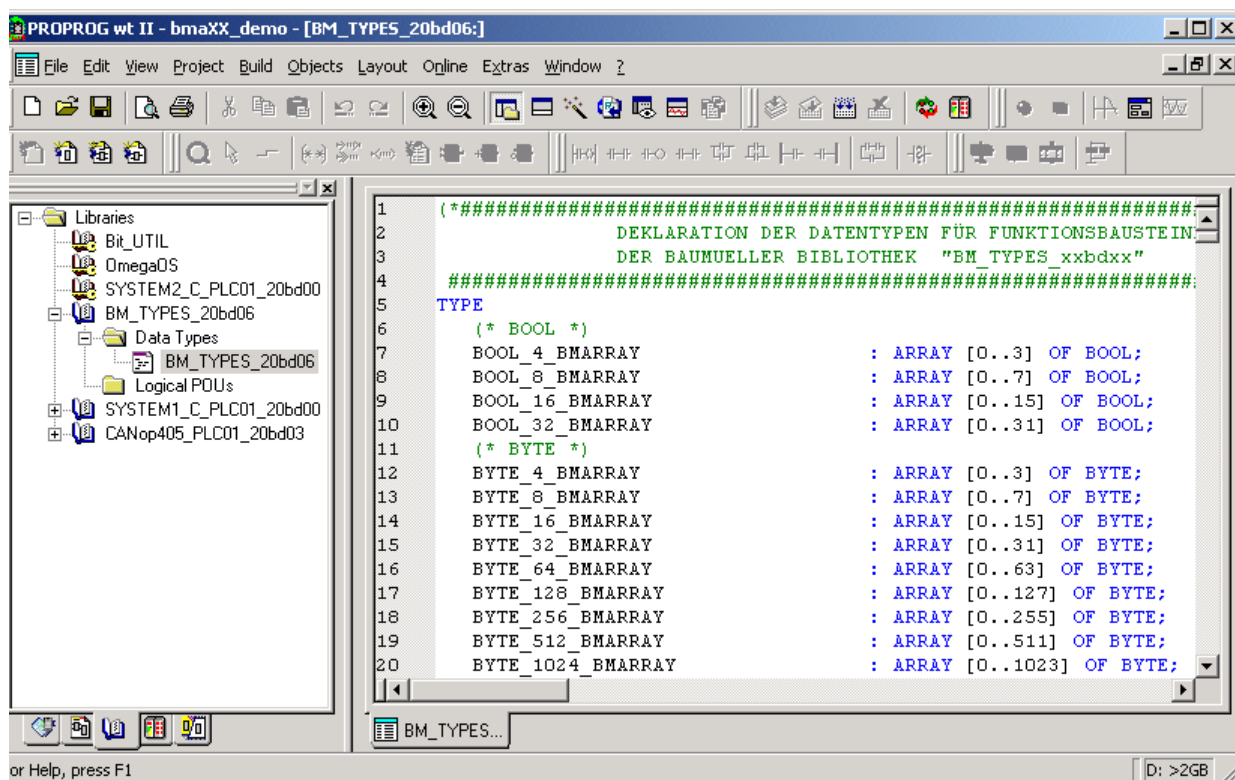


Figure 39: View of Baumüller data types "BM_TYPES_20bd06" in PROPROG wt II that are integrated in your project.

You can access these data types when assigning variables via the variable dialog.
PROPROG wt II:

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

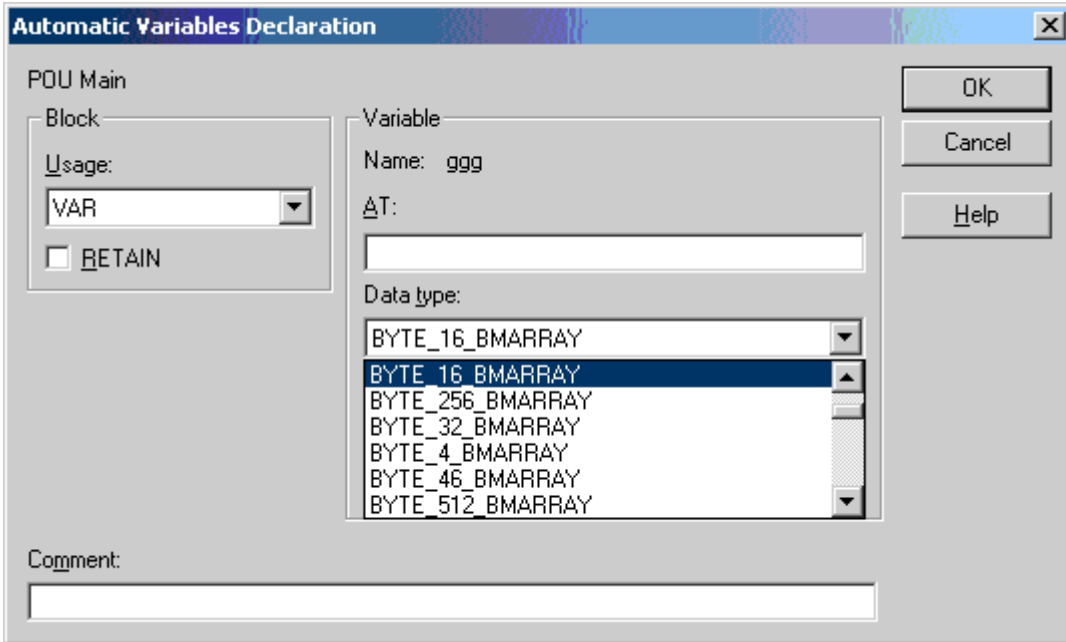


Figure 40: Automatic variable declaration.

ProProg wt III:

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

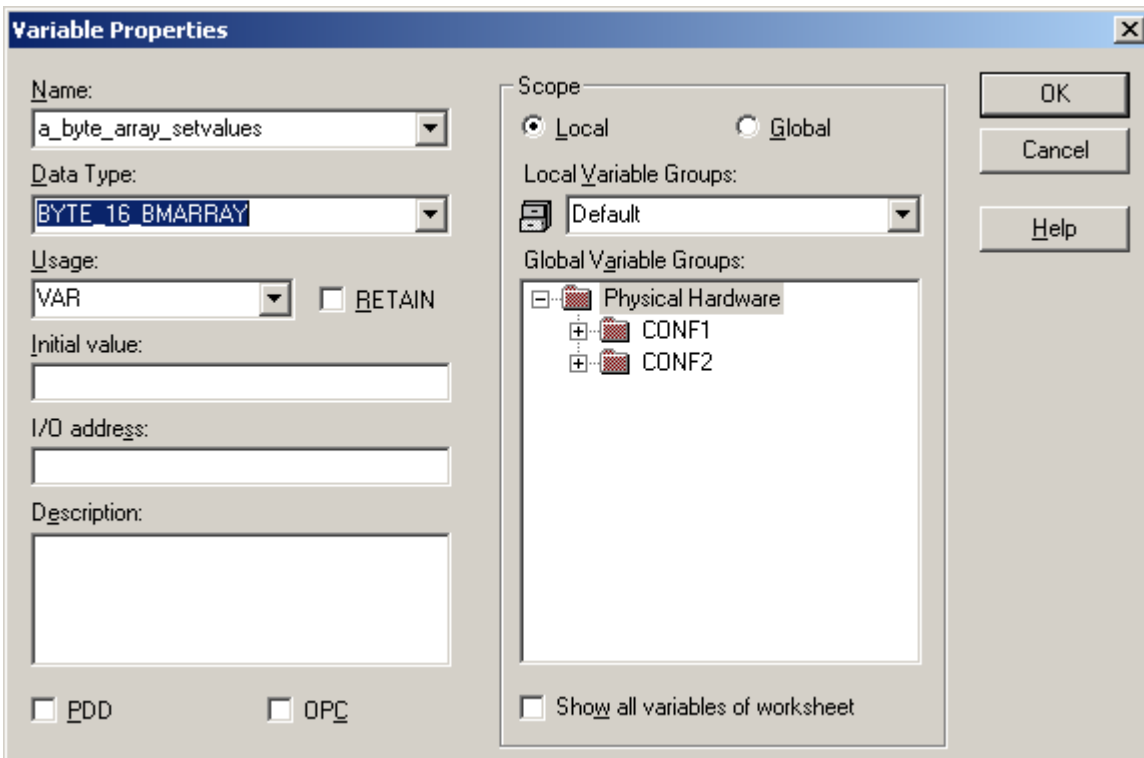


Figure 41: Declaration of variables

When you create a PROPROGRAM project using template BMC_M_PLC01 and BMC-M-PLC-02 respectively, the variables for data exchange are already declared in the global variable worksheet. For the modules (system components left hand of the PLC) you will find one (and sometimes two) variables per module number.

Example:

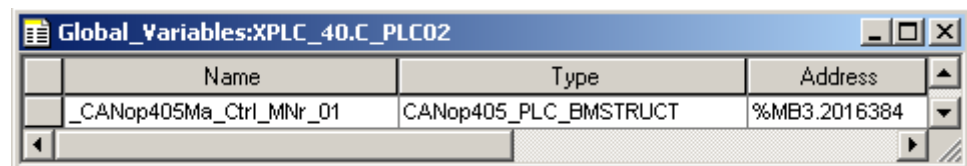
For initialization, system component Ethernet with CANopen-Master (BMC-M-ETH-02) needs settings in various registers.

With the system component with module number 1, this yields the following variable declaration (that is already created in the template):

PROPROGRAM wt II:

```
_CANop405Ma_Ctrl_MNr_01 AT %MB3.2016384 : CANop405_PLC_BMSTRUCT;
```

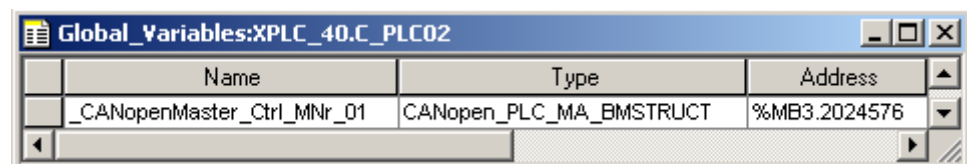
ProProg wt III:



Name	Type	Address
_CANop405Ma_Ctrl_MNr_01	CANop405_PLC_BMSTRUCT	%MB3.2016384

Figure 42: Declaration of variables

ProProg wt III with ProMaster:



Name	Type	Address
_CANopenMaster_Ctrl_MNr_01	CANopen_PLC_MA_BMSTRUCT	%MB3.2024576

Figure 43: Declaration of variables with ProMaster

The complete register structure of CANop405_PLC_BMSTRUCT is stored with its elements and the data types used in BM_TYPES_20bd06 (PROPROGRAM wt II) and BM_TYPES_30bd01 or above (ProProg wt III) respectively (extract):

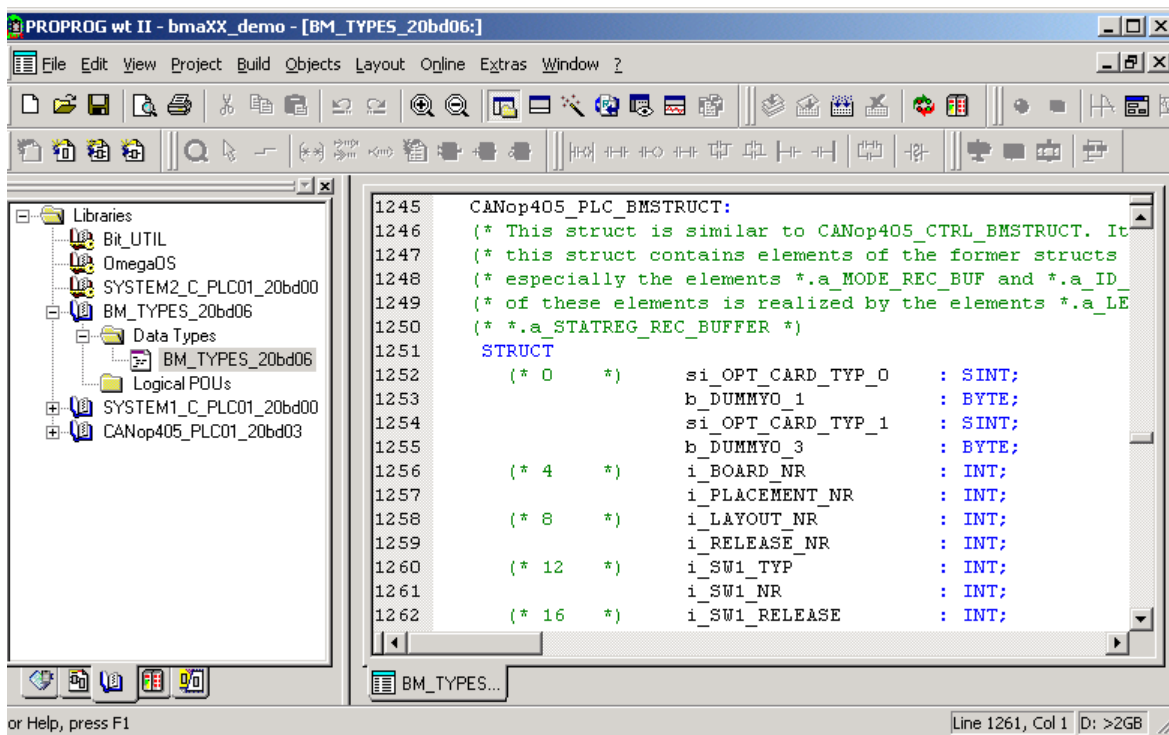


Figure 44: Extract from the register structure of CANop405_PLC_BMSTRUCT

The elements of variable `_CANop405Ma_Ctrl_MNr_01` and `_CANopenMaster_Ctrl_MNr_01` respectively now form the register of the system component with module number 1. For more information on using the system component, see the BMC-M-ETH-02 Operating Instructions and the related Application Manual.

5.5.4 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:

PROPROGRAM wt II:

- **SYSTEM1_C_PLC01_20bd00** (or above)
 - Function block for CBPB (input, timer, output, trigger)
 - USS[®] protocol interface module
 - Code runtime measurement
- **SYSTEM2_C_PLC01_20bd00** (or above)
 - b maXX controller PLC firmware
 - Function block LED8
 - Function block INTR_SET
 - 3964R[®] protocol interface module
 - Function blocks TER_x for terminal communication
- **UNIVERSAL_20bd01** or above (regardless of the hardware)

- Extrapolators, ramp generators, position generators, Min-Max and limitation FBs
- Function block virtual leading axle FB TRAJECTORY_GEN1
- Drive status and control via FB DRIVE1
e.g. for drives connected via a field bus system (e.g. CANopen) to the b maXX controller PLC.

ProProg wt III:

- **SYSTEM1_C_PLC01_30bd00** (or above; for BMC-M-PLC-01)
 - Function block for CBPB (input, timer, output, trigger)
 - USS[®] protocol interface module
 - Code runtime measurement
- **SYSTEM2_C_PLC01_30bd00** (or above; for BMC-M-PLC-01)
 - b maXX controller PLC firmware
 - Function block LED8
 - Function block INTR_SET
 - 3964R[®] protocol interface module
 - Function blocks TER_x for terminal communication
- **SYSTEM1_C_PLC02_30bd02** (or above; for BMC-M-PLC-02)
 - Function block for CBPB (input, timer, output, trigger)
 - Code runtime measurement
- **SYSTEM2_C_PLC02_30bd02** (or above; for BMC-M-PLC-02)
 - b maXX controller PLC firmware
 - Function block LED12
 - Function block INTR_SET
 - 3964R[®] protocol interface module
 - Function blocks TER_x for terminal communication
- **UNIVERSAL_30bd00** or above (regardless of the hardware)
 - Extrapolators, ramp generators, position generators, Min-Max and limitation FBs
 - Function block virtual leading axle FB TRAJECTORY_GEN1
 - Drive status and control via FB DRIVE1
e.g. for drives connected via a field bus system (e.g. CANopen) to the b maXX controller PLC.

5.5.5 b maXX 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)
CAM_PLC02_30bd00 (or above)

- Technology component winder:
 - PROPROG wt II: user library
WINDER_PLC01_20bd01 (or above)

 - ProProg wt III: user library
WINDER_PLC01_30bd01 (or above)
WINDER_PLC02_30bd01 (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_20bd06 and above (PROPROG wt II) and BM_TYPES_30bd01 and above (ProProg wt III) respectively.

5.5.6 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.

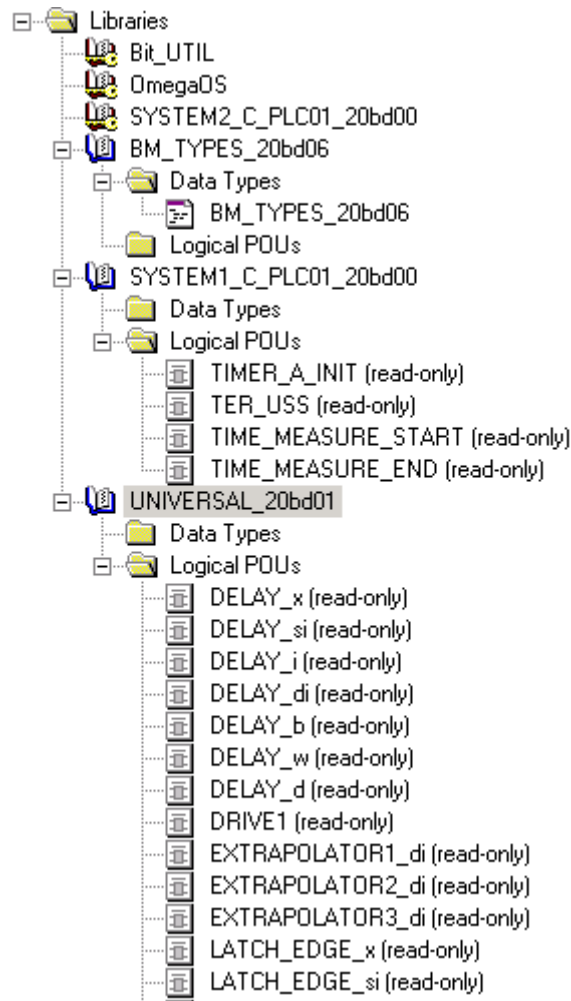


Figure 45: Standard selection of user libraries, firmware and data types using the libraries filter

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.

5.6 CBPB system description for the b maXX controller PLC

5.6.1 Direct access to the CBPB hardware from the b maXX controller PLC

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

A differentiation is made between the CBPB-Master (b maXX controller PLC) and the CBPB-Slave. Only CBPB-Masters can access the system components across the CBPB-BUS. System components cannot access each other let alone the CBPB-Master. Each individual system component can use its own dual-port RAM to query and evaluate the data that the master wrote.

Module number overview of the system components in the b maXX system:

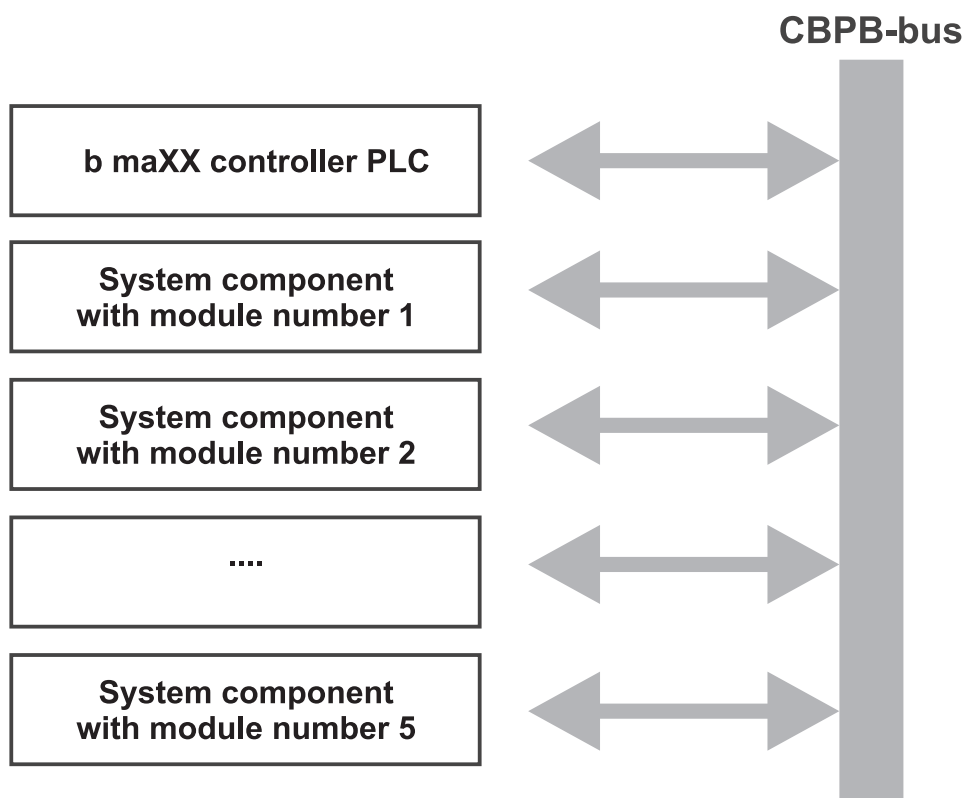


Figure 46: Module number address overview

From the b maXX controller PLC's point of view, it is possible in the application to contact the system components 1 to 5 to communicate with the system components.

NOTE



You can plug in max. 5 system components left hand of the b maXX controller PLC. Each system component gets a number between 1 and 5. The number assignment is independent of the sequence of system components.

Module number	Assigned IEC 61131-3 range (max.)
1	%MB 3.20000000 - %MB 3.20262143
2	%MB 3.30000000 - %MB 3.30262143
3	%MB 3.40000000 - %MB 3.40262143
4	%MB 3.50000000 - %MB 3.50262143
5	%MB 3.60000000 - %MB 3.60262143

NOTE



The b maXX controller PLC can only access on a system component with one of the module number above-named.

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

Global variables of this type are already appropriately predefined in the BMC_M_PLC01 or BMC_M_PLC02 template in PROPROG wt in the "Global_Variables" global variable sheet such that, depending on the system component and the module number 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 controller PLC – will automatically carry out these tasks. This means that users will no longer have any contact with the IEC 61131-3 ranges mentioned above.

For further explanations of the structures and functions of the system components, refer to their respective technical descriptions and application manuals.

Read out of the address (S1, S2, S3) of the power supply unit for b maXX controller PLC

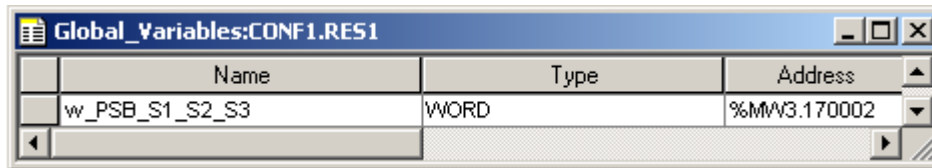
The system component „Power supply unit for b maXX controller PLC“ has the rotary switches S1, S2 and S3 to adjust an address/module number (see Operating Instructions Power Supply Unit for b maXX controller PLC). The application program of the b maXX controller PLC can read out this address/module number.

PROPROG wt II:

The address S1/S2/S3 adjusted at the power supply unit is read out via the global variable w_PSB_S1_S2_S3 created in template BMC_M_PLC01.

```
w_PSB_S1_S2_S3 AT %MW3.170002 : WORD
```

ProProg wt III:



Name	Type	Address
w_PSB_S1_S2_S3	WORD	%MW3.170002

Figure 47: Read out of the address (S1, S2, S3) of the power supply for b maXX controller PLC

In the application program you can activate or deactivate parts of your unitized software dependent of the value of w_PSB_S1_S2_S3, if necessary.

Example for using the address of the power supply unit as b maXX system address:

Your local b maXX system consists of a b maXX controller PLC, a power supply unit and other system components/modules.

You use the three rotary switches as follows:

S1 characterizes the individual machine (e.g. value „1“ for machine 1), S2 the module of the machine (e.g. „2“ for module 2) and S3 the local b maXX system number 3.

Therefore you adjust the rotary switch S1 to 1, S2 to 2 and S3 to 3 at the power supply unit for b maXX controller PLC.

On the b maXX controller PLC the value WORD#16#0123 results for the global variable w_PSB_S1_S2_S3.

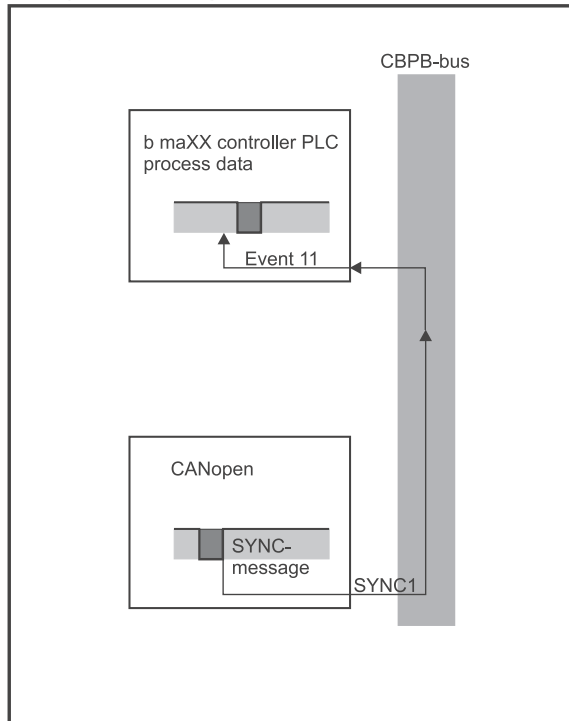
5.6.2 Process data communication between b maXX controller PLC and system components, synchronization to a common CBPB hardware signal source

Process data communication between the b maXX controller PLC and the system components (placed left hand of the b maXX controller PLC) can be synchronized via a CBPB hardware signal.

An EVENT task is applied to the corresponding event (CBPB hardware signal) on the b maXX controller PLC and then the process data communication takes place in this task.

5.6 CBPB system description for the b maXX controller PLC

Process data communication by SYNC-signal 1



Process data communication by SYNC-signal 2

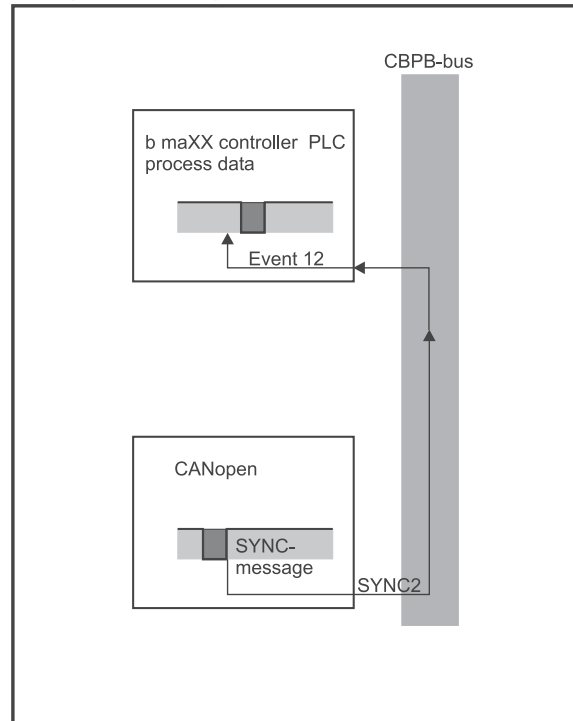


Figure 48: Process data communication by SYNC1- or SYNC2-signal

5.6.3 b maXX controller PLC interrupt sources that the application can use

For the b maXX controller PLC, you can use hardware signals from the CBPB and PLC-internal sources such as the CPU timer 1 and timer 2 or board timer A as interrupt sources, that can be linked to specific program sections of an application via the event selection of the BYPASS EVENT-task.



Information on EVENT bypass event tasks

If you insert an EVENT task type in PROPROP wt, 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 b maXX controller PLC's cold boot/warm restart stage by means of function block "INTR_SET" for events 0, 2, 5, 6, 11, 12 (see standard library SYSTEM2_C_PLC01_20bd00 or higher (PROPROP wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd00 or higher (ProProg wt III)). 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 b maXX controller PLC module that can trigger an interrupt:

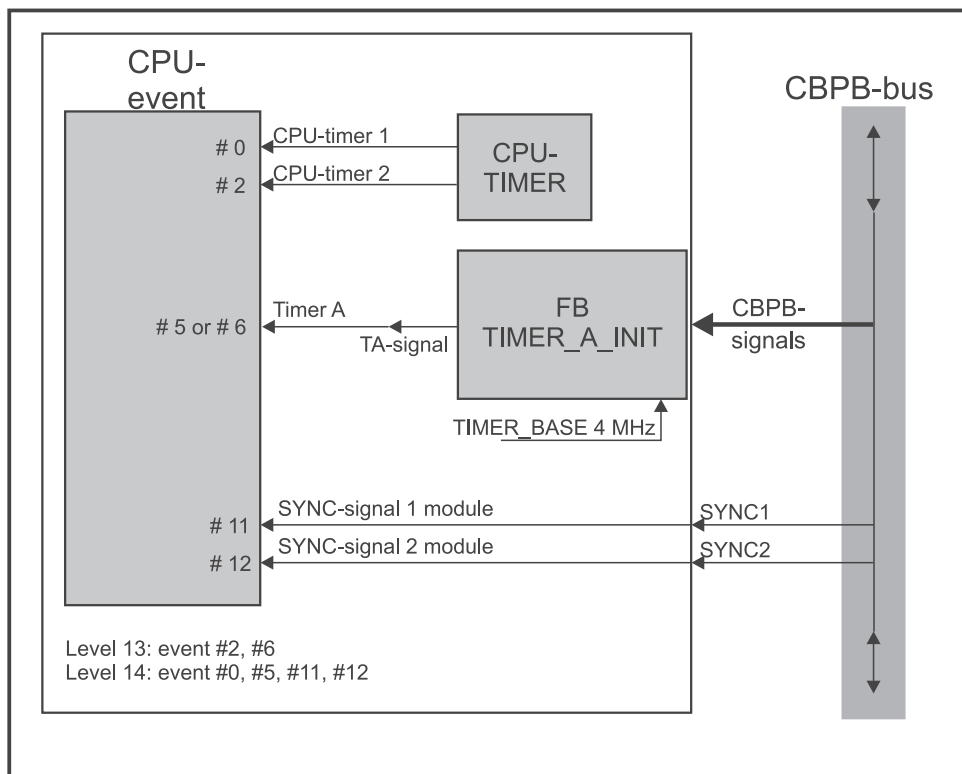


Figure 49: Representation of all EVENTS

Detailed configuration options for the Timer A EVENT by choosing the CBPB signal via the INPUT function of block "TIMER_A_INIT" and applying to the internal TA signal:

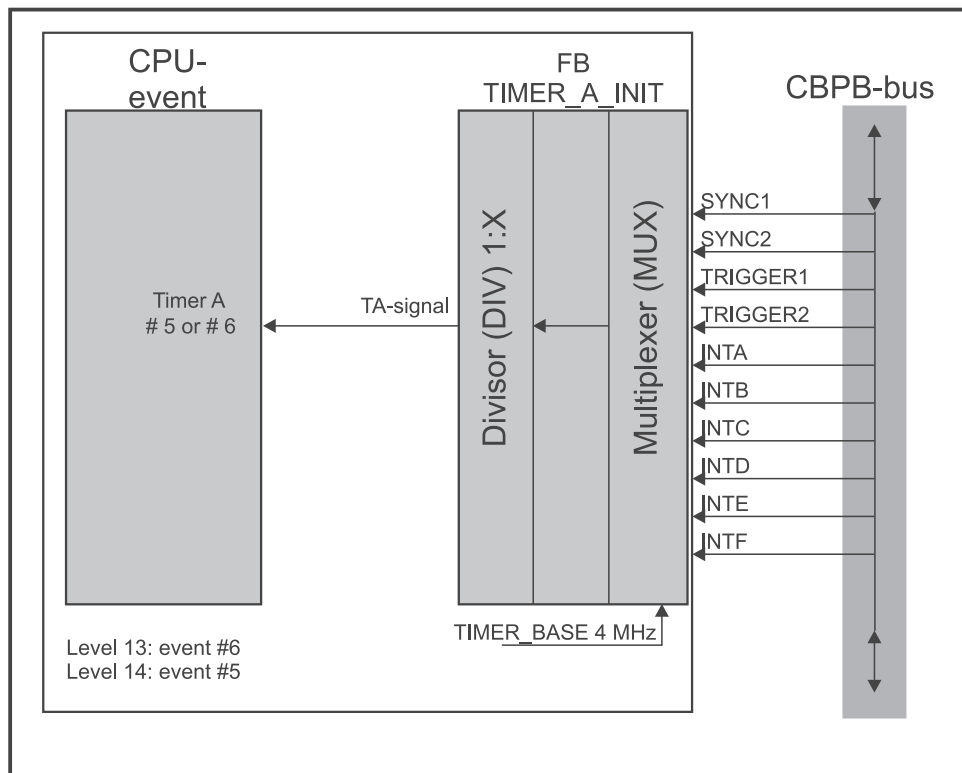


Figure 50: Detailed configuration options by the INPUT function of TIMER_A_INIT

5.6.4 Generating CBPB signals via function block TIMER_A_INIT

You can also configure the b maXX controller PLC for CBPB signal output (b maXX controller PLC -> CBPB); in this case, it can function as a trigger source for other system components (placed left hand of the b maXX controller PLC), for example.

In this connection, the signal source is generated either on the PLC itself or it comes from another CBPB 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 / SYSTEM1_C_PLC02_30bd00 (ProProg wt III) or higher).

Configuration options for generating CBPB signals by injecting the internally generated TA signal to the CBPB signal line via the OUTPUT function of block "TIMER_A_INIT":

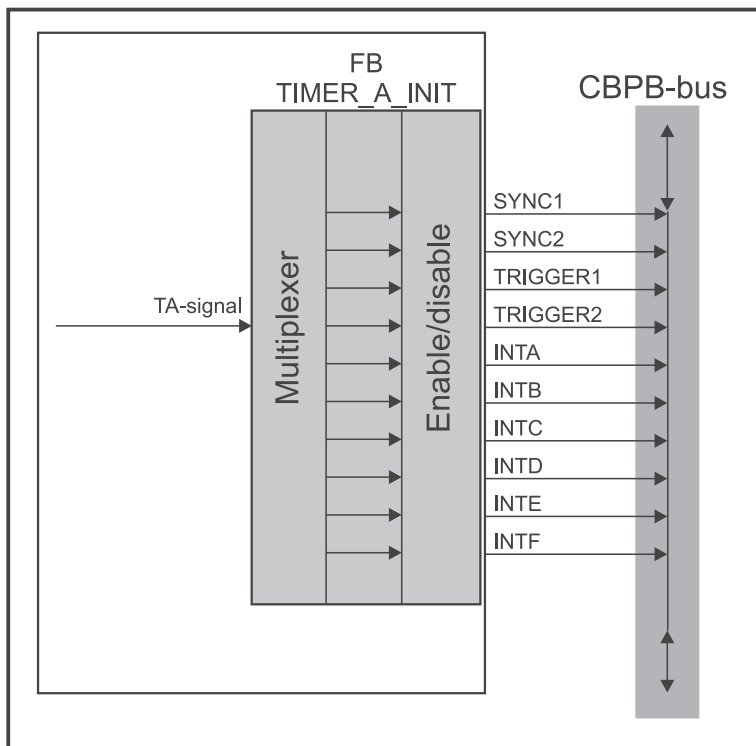


Figure 51: Configuration options for generating CBPB signals by the OUTPUT function of TIMER_A_INIT

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

You can reroute CBPB 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).

5.6.5 "TIMER_A_INIT" function block

5.6.5.1 "TIMER_A_INIT" function block for BMC-M-PLC-01

Description of function block "TIMER_A_INIT" of library SYSTEM1_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 (ProProg wt III) or higher.

The function block for CBPB has several functions:

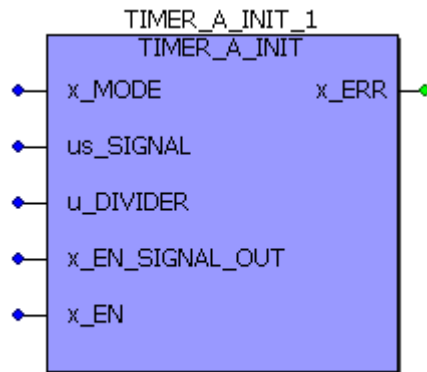


Figure 52: Function block TIMER_A_INIT

Function	Description
INPUT	You can choose a CBPB 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 CBPB output.
TRIGGER	If you combine the TIMER function with the OUTPUT function, you can switch the generated cycle signal to one or more CBPB outputs.
Enable/disable OUTPUT signal	You can disable CBPB 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 CBPB output has been released, the internal TA signal is switched through to the specified CBPB signal destination.

Input us_SIGNAL:

For x_MODE := FALSE: us_SIGNAL :=	Selection of signal source:
USINT#0	4 MHz
USINT#1 to 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

For x_MODE := TRUE: us_SIGNAL :=	Selection of CBPB 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:

$$\text{Internal TA signal} := \text{us_SIGNAL} / (\text{u_DIVIDER} + 1)$$

In the case of **u_DIVIDER = 0**, this means that the external CBPB 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 CBPB output (**x_EN_SIGNAL_OUT := TRUE**) or is disabled again (**x_EN_SIGNAL_OUT := FALSE**).

The internal TA signal can be output to several CBPB outputs by means of several successive block calls and different CBPB 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 CBPB input signal can be divided.

Output **x_ERR := TRUE**:

Selection of the CBPB 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:

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

Minimum value for "u_DIVIDER": UINT#999 => Time = 250 μs

Maximum value for "u_DIVIDER": UINT#65535 => Time = 16384 μs

Formula for calculating u_DIVIDER with the specified time period:

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

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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_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 at u_DIVIDER 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 CBPB outputs.

For this purpose the function block „TIMER_A_INIT“ must be called twice in succession in a cold boot or warm restart POU.

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 CBPB (e.g. a PLC function as a CBPB 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 CBPB-BUS timing.

Formula for calculating the time period:

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

Minimum value for "u_DIVIDER": UINT#8 => Time = 2.25 μs

Maximum value for "u_DIVIDER": UINT#65535 => Time = 16384 μs

Formula for calculating u_DIVIDER:

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

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 (CBPB destination = USINT#11 (Trigger 1)), see the following comments:

OUTPUT special function: Enable/disable CBPB signal:

If you want to output the internally generated TA signal to several CBPB outputs at the same time, the system can call the block several times in succession using the respective CBPB 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_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:

You want input CBPB signal TRIGGER1 to be divided by a value of 10 and then to be switched back to output CBPB 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.

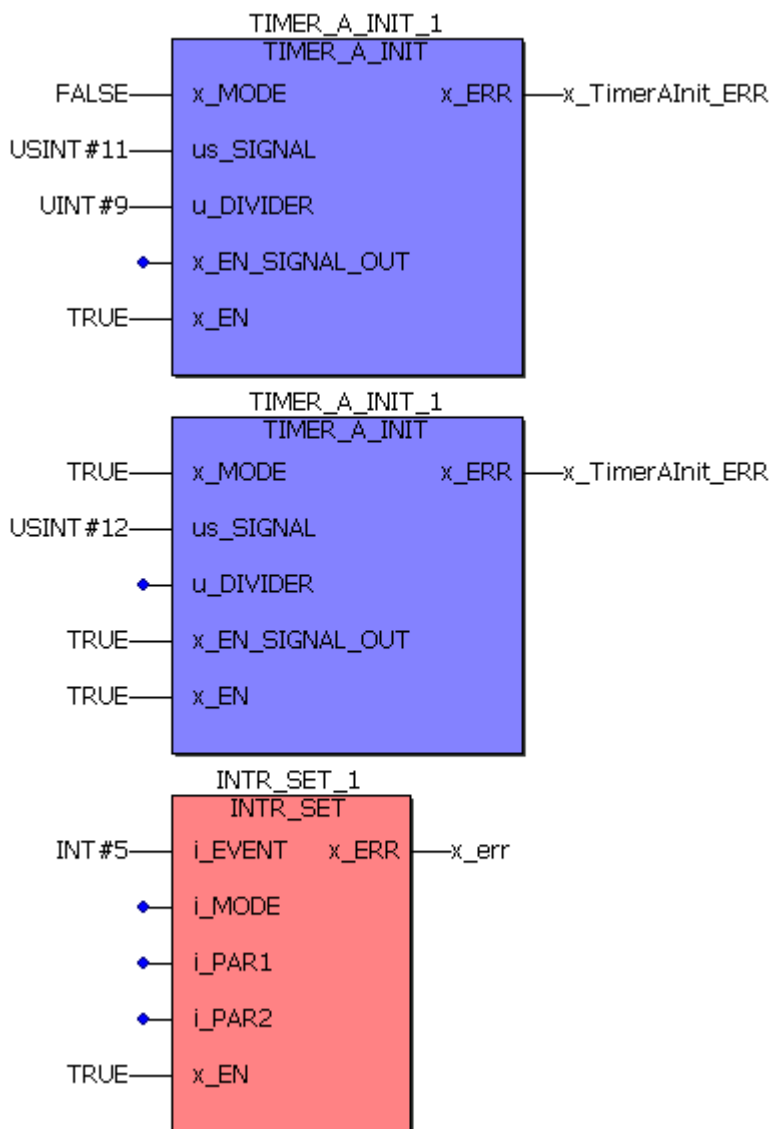


Figure 53: The TRIGGER1 signal triggers every tenth time the TRIGGER2 signal and the TRIGGER1 signal triggers the EVENT BYPASS task „5“.

5.6.5.2 "TIMER_A_INIT" function block for BMC-M-PLC-02

Description of function block "TIMER_A_INIT" of library SYSTEM1_C_PLC02_30bd02 (ProProg wt III) or higher.

The function block for CBPB has several functions:

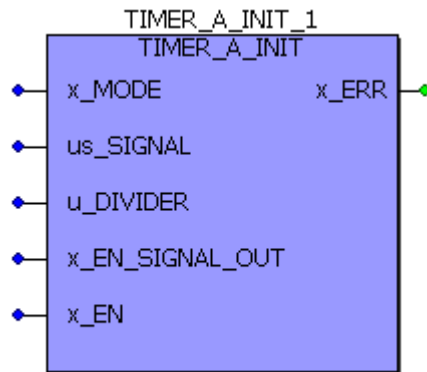


Figure 54: Function block TIMER_A_INIT

Function	Description
INPUT	You can choose a CBPB 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 CBPB output.
TRIGGER	If you combine the TIMER function with the OUTPUT function, you can switch the generated cycle signal to one or more CBPB outputs.
Enable/disable OUTPUT signal	You can disable CBPB 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 CBPB output has been released, the internal TA signal is switched through to the specified CBPB signal destination.

Input us_SIGNAL:

For x_MODE := FALSE: us_SIGNAL :=	Selection of signal source:
USINT#0	1 MHz
USINT#1 to 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

For x_MODE := TRUE: us_SIGNAL :=	Selection of CBPB 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:

$$\text{Internal TA signal} := \text{us_SIGNAL} / (u_DIVIDER + 1)$$

In the case of $u_DIVIDER = 0$, this means that the external CBPB 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 CBPB output ($x_EN_SIGNAL_OUT := TRUE$) or is disabled again ($x_EN_SIGNAL_OUT := FALSE$).

The internal TA signal can be output to several CBPB outputs by means of several successive block calls and different CBPB 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 CBPB input signal can be divided.

Output **x_ERR := TRUE**:

Selection of the CBPB 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 1 MHz is permanently preset. Formula for calculating the time period:

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

Minimum value for "u_DIVIDER": UINT#249 => Time = 250 μs

Maximum value for "u_DIVIDER": UINT#65535 => Time = 65535 μs

Formula for calculating u_DIVIDER with the specified time period:

$$U_DIVIDER = (\text{time period} / 1 \mu\text{s}) - 1$$

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_C_PLC02_30bd02 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 at u_DIVIDER 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#249.

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 CBPB outputs. For this purpose the function block „TIMER_A_INIT“ must be called twice in succession in a cold boot or warm restart POU.

1. Call of the TIMER function:

A timer basis of 1 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 CBPB (e.g. a PLC function as a CBPB 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 CBPB-BUS timing.

Formula for calculating the time period:

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

Minimum value for "u_DIVIDER": UINT#2 => Time = 3 μs

Maximum value for "u_DIVIDER": UINT#65535 => Time = 65535 μs

Formula for calculating u_DIVIDER:

$$u_DIVIDER = (\text{time period} / 1 \mu\text{s}) - 1$$

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#249.

2. Call of the OUTPUT function (CBPB destination = USINT#11 (Trigger 1)), see the following comments:

OUTPUT special function: Enable/disable CBPB signal:

If you want to output the internally generated TA signal to several CBPB outputs at the same time, the system can call the block several times in succession using the respective CBPB 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_C_PLC02_30bd02 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).

See [▶Example application: ◀](#) from page 84 onward.

5.6.6 Sample Configurations

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

The following examples A to C can be used for the majority of applications (see the following chapters [▶Example A: Setting up a CPU-timer 1 interrupt ◀](#) from page 90 onward, [▶Example B: Setting up a board timer A interrupt and triggering of the SYNC-signal 1 ◀](#) from page 91 onward and [▶Example C: Implementing simple serial RS485 communication via a CPU timer ◀](#) from page 95 onward).

5.6.7 Example A: Setting up a CPU-timer 1 interrupt

A CPU-timer 1 interrupt (level 14) with the period 20 ms is intended to generate for the b maXX controller PLC, at what a PLC program part is running in the b maXX controller PLC.

Execute FB INTR_SET with: i_EVENT := INT#0; i_MODE := INT#0; i_PAR1 := INT#400; x_EN := TRUE.

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

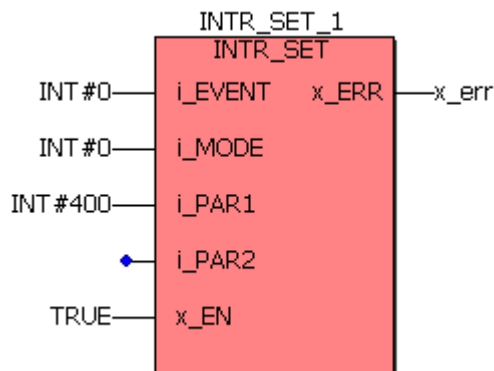


Figure 55: CPU-timer 1 triggers the Bypass Event task every 20 ms

The user program code is executed in an event task assigned to the event "0 CPU-Timer 1 Level 14".

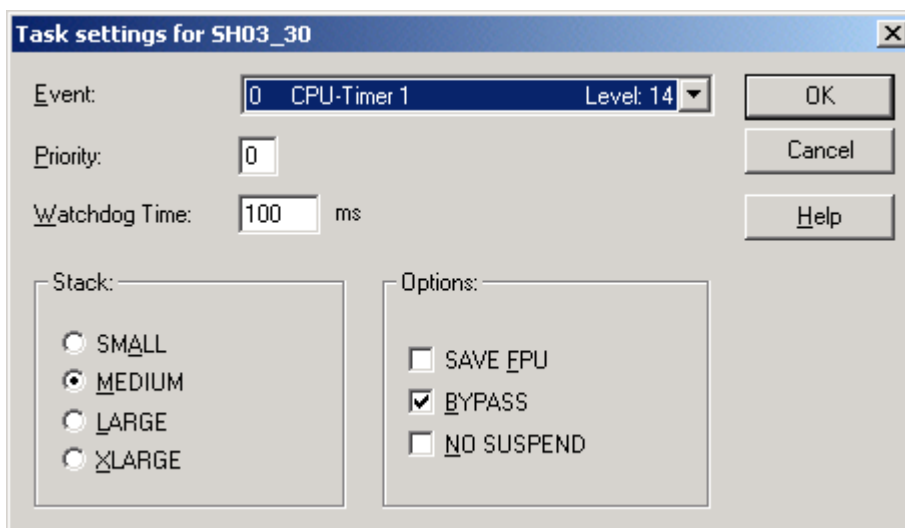


Figure 56: Settings for the Bypass Event task to event 0 CPU-Timer 1 Level 14

5.6.8 Example B: Setting up a board timer A interrupt and triggering of the SYNC-signal 1

At setting up the board timer A it is differentiated between BMC-M-PLC-01 and BMC-M-PLC-02.

5.6.8.1 Board timer A interrupt on BMC-M-PLC-01

The b maXX controller PLC is intended to generate a trigger signal with a period of 2 ms on the SYNC-signal 1 line. A PLC program part (with level 14) is intended to be executed synchronous to 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 μs / 0,25 μs) -1 = UINT#7999; x_EN := TRUE;
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; x_EN := 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.

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

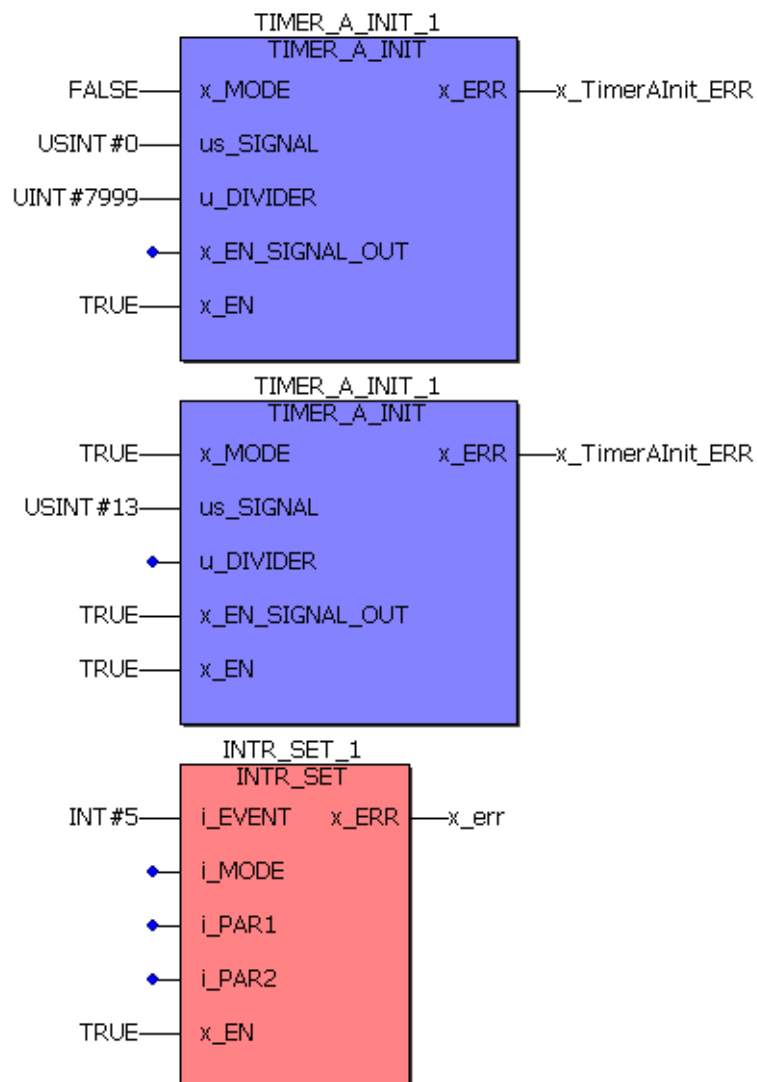


Figure 57: The Board timer A signal triggers the SYNC signal 1 every 2 ms and the Board timer A signal triggers the Bypass event task 5.

The program code is executed in an event task which is assigned to the event "5 Timer A Level 14".

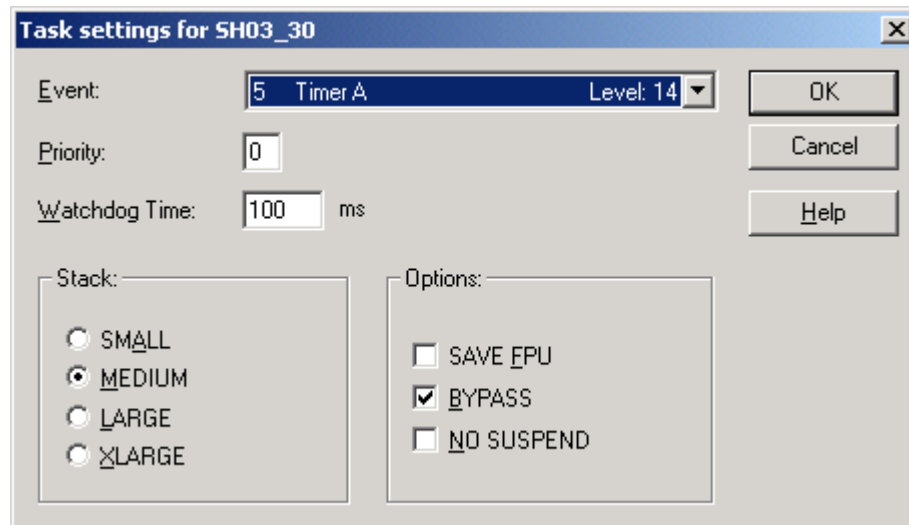


Figure 58: Settings for the Bypass Event task to event 5 Timer A Level 14

5.6.8.2 Board timer A interrupt on BMC-M-PLC-02

The b maXX controller PLC is intended to generate a trigger signal with a period of 2 ms on the SYNC-signal 1 line. A PLC program part (with level 14) is intended to be executed synchronous to 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 µs / 1 µs) - 1 = UINT#1999; x_EN := TRUE;`
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; x_EN := 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.

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

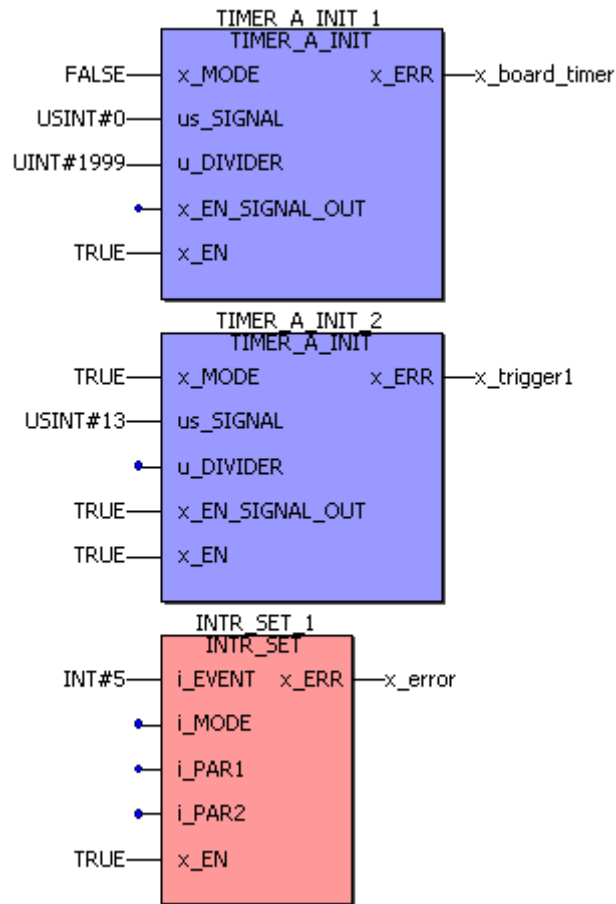


Figure 59: The Board timer A signal triggers the SYNC signal 1 every 2 ms and the Board timer A signal triggers the Bypass event task 5.

The program code is executed in an event task which is assigned to the event "5 Timer A Level 14".

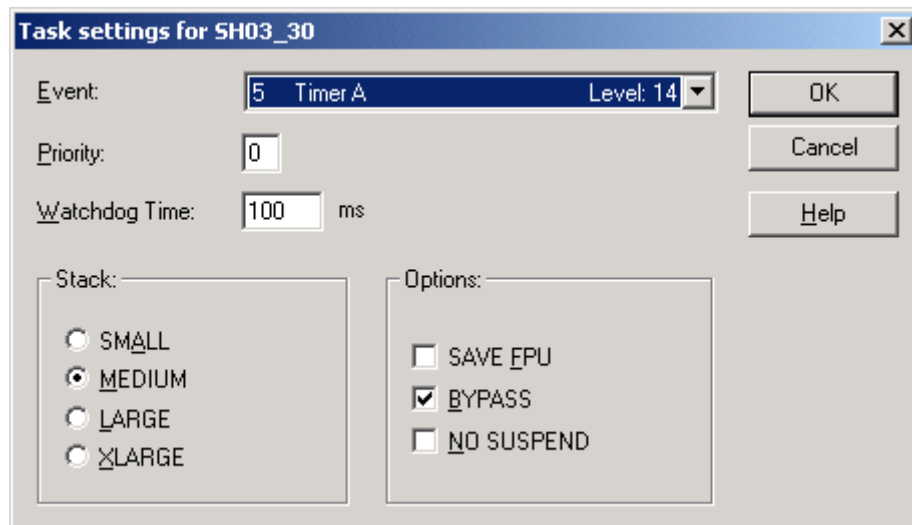


Figure 60: Settings for the Bypass Event task to event 5 Timer A Level 14

5.6.9 Example C: 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 controller PLC interface X2). 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 X2 interface, see the b maXX controller PLC 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 *)

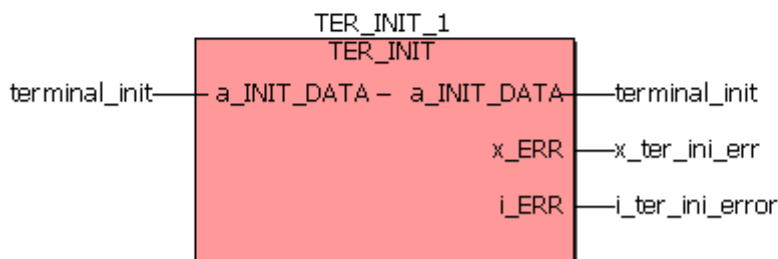


Figure 61: Initializing the serial port

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 X2 interface \(9-pin female Sub-D connector\) on the b maXX controller PLC](#) from page 97 onward or the online description in „SYSTEM1_C_PLC01_20bd00“ (PROPROG wt II) or „SYSTEM1_C_PLC01_30bd00“ / „SYSTEM1_C_PLC02_30bd02“ (ProProg wt III) or higher.

5.6 CBPB system description for the b maXX controller PLC

(* Event #2: CPU timer 2 level 13;
i_PAR1 = #200 => interrupt interval time is 10 mSec *)

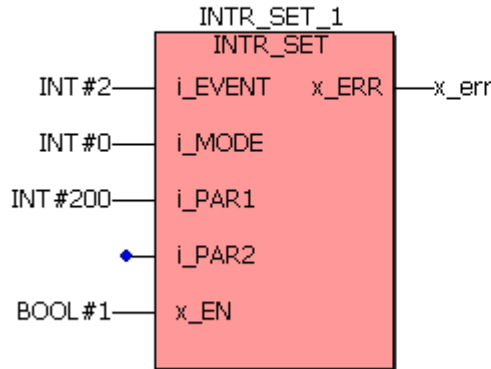


Figure 62: Set up a 10-ms bypass interrupt Level 13 in which the send and receive blocks are to be processed:

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

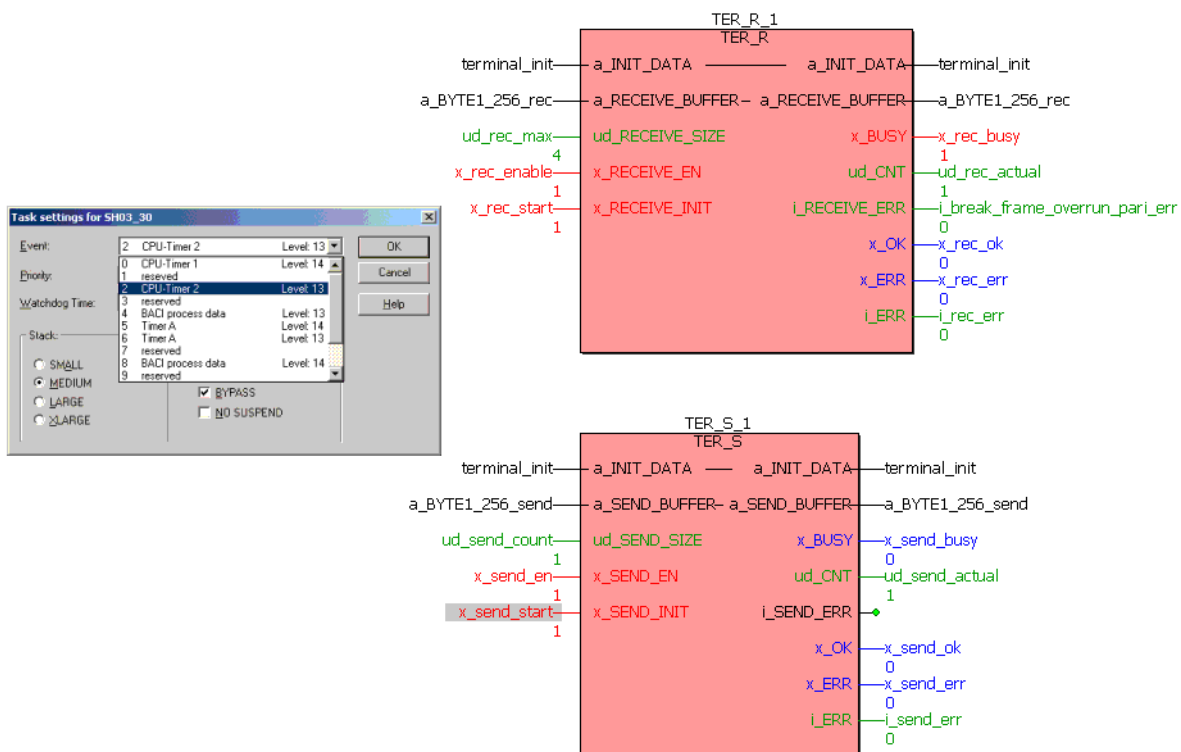


Figure 63: Set up the Bypass event task for the event 2 CPU-Timer 2 Level 13 and calling the send and receive blocks

Brief description

For detailed information on firmware blocks, see [►Description of the blocks for serial RS485 communication via the X2 interface \(9-pin female Sub-D connector\) on the b maXX controller PLC](#) ◀ from page 97 onward or the online descriptions in „SYSTEM1_C_PLC01_20bd00“ (PROPROG wt II) or „SYSTEM1_C_PLC01_30bd00“ / „SYSTEM1_C_PLC02_30bd02“ (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.

5.6.10 Description of the blocks for serial RS485 communication via the X2 interface (9-pin female Sub-D connector) on the b maXX controller PLC

For details on connection assignments and cables of the X2 interface, see the b maXX controller 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 X2 interface.

The +5 V at Pin 2 of interface X2 on the BMC-M-PLC-01 Sub-D socket or of interface X2 on the BMC-M-PLC-02 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 X2 is galvanically separated and optically decoupled.

The RS485 is a possible 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).

Therefore the send driver of a RS485 must be interruptible in principle. Functionally a RS485 includes a RS422.

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 controller 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 controller 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 II firmware library SYSTEM2_C_PLC01_20bd00 or in ProProg wt III firmware library SYSTEM2_C_PLC01_30bd00 or SYSTEM2_C_PLC02_30bd02 or higher (see [▶Function block TER_INIT ◀](#) from page 99 onward) or their online description.

- Configuration option with the b maXX controller 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[®] 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 controller 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 firmware library SYSTEM2_C_PLC01_20bd00 or in ProProg wt III firmware library SYSTEM2_C_PLC01_30bd00 or SYSTEM2_C_PLC02_30bd02 or higher or their online description.

5.6.11 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_C_PLC_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC_30bd00 / SYSTEM2_C_PLC_30bd02 (ProProg wt III) or above.

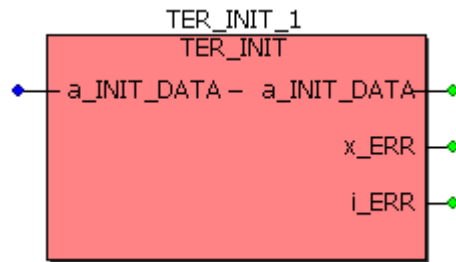


Figure 64: Function block TER_INIT

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 controller PLC 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 [®] 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 ¹⁾
a_INIT_DATA[3]	Baud rate 0: 50 bps ¹⁾ 1: 110 bps ¹⁾ 2: 150 bps ¹⁾ 3: 300 bps ¹⁾ 4: 600 bps ¹⁾ 5: 1200 bps ¹⁾ 6: 2400 bps ¹⁾ 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 ²⁾
a_INIT_DATA[9]	Priority ²⁾ 0: Low priority 1: High priority
a_INIT_DATA[10]	Character delay time (ZVZ) in ms ²⁾

a_INIT_DATA[Index]	Meaning
a_INIT_DATA[11]	Acknowledgment delay (QVZ) in ms ²⁾
a_INIT_DATA[12]	Block delay (BVZ) in ms ²⁾
a_INIT_DATA[13]	Reserved
...	...
a_INIT_DATA[255]	Reserved

1) Not for interfacing to procedure 3964R[®] to data blocks

2) Only for interfacing to procedure 3964R[®] to data blocks

Procedure 3964R[®] 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III) or above.

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[Index]	Meaning
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[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_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_PLC01_30bd00 / SYSTEM2_PLC02_30bd02 (ProProg wt III) or above).

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[Index]	Meaning
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[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

5.6.12 Function block TER_R

Firmware FB TER_R receives characters via the terminal interface (serial port) of the b maXX controller PLC module and stores the characters in an array.

The FB TER_R is contained in the firmware library SYSTEM2_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III) or above.

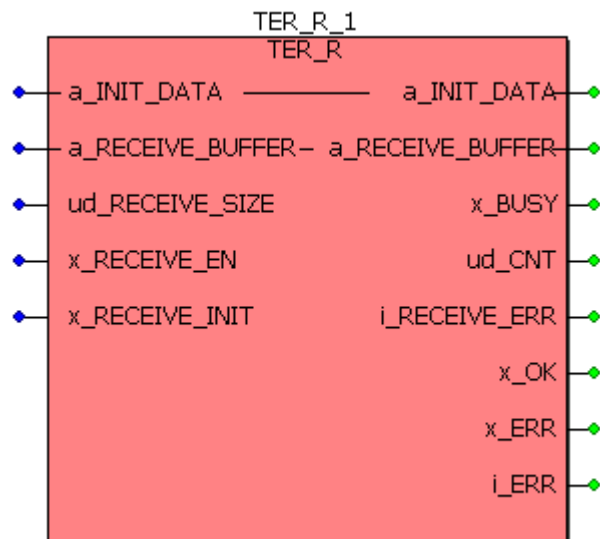


Figure 65: Function block TER_R

Parameter input	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_RECEIVE_BUFFER	ANY ¹⁾	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 ¹⁾	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
x_ERR	BOOL	Error bit
i_ERR	INT	Error number

¹⁾ 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_20bd06 (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

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):

i_ERR	Error on serial port
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.

5.6.13 Function block TER_S

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

The FB TER_S is contained in the firmware library SYSTEM2_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_PLC01_30bd00 / SYSTEM2_PLC02_30bd02 or above.

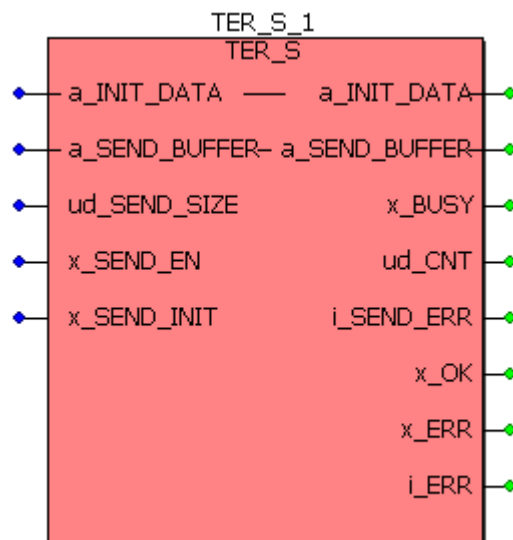


Figure 66: Function block TER_S

Parameter input	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_SEND_BUFFER	ANY ¹⁾	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

Parameter output	Data type	Description
a_INIT_DATA	UINT_256_BMARRAY	Initialization array for the terminal interface
a_SEND_BUFFER	ANY ¹⁾	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

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

- 1) 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_20bd06 (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:

i_ERR	Error on serial port
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.

5.6.14 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III) or above.

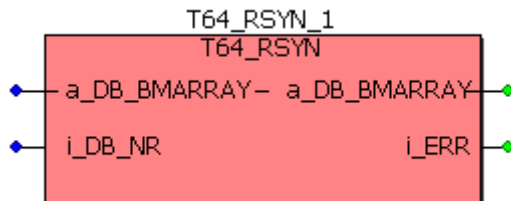


Figure 67: Function block T64_RSYN

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

¹⁾ 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_20bd06 (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.

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 .

5.6.15 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 controller PLC module and another node.

The FB T64_REC is contained in the firmware library SYSTEM2_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III) or above.

Interfacing to procedure 3964R[®] 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[®] is a registered trademark of Siemens AG.

In this connection the terminal interface on b maXX PLC option module is always a slave interfacing to procedure 3964R[®] to data blocks.

For interfacing to procedure 3964R[®] 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).

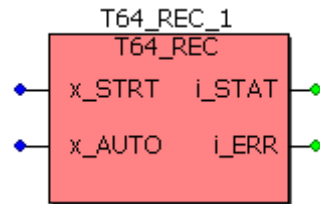


Figure 68: Function block T64_REC

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 PLC option module interfacing to procedure 3964R[®] to data blocks represents the responding slave.

Transferring data using RS485 interfaces on the b maXX controller PLC 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 controller PLC 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 controller PLC module. If a receive error occurs during communication, the system displays error status "-1" at output **i_STAT**.

i_STAT	Internal status of communication of interfacing to procedure 3964R® to data blocks
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.

i_ERR	Error number
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

5.6.16 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 PLC BMC-M-PLC-01.



NOTE

Software interfacing to the USS protocol[®] is available only at BMC-M-PLC-01.

The b maXX controller 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 (ProProg wt III) or above and uses firmware FB USS_SR from firmware library SYSTEM2_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_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_20bd06 (PROPROG wt II) or BM_TYPES_30bd01 (ProProg wt III) or higher are used.

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

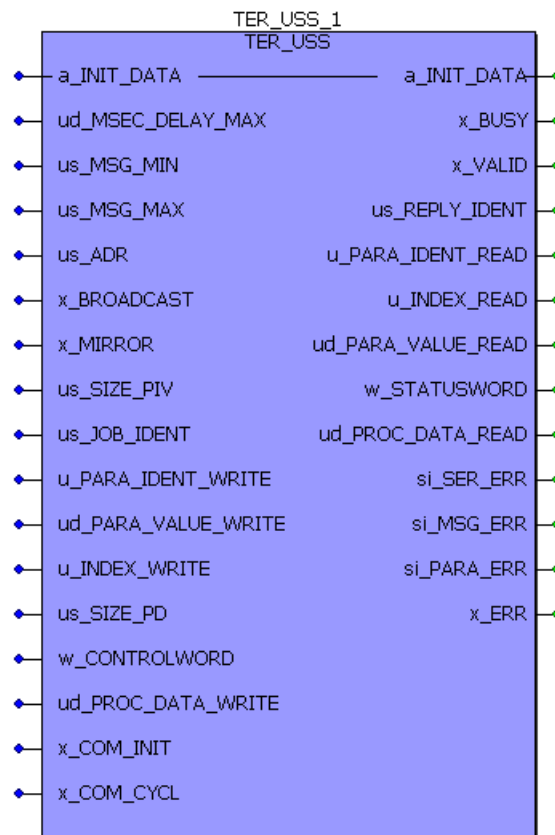


Figure 69: Function block TER_USS

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
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

Parameter output	Data type	Description
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 controller 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 and 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[®] is specified by the PKW and PZD quantity. The settings must be identical for all the nodes in the USS ring so that the system can correctly interpret the contents of the exchanged message frame.

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

us_SIZE_PD	PZD quantity: Subdivision of process data in the USS protocol
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!

us_JOB_IDENT	Job ID (Master -> Slave)
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)

5.6 CBPB system description for the b maXX controller PLC

us_JOB_IDENT	Job ID (Master -> Slave)
4	Read an element from the parameter description (which on is in u_INDEX_WRITE)
5 to ...	(Reserved)

us_REPLY_IDENT	Answer ID (Slave -> Master)
0	No job
1	Parameter wort (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:

si_SER_ERR	Error on serial port
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

si_MSG_ERR	Message frame error on sending/receiving
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

si_MSG_ERR	Message frame error on sending/receiving
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

si_PARA_ERR	Parameter error
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	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

5.7 Code runtimes

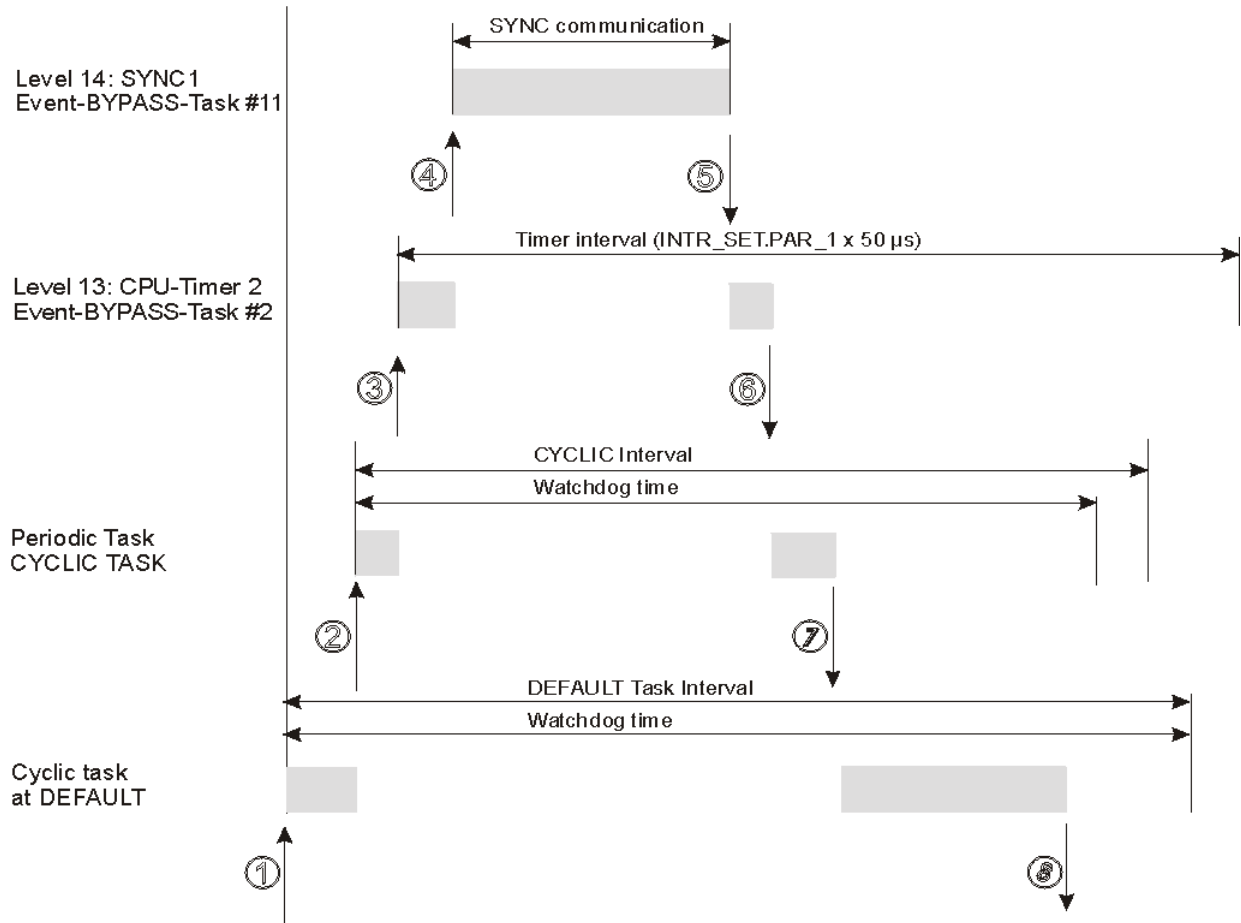


Figure 70: Distribution of code runtimes to resource BMC_M_PLC01

**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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 / SYSTEM1_C_PLC02_30bd02 (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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III)).

It is possible to deactivate a BYPASS event at runtime using FB "INTR_SET" of firmware library SYSTEM2_C_PLC01_20bd00 (PROPROG wt II) or

SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (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.

5.7.1 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 function block TIME_MEASURE_START is contained in the standard library SYSTEM1_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 / SYSTEM1_C_PLC02_30bd02 (ProProg wt III) or above.

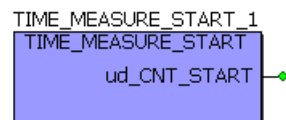


Figure 71: Function block TIME_MEASURE_START

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 controller 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 Bypass event task with the level 14. Runtimes up to a maximum of 10 ms (BMC-M-PLC01) or 15 min (BMC-M-PLC-02) can be determined. In the case of program code with runtimes longer than 10 ms (BMC-M-PLC01) or 15 min (BMC-M-PLC-02), no correct values will be returned.

If the time measurement blocks are used in a Bypass event task with level 13 and if they are interrupted by a Bypass event with level 14, this task will be measured completely!

5.7.2 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 function block TIME_MEASURE_END is contained in the standard library SYSTEM1_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 / SYSTEM1_C_PLC02_30bd02 (ProProg wt III) or above.

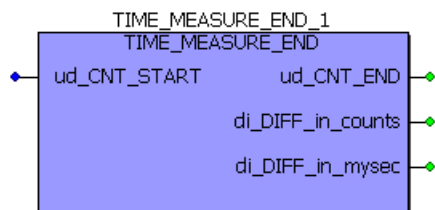


Figure 72: Function block TIME_MEASURE_END

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 controller 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 Bypass event task with the level 14. Runtimes up to a maximum of 10 ms (BMC-M-PLC01) or 15 min (BMC-M-PLC-02) can be determined. In the case of program code with runtimes longer than 10 ms (BMC-M-PLC01) or 15 min (BMC-M-PLC-02), no correct values will be returned.

If the time measurement blocks are used in a Bypass event task with level 13 and if they are interrupted by a Bypass event with level 14, this task will be measured completely!

5.8 Practical information

5.8.1 General PLC error

You will find a list of PLC errors in the Online help of PROPROG wt in:

? \ „Help for PLC“ → Register „Contents“ → Item „PLC specific help“ - „error catalogue“.

5.8.2 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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM1_C_PLC01_30bd00 / SYSTEM1_C_PLC02_30bd02 (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_C_PLC01_20bd00 (PROPROG wt II) or SYSTEM2_C_PLC01_30bd00 / SYSTEM2_C_PLC02_30bd02 (ProProg wt III)).

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.

5.8.3 PLC error "Global ready message (RST signal) is missing!"

The b maXX controller PLC module didn't get the global ready message from the following modules:

- Power supply module for b maXX controller PLC
- Modules which are plugged on left hand of the b maXX controller PLC (e.g. Ethernet with CANopen-Master)

This can be due to the following reason:

- A module is faulty or there is a connection error of a module.
 - Does the error also occur if only the b maXX controller PLC and the power supply unit are used?

5.8 Practical information

You must only plug in and remove the modules when they are deenergized. For assembly and dismantling see the respective Operating Instructions.

Yes: Check whether there are bent pins between the b maXX controller PLC and the power supply unit.

Replace the b maXX controller PLC module or the power supply unit, if necessary.

No Successively assemble the other modules and their connections until you determine the fault module

5.8.4 PLC error "No access to the CX-controller in the power supply unit possible!"

The b maXX controller PLC cannot communicate with the power supply unit.

- A module is faulty or there is a connection error of a module.

→ Check whether there are bent pins between the b maXX controller PLC and the power supply unit.

If the error is generated furthermore replace the b maXX controller PLC module or the power supply unit, if necessary.

5.8.5 PLC error "I/O bus error occurs! Error code: ..."

The error code is specified as 0xAABB. For the meanings of AA and BB see the following table:

Error code		Meaning	Corrective
AA	BB		
01	00	I/O bus: Mapping error (offset 0xFDC/0xFDE false)	The modules of a group (digital input, digital output, analog input, analog output) use more than 512 bytes in the related data area (see b maXX controller PLC Application Manual)
03	00	I/O bus: Command error	An I/O module responds faultily to an I/O bus command: <ol style="list-style-type: none"> 1. The I/O module is faulty. Change the I/O module. 2. No I/O module is plugged. Plug in a I/O module 3. Only the end module is plugged. Plug in an I/O module
04	n	I/O bus: Breakage after I/O module n	Check the I/O module after module n and change it if necessary
05	n	I/O bus: Error at communication with I/O module n	Check I/O module n and change it if necessary

Error code		Meaning	Corrective
AA	BB		
14	n	I/O bus: Bus fault	Check I/O module n and change it if necessary
15	n	I/O bus: Bus fault	Check I/O module n and change it if necessary

UTILIZE MODULES (TO THE RIGHT OF THE B MAXX CONTROLLER PLC)

The I/O modules to the right of the b maXX controller PLC can be configured automatically using either ProProg wt III or PROPROG wt II.

Use the Engineering Framework [►ProMaster ◀](#) from page 131 onward in ProProg wt III.

Use the [►ProModul module configurator ◀](#) from page 146 onward in PROPROG wt II.

6.1 ProMaster

Components/modules connected to the left and right of the b maXX controller PLC can be configured using the ProMaster Engineering Framework system.

Here's how to configure the modules to the right of the b maXX controller PLC.

ProMaster creates project communication variables for the modules in ProProg wt III, thereby allowing module inputs to be read and module outputs to be written.

NOTE



To run ProMaster you need ProProg wt III.

Steps to be taken

Requirements

Assembly, connection and operation of

- b maXX controller PLC
- Power supply unit for b maXX controller PLC
- System components to the left of the b maXX controller PLC are connected (e. g. CAN-open master)
- System components to the right of the b maXX controller PLC are connected (e. g. digital and/or analog I/O modules, end modules)

must be successfully completed.

The following steps must be completed to allow system components to be called and ensure data exchange via the b maXX controller PLC.

- Create a ProProg wt III project for the b maXX controller PLC
- Create a ProMaster Project with b maXX controller PLC
- Configure the b maXX System using ProMaster and associate the ProMaster project to the ProProg wt III project
- Configure the I/Os using ProMaster

Result: Variables list for the initialization and/or operation of these system components.

Now program the application component in the ProProg wt III project for the b maXX controller PLC. Instructions for programming the system components connected to the right of the b maXX controller PLC can be found in the relevant applications manuals.

Example: This chapter features an example for the b maXX controller PLC (BMC-M-PLC-02). In the example we create a configuration with (from left to right) the following system components/modules found to the right of the b maXX controller PLC:

Power supply unit for b maXX controller PLC

- 8 digital outputs +24V DC (DO8000)
- 8 digital inputs +24V DC (DI8000)
- 2 analog outputs 0-10V (AO2010)
- 2 analog inputs 0-10V (AI2010)
- end module (EK0000)

6.1.1 Create a ProProg wt III project for the b maXX controller PLC

6.1.1.1 General

If you haven't created your own project yet for the application – see section [b maXX controller PLC Projekt](#) < from page 35 onward, please create the project "Example_C_PLC02" using the "BMC_M_PLC02" template. For this you need ProProg wt III from Version 1.1 Build 78 and ProMaster from Version 1.1.5.0. You will find the version number for ProProg wt III on the ProProg wt III installation CD cover or in ProProg wt III under the ? \ Info menu item.

Check whether the libraries

- BM_TYPES_30bd01 (or higher)
- SYSTEM2_C_PLC02_30bd02 (or higher)

and, if relevant, any specific libraries used by your system components, can be found in your ProProg wt III project.

If this is not the case, please integrate these libraries into your project. They contain vital data types and functional building blocks for your system components.

**NOTE**

You can also create an "Example_C_PLC01" project using the BMC_M_PLC01 resource. In this case use the template project BMC_M_PLC01 and the BM_TYPES_30bd01 (or higher) and SYSTEM2_C_PLC01_30bd00 (or higher) libraries

6.1.1.2 Example: Creating the "Example_C_PLC02" project

The project "Example_C_PLC02" has been created using the "BMC_M_PLC02" template and linked to the libraries

- BM_TYPES_30bd01
- SYSTEM2_C_PLC02_30bd02.

The example does not feature any system components/modules connected to the left of the b maXX controller PLC. No associated libraries need therefore be linked in.

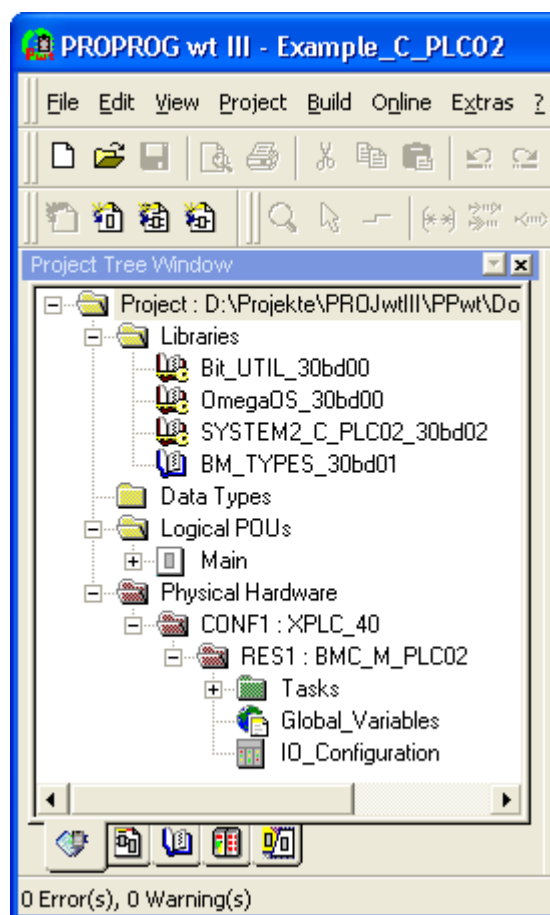


Figure 73: Example: Creating the "Example_C_PLC02" project

6.1.2 Creating a ProMaster project using b maXX controller PLC

In this section a ProMaster project is created using b maXX controller PLC
Open ProMaster:

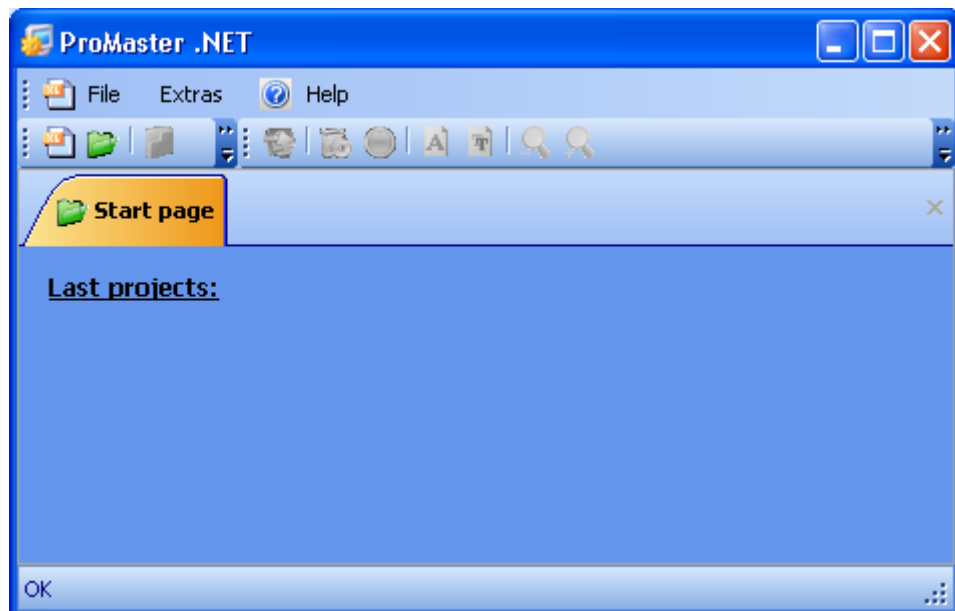


Figure 74: ProMaster without projects

Create a new ProMaster project by clicking on the menu item "File\New Project" to open the "Project Settings" dialog box.

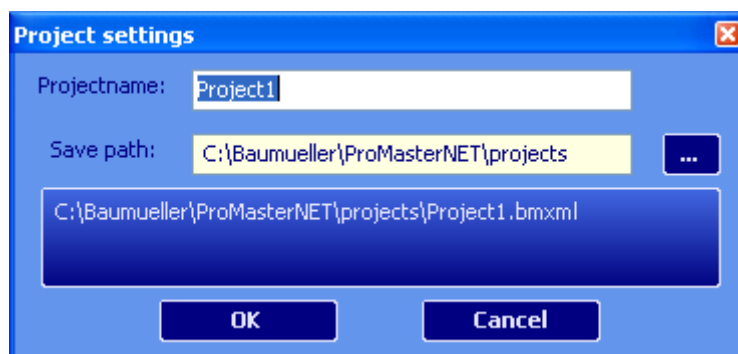


Figure 75: ProMaster – Project name

Enter the name (in the Edit box) and the save location (using the Select Path drop-down list box) of the project. For our sample project we use the name "Example_C_PLC02.bmxml".

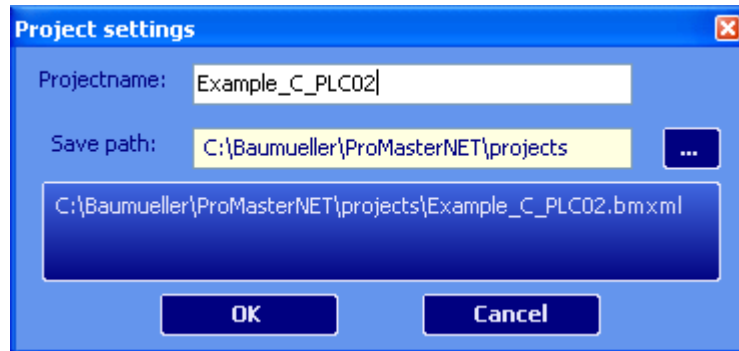


Figure 76: ProMaster – Entering project name

Accept the settings shown in the "Project Settings" dialog box by clicking the OK button. ProMaster now switches to the field bus view.

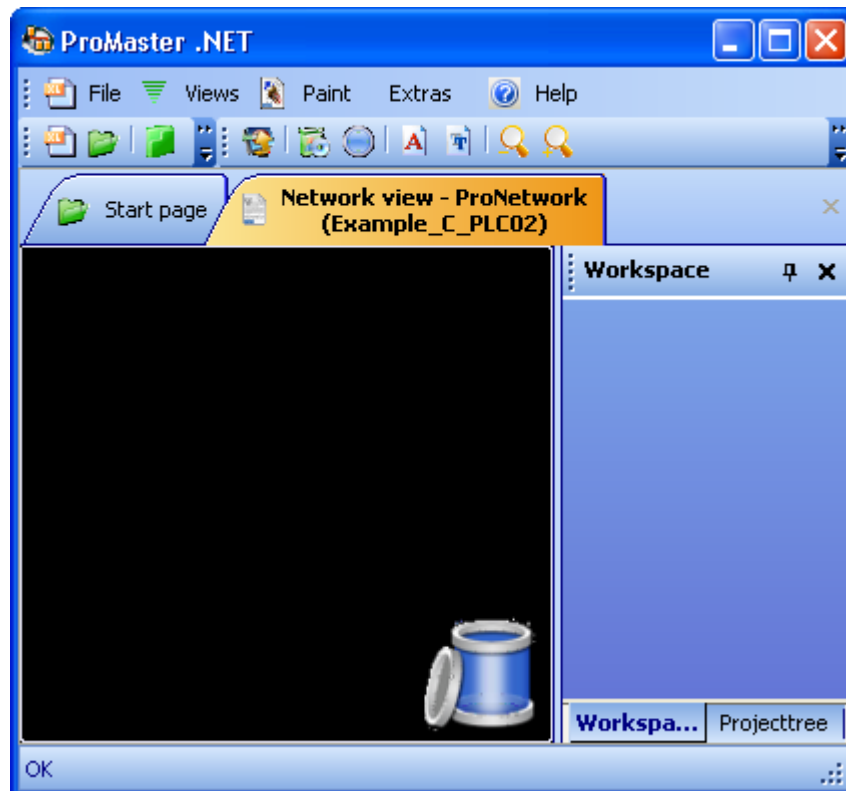


Figure 77: ProMaster – Entering project name

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

In the example we use the

- "b maXX controller PLC CANopen master" device (from the "b maXX controller PLC System" group)

Since we are creating our example in the b maXX controller PLC system without using CANopen master, we delete CANopen master from our b maXX controller PLC system

by clicking on "Delete" under the bus CANOpen master entry in the context menu (the yellow line under the b maXX controller PLC turns red).

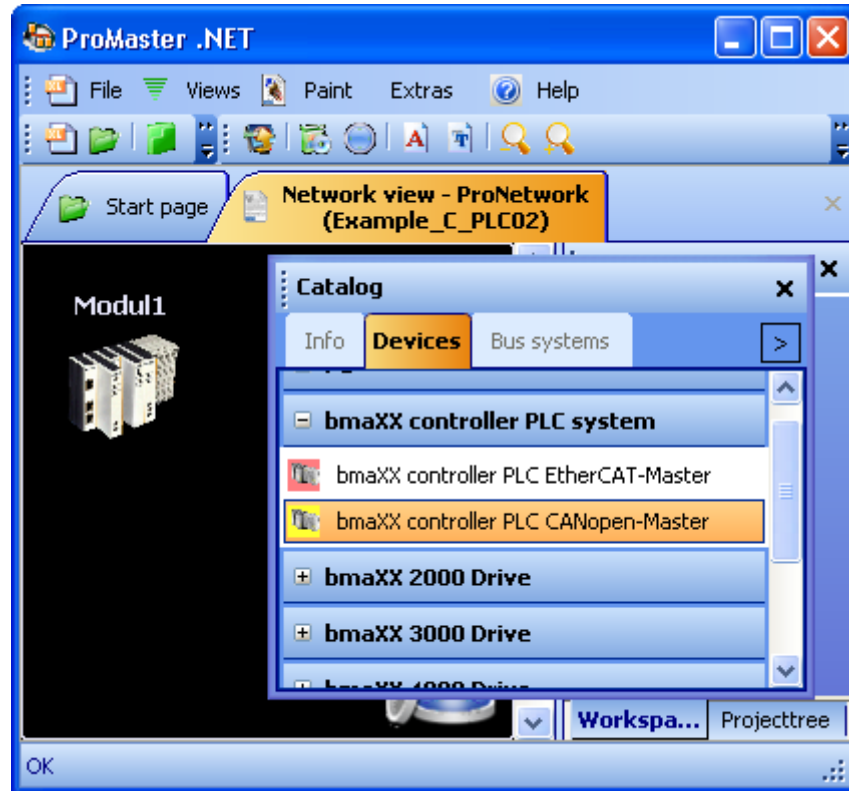


Figure 78: ProMaster – Machine configuration is created

On screen you can now see the b maXX controller PLC. Close the catalog screen for a better view.

You can change the name of the module by clicking on the module and then opening the module's Properties screen via the Properties item in the context menu.

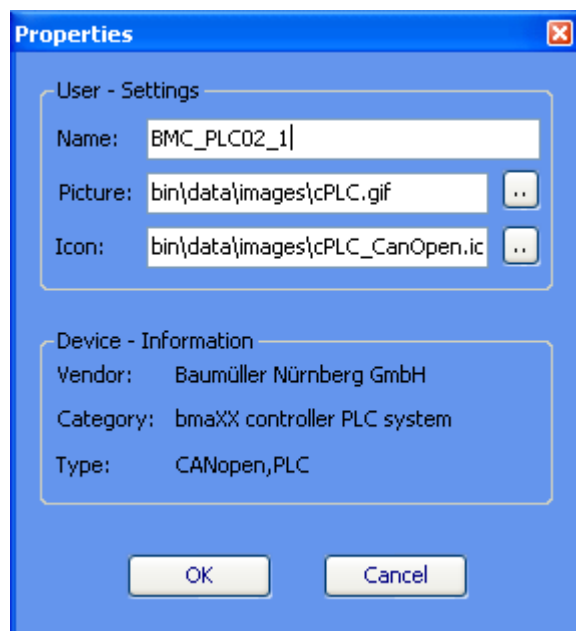


Figure 79: ProMaster – Module properties

In our example we have given the b maXX controller PLC the name BMC_PLC02_1. The new name now appears in the ProMaster network view and in the project tree.

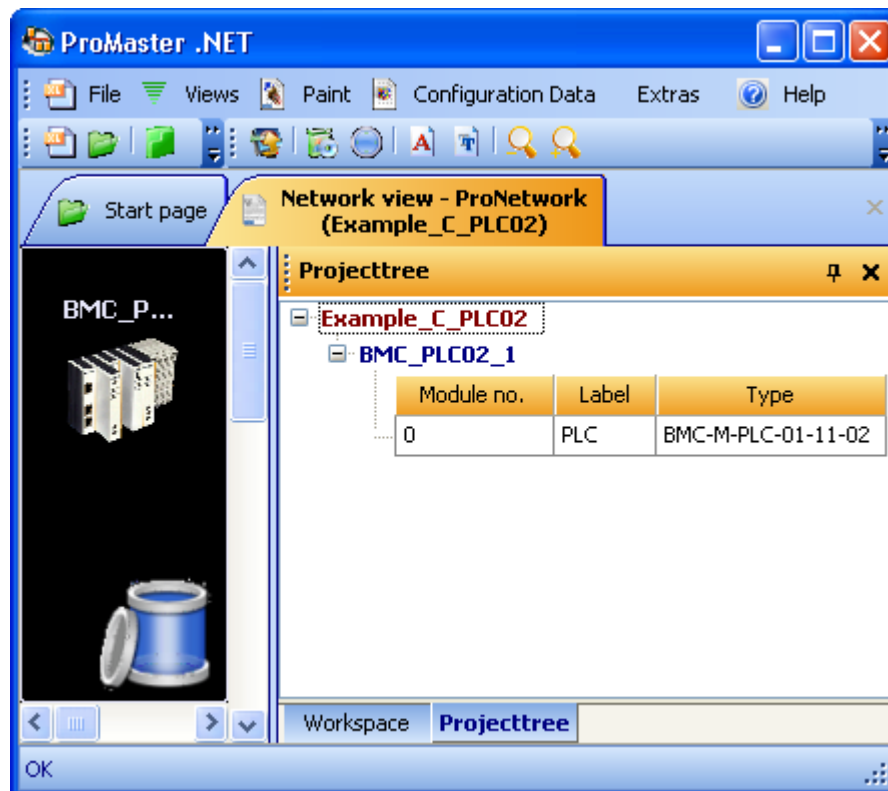


Figure 80: ProMaster – Machine configuration – Changing module name

You can adjust the communication settings for communications between ProMaster and the b maXX controller PLC by clicking on the "BMC_PLC_1" device and selecting "Communication settings" from the context menu. Then save the project by clicking on the "File\Save project" menu item.

6.1.3 Configuring the b maXX System using ProMaster

Now it's time to integrate the ProProg wt III project "Example_C_PLC02.mwt" into the ProMaster project "Example_C_PLC02.bmxml".

In the ProMaster project's b maXX controller PLC network view, open the "PLC configuration" screen by clicking on the device "BMC_PLC02_1" and selecting "Configuration data\PLC (module number 0)\PLC - configuration" from the context menu. Now select the "IEC" tab.

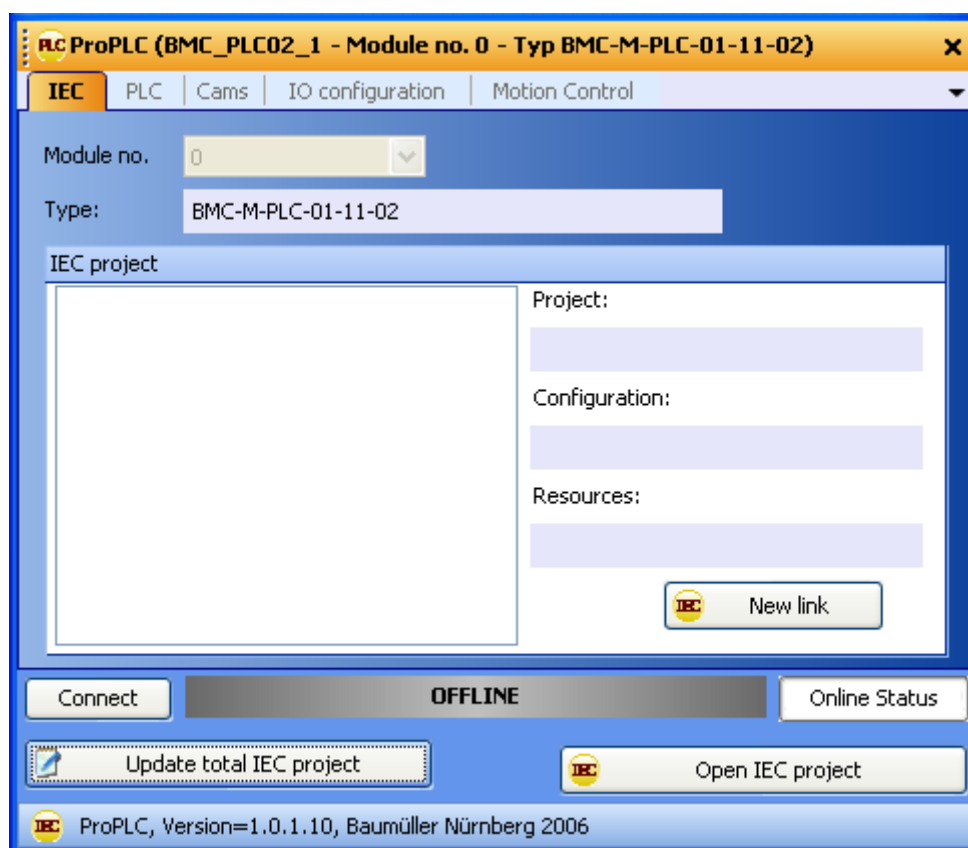


Figure 81: "IEC" tab



NOTE

ProMaster can only open ProProg wt III projects. If you want to use an existing PROPROG wt II project, you first need to open this with ProProg wt III (thereby converting the file). You will now be able to open and use it in ProMaster.

Please note that your PROPROG wt II libraries will be converted at the same time!

Click on the "New link" button and select our sample IEC project "Example_C_PLC02.mwt", which we created in chapter [▶ Create a ProProg wt III project for the b maXX controller PLC](#) ◀ from page 132 onward.

The selected project appears in the "IEC" tab.

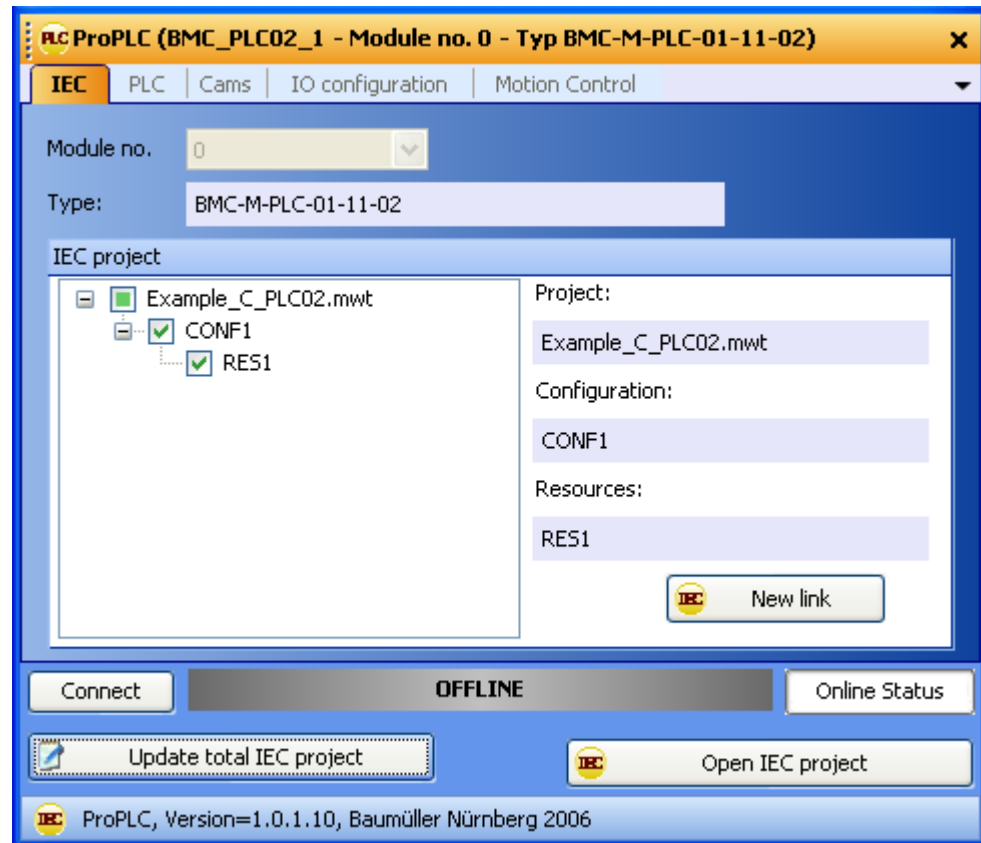


Figure 82: "IEC" tab

Now save the settings you have made in ProMaster using the "File\Save project" menu item.

6.1.4 Configuring the I/Os with ProMaster

In this section we explain how to create an I/O configuration for the b maXX controller PLC and connect it with ProMaster.

In the ProMaster project's b maXX controller PLC network view, open the "Devices view" by clicking on the "BMC_PLC02_1" device and selecting "Configuration data\PLC (module number 0)\I/O configuration" from the context menu.

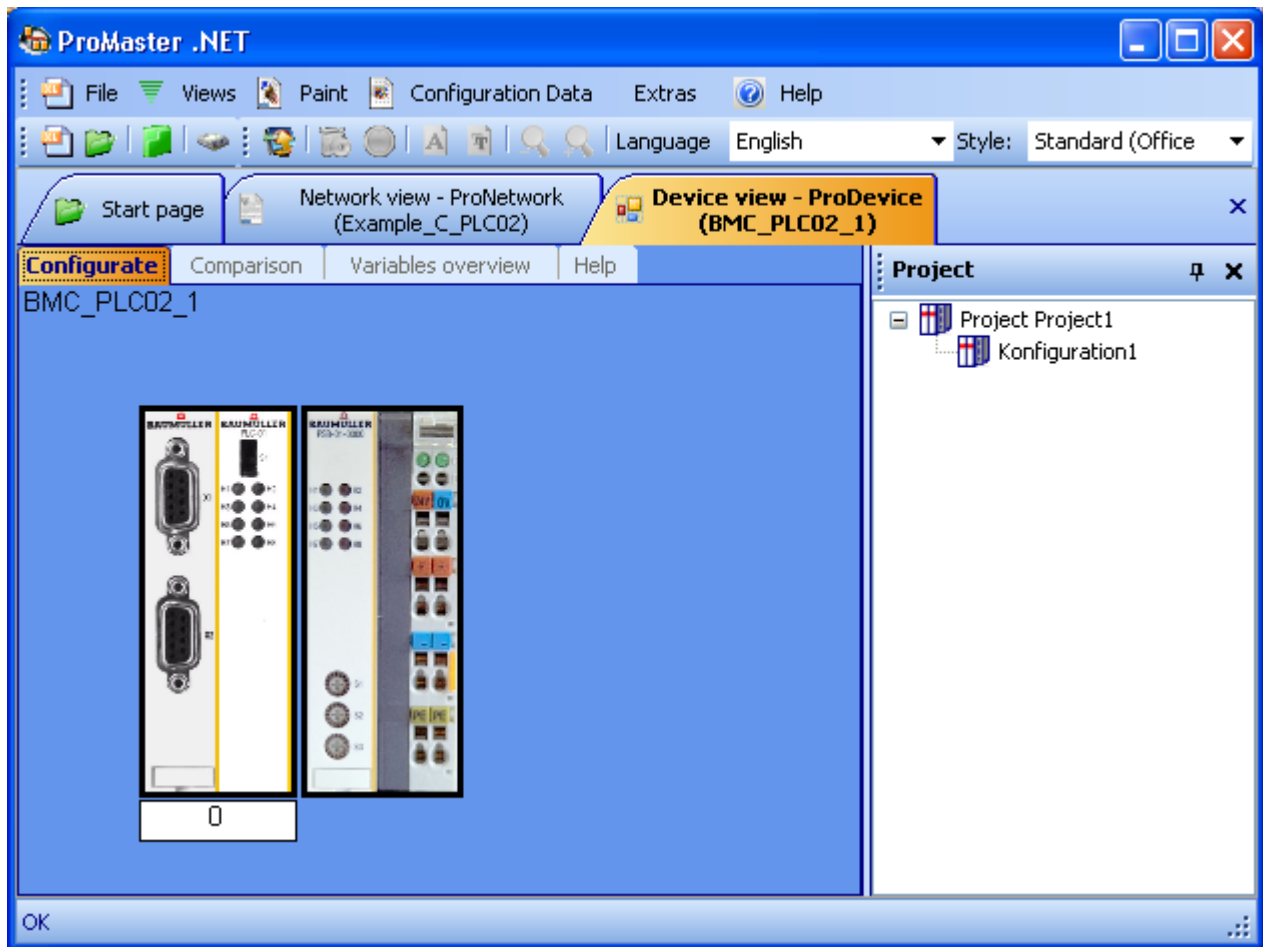


Figure 83: ProMaster - I/O configuration 1

Now open the Baumüller catalog using the "View\Catalog" menu item. Select the I/O modules tab.

For our example we use the following I/O modules:

- DO8000 (8 digital outputs, from the "digital I/O modules" group)
- DI8000 (8 digital inputs, from the "digital I/O modules" group)
- AO2010 (2 analog outputs, from the "analog I/O modules" group)
- AI2010 (2 analog inputs, from the "analog I/O modules" group)

Now drag and drop the I/O module DO8000 (to the right of the b maXX controller PLC power supply unit) into position. The end clamp is automatically positioned to the right of the I/O module. Now drag the I/O module DI8000 between "DO8000" and the end clamp, then AO2010 between DI8000 and the end clamp, and finally AI2010 between AO2010 and the end clamp.

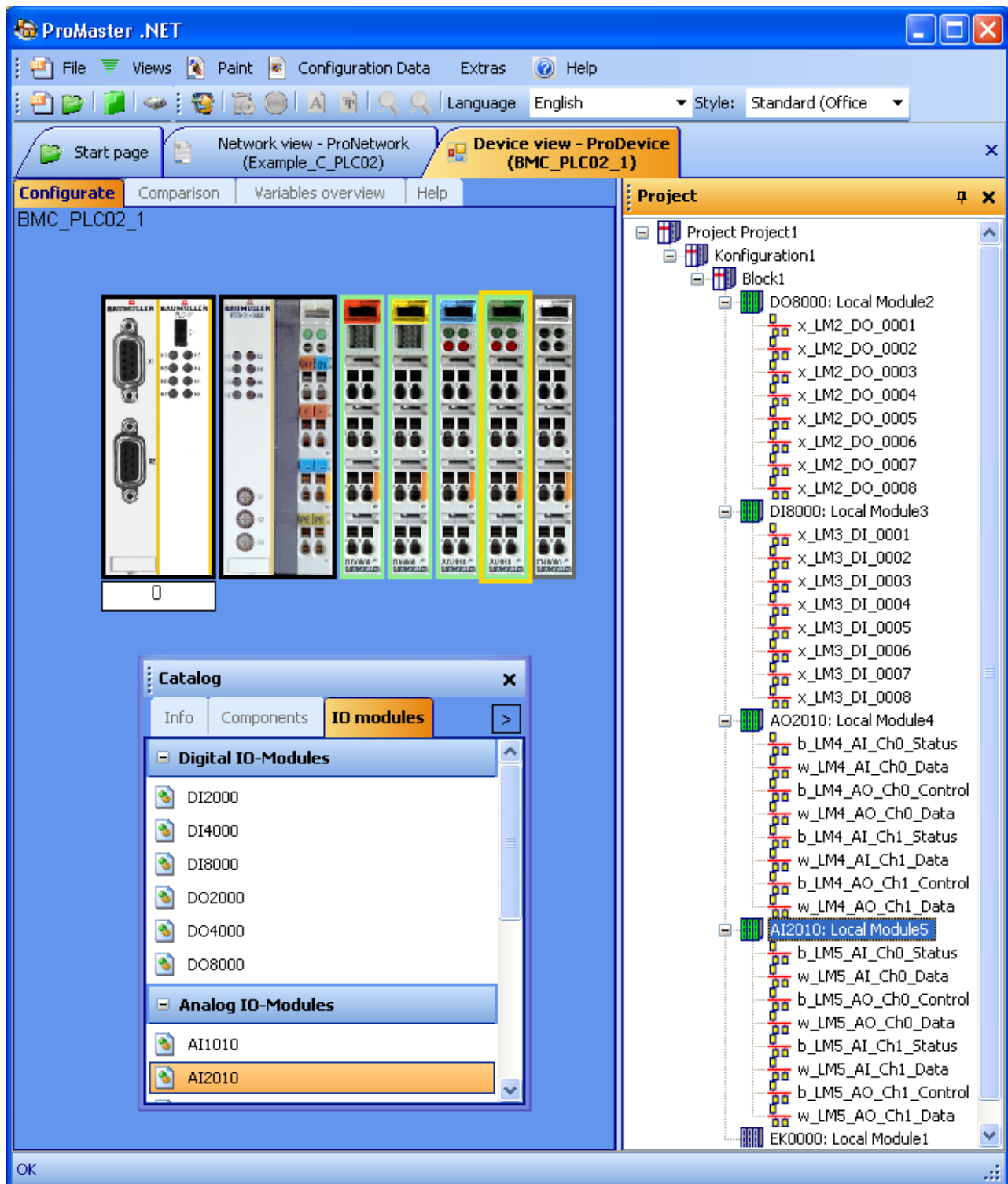


Figure 84: ProMaster - I/O configuration 2

Under "Configurations" you can view the I/O modules and their individual inputs and/or outputs. The IEC variable names of the individual inputs and outputs can either be changed under "Configurations" or in the "Device view"/"IEC variables" tab.

Close the catalog screen for a better view.

Now select the "Compare" tab in the "Device view" and accept the I/O configuration you have just put together in the IEC project for ProProg wt III. To do this use the arrows.

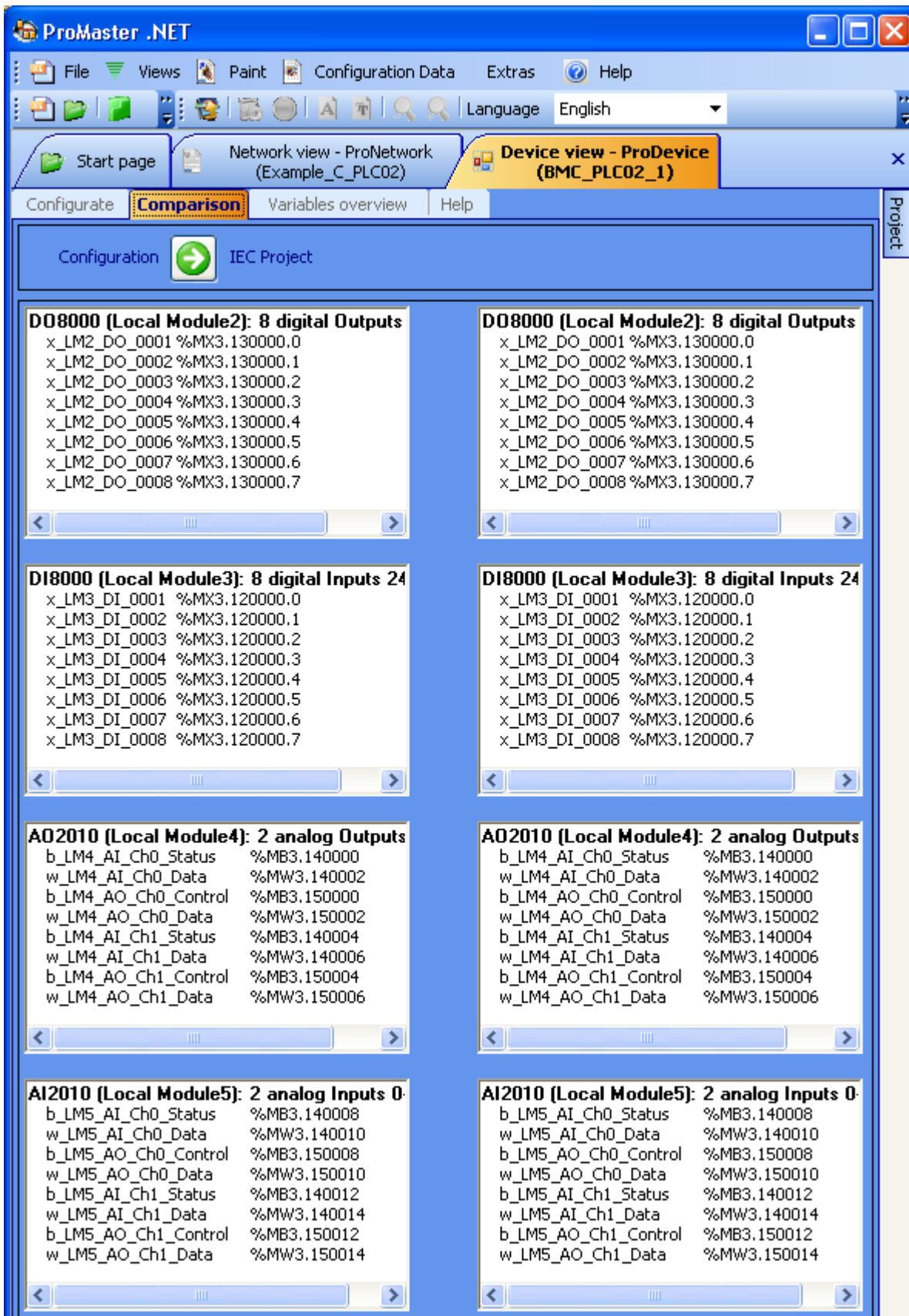


Figure 85: ProMaster – compare I/O configuration

Now return to the "PLC configuration" screen. If you closed it, open it by clicking on the "BMC_PLC_1" device and select "Configuration data\PLC (slot 0)\PLC - configuration" from the context menu.

The "I/O configuration" tab displays information such as IEC project variable names, addresses and comments.

The abbreviation "_LM_" in the default I/O variable names stands for "local module".

The IEC variables are written into the IEC project ("Global_Variables" worksheet) by clicking the "Update I/O variables" button found on the "I/O configuration" tab under "PLC Configuration".

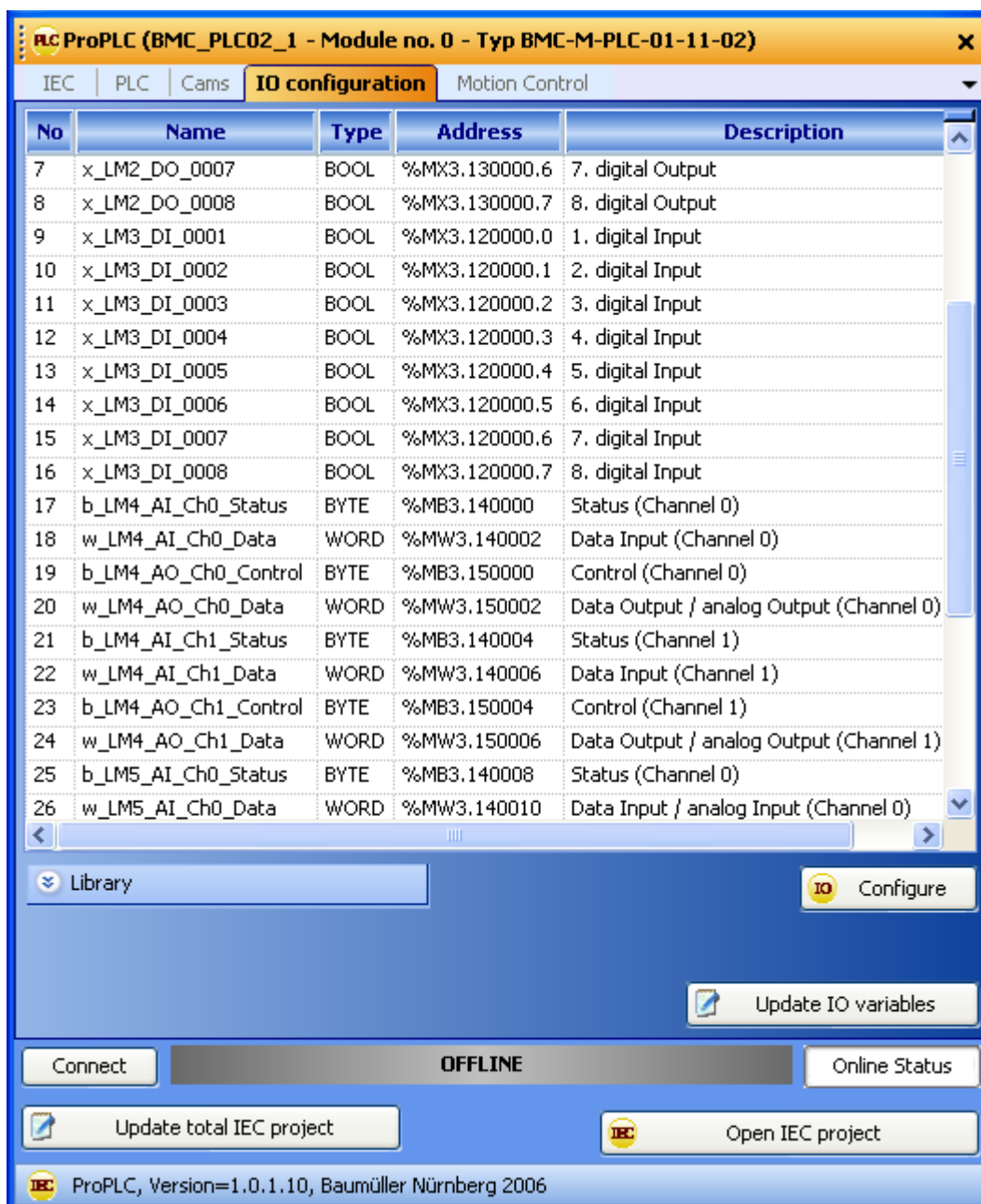


Figure 86: ProMaster - PLC configuration – I/O tab with DO, DI, AI and AO

Now click on the "Update entire IEC project" button in the "PLC configuration" screen. This creates the data for the b maXX controller PLC. This process can take a while, since it includes opening ProProg wt III and writing the configured IEC project variables into the global variables worksheet.

ProMaster has now written the I/O variables into the "Global_Variables" worksheet under the heading "IOVariables". These variables may now be used for programming the ProProg wt III project.

Name	Type	Usage	Description	Address
IOVariables				
x_LM2_DO_0001	BOOL	VAR_GLOBAL	1. digital Output	%MX3.130000.0
x_LM2_DO_0002	BOOL	VAR_GLOBAL	2. digital Output	%MX3.130000.1
x_LM2_DO_0003	BOOL	VAR_GLOBAL	3. digital Output	%MX3.130000.2
x_LM2_DO_0004	BOOL	VAR_GLOBAL	4. digital Output	%MX3.130000.3
x_LM2_DO_0005	BOOL	VAR_GLOBAL	5. digital Output	%MX3.130000.4
x_LM2_DO_0006	BOOL	VAR_GLOBAL	6. digital Output	%MX3.130000.5
x_LM2_DO_0007	BOOL	VAR_GLOBAL	7. digital Output	%MX3.130000.6
x_LM2_DO_0008	BOOL	VAR_GLOBAL	8. digital Output	%MX3.130000.7
x_LM3_DI_0001	BOOL	VAR_GLOBAL	1. digital Input	%MX3.120000.0
x_LM3_DI_0002	BOOL	VAR_GLOBAL	2. digital Input	%MX3.120000.1
x_LM3_DI_0003	BOOL	VAR_GLOBAL	3. digital Input	%MX3.120000.2
x_LM3_DI_0004	BOOL	VAR_GLOBAL	4. digital Input	%MX3.120000.3
x_LM3_DI_0005	BOOL	VAR_GLOBAL	5. digital Input	%MX3.120000.4
x_LM3_DI_0006	BOOL	VAR_GLOBAL	6. digital Input	%MX3.120000.5
x_LM3_DI_0007	BOOL	VAR_GLOBAL	7. digital Input	%MX3.120000.6
x_LM3_DI_0008	BOOL	VAR_GLOBAL	8. digital Input	%MX3.120000.7
b_LM4_AI_Ch0_Status	BYTE	VAR_GLOBAL	Status (Channel 0)	%MB3.140000
w_LM4_AI_Ch0_Data	WORD	VAR_GLOBAL	Data Input (Channel 0)	%MW3.140002
b_LM4_AO_Ch0_Control	BYTE	VAR_GLOBAL	Control (Channel 0)	%MB3.150000
w_LM4_AO_Ch0_Data	WORD	VAR_GLOBAL	Data Output / analog Output (Channel 0)	%MW3.150002
b_LM4_AI_Ch1_Status	BYTE	VAR_GLOBAL	Status (Channel 1)	%MB3.140004
w_LM4_AI_Ch1_Data	WORD	VAR_GLOBAL	Data Input (Channel 1)	%MW3.140006
b_LM4_AO_Ch1_Control	BYTE	VAR_GLOBAL	Control (Channel 1)	%MB3.150004
w_LM4_AO_Ch1_Data	WORD	VAR_GLOBAL	Data Output / analog Output (Channel 1)	%MW3.150006
b_LM5_AI_Ch0_Status	BYTE	VAR_GLOBAL	Status (Channel 0)	%MB3.140008
w_LM5_AI_Ch0_Data	WORD	VAR_GLOBAL	Data Input / analog Input (Channel 0)	%MW3.140010
b_LM5_AO_Ch0_Control	BYTE	VAR_GLOBAL	Control (Channel 0)	%MB3.150008
w_LM5_AO_Ch0_Data	WORD	VAR_GLOBAL	Data Output (Channel 0)	%MW3.150010
b_LM5_AI_Ch1_Status	BYTE	VAR_GLOBAL	Status (Channel 1)	%MB3.140012
w_LM5_AI_Ch1_Data	WORD	VAR_GLOBAL	Data Input / analog Input (Channel 1)	%MW3.140014
b_LM5_AO_Ch1_Control	BYTE	VAR_GLOBAL	Control (Channel 1)	%MB3.150012
w_LM5_AO_Ch1_Data	WORD	VAR_GLOBAL	Data Output (Channel 1)	%MW3.150014

Figure 87: Global variables worksheet showing the ProMaster data

Now compile the IEC project in ProProg wt III using the ProProg wt III menu item "Create new code\project".

Now download the IEC project onto the b maXX controller PLC (ProProg wt III menu item "Online\project control..." → "Send" → Boot project "Send").

Next reset the b maXX controller PLC and then switch the PLC to "RUN" status.
Using the IEC project you can now write outputs and read inputs in ProProg wt III.

6.2 ProModul module configurator

System components placed right hand of the b maXX controller PLC (e.g. I/O modules) are configured with the module configurator.

Communication variables were installed for the modules to read the inputs and to write or set the outputs of the modules.

Steps to be undertaken

Precondition

Assembly, installation and commissioning of

- b maXX controller PLC
- Power supply unit for b maXX controller PLC
- System components plugged in left hand of the b maXX controller PLC (e.g. CANopen-Master)
- System components plugged in right hand of the b maXX controller PLC (e.g. digital I/O modules and/or analog I/O modules, end module)

must be completed successfully.

If you want to activate system components and exchange data via the b maXX controller PLC, you must execute the following steps:

- Create a PROPROG wt II project for the b maXX controller PLC
- Configure the b maXX system with the module configurator
- Configure the system components/modules plugged in right hand of the b maXX controller PLC.

Result:

A list of variables for the initialization and/or commissioning of this system component.

The application part is programmed in PROPROG wt II project for the b maXX controller PLC afterwards. You will find instructions for the programming of the system components plugged in right hand of the b maXX controller PLC in the respective Application Manuals

Example

This chapter is accompanied by an example. We create a configuration (left to right) with the following system components / modules right hand of the b maXX controller PLC.

- Power supply unit for b maXX controller PLC
- 2 digital inputs +24V DC (DI2000)
- 2 digital outputs +24V DC (DO2000)
- 2 analog inputs 0-10V (AI2010)
- 2 analog outputs 0-10V (AO2010)
- End module (EK0000)

6.2.1 Create a PROPROG wt II project for the b maXX controller PLC

6.2.1.1 General

When you have not created an own project for your application in the section [>b maXX controller PLC project <](#) from page 33 onward, create the project with the „Template for b maXX controller PLC“. For this you need a PROPROG wt II version 3.1 Build 274 and the module configurator for PROPROG wt II version 1.0 (or above). You will find the version number on the cover of the product-CD of PROPROG wt II or in the PROPROG wt II software in menu ? \ Info.

Check whether the libraries

BM_TYPES_20bd06 (or above)

SYSTEM2_C_PLC01_20bd00 (or above)

and if necessary the libraries for the system components are available in your PROPROG wt II project.

If not, link these libraries into your project. These libraries contain important data types and function blocks for your system component.

6.2.1.2 Example: Creating the project "Modul_Example"

The sample project „Modul_Example" with the „Template for b maXX controller PLC“ was created. The libraries

BM_TYPES_20bd06

SYSTEM2_C_PLC01_20bd00

are linked.

System components/modules plugged in left hand of the b maXX controller PLC are not used in the example, so that no respective libraries must be linked.

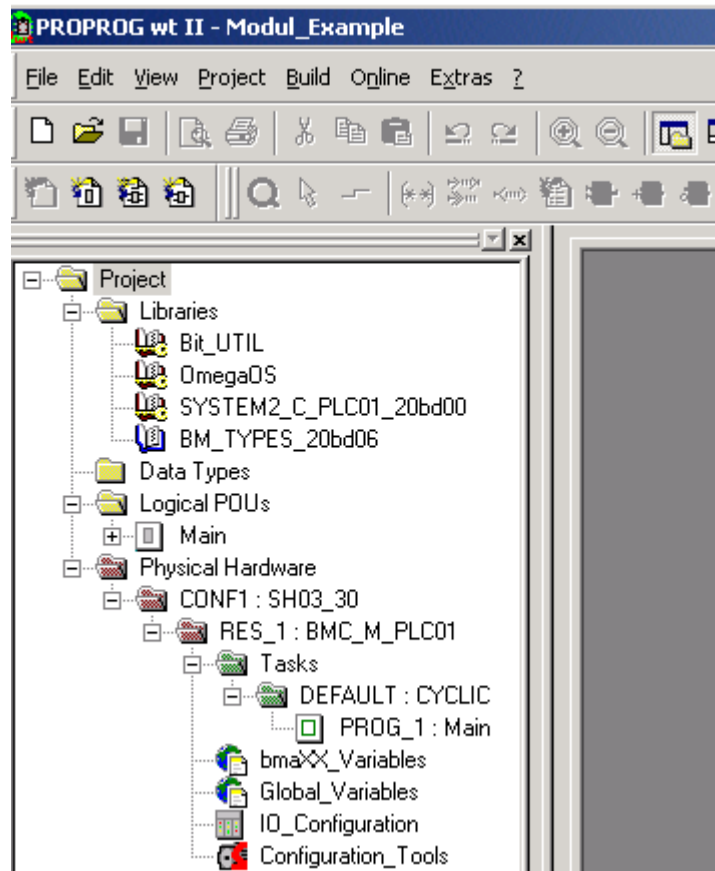


Figure 88: Example: Create the project Modul_Example

6.2.2 Configuration of the b maXX system with the module configurator

In the PROPROPAG wt II project, which you have created already (see [▶Create a PROPROPAG wt II project for the b maXX controller PLC ◀](#) from page 147 onward) you will find the button for the configurators (Configuration_Tools). With doubleclick on this button, the configuration window opens, among other things with the module configurator (ProPLC_Configuration).

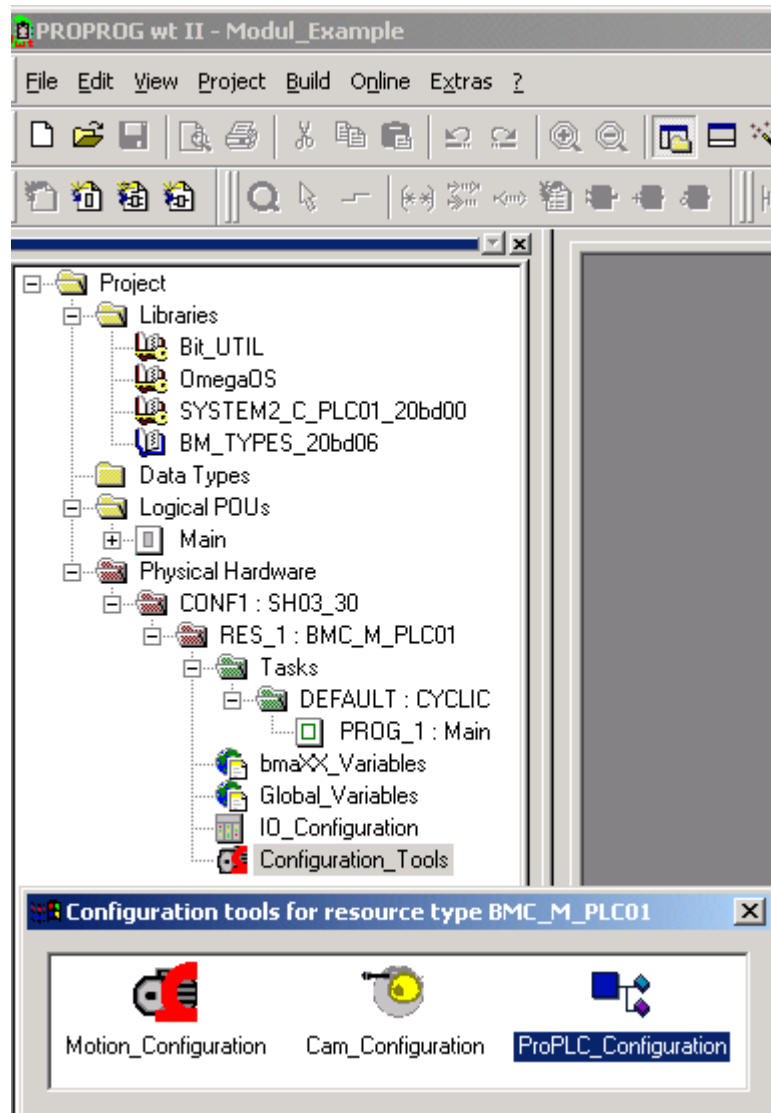


Figure 89: Open the module configurator

Before opening the module configurator, please compile your PROPROGRAM wt II project (menu: Code\Project generate new)

Click on the open button of the module configurator now.

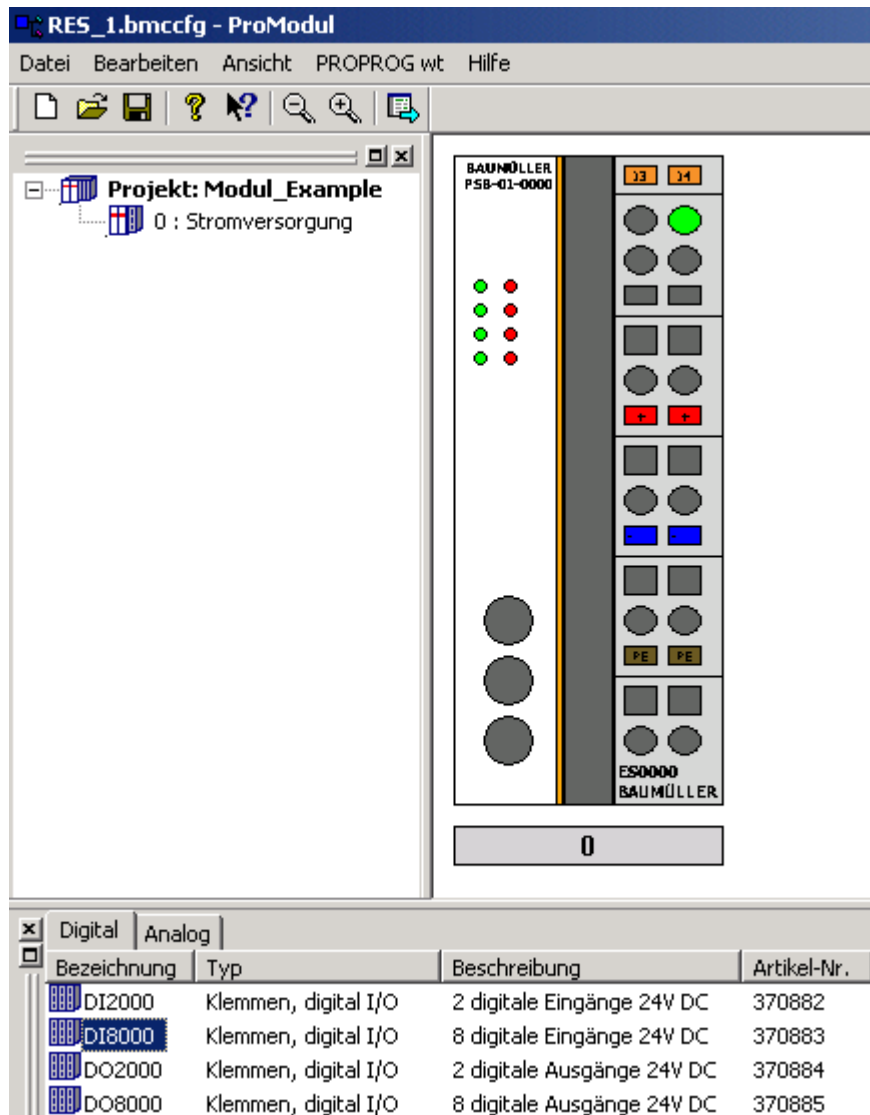


Figure 90: Module configurator

In the project tree of the module configurator the following configuration is set:

- Power supply unit for b maXX controller PLC (power supply)

This configuration displays the beginning configuration. For digital/analog inputs or outputs you must add modules (right hand).

In the following sections we explain how to change, add or delete single modules.

In order to insert a further system component you must do the following:

- Select the register „Digital“ in the window „Catalog“
- Select your module in the window „Catalog“, e.g. 2 digital inputs 24 V DC (DI2000)
- Drag this module per Drag and Drop right hand among the power supply unit.

So the new module is accepted and fitted in the graph.

Repeat these steps, if you want to use other modules.

You can use maximum 64 modules.

Now add more modules for the example:

- 2 digital outputs +24 V DC (DO2000),
- 2 analog inputs 0-10 V (AI2010),
- 2 analog outputs 0-10 V (AO2010).

The configuration of the system components/modules occurs via the according configurators, which you open with the menu item „Configure“ in the context menu. Instructions for this you get from the respective Operating Instructions of the system components/modules and from the Online help of the configurator.

The b maXX system of our example looks at follows:

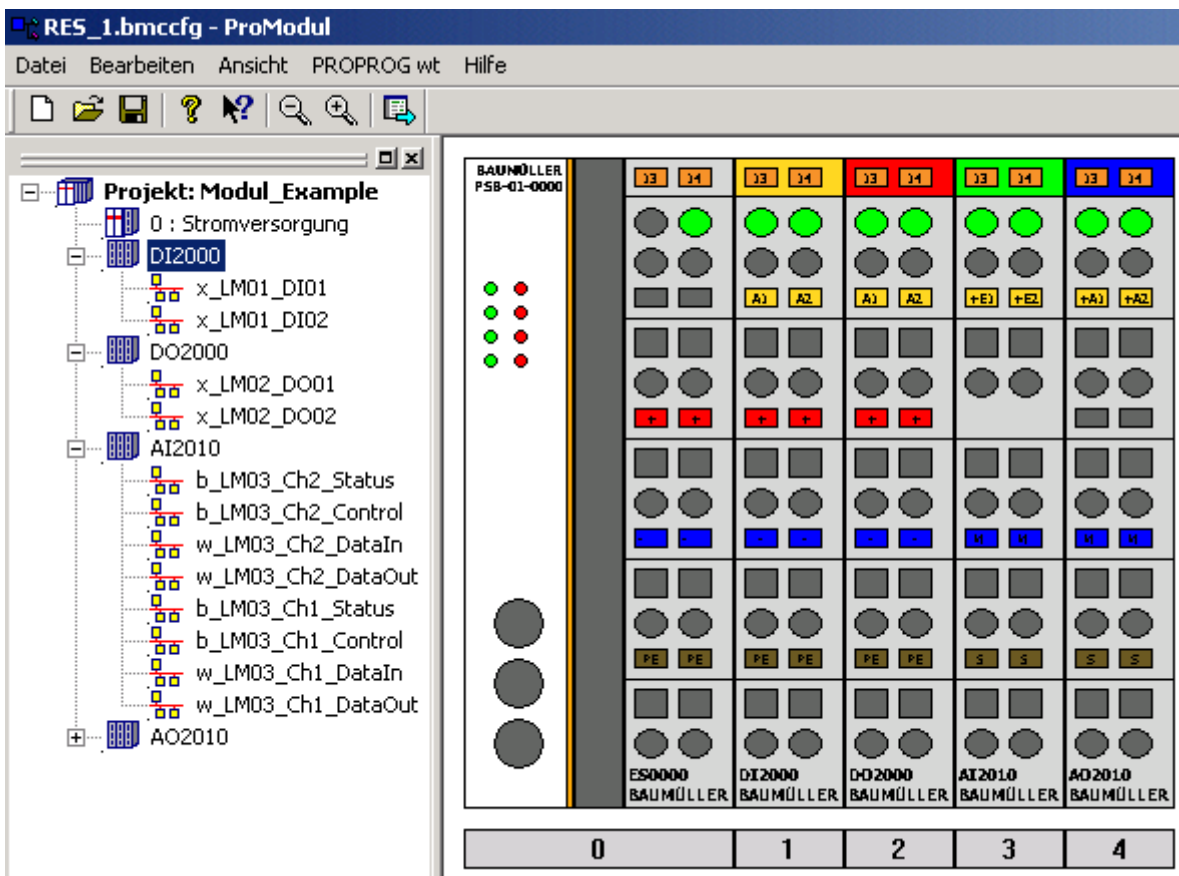


Figure 91: Example - b maXX system in the configurator (without end module)

The variables of the modules get default names. You can change these names by clicking the item „Configure“ in the context menu of each module.

In the example the digital input 1 of the module DI2000 will be changed to the variable name x_MyDigitalInput_0.

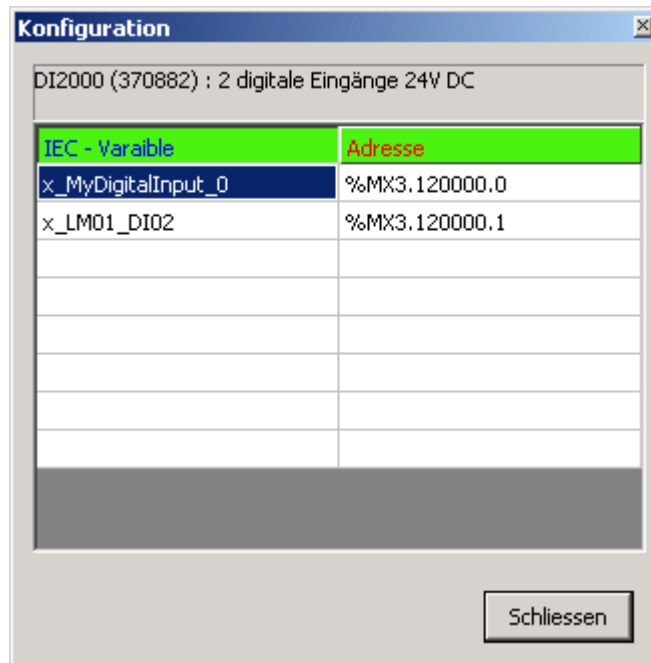


Figure 92: Changing a variable name in the module configurator

In a PROPROGRAM wt II project you use this variable name to evaluate the state of the digital input 1 of the module DI2000.

Close the window „Configuration“.

Accept the configurator data (in the global work sheet „bmaXX_Variable“ in the PROPROGRAM wt II project) with the menu item *PROPROGRAM wt II \ Actualize data* and confirm the safety query with „Yes“ and also the acknowledging message.

Close the module configurator with *File \ Close*. If you confirm the safety query with „Yes“, the configuration which has been set, is stored locally on the hard disk.


```

VAR_GLOBAL
· (*LM.1::DI2000 (370882) ·2·digitale·Eingänge·24V·DC·*)
x_MyDigitalInput_0·AT·%MX3.120000.0::BOOL;
x_LMO1_DI02·AT·%MX3.120000.1::BOOL;
· (*LM.2::DO2000 (370884) ·2·digitale·Ausgänge·24V·DC·*)
x_LMO2_D001·AT·%MX3.130000.0::BOOL;
x_LMO2_D002·AT·%MX3.130000.1::BOOL;
· (*LM.3::AI2010 (370886) ·2·analoge·Eingänge·0-10V·*)
b_LMO3_Ch2_Status·AT·%MB3.140000::BYTE;
b_LMO3_Ch2_Control·AT·%MB3.150000::BYTE;
w_LMO3_Ch2_DataIn·AT·%MW3.140002::WORD;
w_LMO3_Ch2_DataOut·AT·%MW3.150002::WORD;
b_LMO3_Ch1_Status·AT·%MB3.140004::BYTE;
b_LMO3_Ch1_Control·AT·%MB3.150004::BYTE;
w_LMO3_Ch1_DataIn·AT·%MW3.140006::WORD;
w_LMO3_Ch1_DataOut·AT·%MW3.150006::WORD;
· (*LM.4::AO2010 (370890) ·2·analoge·Ausgänge·0-10V·*)
b_LMO4_Ch2_Status·AT·%MB3.140008::BYTE;
b_LMO4_Ch2_Control·AT·%MB3.150008::BYTE;
w_LMO4_Ch2_DataIn·AT·%MW3.140010::WORD;
w_LMO4_Ch2_DataOut·AT·%MW3.150010::WORD;
b_LMO4_Ch1_Status·AT·%MB3.140012::BYTE;
b_LMO4_Ch1_Control·AT·%MB3.150012::BYTE;
w_LMO4_Ch1_DataIn·AT·%MW3.140014::WORD;
w_LMO4_Ch1_DataOut·AT·%MW3.150014::WORD;

END_VAR

```

Figure 93: Example - work sheet bmaXX_Variables

Now you have access to the inputs and outputs of the system components/modules in the PROPROG wt II project.

6.3 Configuration of the system components / modules

Maximum 64 system components and an end terminal may be plugged right hand of the b maXX controller PLC. Taking into account the following chapter these 64 components can consist of any digital and analog single modules.

When assembling the modules you should pay attention, that all digital modules be arranged in groups and all analog modules be arranged in groups for performance reasons.

In the following chapters and in the Online help you get instructions for using the system components / modules and their communication variables.

6.3.1 Digital inputs / digital outputs

A data area of 512 bytes is available in each case for the digital inputs and digital outputs. The maximum number of 64 modules must be considered at the module configuration. I.e. in a system

maximum 64 modules * 8 = 512 digital inputs or alternatively

maximum 64 modules * 8 = 512 digital outputs

can be processed.

You must not pay attention to a sequence at the location of the digital modules in consideration of the performance.

In the b maXX controller PLC every digital input and every digital output is represented by a variable with the data type BOOL. You can indicate the names of the variables in the module configurator, alternatively you use the automatically generated default names of the module configurator.

Digital Input - (DI)

A data area of 512 bytes is available for the digital inputs. The number of modules per system is limited to 64. I.e. you can process

maximum 64 modules * 8 = 512 digital inputs.

The digital inputs are registered via the system components „Digital Input“ (furthermore named DI-modules). You can select between different digital inputs, e.g. modules with 2 digital inputs or 8 digital inputs.

A list of the DI-modules which are available is to be found in the module configurator in register „Digital“.

In our example you can read out the first digital input of the module DI2000 via the variable (with data type BOOL) with the self assigned variable name

```
x_MyDigitalInput_0
```

and the second digital input of the module via the variable (with data type BOOL) with the default name

```
x_LM01_DI02.
```

Digital Output - (DO)

A data area of 512 bytes is available for the digital outputs. The number of modules per system is limited to 64. I.e. you can process

maximum 64 modules * 8 = 512 digital outputs.

The digital outputs are set via the system components „Digital Output“ (furthermore named DO-modules). You can select between different digital outputs, e.g. modules with 2 digital outputs or 8 digital outputs.

A list of the DO-modules which are available is to be found in the module configurator in register „Digital“.

In our example you can write the digital outputs of the system component DO2000 via the variable with data type BOOL

```
x_LM02_DO01 and x_LM02_DO02
```

(see [▶Figure 93](#) on page 153). You set the output to +24 V DC by writing on the variable the value TRUE.

6.3.2 Analog inputs / analog outputs

A data area of 512 bytes is available in each case for the analog inputs and analog outputs.

It must be pointed out, that every analog channel occupies all the same memory both in the output area and in the input area. In addition the maximum number of a total of 64 modules must be attended to the module configuration.

You must not pay attention to a sequence at the location of the analog modules in consideration of the performance.

In the b maXX controller PLC every analog input and every analog output is represented by two variables with the data type WORD. One of the variables contains the value (for the analog input or analog output), the other variable is reserved for special communication tasks. As aforementioned it must be pointed out, that the two words are represented both in the input area and in the output area

You can specify the names of the variables for the value (of the according analog input or analog output) in the module configurator, alternative you use the default names of the module configurator.

Example 1:

32 analog output modules per 4 analog outputs:

For each analog output a memory area of two words (= 4 bytes) is needed, i.e. in the data area 512 bytes are occupied for this configuration

$$32 * 4 \text{ analog outputs} * 4 \text{ bytes} = 512 \text{ bytes}$$

Additionally you could integrate up to 32 digital modules in the system.

Example 2:

32 analog input modules per 4 analog inputs:

$$32 * 4 \text{ analog inputs} * 4 \text{ bytes} = 512 \text{ bytes,}$$

Additionally you could integrate up to 32 digital modules in the system.

Example 3:

Combination from, e.g.:

16 analog output modules per 4 analog outputs and
16 analog input modules per 4 analog inputs

$$\begin{aligned} &16 * 4 \text{ analog outputs} * 4 \text{ bytes} \\ &+ 16 * 4 \text{ analog inputs} * 4 \text{ bytes} \\ &= 512 \text{ bytes.} \end{aligned}$$

Now you cannot insert another analog module.

Additionally you could integrate up to 32 digital modules in the system.

Example 4:

62 analog output modules per 2 analog outputs and 1 analog output module per 4 analog outputs:

For each analog output a memory area of two words (= 4 bytes) is needed, i.e. in the data area 512 bytes are occupied for this configuration

$$[(62 * 2) + (1 * 4)] \text{ analog outputs} * 4 \text{ bytes} = 512 \text{ bytes}$$

Additionally you could integrate another digital module in the system.

Example 5:

62 analog input modules per 2 analog inputs and 1 analog input module per 4 analog inputs:

$$[(62 * 2) + (1 * 4)] \text{ analog inputs} * 4 \text{ byte} = 512 \text{ bytes}$$

Additionally you could integrate another digital modules in the system.

Example 6:

Combination from, e.g.:

16 analog output modules per 4 analog outputs and
30 analog input modules per 2 analog inputs and
1 analog input module per 4 analog inputs.

$$\begin{aligned} & 16 \text{ analog outputs} * 4 \text{ bytes} \\ & + [(30 * 2) + (1 * 4)] \text{ analog inputs} * 4 \text{ bytes} \\ & = 512 \text{ bytes} \end{aligned}$$

Now you cannot insert another analog module. Additionally you could integrate up to 17 digital modules in the system.

Analog Input - (AI)

The analog inputs are registered via the system components „Analog Input“ (in the following named AI-modules). You can select between different analog inputs, e.g. modules with 2 analog inputs or 4 analog inputs.

A list of the AI-modules which are available is to be found in the module configurator in register „Analog“.

In our example you can read out the analog input of the system component AI2010 via the variables

```
w_LM03_Ch1_DataIn and w_LM03_Ch2_DataIn
```

(see [▶Figure 93](#) on page 153). At an input voltage of +2,5 V DC you get the value WORD#16#1FFF in the respective variable.

Analog Output - (AO)

The analog outputs are set via the system components „Analog Output“ (in the following named AO-modules). You can select between different analog outputs, e.g. modules with 2 analog outputs or 4 analog outputs.

A list of the AO-modules which are available is to be found in the module configurator in register „Analog“.

In our example you can write the analog outputs of the system component AO2010 via the variables

```
w_LM04_Ch1_DataOut   and   w_LM04_Ch2_DataOut
```

(see [▶Figure 93](#) on page 153). You obtain an output voltage of +5 V DC at the output by writing the value WORD#16#3FFF on the respective variable.

6.3.3 Alarm output

If a hardware output is necessary for the duly function of the PLC, this can be realized with a digital output module. The output must be set to 1 (TRUE) for a duly operation condition. If an error occurs the Output must be set to 0 (FALSE). If the output terminal gets no signal over 100 ms, the terminal switches its outputs to 0 (FALSE).



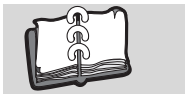
APPENDIX A ABBREVIATIONS

CAN	Controller Area Network
CBPB	Controller Based Parallel Bus
CPU	Central Processing Unit
EMC	Electromagnetic compatibility
DPRAM	Dual Ported RAM
DRAM	Dynamic RAM
EN	European standard
ESD	Electrostatic sensitive device
EXT, ext	External
I/O	Input/output
I/O-bus	Bus for the input and output modules
LED	Light Emitting Diode
NOVRAM	Non-volatile RAM
OPC	OLE for Process Control (OLE: Object Linking and Embedding)
PLC	Process loop control
PROPROG wt II	Programming-tool of the b maXX PLC
RAM	Random access memory
RISC	Reduced Instruction Set Computers
SDRAM	Synchronized Dynamic RAM
SW	Software
USS®	Trademark of Siemens, universal serial interface



Revision index

Revision level	State	Modifications
5.04019.02	07.07.2006	Upgrading for BMC-M-PLC-02
5.04019.03	15.02.2007	New: Chapter 4.3 and 6.3.3



Revision index



Index

A		F	
Activation	23	FB TIME_MEASURE_END	58
Address		FB TIME_MEASURE_START	58
read out power supply	74	Firmware library	54
Alarm output	157	Flag address	
ASCII protocol	102	absolute	51
B		Function block	56
b maXX controller PLC		INTR_SET	58
data types	65	TER_INIT	99
b maXX PLC		Function block TIMER_A_INIT for BMC-M-PLC-01	79
Firmware	56	Function block TIMER_A_INIT for BMC-M-PLC-02	85
Basic functionality	69	G	
Baumüller	7	Global variable worksheets	36
BM4-O-PLC-01	5	I	
BMC-M-PLC-01		I/O bus	29, 128
Display elements	16	I/O configuration	36
Function test	23	INTR_SET	58
Initialisation	16	L	
LED display	16	LED function block	62, 64
Operation	18	Level	
S1 switch/pushbutton	18	Interrupt	53, 59
BMC-M-PLC-02	5	N	
Communication	20	New project	33
Display elements	19, 65	P	
Function test	25	Personnel	13
Initialisation	20	qualified	13
LED display	19, 65	PLC errors	126
Operation	21	Process data communication	75
S2 switch/pushbutton	22	Programming languages	31
Board function	57	Project	
C		New, open	33
CAM_PLC_20bd00	70	ProMaster	131
CAM_PLC01_30bd00	70	Property	
CAM_PLC02_30bd00	70	Event Task	52
CBPB	29	PROPROG wt II	31
CBPB hardware signal	75	PROPROG wt II library	72
Communication source	36	Q	
Communication via RS232	37	Qualified Personnel	13
Control Dialog for Resources	41	R	
Create project		Real-time response	52
with ProMaster	134	Resource	34, 36
D		Settings	37
Data Area	36, 49	Resource BMC_M_PLC01	34
Data communication	30	Resource BMC_M_PLC02	34
Directory path for libraries	56	Resources	
Documentation worksheets	36	E	
E		Event Task	52



Index

control dialog	41
Responsibility and liability	13
Retain data	48
Retain flag	50
RS485 interface	98
RUN	17, 18, 21, 22, 25

S

Send boot project	25
Send project to target system	43
Specialist	13
Standard library	69
STOP	24
Stop	17, 18, 21
Switch/pushbutton	19, 22
Switch/pushbutton S1	24
System data	48
System flag	50
SYSTEM1_C_PLC01_20bd00	58, 69
SYSTEM1_C_PLC01_30bd00	58, 70
SYSTEM1_C_PLC02_30bd02	58, 70
SYSTEM2_C_PLC01_20bd00	69
SYSTEM2_C_PLC01_30bd00	70
SYSTEM2_C_PLC02_30bd02	70

T

Technology component	
Cam disk	70
Winder	71
TER_INIT	99
Terms	
Definition	5
TIME_MEASURE_END	58
TIME_MEASURE_START	58
TIMER_A_INIT for BMC-M-PLC-01	79
TIMER_A_INIT for BMC-M-PLC-02	85
Transmission time	30
Two-wire/four-wire	99

U

UNIVERSAL_20bd01	69
UNIVERSAL_30bd00	70
User flag	50
User library	54
Inserting in a project	71

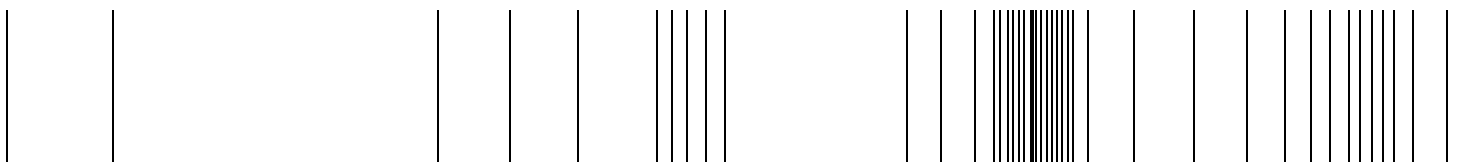
W

Warranty and Liability	14
Watchdog	48
WINDER_PLC_20bd00	71
WINDER_PLC01_30bd01	71
WINDER_PLC02_30bd01	71

Z

ZWT format file	72
-----------------	----

be in motion



Baumüller Nürnberg GmbH Ostendstraße 80-90 90482 Nuremberg Tel: +49(0)911-5432-0 Fax: +49(0)911-5432-130 www.baumueller.de

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.