Application Manual

Language **English**
Translation
Document-No. 5.08004.02
Part No. 422420
Status 26.08.2009

be in motion be in motion

BAUMÜLLER

b maXX® Systems

b maXX safe PLC

BMC-M-SAF-02

E	5.08004.02

Read the Operating Instructions before beginning

Baumüller Nürnberg GmbH

Ostendstr. 80 - 90
90482 Nürnberg
Germany

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

E-Mail: mail@baumueller.de
Internet: www.baumueller.de

Table of Contents

1

DOCUMENT HISTORY

Revision level	State	Modifications
5.08004.01	31.03.2009	Initial document
5.08004.02	26.08.2009	Editorially checked contents

2 PREFACE

2.1 Information on the Application Handbook

This Application Handbook for the b maXX safe PLC is an important component of your b maXX system; for that reason, read this documentation thoroughly in its entirety, not only in interest of your own safety.

Furthermore, the local accident prevention legislation and general safety regulations applying to the device's field of application must also be complied with.

Read the Operation Manual completely, in particular the chapter on safety instructions, before beginning any work on the device. The Operation Manual is a component of the product and must be kept accessible to personnel in the immediate vicinity of the device at all times.

2.2 Legend

Warning notices

Warning notices are marked by symbols in this Application Handbook. The notices are introduced by signal words which express the extent of the hazard.

Comply with the notices under all circumstances and act with caution in order to avoid accidents, personal injury and property damage.

DANGER!

....notifies of an imminent dangerous situation which will lead to death or serious injuries if not avoided.

WARNING!

....notifies of a potentially dangerous situation which can lead to death or serious injuries if not avoided.

CAUTION!

....notifies of a potentially dangerous situation which can lead to minor or slight injuries if not avoided.

CAUTION!

....notifies of a potentially dangerous situation which can lead to property damage if not avoided.

Recommendations

NOTICE!

....draws attention to useful tips and recommendations as well as information for efficient and trouble-free operation.

2.3 Limitation of liability

All statements and instructions in this Application Handbook have been compiled in compliance with the applicable standards and legislation while taking the current level of technology and our long-term experience and findings into account.

The manufacturer assumes no liability for damages resulting from:

- failure to observe the Operation Manual
- application for purposes other than those intended
- use by untrained personnel

The actual scope of materials delivered can vary from the explanations and illustrations described here in the event of custom designs, the use of additional ordering options or due to the most recent changes in technology.

The user assumes the responsibility of conducting maintenance and commissioning in accordance with the safety regulations of the applicable standards and all other relevant national or regional legislation relating to conductor dimensioning and protection, grounding, circuit breakers, overvoltage protection, etc.

The person who conducted the assembly or installation shall be accountable for damages occurring during assembly or connection.

2.4 Preliminary information

CAUTION!

The following shall apply if the document you are reading is designated as preliminary information:
This version pertains to preliminary technical information which the user of the described devices and functions should receive ahead of time, in order to be able to adjust to potential changes and/or functional expansions.
This information is to be seen as preliminary, since it has not yet been subjected to the Baumüller internal review process. In particular, this information is still subject to changes, meaning that this preliminary information cannot be construed as legally binding. Baumüller assumes no liability for damages resulting from this potentially incorrect or incomplete version.
Should you detect or suspect content-related and/or serious formal errors in this preliminary information, please contact the contact person assigned to you and inform us of your findings and comments, so that they can be taken into account and potentially incorporated during the transition from the preliminary information to the final (reviewed by Baumüller) information. The obligations specified in the following section under “Obligations” do not apply to preliminary information.

2.5 Copyright

Treat the Application Handbook as confidential. It is intended exclusively for those working with the device. It is not permissible to transfer the Application Handbook to third parties without the written approval of the manufacturer.

2.6 Further applicable documents from other manufacturers

Components from other manufacturers are built into the device. Hazard evaluations for these bought-in parts have been conducted by the applicable manufacturers. The conformity of the designs with the applicable European and national legislation has been declared by the respective component manufacturers.

2.7 Replacement parts

WARNING!

Improper or defective replacement parts can lead to damage, malfunctions or total failure as well as jeopardize safety.

Therefore:

- Only use original replacement parts from the manufacturer

Procure replacement parts from authorized dealers or directly at the manufacturer.

2.8 Disposal

If no return or disposal agreement has been made, dismantled components can be taken for recycling after proper disassembly.

2.9 Warranty conditions

The warranty conditions can be found as a separate document in the sales documentation.

The operation of the devices described here in accordance with the specified methods/ procedures/requirements is permissible. Everything else, even the operation of devices in installation positions not depicted here, for instance, is not permissible and must be clarified with the factor on a case-by-case basis. The warranty will be rendered null and void if the devices are operated differently than described here.

2.10 Customer service

Our customer service is available for technical support.

Information on the competent contact person can be found at any time via telephone, fax, E-mail or over the internet.

2.11 Terms used

The terms “PLC” or “BMC-M-SAF-02” are also used for the product “**b maXX safe PLC**.”

The terms “module” or “power supply” are also used in this documentation for the Baumüller product “power supply for b maXX controller/safe PLC.”

The term “b maXX System” is also used for the product consisting of “power supply for b maXX controller/safe PLC,” “b maXX safe PLC” and further system components.

A list of the abbreviations used can be found in ▹Appendix A - Abbreviations◃ on page 165.

2.12 Certification

The programmable safety controller b maXX safe PLC from Baumüller Nürnberg GmbH has been developed in accordance with the standards specified in ▹Approvals, directives and standards ◃ on page 19 and certified by TÜV Rheinland.

Approval no. 968/EZ 358.00/09

Test report: 968/EZ 358.00/09

2.12 Certification

3

FURTHER APPLICABLE DOCUMENTS

Name	Contents
b maXX safe PLC Operating Instructions 5.07020	Description, installation and commissioning
b maXX controller PLC Application Manual 5.04019	Programming of standard PLC

4

DEFINITION OF TERMS

Abbreviation / Term	Definition
ProSafety	Safe programming system for safety control and safe programming
Safety PLC	Safety functional device of the safety control device
Standard PLC	Non-safety related standard functional devices of the safety control device
ProMaster	Engineering framework for Baumüller automation products (see chapter ▷Programming, configuration and parameterization of safe functions ◁ from page 39 onward)
ProPLC	Configurator in ProMaster for the standard PLC
ProSafetyPLC	Configurator in ProMaster for the safety PLC
OmegaSAFE	Running time system of the safety PLC also safety PLC or W1 and OmegaSAFE channel 2.
SafeOS	Safe IEC running time system component of OmegaSAFE
HNF	Low-level firmware
FSoE	Functional safety over EtherCAT
Copymanager	Handler in grey channel for data exchange between safe stacks and various communication media.
Safe stack	Firmware module enabling access to local and decentral I/Os via a particular safety protocol, such as FSoE
CRC	Cyclic redundancy check
Remote-IO	Decentral IO via bus coupler and field bus connected with the control

Abbreviation / Term	Definition
Safety ID	One-to-one identification number for safety devices which communicate with the safety control via safety protocol. It is used in safe IEC application programming and in safe parametrization. Depending on the ID allocation procedure used, the number must be one-to-one system-wide or in relation to the safety control or system.
Device ID	One-to-one identification number for safety devices which communicate with the safety control via safety protocol. It is set on each safety device. The device ID must be one-to-one system-wide. Depending on the ID allocation procedure used, the device ID is equal to the safety ID or is generated by a parameterizable algorithm from the safety ID and the safe station number of the safety control (see ▹Address/ID allocation and safe station numbers ◃ from page 156 onward).
Safe station number	A number controlled via safe procedure in the safety control which can affect both the functionality of the safe application program and allow identical application programs to be run on multiple safety controls in identical machine modules (stations), yet achieving communication relations which are clearly safe (see ▹Address/ID allocation and safe station numbers ◃ from page 156 onward).
FB	Functional module in the IEC programming system

5

USE OF THIS HANDBOOK

This safety handbook contains information on the intended use of the Baumüller b maXX safe PLC.

Knowledge of the regulations and proper technical implementation of the safety instructions in this handbook by qualified personnel are prerequisites for the safe installation, commissioning and safety during the operation and maintenance of the Baumüller b maXX safe PLC. Unqualified interference with the devices during shutdown or use of the safety functions or failure to comply with the instructions of this handbook can lead to serious personal injury, property damage or environmental harm, for which Baumüller assumes no liability.

Baumüller control devices are developed, manufactured and tested in compliance with the applicable safety standards. They may only be used under the specified environmental conditions and only in connection with approved external devices.

5.1 Documentation

This handbook describes the implementation of safety technology applications with Baumüller b maXX safe PLC and the safe programming environment ProSafety.

In addition to this handbook, the operating manual and the programming handbook are to be used with special care before installing the control system.

Additional system documentation

Document number	Contents
5.07020	Operating manual
	All documents which do not describe the safe functionality and applications

6

APPROVALS, DIRECTIVES AND STANDARDS

The b maXX safe PLC fulfills the functional safety requirements of the standards specified in the following:

Safety engineering standards and directives	**Area of application**	**Approvals**
IEC 61508, Parts 1-7	Functional safety of safety-related electric, electronic and programmable electronic systems	up to SIL 3
DIN EN ISO 13849-1	Safety-related components of control units	up to performance level E
EN 954-1	Safety-related components of control units	up to category 4
IEC 62061 Appendix E	Functional safety of security-related electric, electronic and programmable electronic systems Fulfilment of increased stability requirements in accordance with Appendix E	
Additional standards	**Area of application**	
EN 61131-2	General device requirements and tests for control systems	
EN 50178	Equipping of high voltage equipment with electronic utilities Use of ventilation and leakage paths	
EN 60204	Electrical machine equipment	

7

SYSTEM ARCHITECTURE

7.1 System overview

The most important system components are listed here again in brief.

A safety control device can consist of the following components which can be installed strung together on a top-hat rail:

- Safety control PLC module in 2-channel version which allows standard applications to be run on the first channel.
- Power supply (supplied in 24 V) which is attached to the right of the safety control. It supplies both the safety control PLC module as well as the communication modules with low-voltage power (such as 5 V). Local IO modules can be attached in variable number to the power supply on the right. In doing so, safe IO modules can be attached to standard IO modules in mixed order.
- Communication modules, which can be attached on the left of the safety control. The following types are currently available: ethernet, EtherCAT and CANopen. The component groups can be arranged as bus masters or bus slaves, depending on type.

Devices which the safety control can communicate with include: bus couplers for decentral IOs and drives, as well as additional devices in the standard area with ethernet or RS485 interface (PCs, displays, sensors).

A PC is used for the configuration, programming and parameterization of the safety control. The ProMaster engineering system is the main program used. It is described in the chapter ▹Programming, configuration and parameterization of safe functions ◃ from page 39 onward.

The control system has following communications paths as gray channels on which safe IO data can be exchanged by means of safe communications protocols:

- IO-bus communication to local safety IO modules on the safety control, which are activated using the power supply
- EtherCAT communication to decentral safety IO modules on the EtherCAT coupler, which are activated by means of the EtherCAT master
- Ethernet communication to safe drives, which are activated by means of the ethernet interface of the EtherCAT master (the application-specific Pecom protocol, which uses UDP telegrams, is implemented over the ethernet)

- EtherCAT communication to safe drives, which are activated by means of the EtherCAT master.

The safe stack(s) are run on the Safety PLC and transmit the safe IO data via safe protocol.

7.2 Field bus connection

The following connections are currently available: Ethernet, EtherCAT and CANopen. The component groups can be arranged as bus masters or bus slaves depending on type.

7.3 Internal communication interfaces

Communication to the system components (modules) connected to the left of the b maXX safe PLC and the appendent power supply is carried out by means of CBPB (Controller Based Parallel Bus).

Communication to the system components connected to the right of the b maXX safe PLC is carried out via I/O bus.

7.3.1 CBPB (Controller Based Parallel Bus)

The CBPB is an internal communication interface for data exchange between the b maXX safe PLC and the system components connected to the left of the b maXX safe PLC.

Accesses to the system components (which are connected to the left of the power supply of the b maXX safe PLC) are represented to the user as accesses to the DPRAM of the respective system components.

Parameterizing the interface is not necessary.

You can obtain instructions on programming and/or parameterizing the respective system components in the application handbooks of the respective system components.

7.3.2 I/O bus

The I/O bus is an internal communication interface for data exchange between the b maXX safe PLC and the system components connected to the right of the power supply of the b maXX safe PLC.

Accesses to the system components (which are connected to the right of the power supply of the b maXX safe PLC) are represented to the user as accesses to the inputs and outputs of the respective system components.

Parameterizing the interface is not necessary.

8

Application of safety technology standards

Two B standards harmonized under the Machinery Directive are used in the implementation of safety-oriented applications. The main contents which are relevant in connection with control systems are summarized in the following chapters. The chapter ▷General safety technology applications ◁ from page 31 onward describes typical safety technology applications with different safety classifications.

8.1 Security classification according to standards

The Application Handbook serves in the implementation of security-related applications in conformity with the standards. As an introduction, the basic principles of standards which are useful in creating safety applications will be summarized in this chapter. **The information presented here is no replacement for training courses in acquiring the required skills in working with “functional security”. It merely serves as support in creating applications.**

The applicable safety technology standards in mechanical and systems engineering are the B standards harmonized in the Machinery Directive:

- DIN EN ISO 13849
- IEC 62061

If C standards apply to a particular machine or system, they must also be taken into account, since they have higher priority than the B standards.

DIN EN ISO 13849-1 is the successor standard to EN 954-1. During the transition period up to November 29, 2009, EN 954-1 will remain valid parallel to DIN EN ISO 13849-1. Whereas EN 954-1 ranks the safety control of a machine in five categories according to the anticipated risk, additional quantitative methods and criteria for assessing functional safety were incorporated into DIN EN ISO 13849-1. Characteristic, pre-calculated structures are available to the user for risk assessment.

While DIN EN ISO 13849-1 is designed to be a practical standard for safety-related control parts in mechanical engineering applications, EN 62061 provides a framework for the functional safety of safety-related electrical control systems and their subsystems on machines. It is a sector application standard for machines and is derived from the basis standard IEC 61508.

8.2 Risk graph according to DIN EN ISO 13849-1

The risk graph presents a method of classifying an existing risk in the risk assessment framework.

The risk graph from DIN EN ISO 13849-1 is depicted in Figure 1. In assessing the risk, it is assumed that no safety components have been installed. The risk is assessed according to the severity of the injury (S), the frequency or duration of the exposure (F) in the danger area and the possibility of avoiding the hazard or limiting the damage (P). The required performance level for each safety application results from the classification of the specified parameters. It presents a measurement for risk reduction.

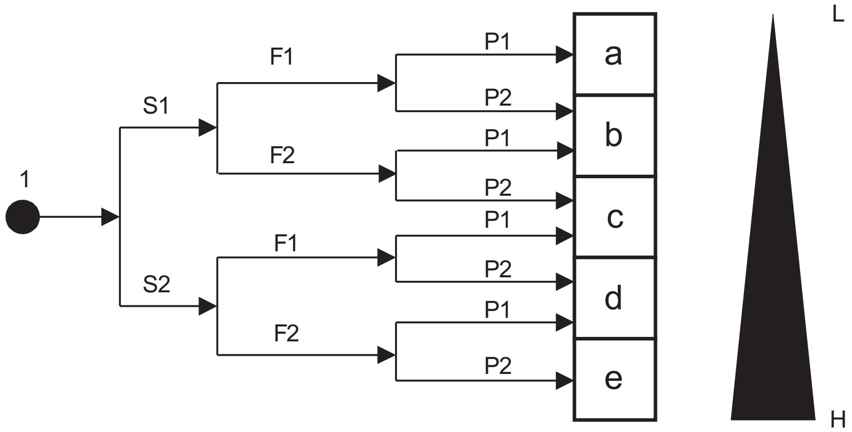


Figure 1: Risk graph in accordance with DIN EN ISO 13849-1

Legend

1	Starting point for the risk reduction contribution assessment
L	Low risk reduction contribution
H	High risk reduction contribution
PLr	Required performance level

Risk parameters:

S	Severity of the injury
S1	Minor (injury which is normally reversible)
S2	Severe (injury which is normally irreversible including death)
F	Frequency or duration of exposure to danger
F1	Rare to infrequent and/or time of exposure to danger is short

F2	Frequent to prolonged and/or time of exposure to danger is long
P	Possibility of avoiding the hazard or limiting the damage
P1	Possible under certain conditions
P2	Hardly possible

Example:

The risk analysis for a machine yields the following parameters:

- severe injury = S2
- frequent exposure in the danger area = F2
- avoiding the hazard is hardly possible = P2

The parameters determined result in a performance level of E as the requirement for the safety system.

8.3 Classification according to performance level according to DIN EN ISO 13849-1

A risk assessment is carried out for the control system which has to achieve the established risk reduction. It must provide proof that the system can attain the performance level determined in the scope of the risk analysis.

Category, $MTTF_d$ (Mean Time To Failure $_{dangerous}$) and DC_{avg} (Diagnostic Coverage $_{average}$) are consulted for the performance level assessment according to DIN EN ISO 13849-1.

In order to attain a desired performance level, it is first necessary to determine a category. This category represents the basis parameter and is instrumental in determining the architecture of the control system. It establishes the fundamental behavior of the control system in relation to the effects of failures. The possible categories (B through 4) with the allocated architectures are specified in DIN EN ISO 13849-1.

The $MTTF_d$ values are primarily manufacturer's specifications. They are specified for control systems as well as components such as switches and sensors. If no $MTTF_d$ values are specified for components of the safety system, the characteristic values can be determined from tables (see DIN EN ISO 13849-1).

Three value ranges have been established for grading the $MTTF_d$ values.

low	3 years $\leq MTTF_d <$ 10 years
middle	10 years $\leq MTTF_d <$ 30 years
high	30 years $\leq MTTF_d \leq$ 100 years

The middle degree of diagnostic coverage is a measurement for the effectiveness of the diagnosis. It specifies the ratio of the dangerous failure detected to the total number of dangerous failures.

The value for the middle degree of diagnostic coverage is specified in four levels.

none	$DC_{avg} < 60\%$
low	$60\% \leq DC_{avg} < 90\%$
middle	$90\% \leq DC_{avg} < 99\%$

high	99% $\leq DC_{avg}$

DIN EN ISO 13849-1 makes it possible to conduct a simplified procedure to estimate the performance level by means of applying the parameters described.

In order to attain a performance level of D, for example, an architecture of at least category 2 will be necessary (see Figure 2). Furthermore, the system must display a $MTTF_d$ value = high and a middle degree of diagnostic coverage = low. If a middle degree of diagnostic coverage = middle is reached for the same architecture (category 2), a $MTTF_d$ = middle will be adequate for a performance level of D.

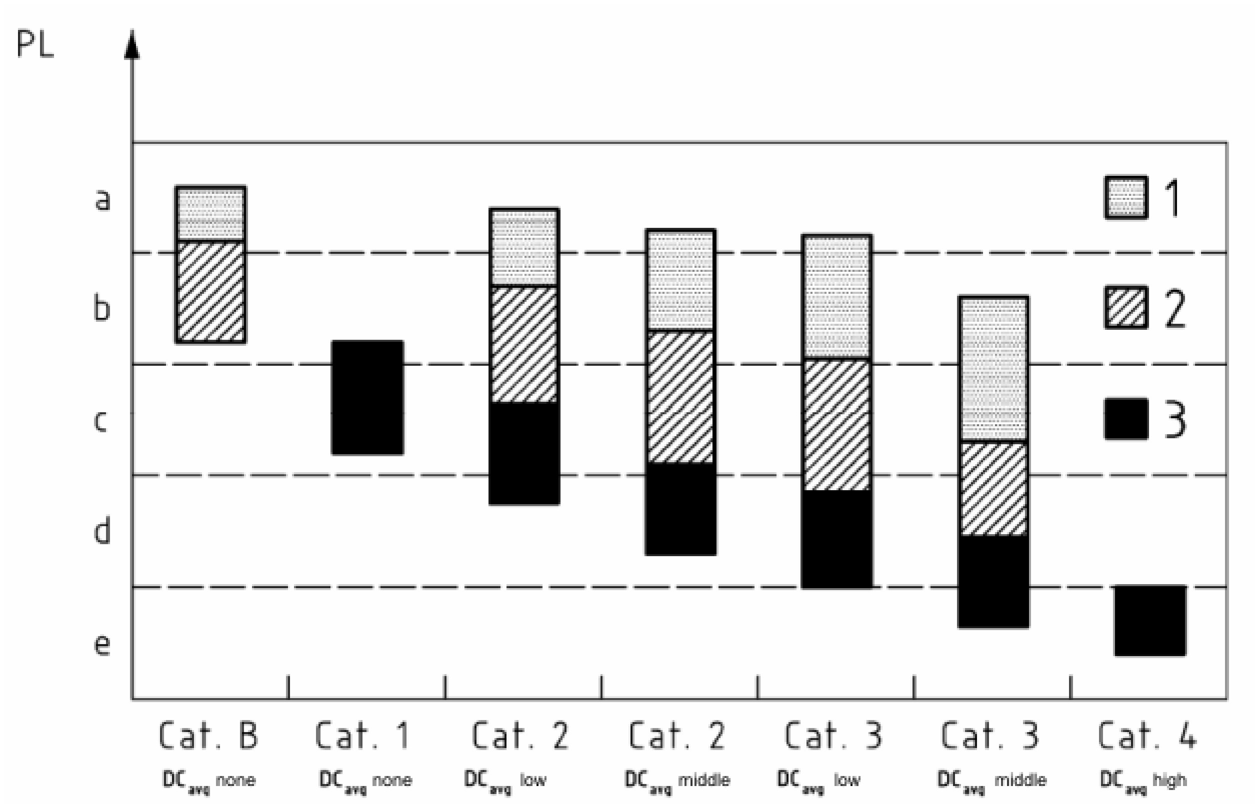


Figure 2: Performance level classification according to DIN EN ISO 13849-1

Legend

PL	Performance Level
1	$MTTF_d$ of each channel = low
2	$MTTF_d$ of each channel = middle
3	$MTTF_d$ of each channel = high

8.4 Classification according to SIL corresponding to EN 62061

EN 62061, which is derived from the basis standard EN 61508, is a sector application standard. The safety requirements are classified according to the safety integrity level (SIL).

As with DIN EN ISO 13849-1, the determination of the safety integrity level is carried out by means of risk analysis. The establishment of the required safety integrity level is carried out according to an SIL allocation table. The exact course of action in establishing the safety integrity level can be found in EN 62061.

The assessment of whether a safety system meets the requirements of the established SIL is carried on the basis of hardware failure tolerance (HFT), the fraction of safe failures (SFF = Safe Failure Fraction) and the probability of dangerous failures per hour (PFH = Probability of Failure per Hour).

A hardware failure tolerance value of N indicates that N+1 failure could lead to a loss of the safety function. An HFT = 1 stands for a two-channel system. This means that the loss of the safety function can only take place upon the second failure and not before it.

The safe failure fraction is produced by the following calculation:.

$$SFF = (\Sigma\lambda_S + \Sigma\lambda_{DD}) / (\Sigma\lambda_S + \Sigma\lambda_D)$$

$\Sigma\lambda_S$ sum of safe failures (i.e. relay contacts are not closing)

$\Sigma\lambda_{DD}$ sum of dangerous failures detected by the diagnosis

$\Sigma\lambda_D$ sum of all dangerous failures (i.e. relay contacts are not opening)

Table 1: Classification according to SIL

Fraction of safe failures (SFF)	**Hardware failure tolerance**		
	0	**1**	**2**
< 60%	not allowed	SIL 1	SIL 2
60% through 90%	SIL 1	SIL 2	SIL 3
90% through 99%	SIL 2	SIL 3	SIL 3
≥ 99%	SIL 3	SIL 3	SIL 3

The PFH value specifies the probability of a dangerous failure per hour. Consult the allocation according to Table 2 to assess whether a desired safety integrity level has been reached with the intended control system in the risk assessment.

Table 2: Allocation of SIL and PFH

SIL	PFH (Probability of Failure per Hour)
3	10^{-8} to < 10^{-7}
2	10^{-7} to < 10^{-6}
1	10^{-6} to< 10^{-5}

If the arrangement depicted in ▷Figure 3◁, consisting of a sensor, a control and power contact is selected for a safety application in accordance with SIL 2, for example, compliance with SIL 2 can be proven by forming the total sum of the PFH values of all subsystems.

PFH total = $2 \times 10^{-7} + 1 \times 10^{-7} + 2 \times 10^{-7} = 5 \times 10^{-7} < 10^{-6}$

→ Requirement met.

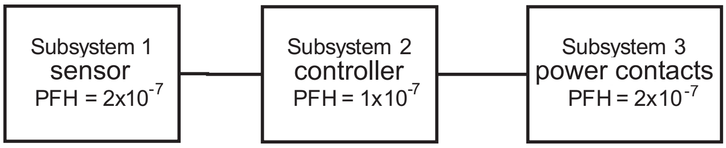


Figure 3: Interlinking of subsystems

Safety integrity level and performance level can be brought into relation with one another. However, it involves a rough allocation which is only usable for estimates and does not apply to all boundary conditions. For those reasons, fundamentally separate assessments according to DIN EN ISO 13849-1 and EN 62061 are required. However, both standards are organized in such a way that the results calculated from one standard can be used again in the respective other standard.

Table 3: Relation between performance level and safety integrity level

PL	SIL / (high requirement rate)
a	No correspondence
b	1
c	1
d	2
e	3

8.5 Definition of cycle time, reaction time and maximum reaction time

In accordance with IEC/TR 62513, the use of communication systems in security-oriented applications requires knowledge of the connections between cycle time, reaction time and maximum reaction time. The technical report explains the connections between the specified safety-related parameters, which are to be taken into account under all circumstances.

▷Figure 4◁ shows the typical arrangement of safety-oriented components within a communication network.

Figure 4: Chain for determining maximum system reaction time

8.5.1 Cycle time

The system components depicted in ▷Figure 4◁ each have a cycle time. The cycle time describes the internal processing time of the respective components. The communication cycle time is composed of the times $T_{Bus\ 1}$ and $T_{Bus\ 2}$.

8.5.2 System reaction time

The system reaction time results from interlinking the individual cycle times.

$$T_{Reaction} = T_{Sensor} + T_{Filter} + T_{Cycle\_Input} + T_{Bus\ 1} + T_{Cycle\_PLC} + T_{Bus\ 2} + T_{Cycle\_Output} + T_{Actor}$$

8.5.3 Maximum system reaction time

Since the processing cycles of the individual components are not synchronized with one another, the maximum (worst case) reaction time for each system must be determined.

It can be defined by using the following formula.

$$T_{Reaction} = T_{Sensor} + T_{Filter} + \mathbf{2x}T_{Cycle\_Input} + \mathbf{2x}T_{Bus\ 1} + \mathbf{2x}T_{Cycle\_PLC} + \mathbf{2x}T_{Bus\ 2} + \mathbf{2x}T_{Cycle\_Output} + T_{Actor}$$

The equation specified depends on the system. Depending on the bus system used, the number of cycles run through can be reduced. If, for example, the communication cycles are synchronized with the PLC processing cycle, the factor 2 of the PLC cycle can be omitted.

8.5 Definition of cycle time, reaction time and maximum reaction time

The determination of the system reaction time of the Baumüller b maXX safe PLC under use of the scheduled system components is explained with corresponding examples in the chapter ▹Calculation of the maximum reaction time for a safety function ◃ from page 161 onward.

!	**CAUTION!** The maximum reaction time of the system must be determined under all circumstances.

!	**CAUTION!** The system must always be set up so that the maximum system reaction time lies within the defined reaction time of the safety functions.

9

GENERAL SAFETY TECHNOLOGY APPLICATIONS

The sample applications presented in this chapter represent typical applications in safety technology.

9.1 Safety-oriented switching functions

Safety-oriented switching functions form the basis of safety technology. They are used in various applications. The most frequently used switching functions are described in the following chapters.

9.1.1 Safe shutdown functions

Shutdown functions are used in drive technology, for example.

The simplest form of shutdown for a drive is carried out by means of a gate. This separates the drive from the energy supply upon the safety function’s request. The shutdown depicted on ▷Figure 5◁ on page 32 is suitable for the safe stop functions STO and SS.

Two positively driven power contacts are used to shut off the energy. The gates are equipped with positively driven auxiliary contacts which are designed as break contacts. The drive can only be restarted with the button once the contacts of both gates drop when the drive is shut down. An individual failure will thus be recognized. Due to the dual-channeled nature, the positively driven contacts and the diagnostic function via feedback loop, applications up to SIL 3 and PL E can be executed with this arrangement.

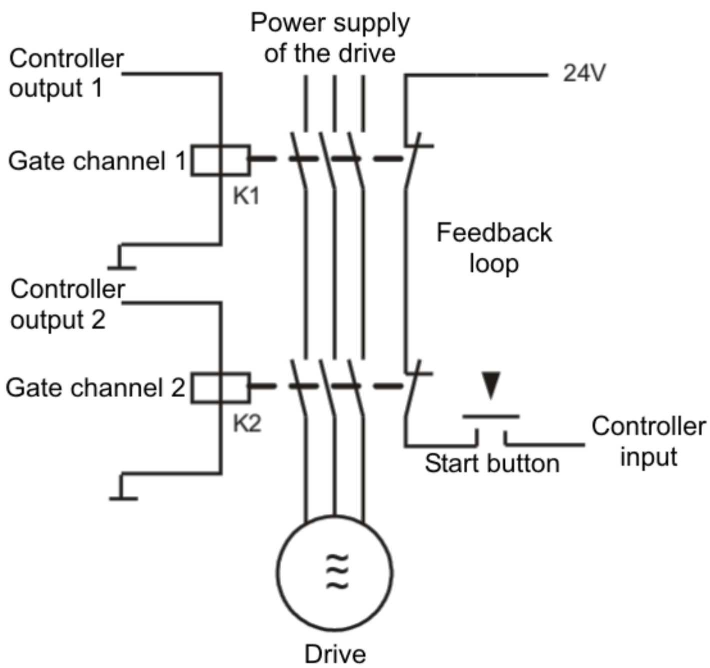


Figure 5: Dual-channel shutdown with contact test

Modern drive solutions frequently dispense with the safe shutdown of the energy supply to stop the drive. One of the basic principles in shutting down a drive lies in safely shutting off the impulse pattern necessary to generate a rotating field. The safe impulse cut-off principle is mostly implemented locally in the power element of the drive module (see ▹Monitoring of guard doors◃ page 35).

9.1.2 Safe shutdown by means of emergency stop devices

The design guidelines for emergency stop devices are established in the standard DIN EN ISO 13850. The emergency stop function is one of the most important safety functions.

WARNING!

Emergency stop functions have a very low requirement rate.

Therefore:

- Have emergency stop functions tested regularly by the user

Emergency stop functions come in either single-channel or dual-channel design, depending on the risk assessment. All risk reduction classifications (SIL, PL) specified in the example applications are estimates which are to be verified individually. The single-channel activation unit of an emergency stop devices is depicted in ▹Figure 6. Applications up to SIL 1 or performance level b through c can be reached with a single-channel emergency stop device.

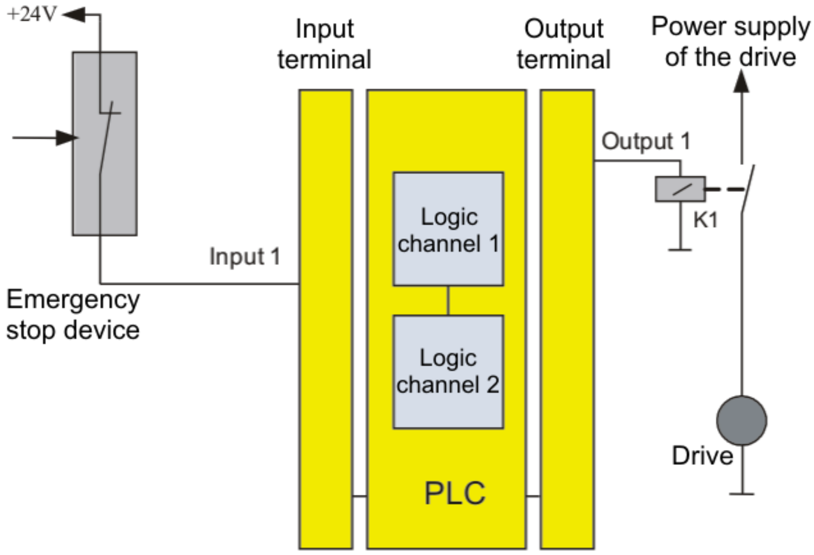


Figure 6: Single-channel emergency stop device

Safety shutdowns up to PL E and SIL 3 can be conducted with a dual-channel device.

A typical arrangement for the use of dual-channel emergency stop devices is depicted in ▹Figure 7◃ page 34.

The safe input terminals depicted in ▹Figure 7◃ is a terminal with a low degree of diagnostic coverage which has no short circuit monitoring for the wiring of the emergency stop button. Since the system is a dual-channel design, it has a hardware failure tolerance of HFT = 1. The power relay is turned on by means of a common safe output. Safe outputs distinguish themselves in that they have an internal dual-channel shutdown. The outputs can thus also be considered to be dual-channel components. Applications up to SIL and a performance level of D can be attained with the arrangement shown if the interlinking of all system components comply with the value range defined for SIL 2 for PFH/PFD or the $MTTF_d$ value defined for performance level D.

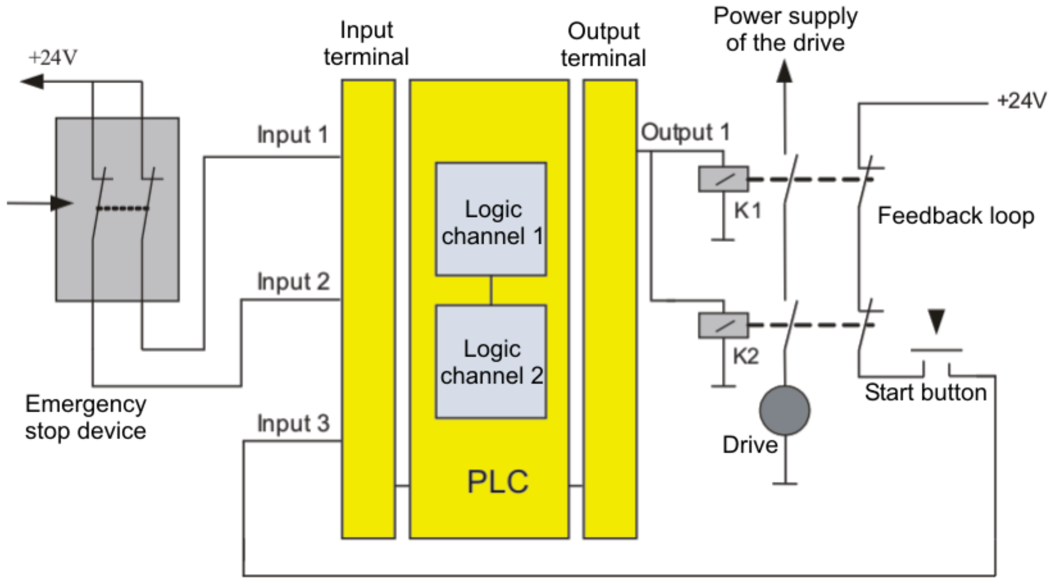


Figure 7: Typical arrangement of a dual-channel emergency stop button without short circuit monitoring

In order to cover the increased safety requirements for SIL 3 and PL E, the diagnostic properties of the system must be improved. For this purpose, a dynamic sampling of the input signals is conducted. The dynamic sampling is carried out by means of test pulses which cause a cyclic interruption of the emergency stop device supply. The test pulses are delayed as shown. Short circuits in the emergency stop button wiring can be detected at the desired system potentials by monitoring the timing signals in the safe input terminals.

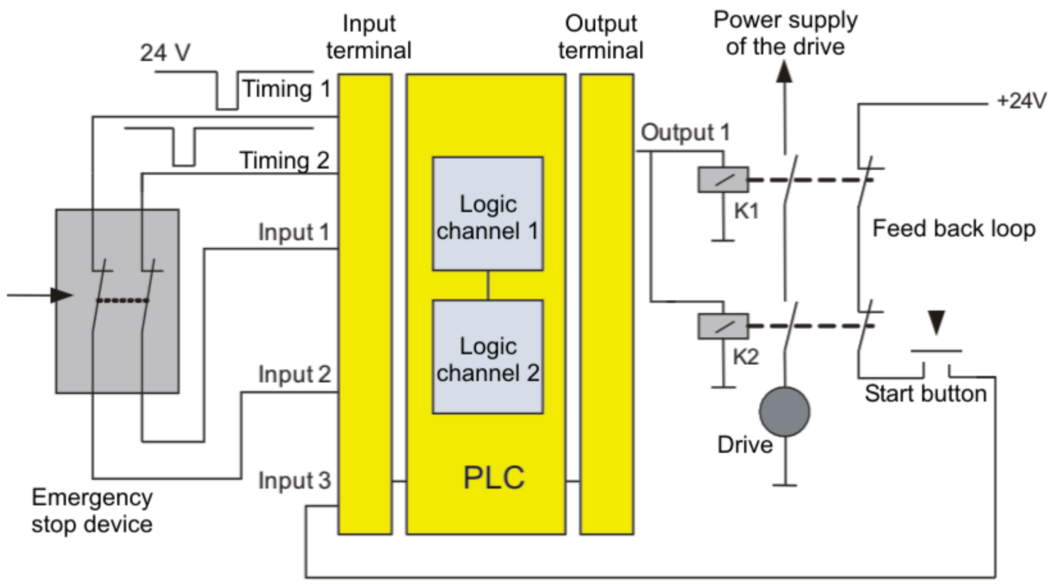


Figure 8: Dual-channel emergency stop button with timing monitoring

9.1.3 Safe shutdown by means of sensors with OSSD outputs

An application with a safety light grid is depicted in ▹Figure 9◃ on page 35. LIke other electronic safety sensors, the safety light grid is equipped with so-called OSSD outputs (OSSD = Output Signal Switching Device). These are semiconductor outputs which are momentarily turned off. The resulting output impulse is evaluated by the sensor's internal diagnostic functions. If the impulse on one channel remains out, there is either a failure in the sensor or the output line has shorted out to another line. In the event of a failure, the safe status is engaged by shutting down the intact output. In this type of operation of the safe input terminals, the timing sensitivity of the input is shut off by the commensurate parameterization. The OSSD test pulses must be inhibited by the input switching of the safe terminals.

WARNING!

Make absolutely certain that the correct parameterization is present for each input. Timing signals which have accidentally been deactivated impair diagnosis with electromechanical switches. This failure is not detected by the system.

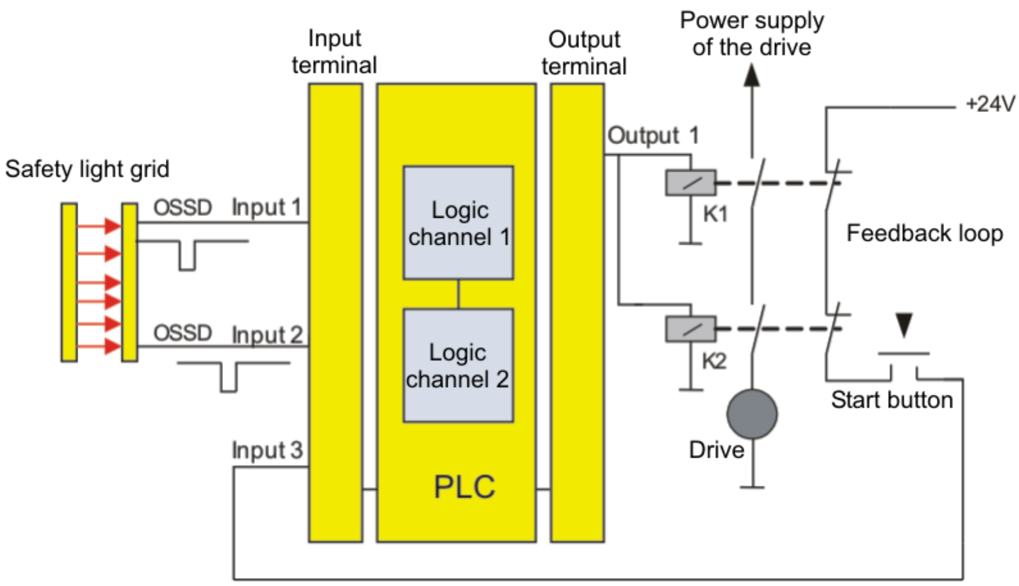


Figure 9: Dual-channel sensor with OSSD output

9.1.4 Guard door monitoring

Guard doors are monitored by position switches or sensors. Guard doors are additionally equipped with a guard locking, depending on the application. Combined units equipped with position recording and guard locking in a single device are frequently used.

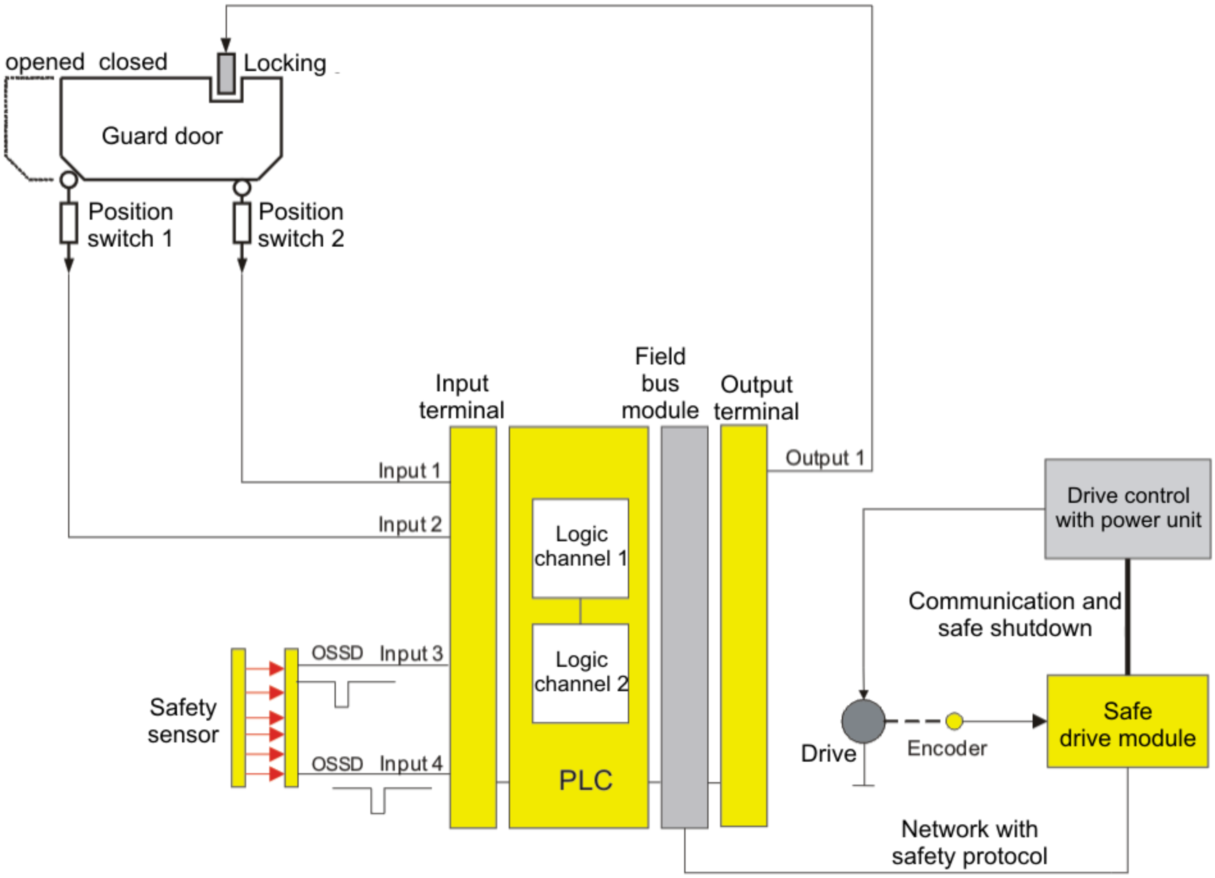


Figure 10: Guard door monitoring with guard locking

A typical guard door monitoring is depicted in ▹Figure10◃. As opposed to the previous examples, the motion and safe shutdown of the drive is monitored by a decentral safe drive module. The communication connection is carried out via network (such as EtherCAT), enabling the transmission of safety-oriented data (such as safety protocol FSoE).

If a safe drive function such as STO (Safety Torque Off), SS1 (Safe Stop 1) or SLS (Safely Limited Speed) is required by a safety sensor, as shown in ▹Figure 10◃ on page 36, this command is transmitted over the network to the safe drive module. The safety module conducts the required function and reports the status attained, such as the drive's standstill, to the safe control. The guard locking is disengaged by the safety control and the guard door can be opened. Safe locking is active in powerless condition. That means that it must be supplied with power in order to open it. As long as the guard door is open, the drive cannot start running. The control will only send a release command to the decentral safe drive module once the safety control (PLC) has detected the correct function and position of the position switch and the locking is active.

A change of status must be carried out for both position switches when opening and closing the guard door.

Applications up to PL E and SIL 3 can be achieved by means of the arrangement shown.

Enabling switching

During special types of machine operation such as set-up operation or maintenance work, it is frequently necessary to override the effect of the protective devices. During such work, the operator is exposed to increased danger. The operator holds an enabling switch in his hand in order to be able to shut down the machine as quickly as possible in the event of danger. This enabling switch typically has three mechanical positions. THe safety contacts to release the machine are only active in the middle position. If the operator lets go of the enabling switch or pushes it all the way down (panic position), the machine is shut down. The wiring or an enabling switch corresponds to the wiring of an emergency stop device.

The properties of enabling switches are defined in EN 60947-5-8.

9.1.5 Two-hand operation

The processing operation may never be running if a hand is in a danger zone when work pieces or semi-finished products are inserted manually in a production process. Presses are typical machines with which a person first places a metal part precisely and then begins the processing operation. Two-handed control which only triggers the start of the course of motion when one actuates a start button separately with both hands can be set for safeguarding purposes.

The control console with two-hand control is positioned far enough from the danger zone of the press that the press cannot be reached directly after the press run has been started.

Both of the two-hand buttons for the two-handed control are arranged so that they cannot be operated with a single hand under any circumstances. The function can only be started when both buttons are pushed simultaneously within 0.5 seconds. These two properties prevent the potential manipulation of the safety function.

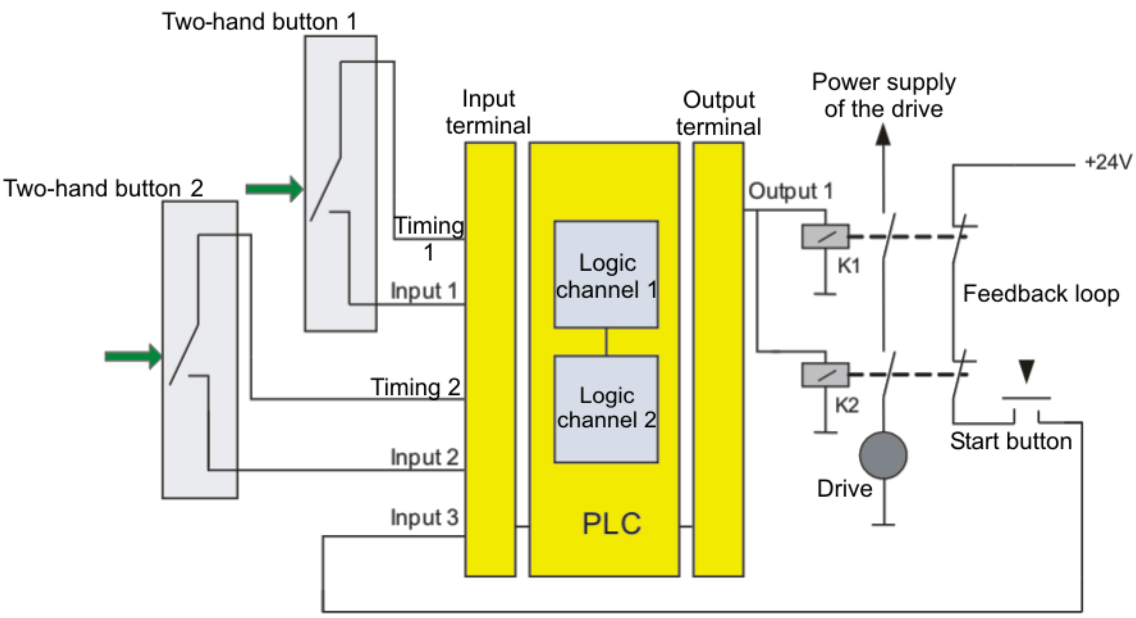


Figure 11: Two-hand operation

10

PROGRAMMING, CONFIGURATION AND PARAMETERIZATION OF SAFE FUNCTIONS

10.1 Programming with the safe programming environment ProSafety

The safe programming system serves for the development of safe applications for safety controls.

It is based on the IEC 61131-3 standards and meets the safety requirements defined in IEC 61508 for the development process.

The programming system contains a code system for developing the safety control program with the help of the graphic programming languages FBD and LD, a table-oriented variable editor for administering the variable as well as a cross-reference window and many other useful functions for the various phases in developing a safe application: **processing, compiling, sending, controlling the safety control, debugging** the safe application, **printing**, etc.

The system offers the possibility of integrating **libraries**: defined safe standard function blocks such as EMERGENCY STOP, safety door or two-hand control are available in cooperation with PLCopen.

The **user administration** enables the access rights to project changes to be limited to authorized programmers and also provides a chronicle of which changes have been made by which users.

An integrated **bus navigator** tool shows the connected safe devices with the related descriptions, safety recognitions and I/O signals (the bus project is created in ProMaster). In the device detail area of the bus navigator window, the user can allocate the I/O signals of the safe devices during the insertion of global I/O variables in to code in a simply manner by using drag and drop. The bus navigator has a **device parameterization editor** for the parameterization of the connected safe devices.

10.1.1 Programming system ProMaster

As an engineering framework, ProMaster is the development environment for automatization solutions with the b maXX device series.

The b maXX device series primarily encompasses control systems, drive systems and I/O modules as well as graphic operator displays.

ProMaster provides the network view, IEC programming, cam disk draft and the configurators needed for field bus systems such as CANopen, EtherCAT as a real time ethernet system and for the modular b maXX devices including the I/O modules and coupler and much more.

10.1.2 Installing the ProMaster programming systems

NOTICE!

In the following explanations, the depictions on the screen may deviate slightly from the displays with newer program versions.

Place the CD in your CD-ROM drive.

Run the setup file. The installation assistant which guides you through the installation process will appear. First, select the language for our installation.

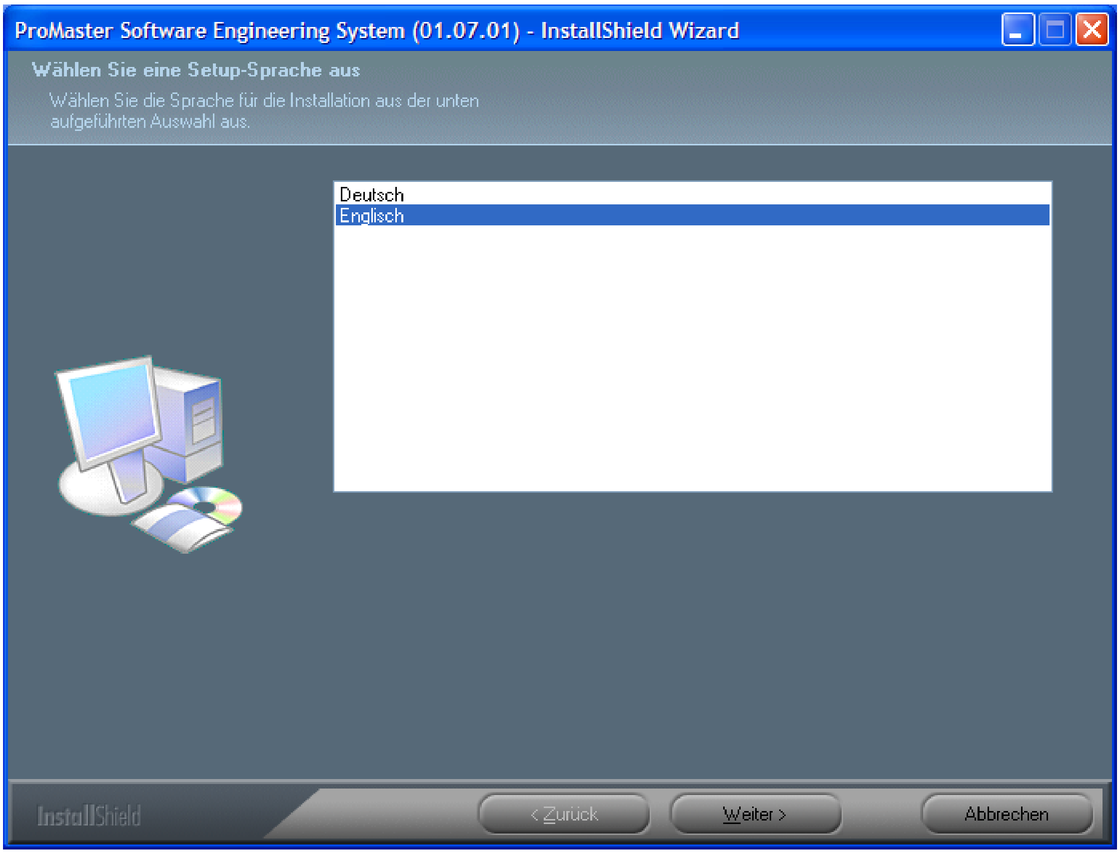


Figure 12: Language selection

Click on 'Next' ('Weiter').

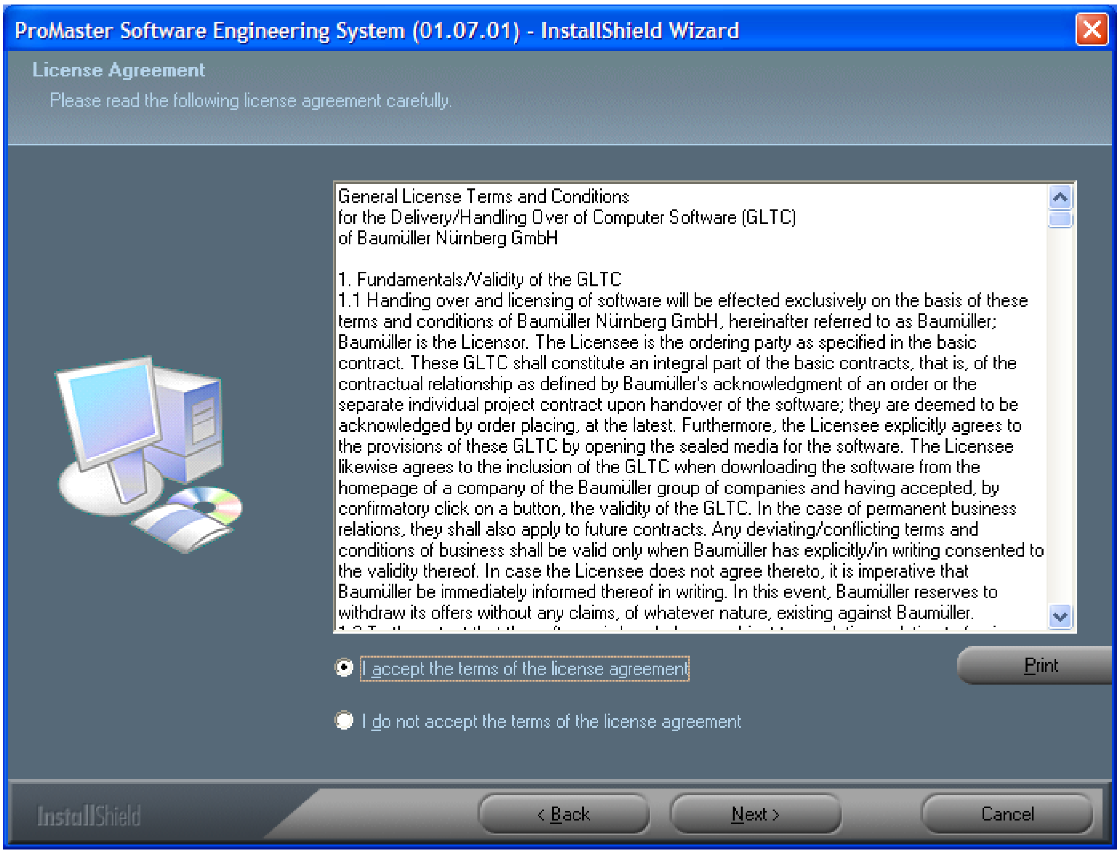


Figure 13: License conditions

Read the license conditions thoroughly. Use the scroll bar to read the rest of the conditions. Highlight the option “I accept the terms of the license agreement” and click on “Next” in order to accept the license conditions and install the software.

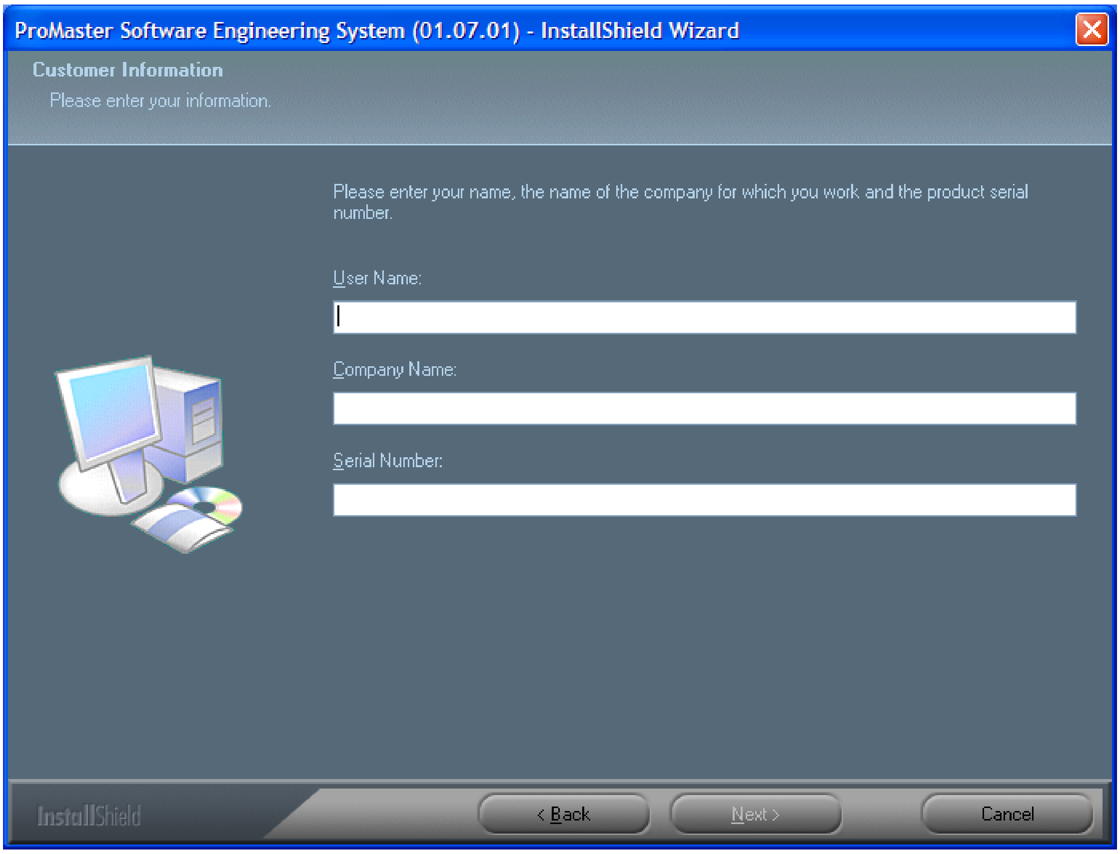


Figure 14: Enter user name and serial number

Enter your user name, company and serial number here. The serial number can be found on the left side of the original CD case. Then click on 'Next'.

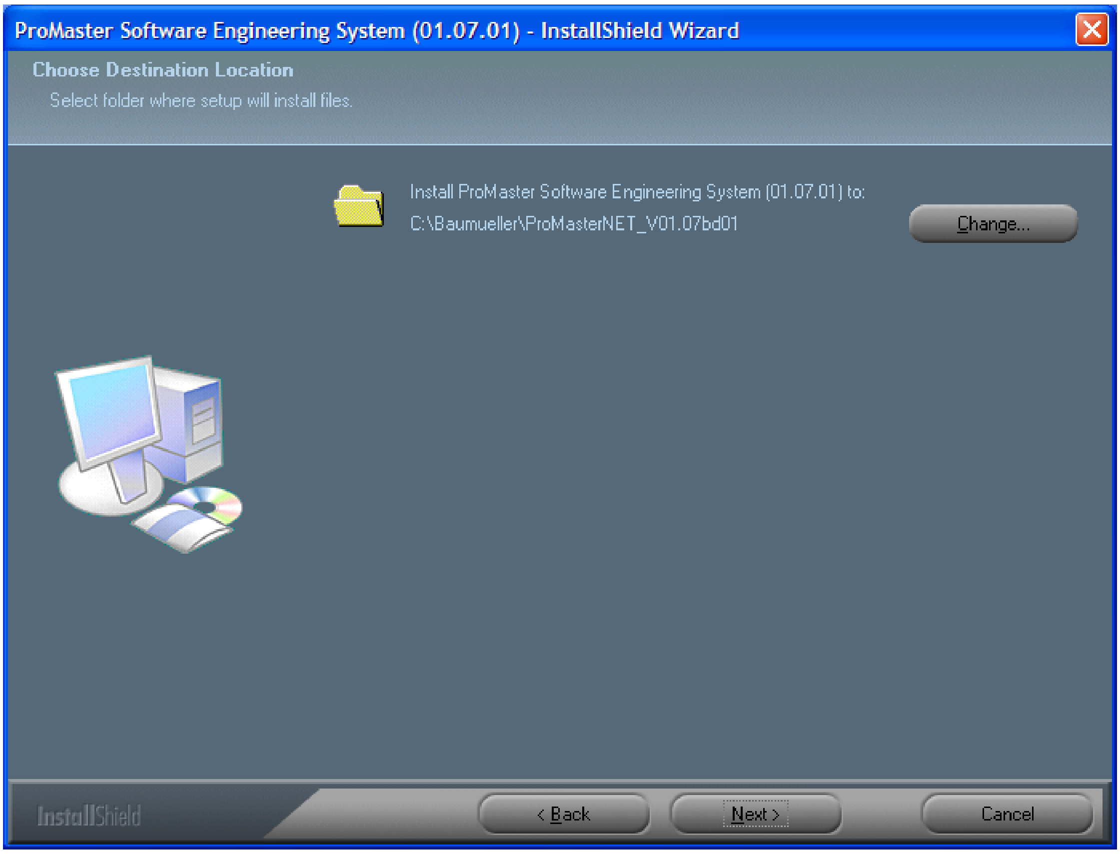


Figure 15: Installation directory selection

Select the directory in which the safe programming system is to be installed. The standard path is already set up.

Click on 'Next'.

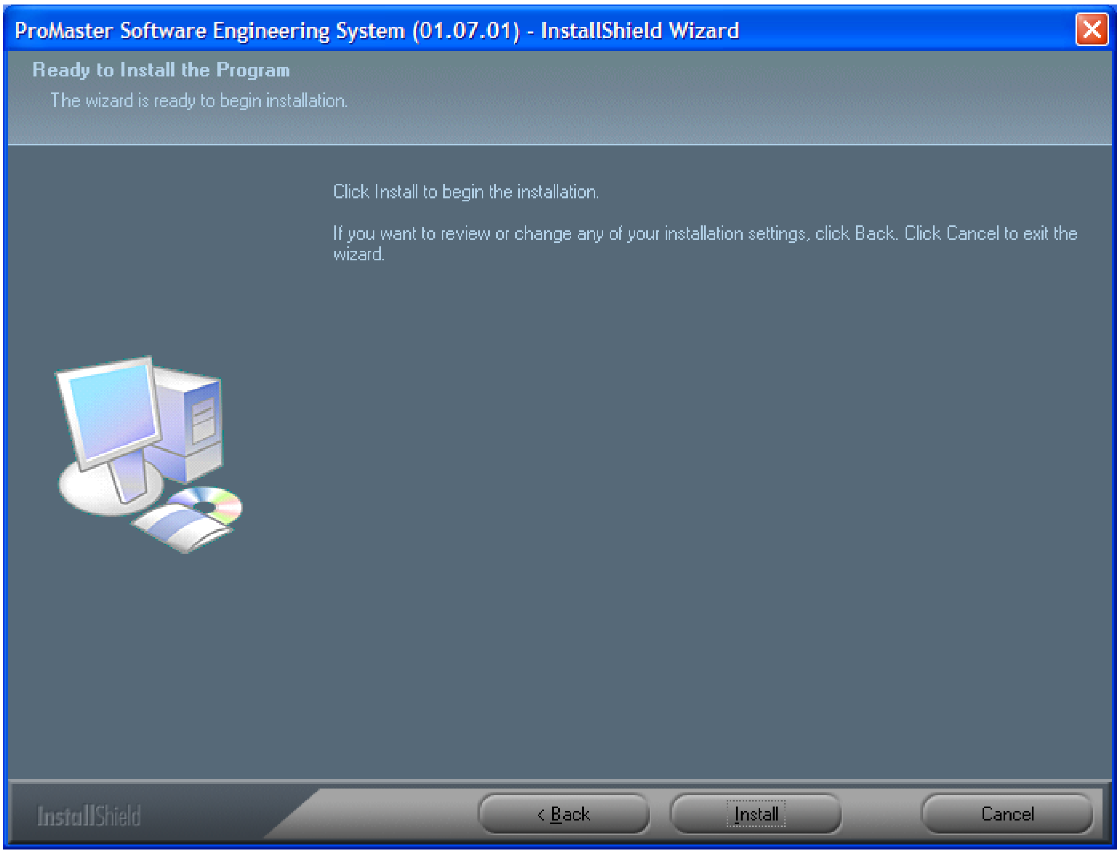


Figure 16: Start installation

Click on 'Install' in order to commence the installation. The installation process can be aborted by hitting 'Cancel'. If you wish to change entries, click on 'Back' to return to the previous steps.

After successful installation, click on 'OK' in order to close the installation assistant. You can start the programming system immediately without having to reboot the computer.

10.1.3 Starting the ProMaster programming systems

The following image will initially appear once the program has been launched:

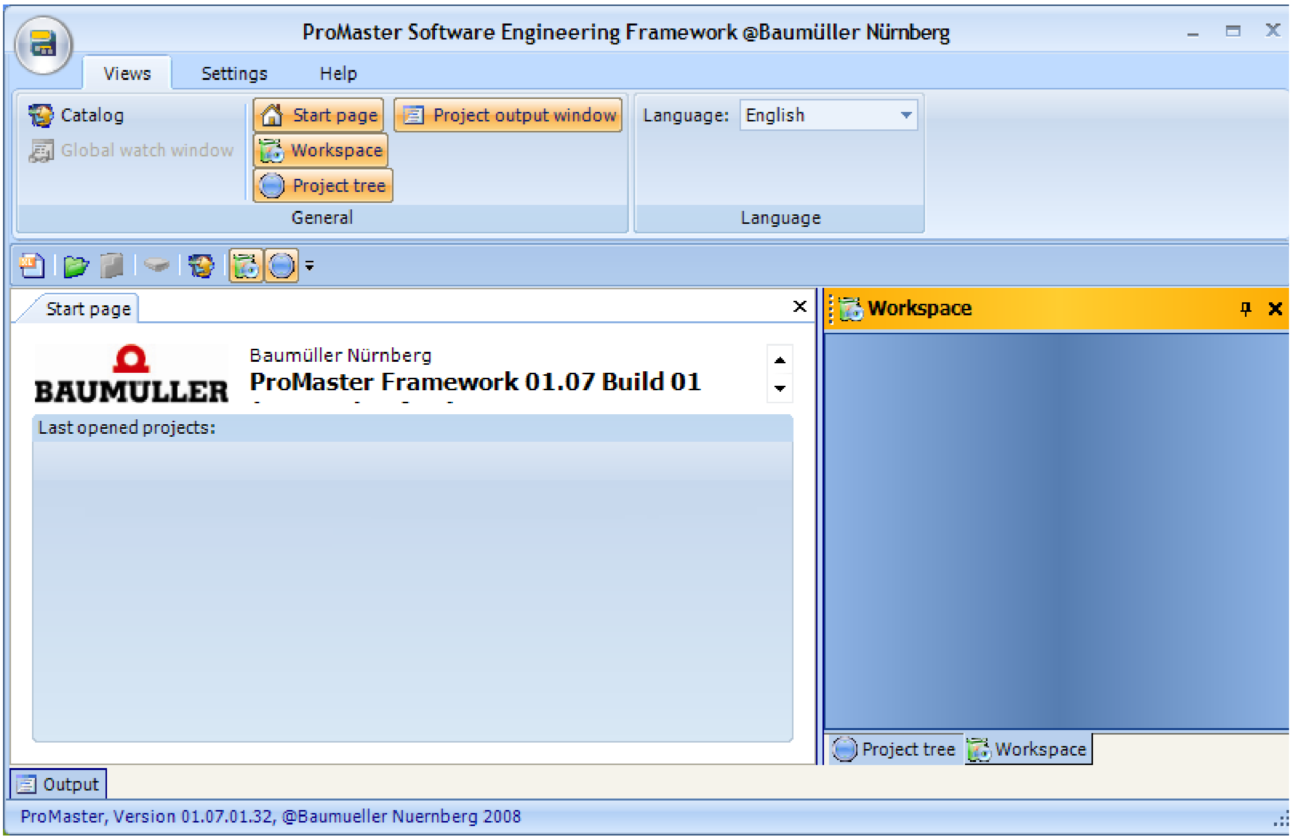


Figure 17: ProMaster start screen

10.1.4 Starting a project

WARNING!

Errors in project planning cannot be entirely precluded by the safe programming system. The user assumes accountability for the project planning!

Therefore:

- Keep at a sufficient distance from moving machine parts/line parts.
- Activate the safety devices of the machine parts/line parts under all circumstances.

Click on the ‘New project’ icon:

The following dialog will then appear:

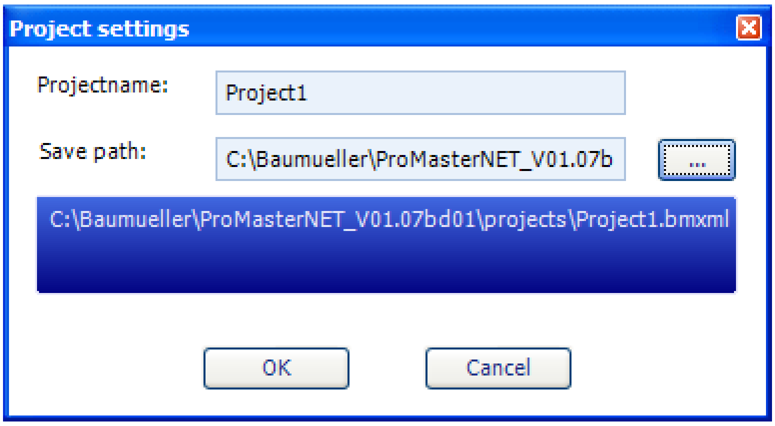


Figure 18: ‘Project settings’ window

In this dialog, you select a project name, “Project”, for example, and a storage location (the default setting is C:\Baumueller\ProMasterNET_V0x.yz\projects). The project file and a further subdirectory with the project name is established in this directory.

NOTICE!

When selecting the project name, please make sure that a subdirectory with the same project name does not already exist, since your project in ProMaster will ultimately contain a great number of files and it will be easier to have a good overview if only one ProMaster project has been created in this subdirectory.

The project will be created upon confirmation with “OK” and the network view will open automatically.

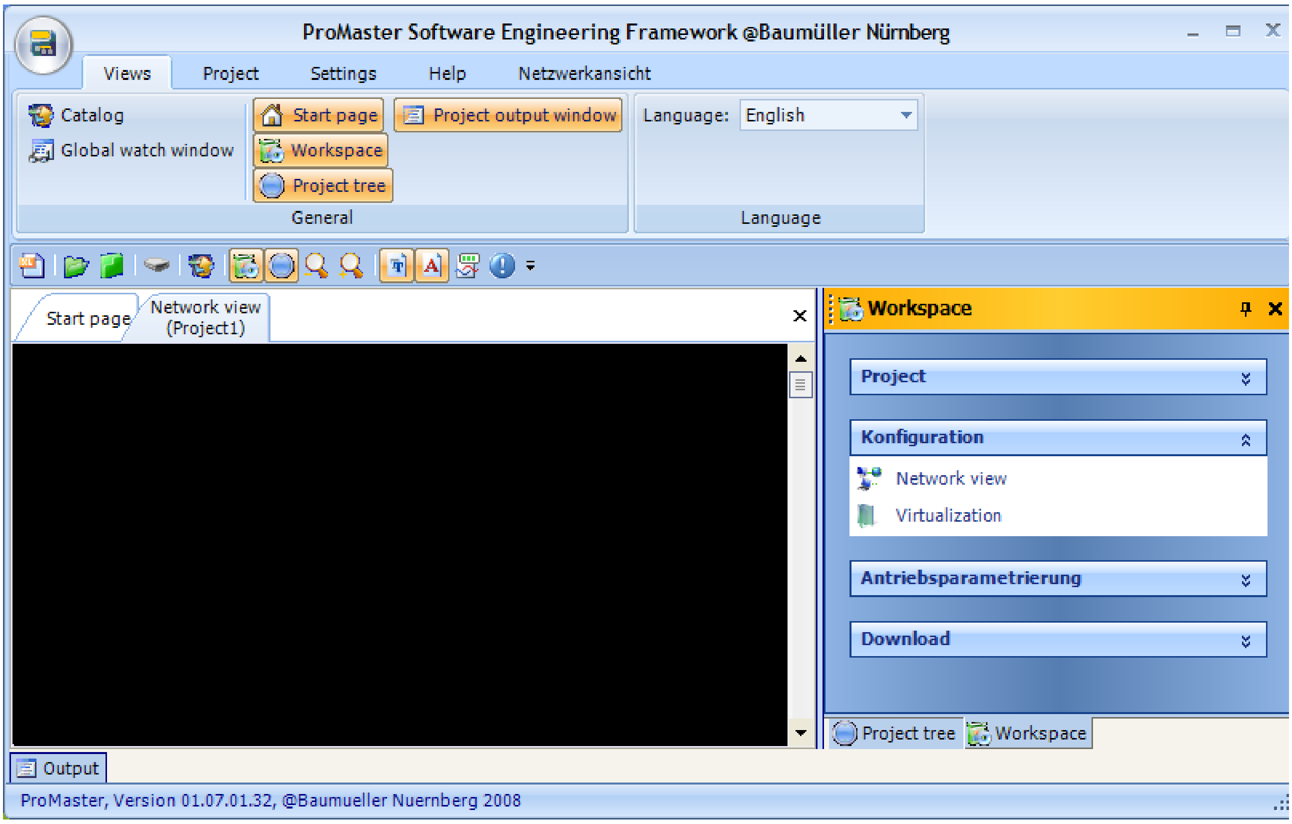


Figure 19: Network view

In the network view, you can now establish your network structure with all devices used. You will first have to open the catalog in order to place devices on the black drawing area. To do this, select the entry “Catalog” under “Views” in the main menu.

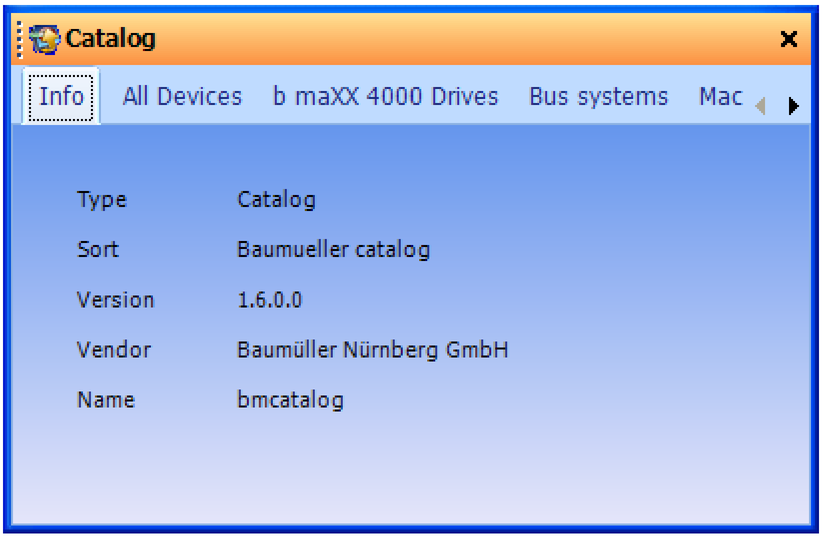


Figure 20: Catalog window

An additional window with the catalog will then appear, in which you click on the “All devices” tab:

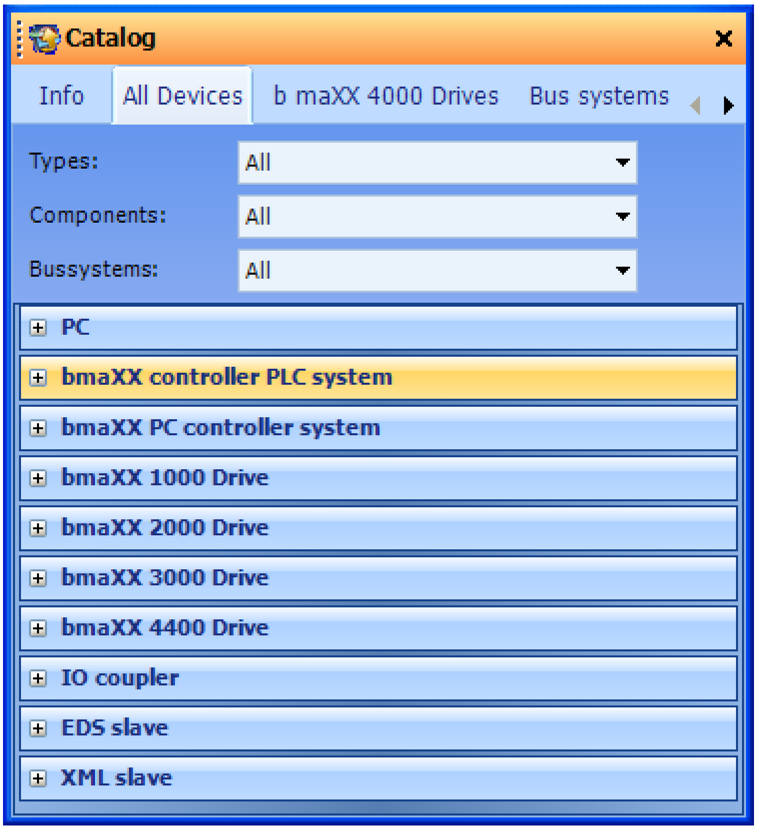


Figure 21: ‘All devices’ catalog

Click on ‘b maXX controller PLC System’ for our example.

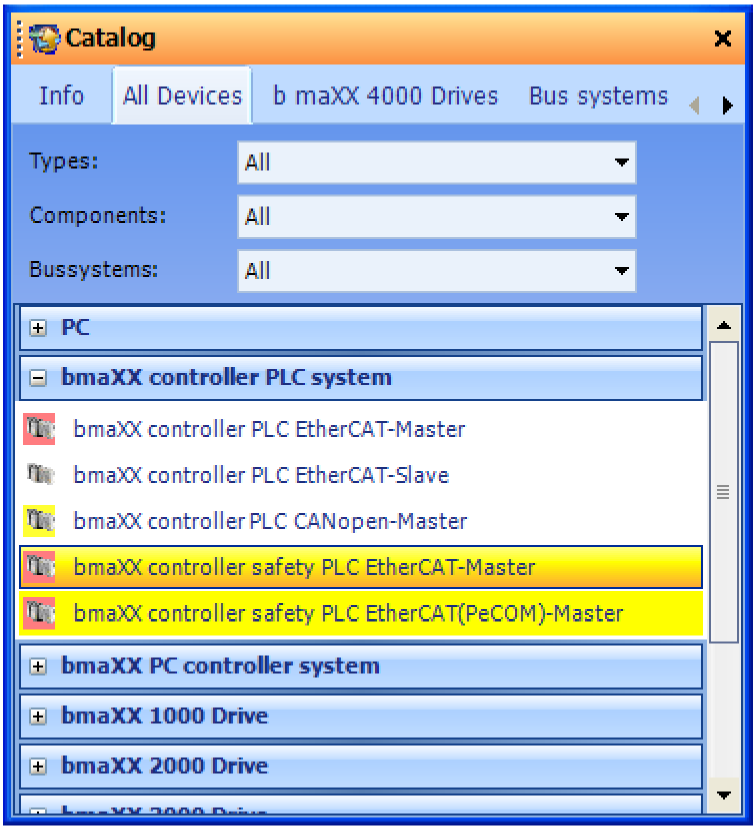


Figure 22: 'All devices' catalog and 'b maXX controller PLC System' opened

You can now drag the desired devices from the catalog into the drawing area with the mouse using drag and drop. The devices will be arranged automatically in the process.

In the following example, we take the template "bmaXX controller safety PLC EtherCAT-Master" as the EtherCAT master. Select this entry in the catalog, drag it onto the black drawing area with the mouse and release the mouse button. A b maXX controller PLC system with an EtherCAT master will then be added to the project.

The drawing area will then look like this:

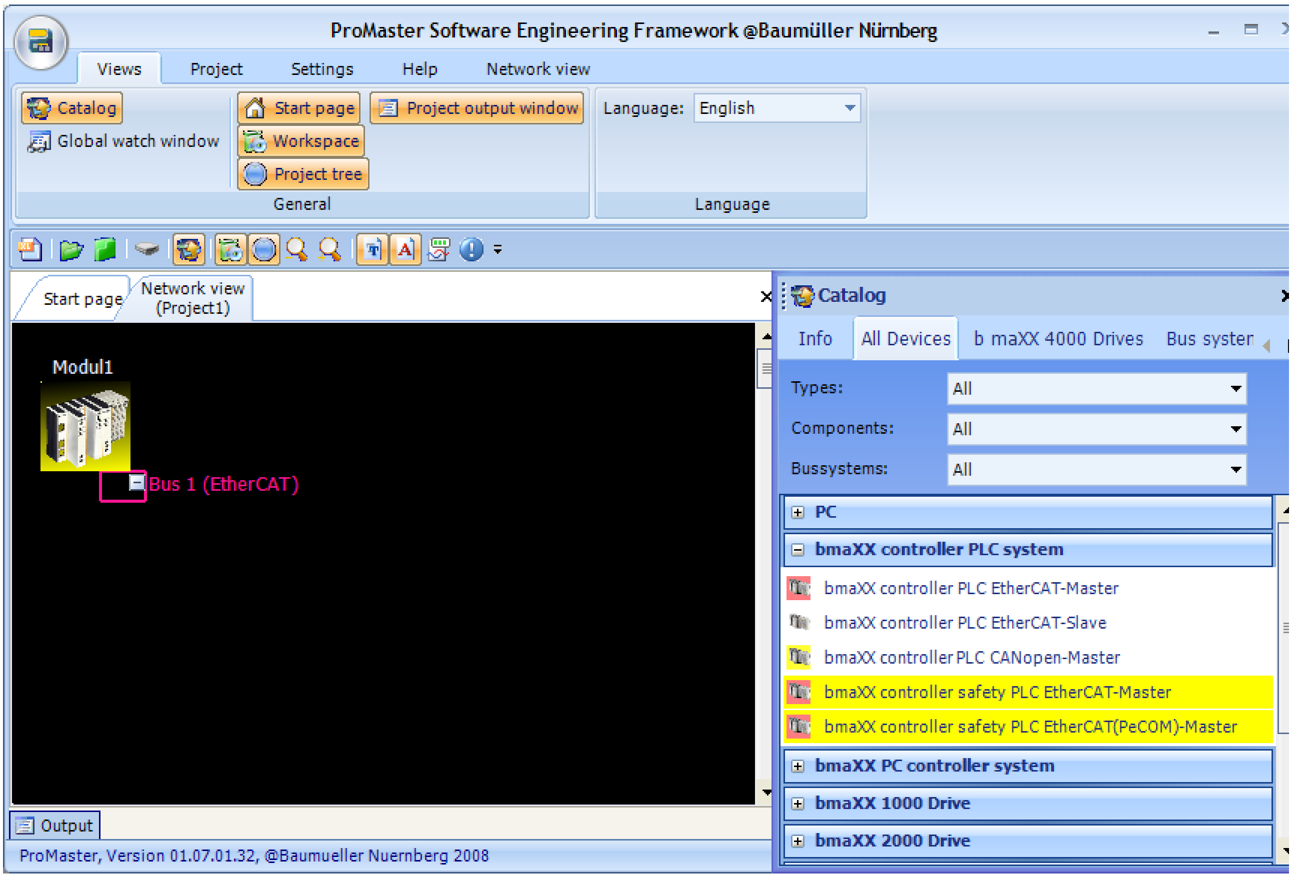


Figure 23: Network view with the EtherCat master module

The device has automatically received the name “Modul1”.

Now, close the catalog and select the module in the network view by clicking on the module. The module will receive a white dashed frame.

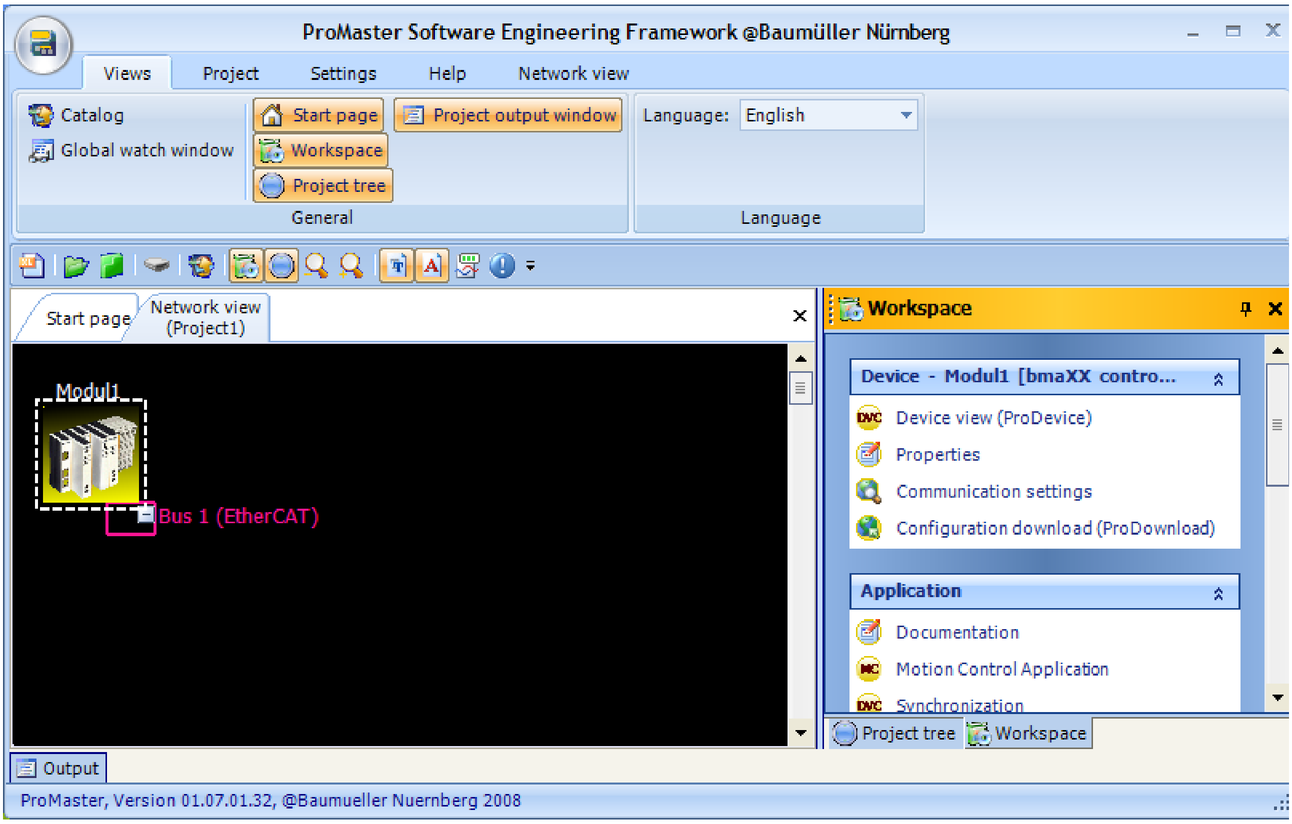


Figure 24: Device view with the EtherCat master module

Click on ‘Device view (ProDevice)’ at the right of the ‘Workspace’. The selected module with its individual components will then appear.

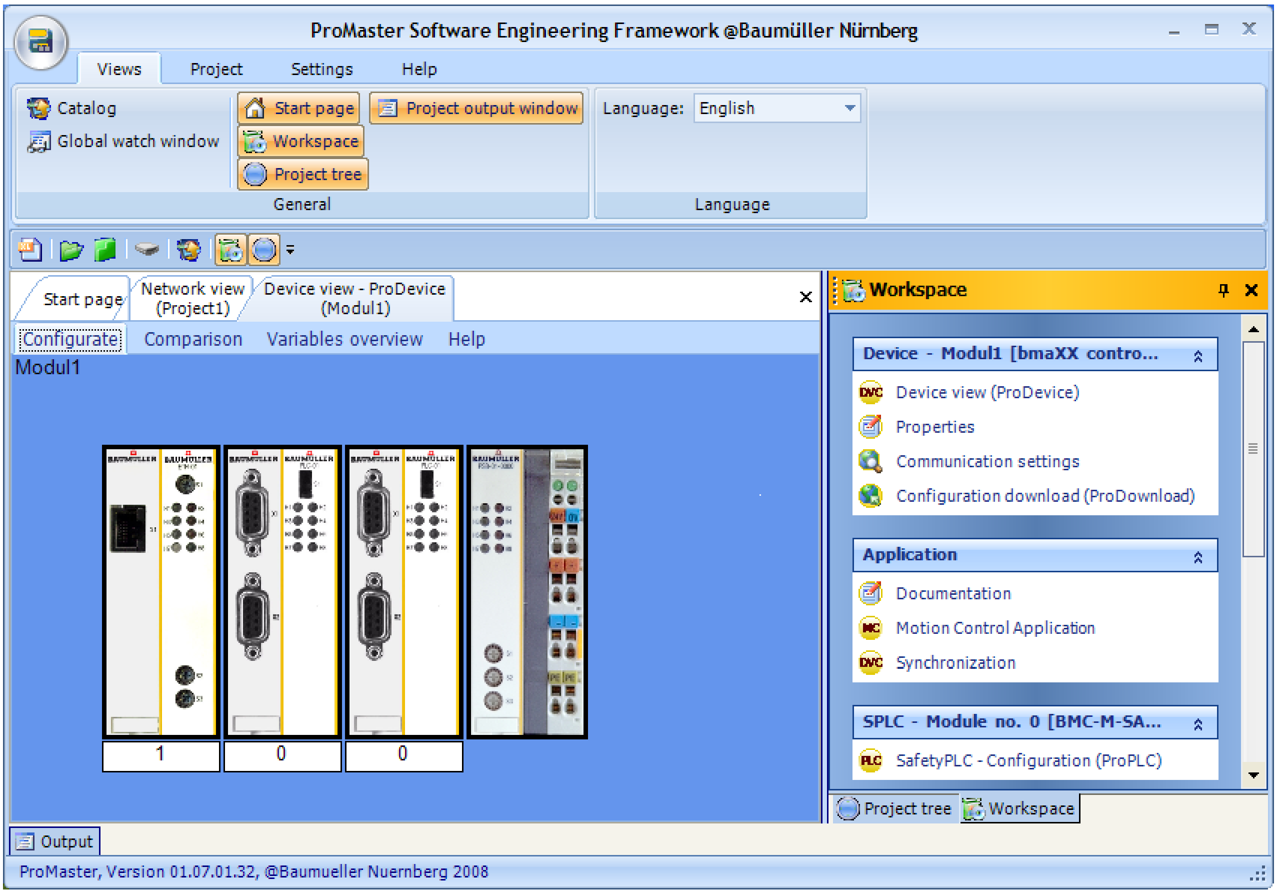


Figure 25: Device view EtherCat master module

Open the catalog again and select the I/O module tab.

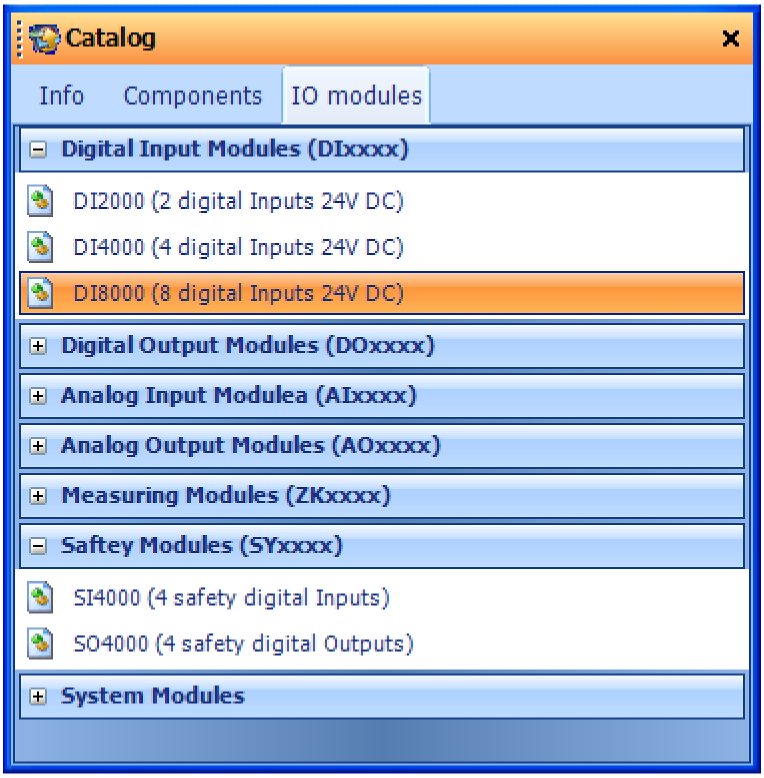


Figure 26: 'Catalog' window, I/O modules opened

Click on the 'Digital input modes'. The subdirectory shows all digital input modules present. For the example, you can pull up an 8-channel DI8000 input model from the right into the module shown in device view by using drag and drop.

NOTICE!

The final module EK0000, which is absolutely necessary, will automatically appear after the first I/O module is inserted.

For the example, drag another DO8000 digital output module and another SI4000 digital input module from the directory 'Safety module SYxxxx' and a SO4000 safe digital output module into the device view. Once the catalog has been closed, the device view will look like this:

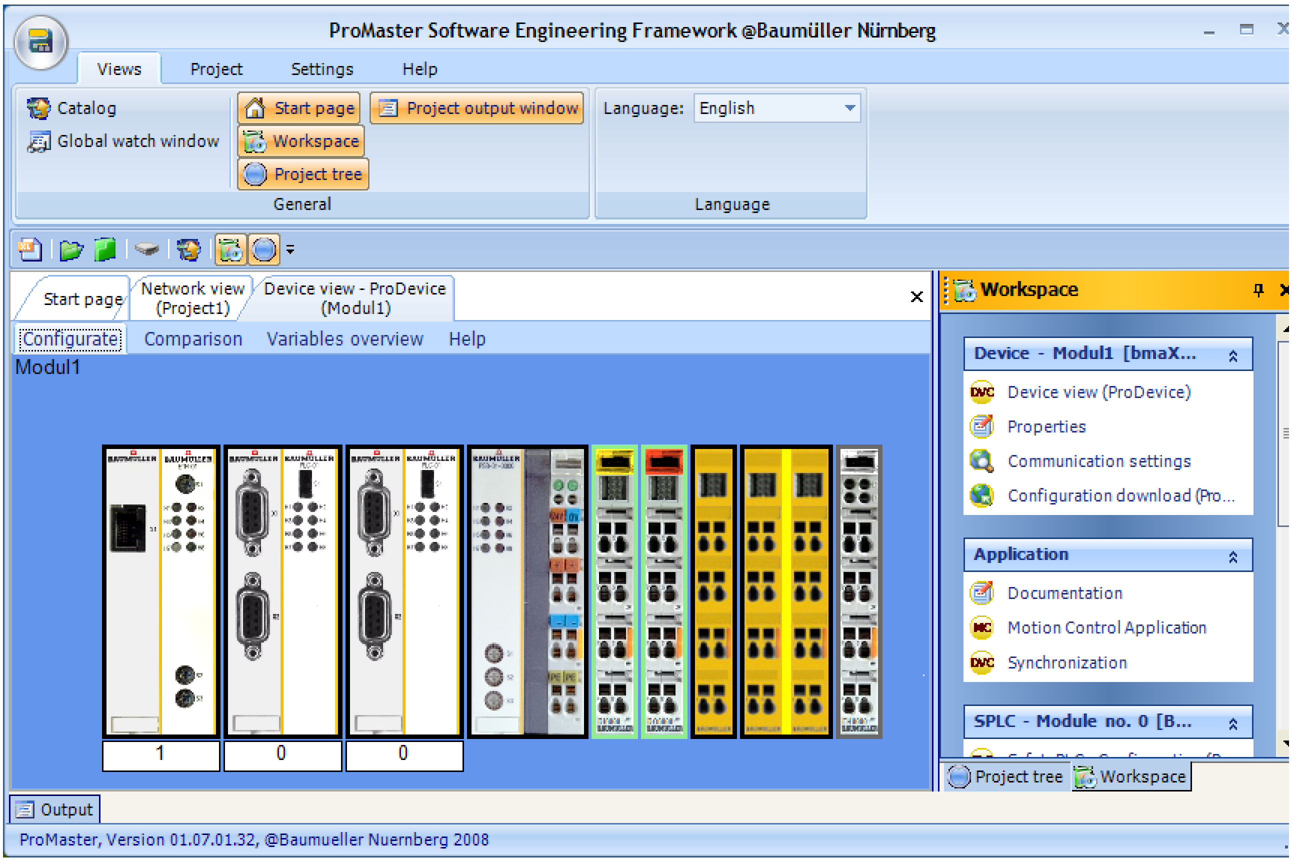


Figure 27: Device view EtherCAT master module I/O modules

Click the menue ‘Comparison’ in the device view.

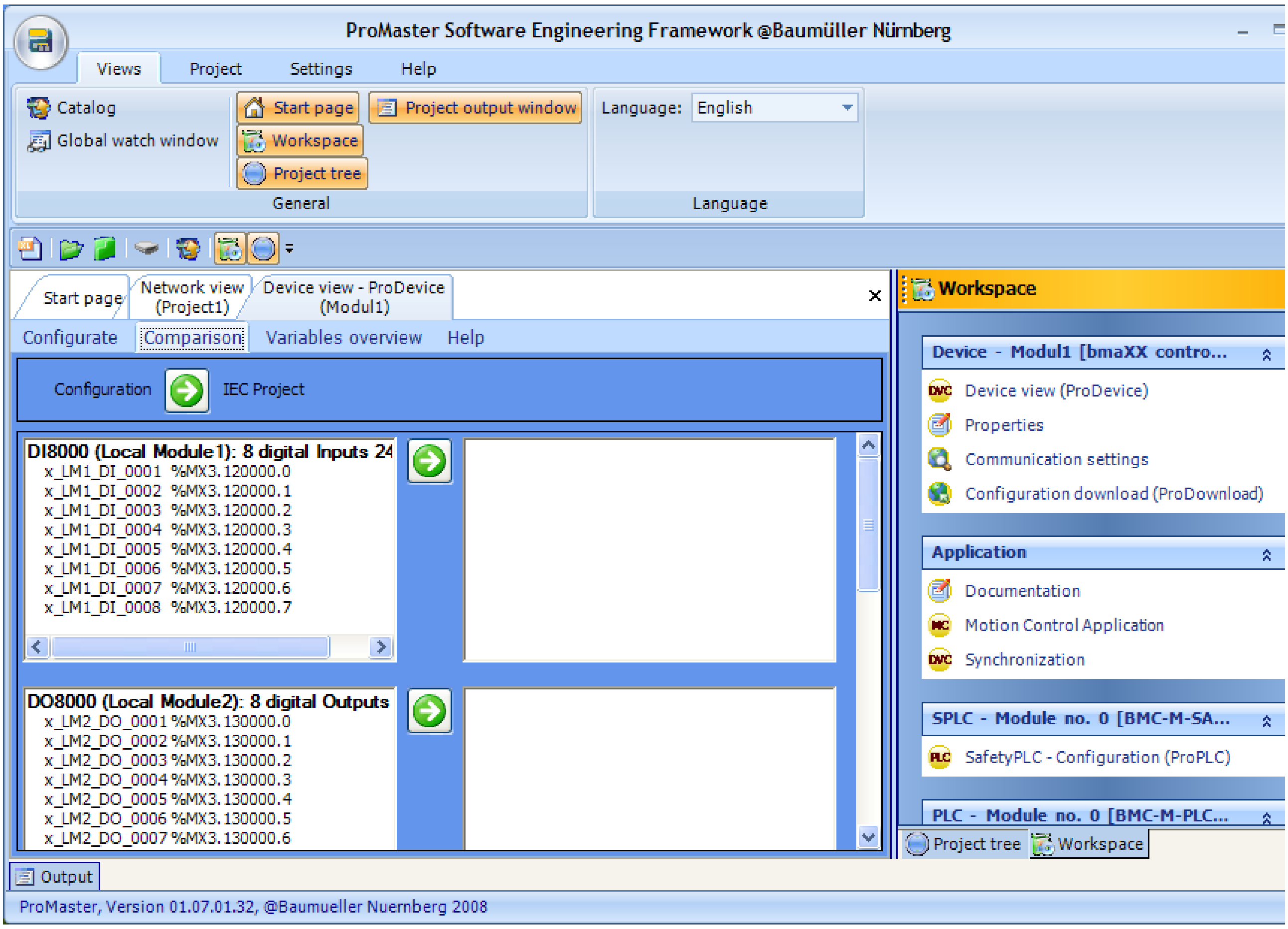


Figure 28: Device view EtherCAT master module I/O modules

Click on the green arrow in order to assume the current configuration in the IEC project.

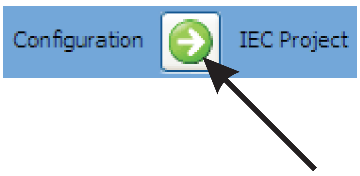


Figure 29: Assuming the configuration in the IEC project

Now close the ‘Device view’. It will remain in the ‘Network view’. Click on the ‘Communication settings’ in the ‘Workspace’ at the right. The window ‘Port parameter’ will open.

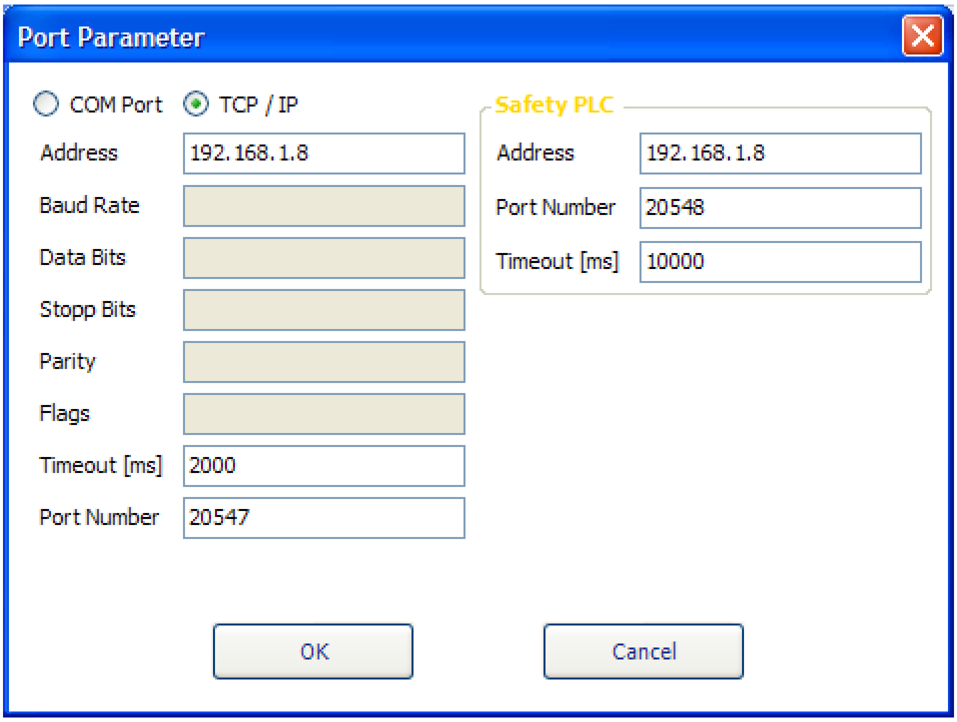


Figure 30: The 'Port parameter' window

First, select `TCP/IP' and enter the IP address (it is automatically overtaken for 'Safety PLC').

Now start the ProSafety program either with double click on the desktop or with the start menue.

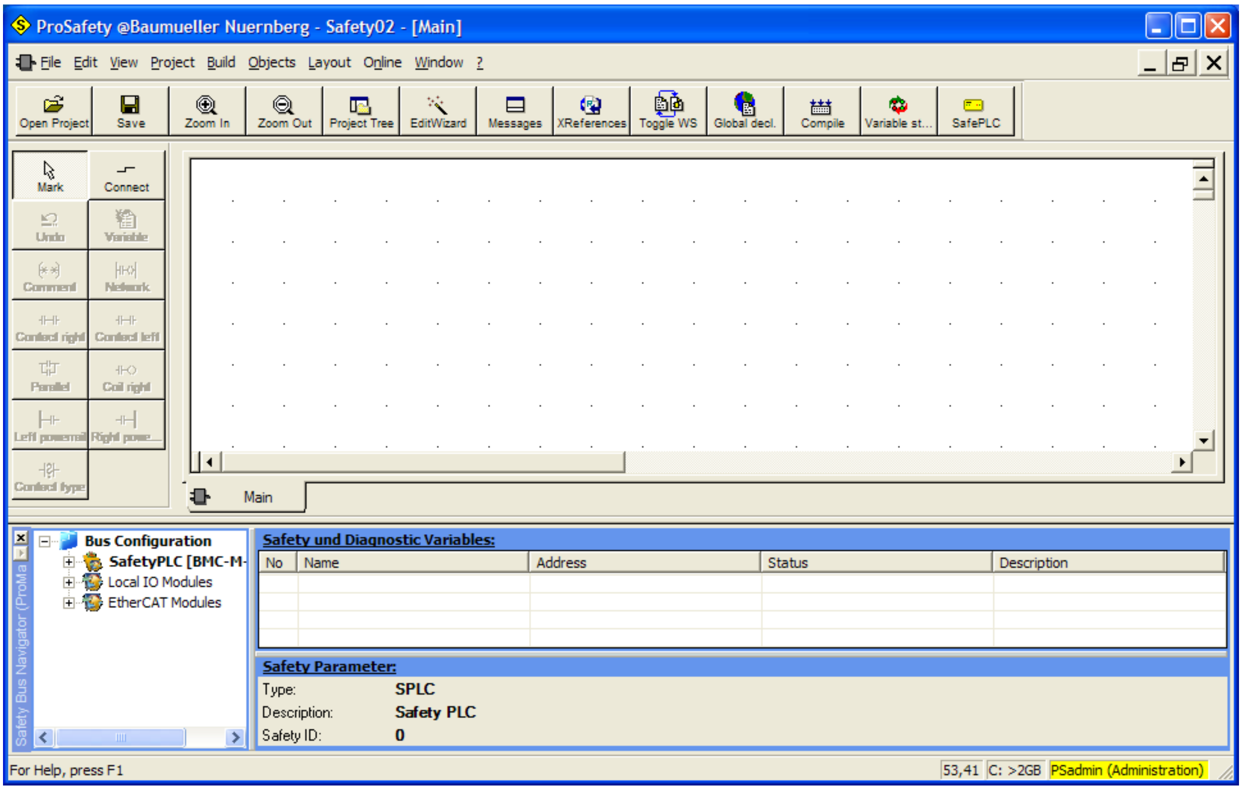


Figure 31: ProSafety start screen

10.1.4.1 User administration - Adding a new user

Once the safe programming system has been started, there will be not be any registered users yet. A message will appear, notifying that at least one user must be registered in the 'Administration' group before you can log in:

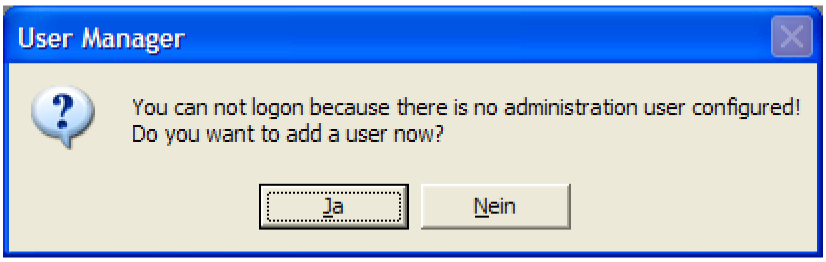


Figure 32: The ‘User administration’ window: Adding a new user

You will first have to define an administrator user in order to create a sample project and use all of the features of the safe programming system.

This means that you will generally have to decide which of the participating people will have administrator rights and thus have access to all of the functions of the safe programming system.

1 Click on ‘Yes’ in the dialog above to open the 'User administration dialog:

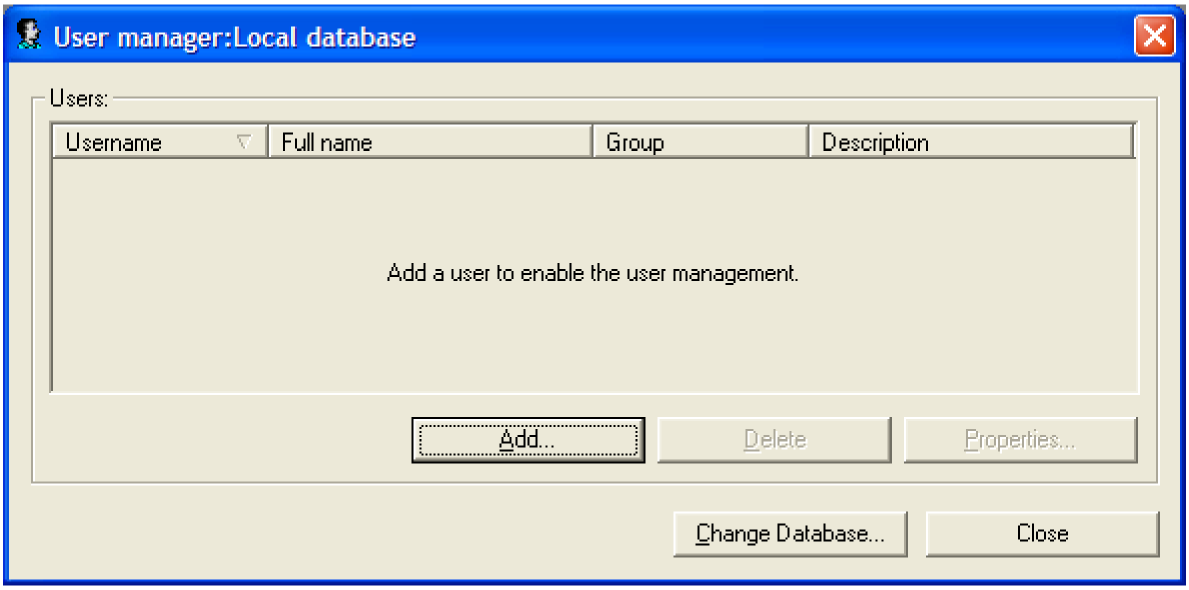


Figure 33: The ‘User administration’ dialog for registering a new user in the safe programming system

2 Click on the ‘Add...’ button in order to open the ‘Add user’ dialog.

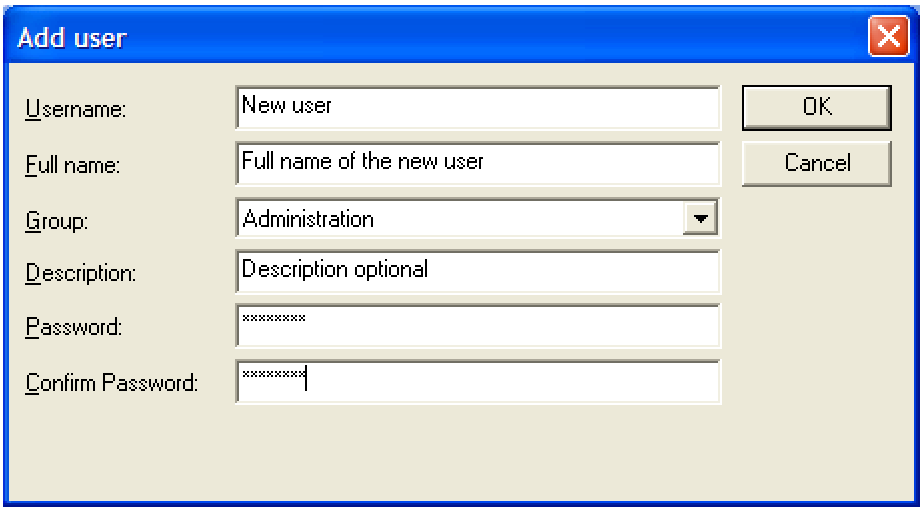


Figure 34: 'Add user' dialog with the new user’s data

3 Enter the required user data.

All of the fields except the ‘Description’ field have to be filled out. Otherwise, an error message will appear when the dialog is confirmed. The password must be at least six digits long.

4 Confirm the dialog with ‘OK’. The ‘User administration’ dialog will now receive the newly established user ‘New user’.

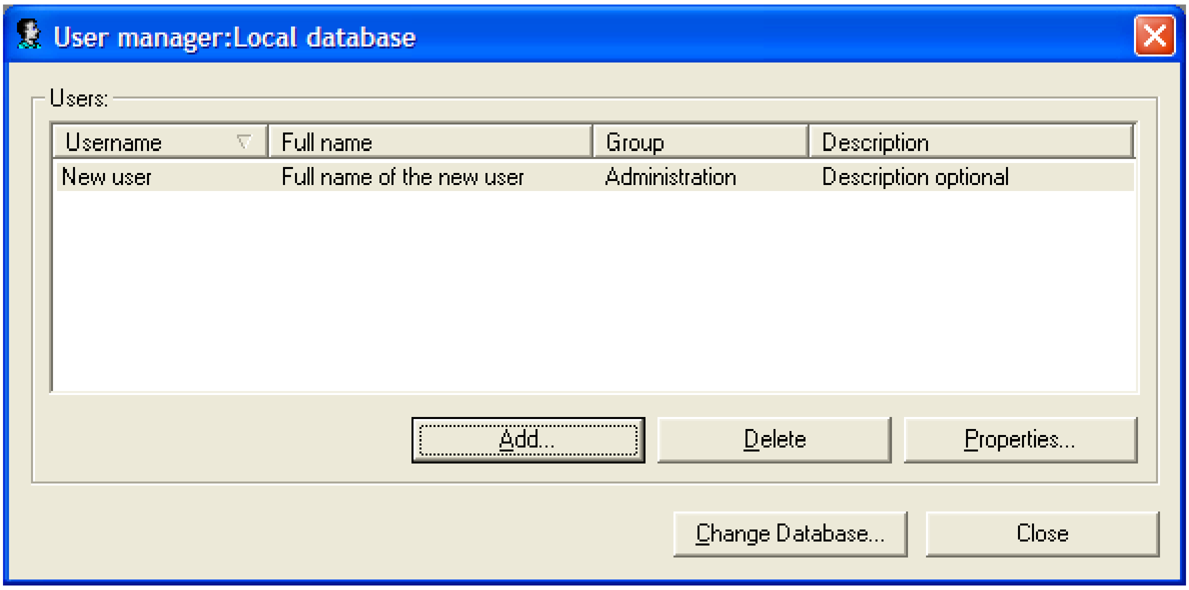


Figure 35: The ‘User administration’ dialog with the newly established user

5 Click on the ‘Close’ button to close the ‘User administration’ dialog.

The safe programming system’s user interface will be activated and you can commence with the development of a new project.

NOTICE!

The log-in procedure will be activated as soon as at least one administrator user has been registered, that is, you will have to log in with a valid user name/password combination each time you launch the safe programming system.

The course of action described in this chapter also applies to the establishment of new users in the ‘Development’, ‘Commissioning’ and ‘Maintenance’ groups.

We would now like to develop a sample project with the programming language function block diagram (FBD).

NOTICE!

For best results, please use the same identifier and name as in this handbook. We have largely selected descriptive variable names to increase the understandability of the project.

DESCRIPTION OF THE SAMPLE PROJECT

The work piece is held in a processing machine by clamping the clamp jaws. If the clamp jaws are released, the work piece can be removed.

The engagement area for changing work pieces is secured by a guard door. Interference behind the protective device is prevented by constructional measures. The processing machine must immediately be shut down upon opening the guard door. This prevents the machine from being accessed while a tool is running. An EMERGENCY STOP switch is installed.

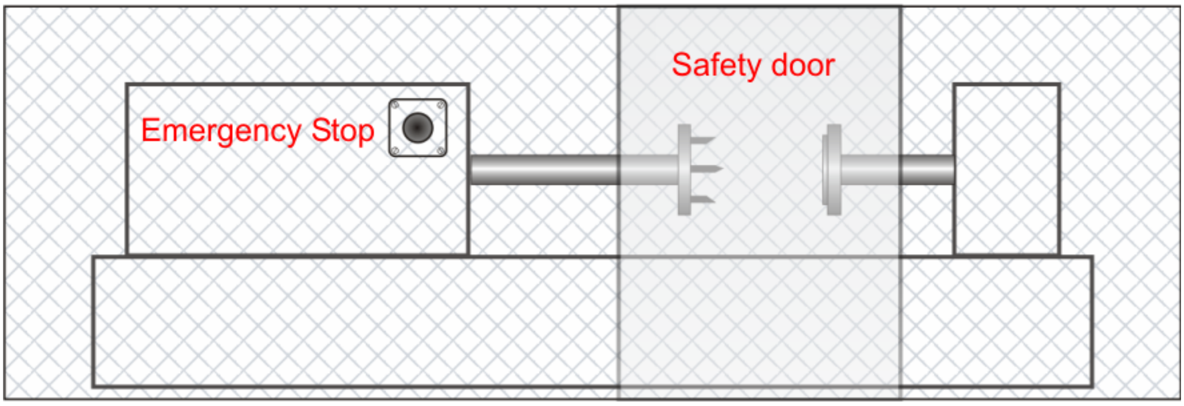


Figure 36: Processing machine

The following pages give you an overview of the entire project which we will be developing in this handbook. The step-by-step instructions describe the function of the sample project and its programming (see ▹Phase 1: Create a new project with the help of the project assistant ◃ from page 66 onward).

Our sample project will contain the code spreadsheet 'Main' (see ▹Code spreadsheet of the POU 'Main': ◃ on page 62). Furthermore, the 'PLCopen_SF' library, which contains special safe function blocks, is also to be integrated.

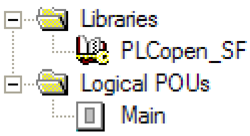


Figure 37: Project tree of the sample project

All global and local variable declarations as well as the code spreadsheet are depicted in the following.

Global variable declarations in the project:

		Name	Data type	Description	Terminal	Init
1	⊟	NewGroup				
2		ES_01_Input	SAFEBOOL		Safety Module SI4000 [ID: 1] - Input 1	
3		SDoor_01_Input1	SAFEBOOL		Safety Module SI4000 [ID: 1] - Input 2	
4		Valve_01	SAFEBOOL		Safety Module SO4000 [ID: 2] - Output 1	
5		Y_M_01_Input	SAFEBOOL		Safety Module SI4000 [ID: 1] - Input 3	
6		Y_M_02_Input	SAFEBOOL		Safety Module SI4000 [ID: 1] - Input 4	
7		M_01_Output	SAFEBOOL		Safety Module SO4000 [ID: 2] - Output 2	
8		x_junctionVarIn_01	BOOL		Normal Module # 0 Input Terminal 0	

Figure 38: Global variable spreadsheet with all variable declarations and allocations of connecting terminals

Local variable declarations of the POU 'Main':

	Name	Data type	Usage	Description	Init
1	⊟ NewGroup				
2	Emergency_Stop_01	SF_EmergencySto...	VAR		
3	Enable_From_Device_OK	BOOL	VAR		
4	S0_Reset	BOOL	VAR		
5	Error_Emergency_Stop	BOOL	VAR		
6	DiagCode_Emergency_Stop	WORD	VAR		
7	SDoor_01	SF_GuardMonitorin...	VAR		
8	Protective_Equipment_OK	BOOL	VAR		
9	Error_SDoor	BOOL	VAR		
10	DiagCode_SDoor	WORD	VAR		
11	Enable_Contactor_Monitoring	SAFEBOOL	VAR		
12	Contactor_Monitoring_01	SF_EDM_V1_00	VAR		
13	Error_CMonitoring	BOOL	VAR		
14	DiagCode_CMonitoring	WORD	VAR		
15	Enable_Contactor_01	SF_OutControl_V1...	VAR		
16	M_01_Control_Output	BOOL	VAR		
17	Enable_Contactor_OK	BOOL	VAR		
18	Error_Enable_Contactor	BOOL	VAR		
19	DiagCode_Enable_Contactor	WORD	VAR		

Figure 39: Local variable declarations of the POU 'Main'

Code spreadsheet of the POU 'Main':

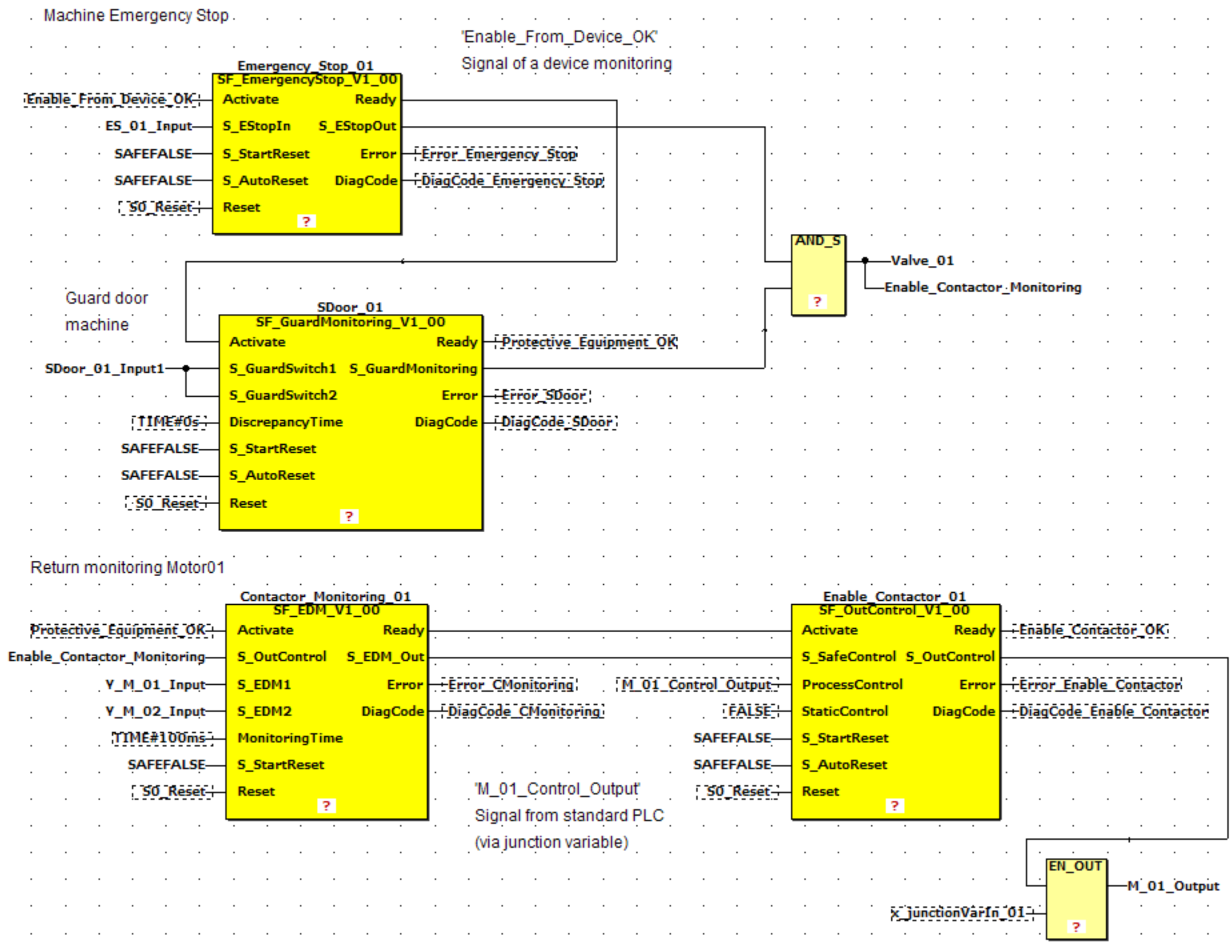


Figure 40: Code spreadsheet of the POU 'Main'

We now have an overview of how the sample project will look once it has been finished and can begin to create a new project in the safe programming system.

We will first provide some general information on use before proceeding to the step-by-step instructions.

GENERAL INFORMATION ON VARIABLES AND FUNCTION MODEL INSTANCES IN THE SAFE PROGRAMMING SYSTEM

Inserting variables and function block instances

Variables and function block instances as well as constants can be inserted directly into the code in the safe programming system during development.

The dialog 'Variable' is used to insert one of these objects.

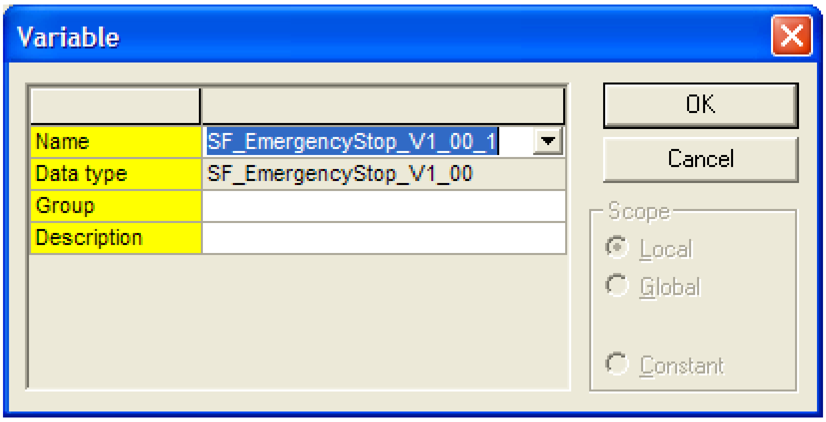


Figure 41: The ‘Variable’ dialog for inserting variables, function block instances and constants

NOTICE!

The ‘Variable’ dialog can contain different elements depending on the type of object to be inserted.

The ‘Variable’ dialog is used when you

- define new variables and insert them in the code or when you insert already declared variables.
- draw a variable of a terminal from the bus navigator into the code (since a global variable must be assigned in this case).
- replace an existing variable.
- declare function block instances.
- insert a constant (literal) into the code.
- change the properties of existing variables.
  The dialog will appear when editing the code when you double click on a variable, a contact or a coil or select the menu point ‘Object properties...’ from the context menu of the object.

The related declarations will automatically be inserted into the corresponding variable spreadsheet (see ▷Page 64) when inserting new variables or function block instances in the code. In the process, it makes no difference if the variable workplace is open or not.

NOTICE!

Replacing variables

Existing variables in the code can be replaced, even if they are already assigned to an LD object or are connected to the input or output of a function or function block. Moreover, it makes no difference if the variable should be replaced by a new variable or an already existing one.

Again, the ‘Variable’ dialog is used to replace variables.

Proceed as follows:

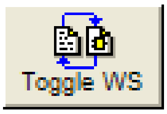
- Double click on the variable to be replaced.
  The 'Variable' dialog will be opened.
- Select either an existing variable in the list field ‘Name’ or provide a new variable name and define the properties of the variable.
- Confirm the 'Variable' dialog with 'OK'.
  The variable will be replaced.

Variable spreadsheets

Instead of declaring variables and function block instances directly when processing the code using the ‘Variable’ dialog (▹Inserting variables and function block instances ◃ on page 62), you can also insert the declarations into the variable spreadsheet “manually” and use these variables/instances in the code spreadsheet later.

However, we recommend declaring new variables directly when inserting them into the code by using the ‘Variable’ dialog, since the declaration and the insertion are carried out in a single step this way.

Variable spreadsheets are opened as follows



The local variable declarations used in the code spreadsheet are contained in the local variable spreadsheet.

In order to open the local variable declarations directly from the project tree, click on the POU symbol with the right mouse button and select ‘Variable spreadsheet’ in the context menu.

Alternatively, you can open the code spreadsheet first and then hit <CTRL>+<D> or click on the ‘‘Switch AB’ while the code spreadsheet is active.

The declaration spreadsheet for global variables is not visible in the project tree. Click on the ‘Global decl.’ symbol in the symbol bar in order to open it.

For quick and simple access the variable spreadsheet in which a particular variable is being used, click on the desired variable in the code with the right mouse button and select ‘Go to variable name definition’ in the context menu.

The related variable spreadsheet will then be opened and the corresponding variable will be highlighted.

Chief advantages of the variable spreadsheet

The variable spreadsheets are used in practice as variable tables. Thus, the declarations are not stated in pure text form (as described in the IEC), but in table form, which makes the declarations much easier to manage. Each table contains a declaration of a variable or instance, each column in the table stands for a variable property (i.e. an element of the declaration). This way, the table reflects the complete declaration syntax in accordance with IEC 61131.

All processing functions in the variable table can be conducted via either the context menu or the 'Edit' menu.

In the variable table, you can

- create new variables or variable groups and process or delete existing objects.
- cut, copy and paste elements within the table.
- cut connected connecting terminals, that is, separate terminals (only in the global variable spreadsheet).

INFORMATION ON NON-SAFE VARIABLES

Addressing I/Os

The safe programming system supports the strict separation of safe and non-safe logics (networks). Non-safe variables are represented with in a dashed frame in code spreadsheets.

CAUTION!

Please follow the following rules for I/Os:

- Safe signals may not be laid on non-safe physical outputs in order to conduct safe functions.
- Non-safe inputs may only be used in safe applications for programming the shutdown via EN_OUT or with function blocks which were specially developed for connection with non-safe signals (for more on this, read the instructions and detailed information in the documentation of the individual function blocks).
- When using safe physical inputs without safety sensors, the data type in the global variable spreadsheet must be set on non-safe (i.e. BOOL instead of SAFEBOOL) in order to define these inputs as non-safe. Otherwise, the safe programming system’s data type inspection will not recognize the incorrect use of these non-safe signals in safe usage.

Use in sample project:

In our sample project, the non-safe global variable 'S0_Reset' (data type: BOOL) is used in order to restore all function blocks.

CAUTION!

Device diagnosis

The input signals of the device diagnosis can be non-safe variables. These are only allowed for the group shutdown and may not be used for functions relevant to safety.

10.1.4.2 Phase 1: Create a new project with the help of the project assistant

The project assistant guides you through the creation of a new projects in 3 steps. In the process, you must determine the name and path of the project, the related bus configurator project and the safe startup component group used.

1 Select the menu item 'File > New project'.

The 'Project assistant' dialog will appear (step 1 of 3) (see ▷Figure 42◁ on page 67).

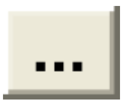
2 Enter the desired project name ('Safety02') in the first entry field (see ▷Figure 42◁). The project assistant will save the project in the file 'Safety02.swt' and create a subfolder of the same name in which the code spreadsheets, variable files, etc., are saved.

NOTICE!

In accordance with the applicable rules for projects, the project name and project path may not contain spaces or special characters. The project name may consist of not more than 24 characters.

The standard path for projects will automatically be entered when the project assistant is started.
If you wish to save the project in a different path, enter it in the second entry field as follows:

3 Click on the search button.



The dialog ‘Find folder’ will appear.

4 Select a directory for the new project and click on 'OK'.

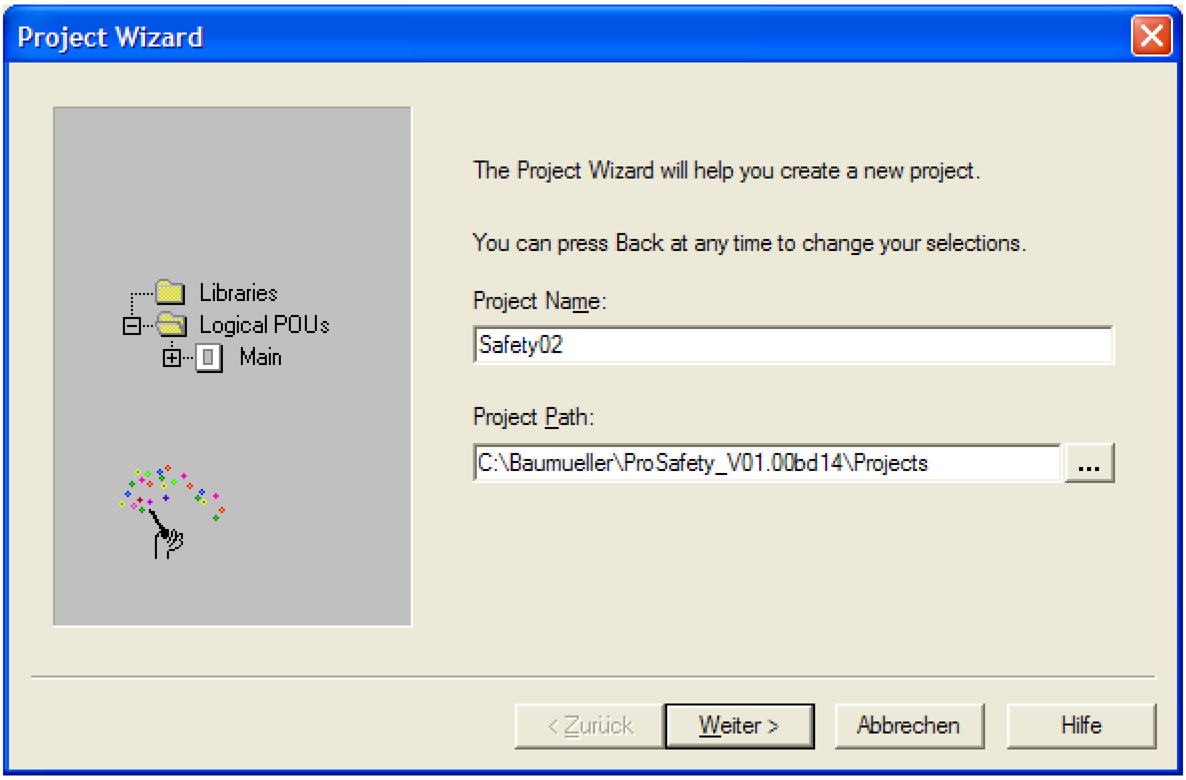


Figure 42: ‘Project assistant’ dialog for entering the project name and project paths

5 Click on the ‘Next’ (’Weiter’) button in order to proceed.

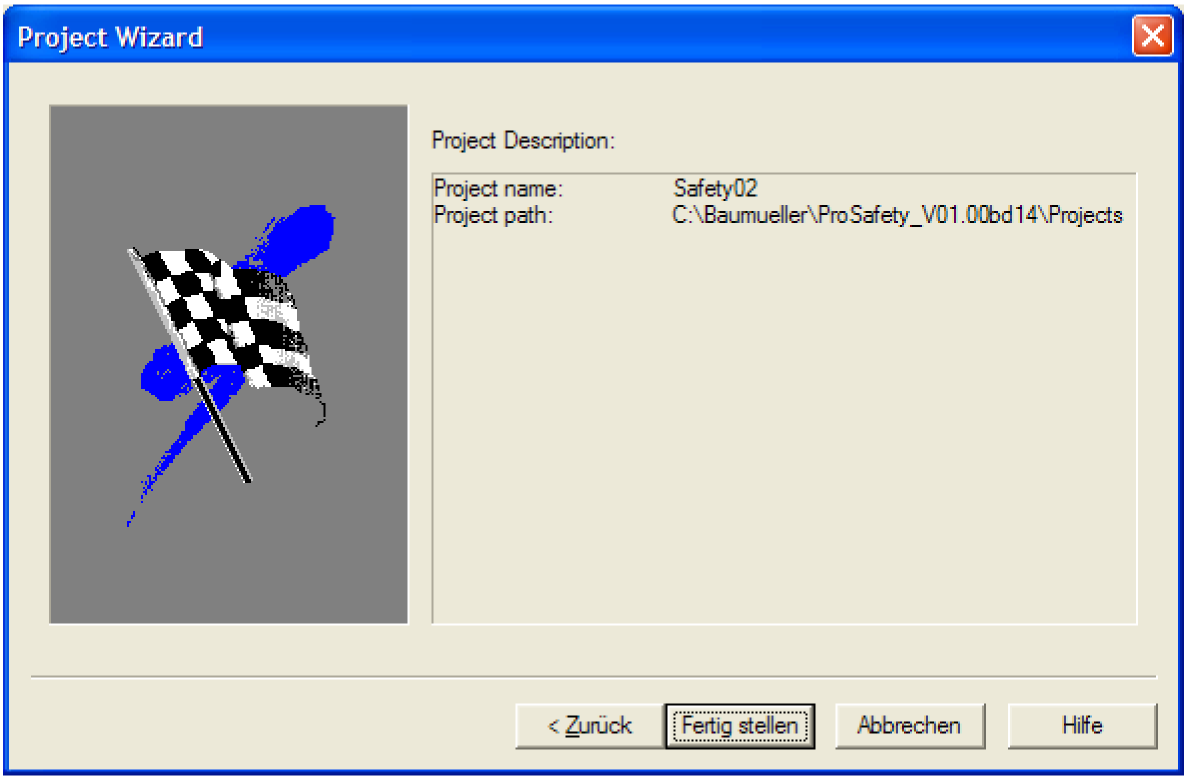


Figure 43: ‘Project assistant’ dialog with a summary of the project settings

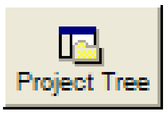
This dialog shows a summary of the project settings which you have established in steps 1 and 2. If you have entered an impermissible name, the error report (‘INVALID NAME’) will appear and the ‘Finish’ button will be inactive. In such case, check the incorrect entry.

In order to correct the error, simply go back to the corresponding dialog by repeatedly clicking on the ‘Back’ button. Make sure that all designation rules are followed.

6 If no error message is displayed, click on the ‘Finish’ (’Fertig stellen’) button.

The empty code spreadsheet of the program POU ‘Main’ will open and the ‘Bus navigator’ window will show the bus devices of the allocated bus configurator project.

7 Click on the 'project tree' symbol in the symbol bar in order to open the project tree.



In the project tree, you will see the newly created project with the program POU ‘Main’ in the sub-tree 'Logical POUs'.

Figure 44: Automatically created project in the project tree

Now close ProSafety. You will now find yourself back in the ProMaster program. Click on the 'SafetyPLC - configuration (ProPLC)' in the 'Workspace' area.

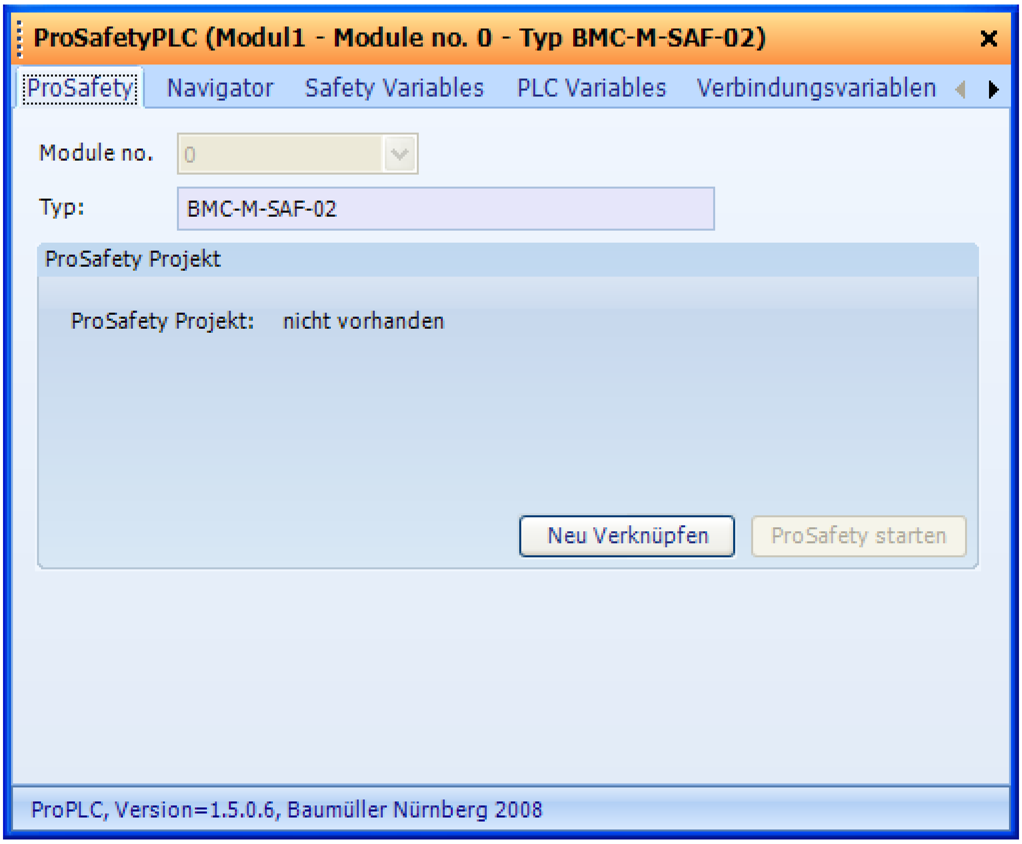


Figure 45: SafetyPLC - configuration

Click on ‘New link’ in order to link the ProMaster project to the ProSafety sample project ‘Safety02’. The Explorer will open:

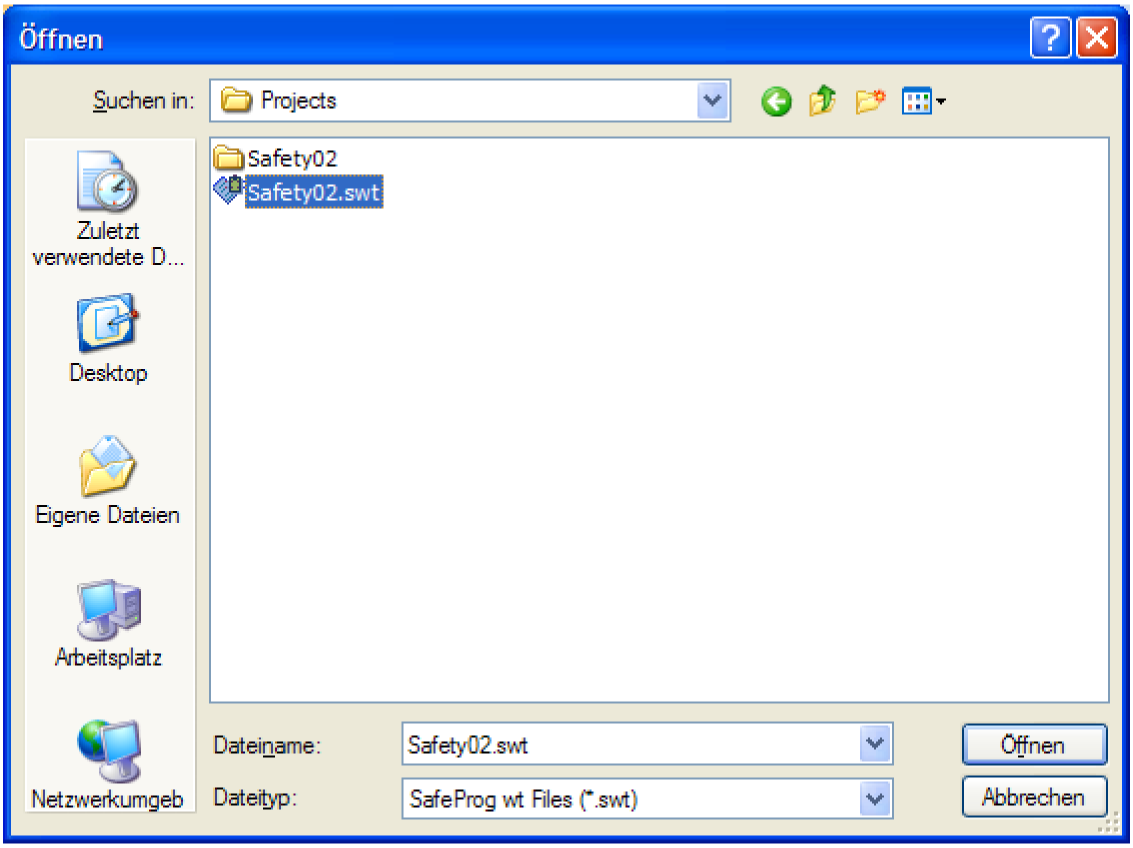


Figure 46: Opening the project

The safety project ‘Safety02’ can be found under ‘Baumüller\ProSafety_V01.xxb-dyy\projects’. Select ‘safety02.swt’ and open the file.

The ‘ProSafetyPLC’ window will open again with the linked safety project ‘Safety02’.

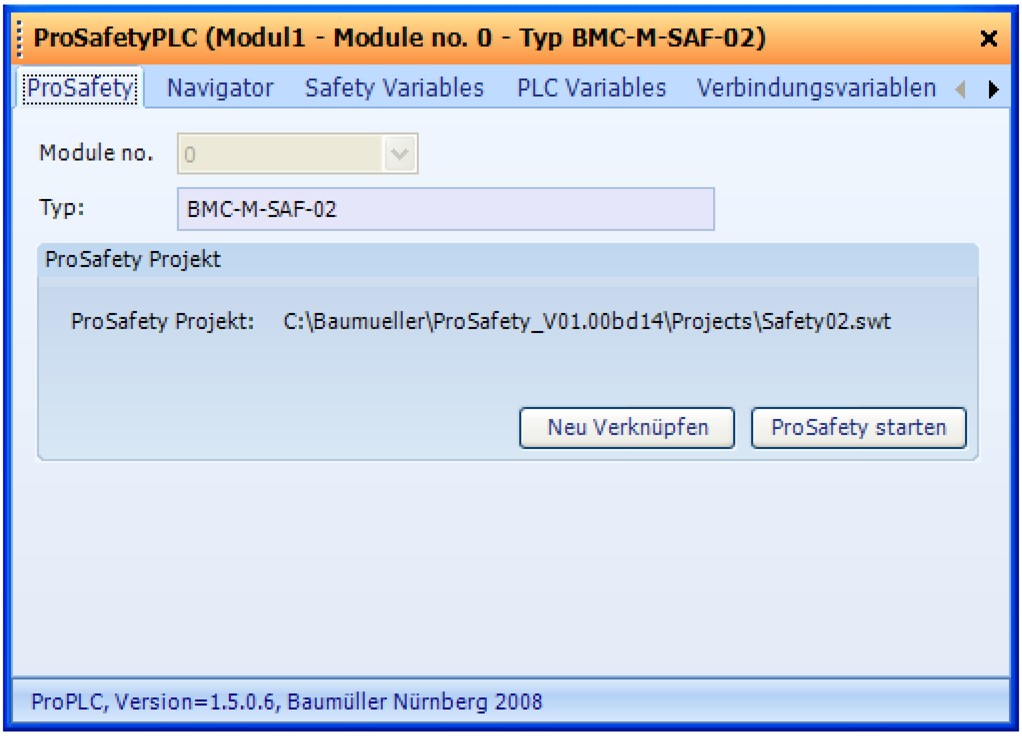


Figure 47: ProSafetyPLC - configuration with linked ProSafety project

Now hit <F9> in order to update the ProMaster project. By doing so, the current bus configuration of ProMaster will be overtaken in ProSafety. Then click on ‘Start ProSafety’ in the window.

The ‘ProSafety’ program starts with login dialog:

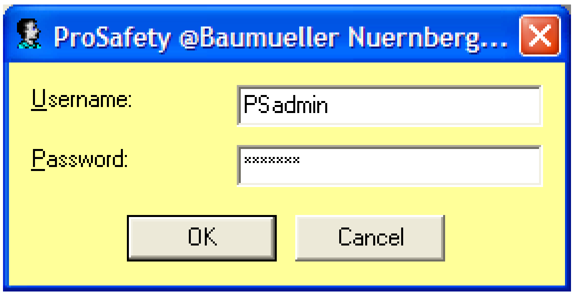


Figure 48: ProSafety login dialog

Enter your user name and password and hit ‘OK’.

The following notice will appear:

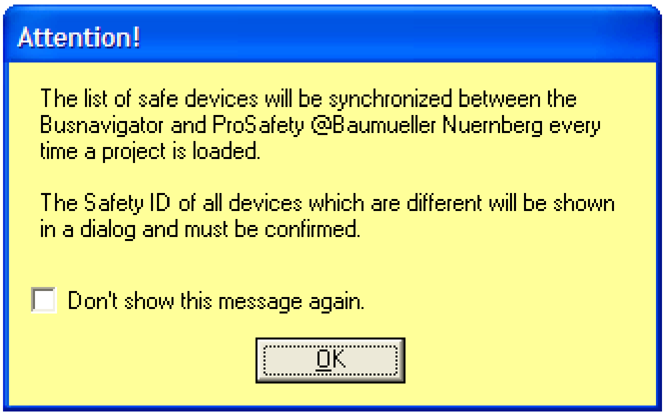


Figure 49: Notice

Acknowledge this notice by hitting ‘OK’. A confirmation of the safe devices changed in comparison with the general project template will then be requested. Here, you have to confirm the changes by clicking in the individual check boxes.

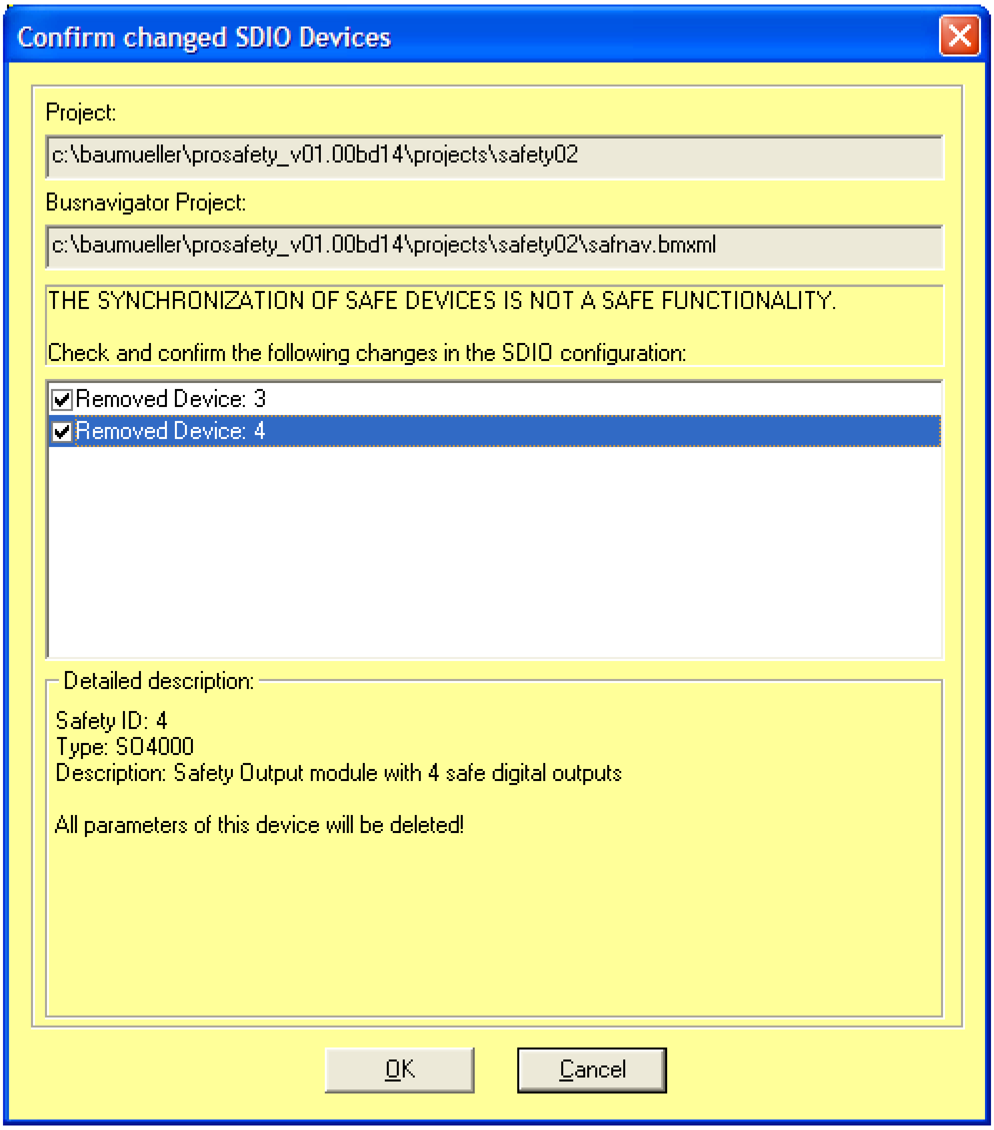


Figure 50: Confirmation of the safe devices changed

Click on 'OK'.

10.1.4.3 Phase 2: Parameterization of the safe devices

Once we have created the new project, we will want to parameterize the safe device of the bus configurator project according to our requirements in phase 2.

The bus configurator contains all modules specified in ProMaster, such as PLC, I/O modules and the devices present in the bus.

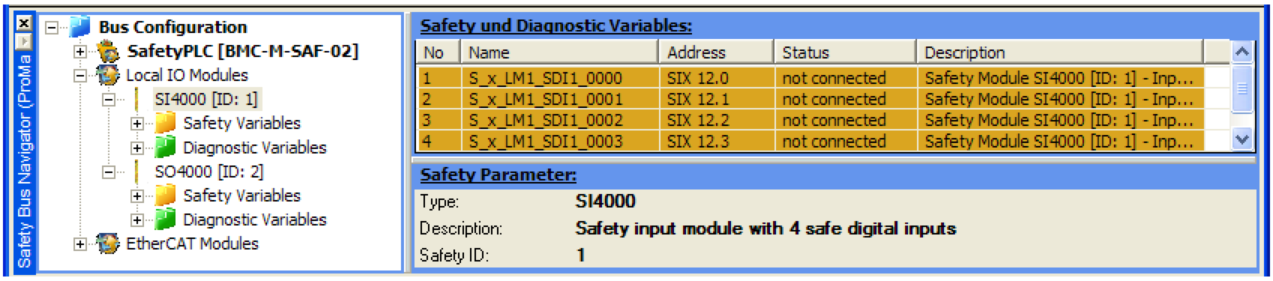


Figure 51: Bus configurator

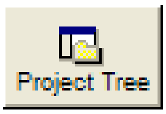
For parameterization, select all safe devices in the project tree in sequence and then enter the safe parameters for the devices in the 'Safety parameters' list. A description of the safe parameters for the PLC can be found in the chapter ▷Parameterization of the safe device parameterization editor) ◁ from page 150 onward, and in the respective operating manuals for other devices.

10.1.4.4 Phase 3: Integration of a library in the project tree

Since we are using function blocks which are from a library, we will have to incorporate the library in our sample project before we can use the safe function blocks need.

Proceed as follows:

1 If it is not already open, click on the 'Project tree' symbol in the symbol bar to open the project tree.



2 Highlight the file symbol of the 'Libraries' sub-tree:

3 Click on the highlighted file with the right mouse button in order to open the context menu and select the menu item 'Insert Library...'.

Alternatively, you can also select the menu item 'Project > Add Library'.

The 'Add Library' dialog will appear in both cases.

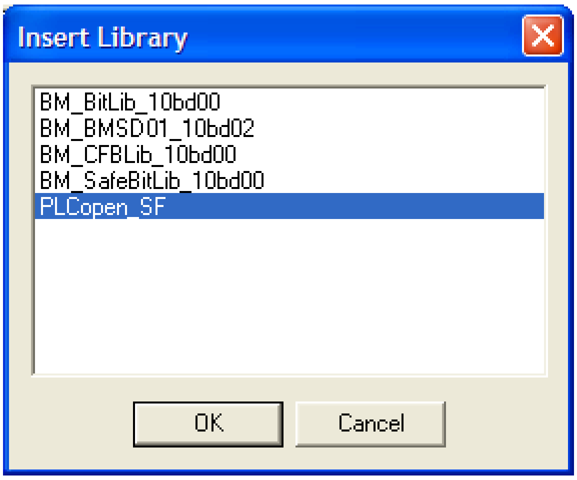


Figure 52: 'Add library' dialog

4 Select the library 'PLCopen_SF' and click 'OK'.

The library symbol will be inserted into the project tree:

The library has now been integrated into the sample project and the function blocks contained can be used for programming the code spreadsheets.

10.1.4.5 Phase 4: Developing the code

Once we have created the new project and integrated the library which contains the function blocks need, we will begin with phase 4, the development of the project code.

The steps in this section describe how you

- declare and insert the safe function block 'Emergency_Stop_01' and all related variables (see step 1 on ▹Page 76).
- declare and insert the safe function block 'SDoor_01' and all related variables (see step 2 on ▹Page 85).
- insert the safe AND connection ('AND_S' function) and connect the objects with the help of connection mode (see step 3 on ▹Page 93).
- insert and connect individual variables in the code (see step 4 on ▹Page 96).
- declare and insert the safe function block 'Contactor_Monitoring_01' and all related variables (see step 5 on ▹Page 99).
- declare and insert the safe function block 'Enable_Contactor_01' and all related variables (see step 6 on ▹Page 104).
- connect function blocks in connection mode (see step 7 on ▹Page 112).
- insert descriptive texts (commentaries) in the code spreadsheet (see step 8 on ▹Page 113).

A general description of the sample project with illustrations can be found in ▹Code spreadsheet of the POU 'Main': ◃ from page 62 onward.

Step 1 **DECLARING AND INSERTING THE SAFE FUCTION BLOCK 'Emergency_Stop_01' AND ALL RELATED VARIABLES**

We now want to declare and insert the safe EMERGENCY STOP function block 'Emergency _Stop_01' and all of the input and output variables necessary for it:

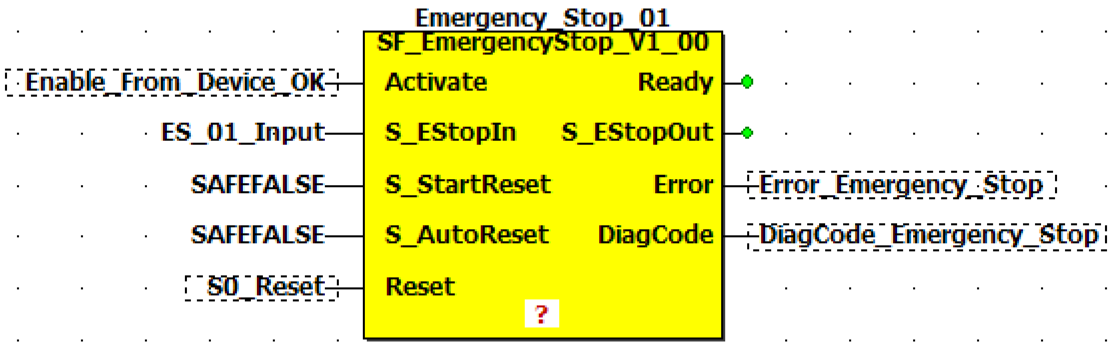


Figure 53: EMERGENCY STOP function block 'Emergency _Stop_01' to be declared and inserted

A detailed description of the safe function block 'SF_Emergency_Stop_V1_00' can be found in the handbook on components. You can open this handbook by place the cursor on the component and right clicking. The following window will open:

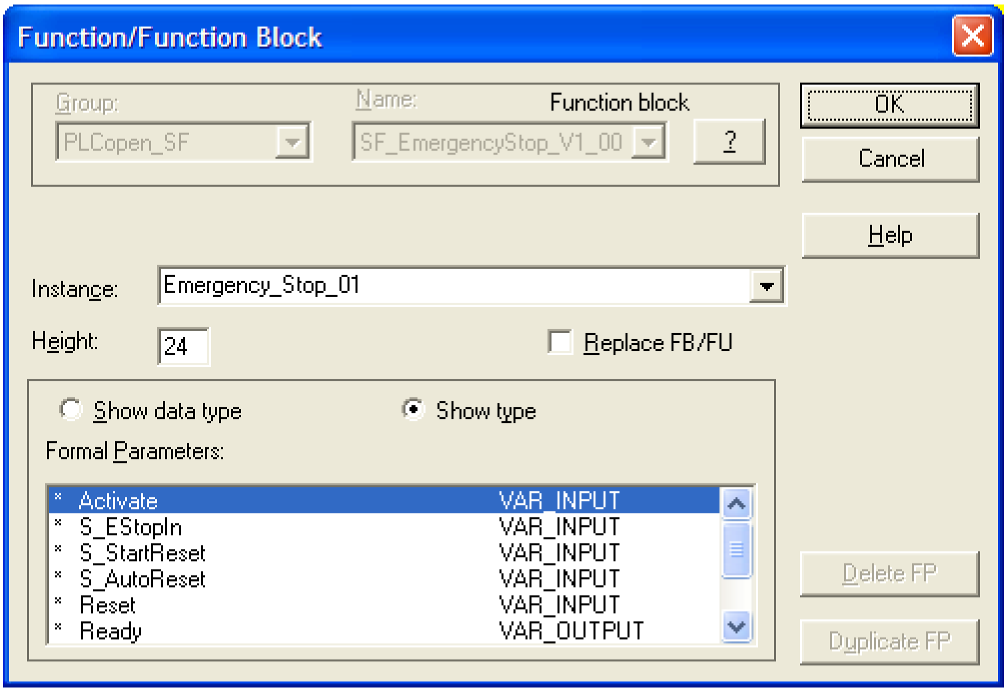
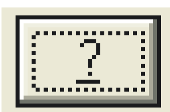


Figure 54: 'Function/function block' window

Click on the symbol

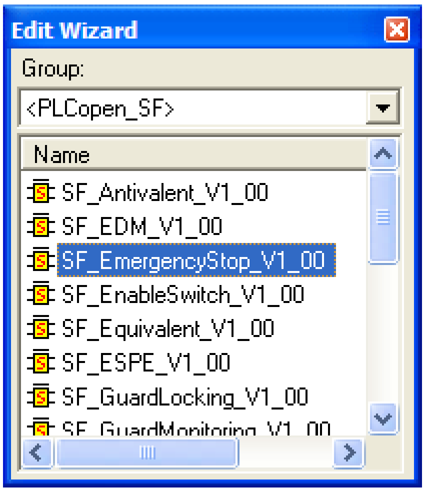


and you will receive a detailed description of the function block.

1 Make sure that the code spreadsheet of the program POU ‘Main’ is the active window. If this is not the case, click on the code spreadsheet (a cross will appear).

2 If the editor assistant has not yet been opened, click on the 'Editor assistant' symbol in the symbol bar:

3 Open the 'PLCopen_SF' group in the editor assistant (contains of the function blocks available in the library 'PLCopen_SF').



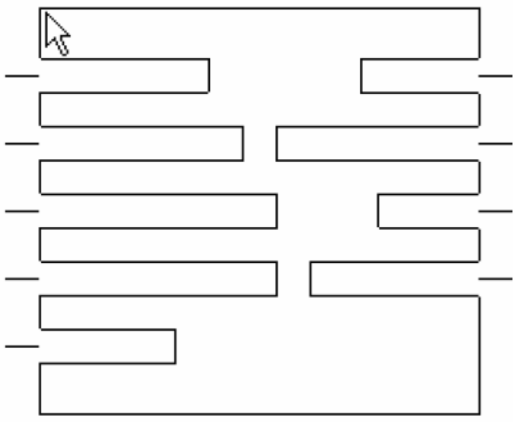
4 Click the entry 'SF_EmergencyStop_V1_00' with the left mouse button and hold the mouse button down.

5 Drag the cursor onto the code spreadsheet (which appears as a plus symbol) and release the mouse button.

The 'Variable' dialog will appear.

6 Enter the instance name ‘Emergency_Stop_01’ in the combo-box ‘Name’, select the group ‘NewGroup’ in the list field ‘Group’ and confirm the dialog with ‘OK’.

The schematic outline of the function block (we call it a “shadow”) will appear at the cursor.



7 Move the object in the upper left corner of the code spreadsheet and click with the left mouse button in order to deposit it.

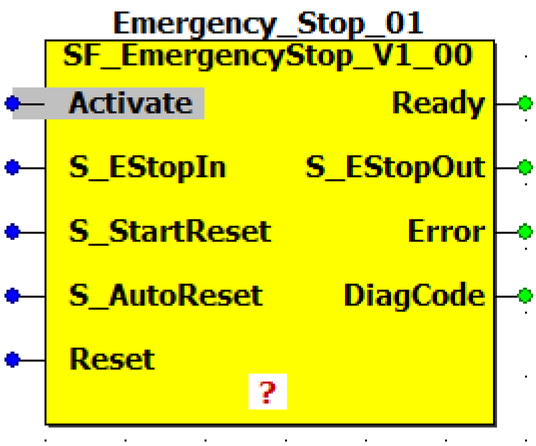


Figure 55: Code spreadsheet 'Main' with inserted function block 'Emergency_Stop_01'

We now want to insert a variable and connect it to the input ‘Activate’.

8 Double click on the blue connection point of the formal parameter 'Activate'.

The ‘Variable' dialog will appear.

Use the following settings in the ‘Variable' dialog for the parameters of the new variable (Name: 'Enable_From_Device_OK'):

Parameter	**Setting**
Area	Local
Name	Enable_From_Device_OK
Type	BOOL
Group	NewGroup
Use	VAR

9 Click on 'OK' in order to confirm the dialog 'Variable'.

The variable ‘Enable_from_Device_OK' is inserted at the ‘Activate’ input of the function block and the declaration of the variables is automatically added in the variable spreadsheet of the POU ‘Main’ (group 'NewGroup').

The dashed border of the ‘Enable_from_Device_OK' variables indicates that it is a non-safe variable (data type 'BOOL').

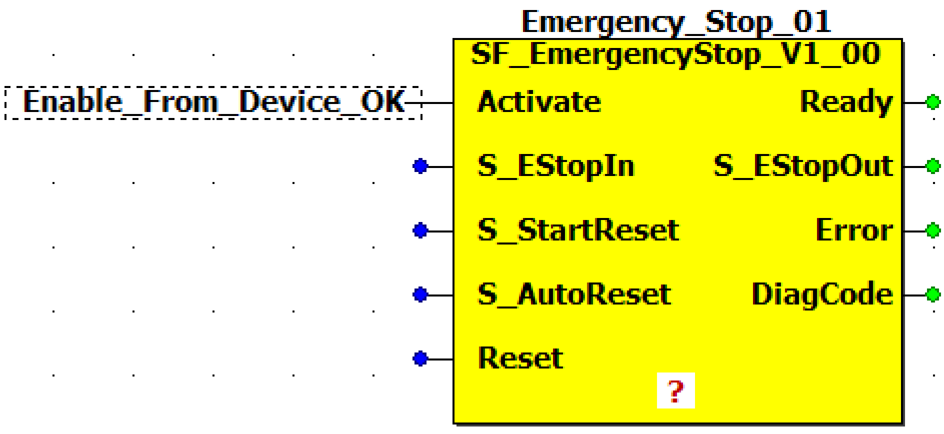


Figure 56: FB SF_EmergencyStop_V1_00 with non-safe variable (dashed border)

NOTICE!

If the variable cannot be inserted at the input of the function block, you will potentially have to relocate the function block in order to have more space at the left edge. To do this, highlight the function block by drawing a rectangle with the mouse (while holding the left mouse button down). Relocate the function block and let go of the mouse button in order to deposit it in the position desired.

We now want to insert and connect a global variable which is assigned to a terminal by dragging the signal into the code from the list of available terminals.

10 Open the directory of the module in the bus configurator from which you want to connect the global variable to the input of the function block.

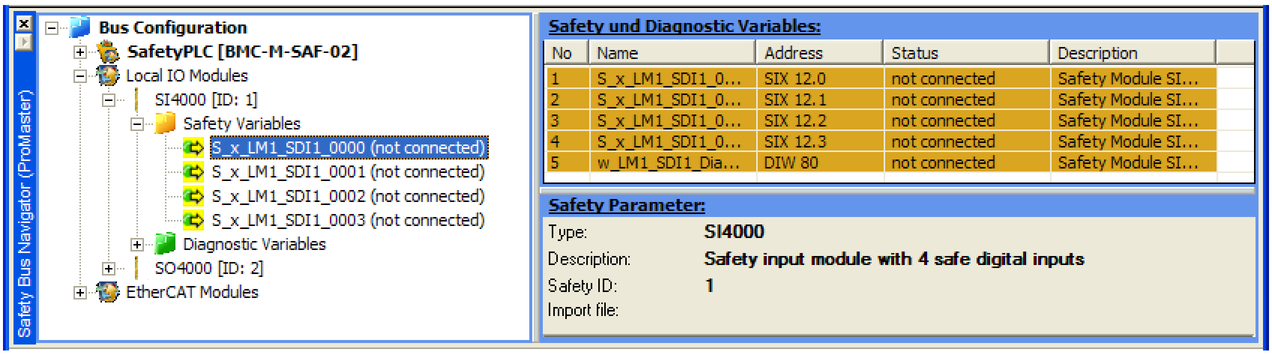


Figure 57: Bus configuration with safe device highlighted

11 Hold the left mouse button down and drag the variable into the code spreadsheet. You can drag the variable directly from the bus configurator or from the list of safety variables.

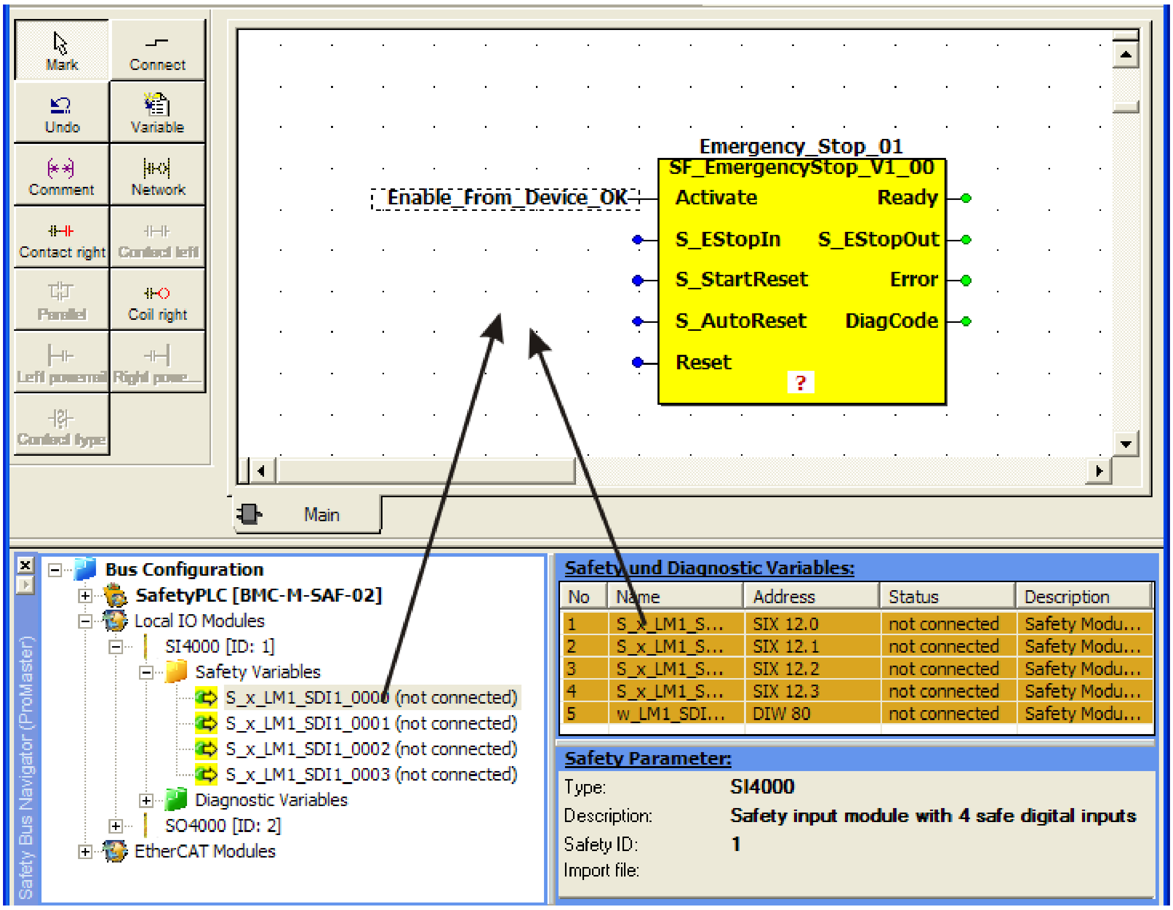


Figure 58: Dragging a variable from a terminal into the code

12 Release the mouse button. The dialog 'Variable' will appear. Here, you will have to declare the new variable which will be inserted into the code and linked to terminal SI4000 of the safe device.

Use the following settings in the dialog 'Variable' for the parameters of the new safe variables (Name: 'ES_01_Input'):

Parameter	Setting
Area	Global
Name	ES_01_Input
Type	SAFEBOOL
Group	NewGroup
Object type	Variable

13 Click on 'OK' in order confirm the dialog 'Variable'.

The variable will appear as a rectangle at the cursor.

14 Drag the variable onto the blue formal parameter 'S_EStopIn' as shown in the following illustration:

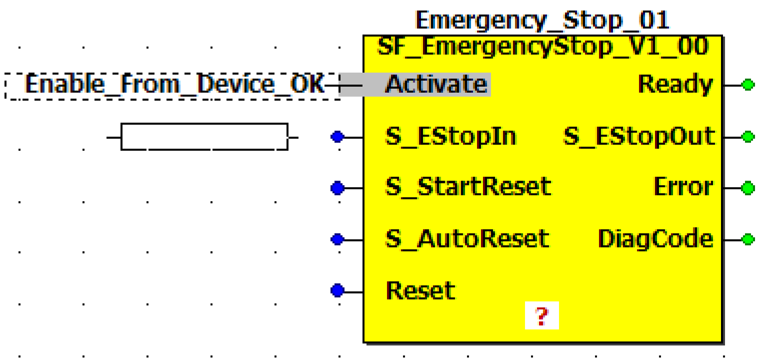


Figure 59: Dragging an object onto a formal parameter

15 Press the left mouse button in order to deposit the variable and create the connection.

The global variable 'ES_01_Input' will be inserted at the input 'S_EStopIn' of the function block, as shown in the following illustration. Since it is a safe variable (data type 'SAFEBOOL'), it will be inserted without a dashed border.

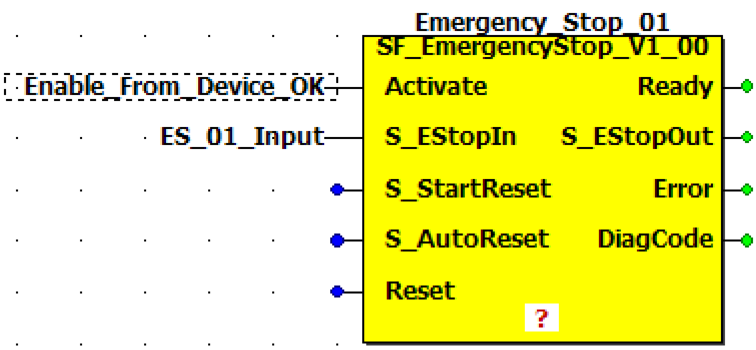


Figure 60: FB SF_EmergencyStop_V1_00 with the global variable 'NA_01_Input'

The name of the connected variables in the directory of the variables of the related terminals will be entered into the list of safety variables of the bus navigator.

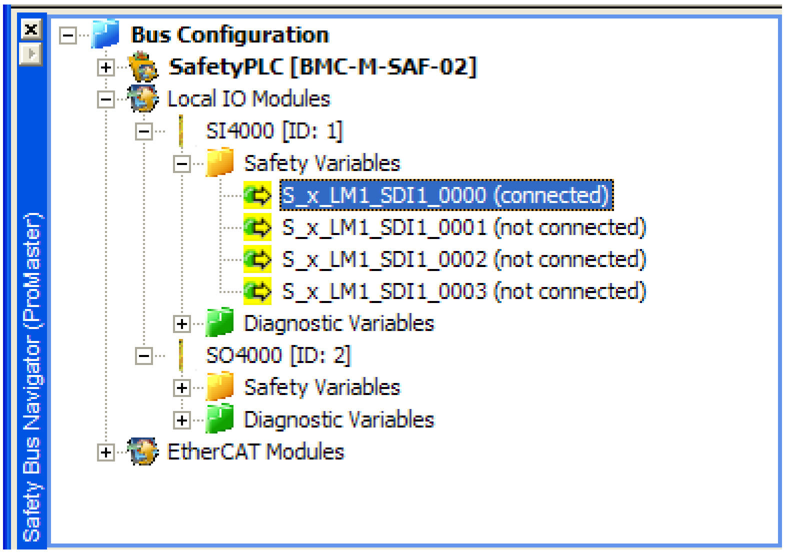


Figure 61: Name of connected variables in the bus navigator

NOTICE!

If you have accidentally connected a terminal with an incorrect variable, you can delete this connection. To do this, click on the variable in the code with the right mouse button and select ‘Go to definition of ...(variable name)’ in the context menu.

The variable spreadsheet in which the variable is located will open and the corresponding variable will be highlighted in it.

Figure 62: Global variable spreadsheet

In order to disconnect the connection, select the menu item 'Edit > Disconnect terminal'.

We now want to insert constants and connect them with the inputs of the function blocks.

16 Double click on the blue connection point of the formal parameter 'S_StartReset'.

The 'Variable' dialog will appear.

17 Since we want to insert a constant, you will have to activate the option 'Constant' and select the entry 'SAFEFALSE' in the list field 'Name'.

18 Click on 'OK' in order to confirm the dialog 'Variable'.

The constant 'SAFEFALSE' will be inserted at the output 'S_StartReset' of the function block.

19 Repeat steps 16 through 18 for the input 'S_AutoReset' of the function block.

You code should now look like this:

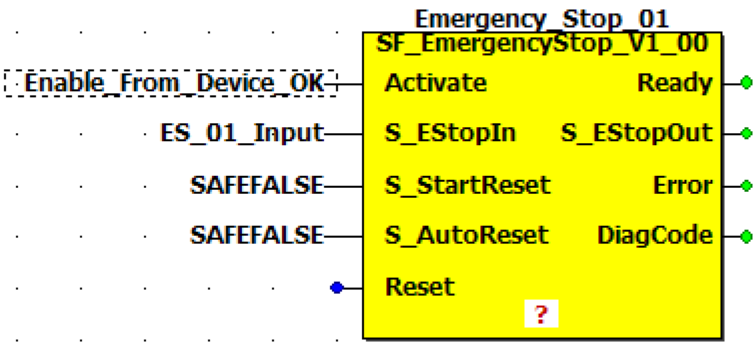


Figure 63: FB SF_EmergencyStop_V1_00 with constants inserted

We now want to insert a further variable and connect it to the input 'Reset' of the function block.

20 Double click on the blue connection point of the formal parameter 'Reset'.

The 'Variable' dialog will appear.

Use the following settings for the parameters of the new variable (Name: 'S0_Reset') in the 'Variable' dialog:

Parameter	Setting
Area	Local
Name	S0_Reset
Type	BOOL
Group	NewGroup
USe	VAR

21 Click on 'OK' in order to confirm the 'Variable' dialog and drag the variable onto the blue formal parameter 'Reset'.

22 Press the left mouse button in order to deposit the variable and create the connection.

The name of the connected variables in the directory of the variables of the related terminal will be entered into the list of the safety variables of the bus navigator.

We now want to insert and combine two local variables.

23 Double click on the green connection point of the formal parameter 'Error'.

The 'Variable' dialog will appear.

Use the following settings for the parameters of the new variable (Name: 'Error_Emergency_Stop') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Local
Name	Error_Emergency_Stop
Type	BOOL
Group	NewGroup
Use	VAR

24 Click on 'OK' in order to confirm the 'Variable' dialog.

25 Double click on the green connection point of the formal parameter 'DiagCode'.

The 'Variable' dialog will appear.

Use the following settings for the parameters of the new variable (Name: 'DiagCode_Emergency_Stop') in the 'Variable' dialog:

Parameter	Einstellung
Area of validity	Local
Name	DiagCode_Emergency_Stop
Type	WORD
Group	NewGroup
Use	VAR

Your code should now look like this:

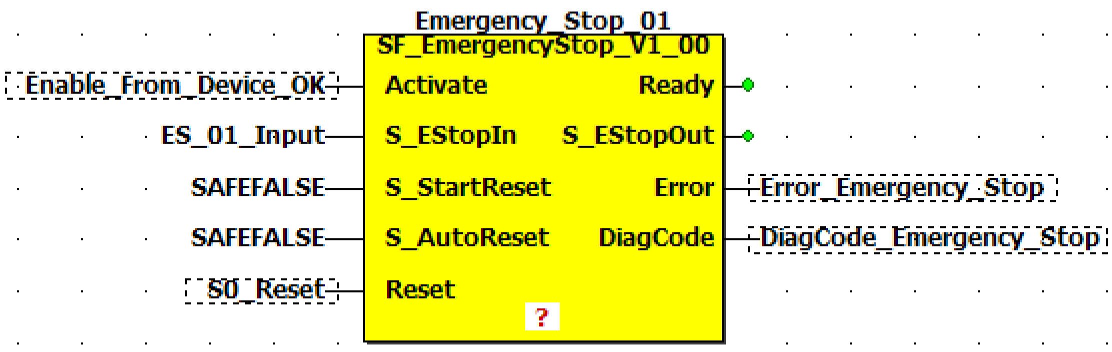


Figure 64: FB SF_EmergencyStop_V1_00 with outputs inserted

Step 2 **DECLARING AND INSERTING THE SAFE FUNCTION BLOCK 'SDOOR_01' AND ALL RELATED VARIABLES**

We now want to declare the function block for the guard door 'SDoor_01' and all necessary input and output variables related to it:

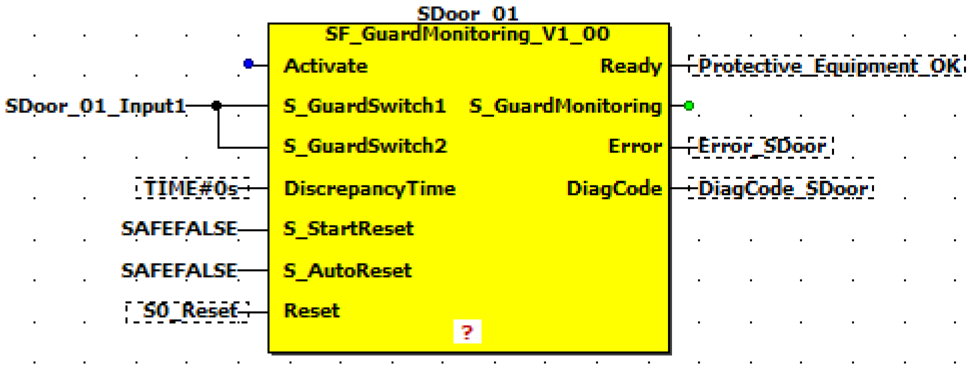
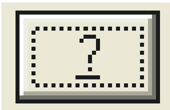


Figure 65: Preview of the function block for the guard door 'SDoor_01' which is to be declared and inserted

A detailed description of the safe function block 'SF_GuardMonitoring_V1_00' can be found in the handbook on the module. You can open this by going to the module with the cursor and clicking at the right. The 'Function/function block' window will open:

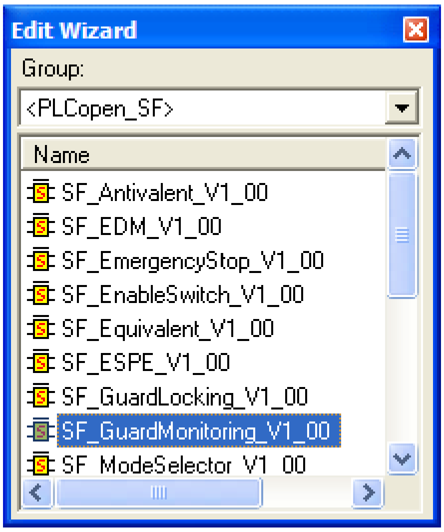
Click on the symbol



and you will receive a detailed description of the function block.

1 Activate the editor assistant by clicking on a free position in the code spreadsheet with the left mouse button.

2 In the editor assistant, click on the entry 'SF_GuardMonitoring_V1_00' with the left mouse button and hold the mouse button down.



3 Drag the cursor onto the code spreadsheet (which will appear as a plus symbol) and release the mouse button.

   The 'Variable' dialog will appear.

4 Enter the instance name 'SDoor_01' in the combo-box 'Name' and confirm the dialog with 'OK'.

   The schematic outline of the function block will appear at the cursor.

5 Drag the object into the code under the EMERGENCY STOP function block and push the left mouse button to deposit it.

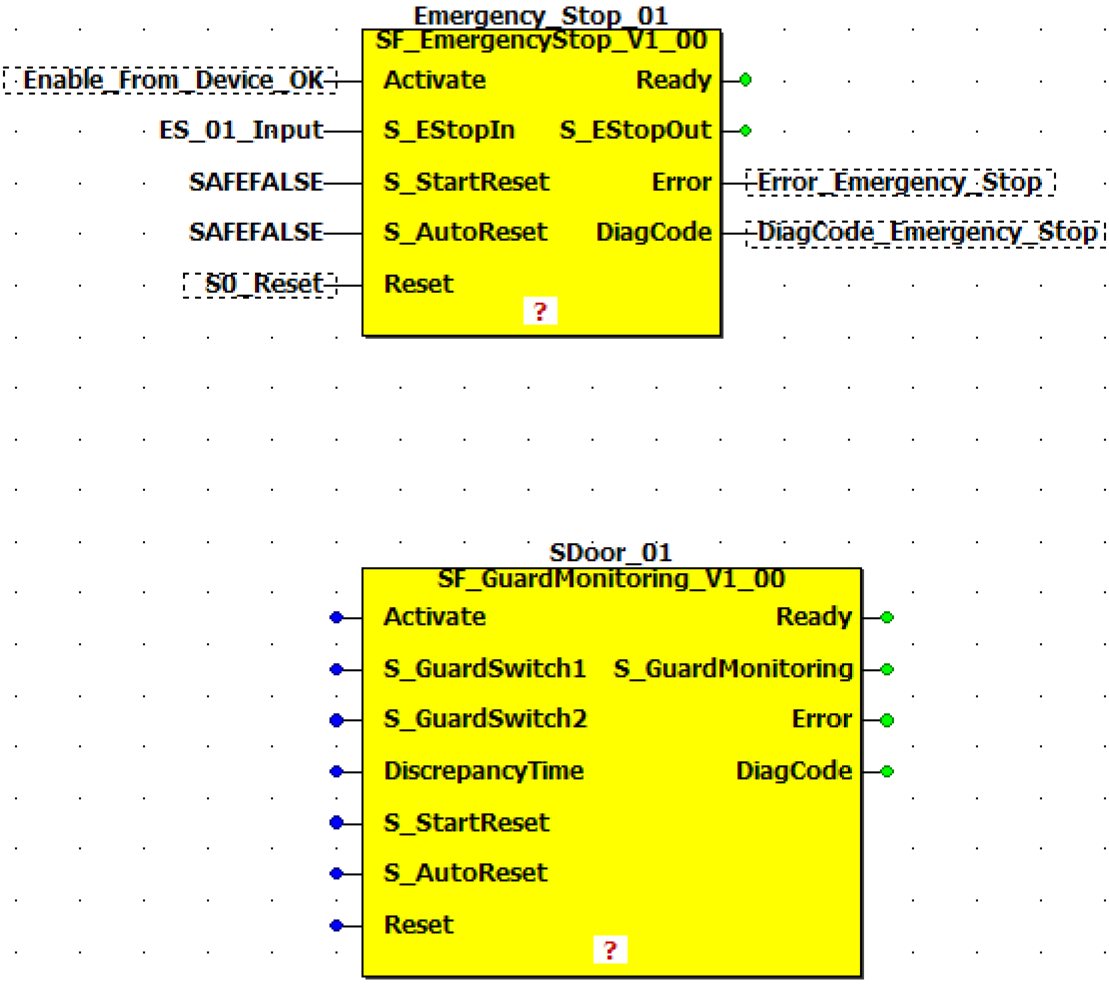


Figure 66: 'Main' code spreadsheet with function block 'SDoor_01' inserted.

We now want to insert and connect a global variable, which is assigned to a terminal by dragging the signal from the list of available terminals into the code.

6 Search for the desired I/O terminal SI4000 in the directory tree of the bus configurator and open the subdirectory with the variables.

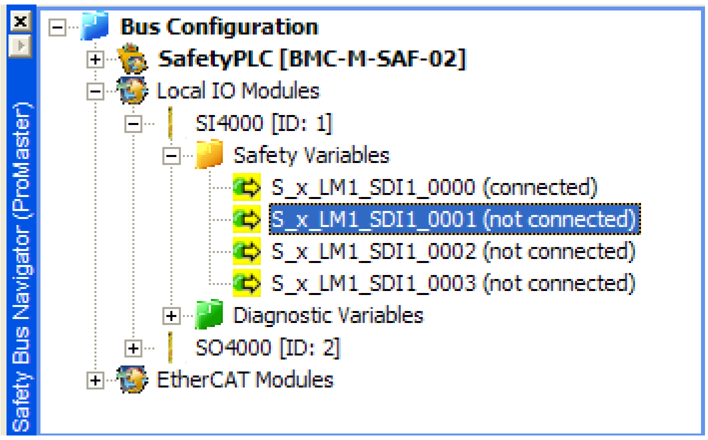


Figure 67: Drag the variable from the bus configurator onto the code spreadsheet

7 Click on the variable of the terminal SI4000 and with the left mouse button and hold the button down while you drag the variable into the code

8 Release the mouse button. The 'Variable' dialog will appear. Here, you will have to declare the new variable which is to be inserted into the code and linked with the terminal SI4000 of the safe device.

Use the following settings for the parameters of the new variable (Name: 'SDoor_01_Input1') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Global
Name	SDoor_01_Input1
Type	SAFEBOOL
Group	NewGroup
Object type	Variable

9 Click on 'OK' in order to confirm the 'Variable' dialog.

The variable will appear as a rectangle at the cursor.

10 Drag the variable on onto the blue formal parameter 'S_GuardSwitch1', as shown in the following illustration:

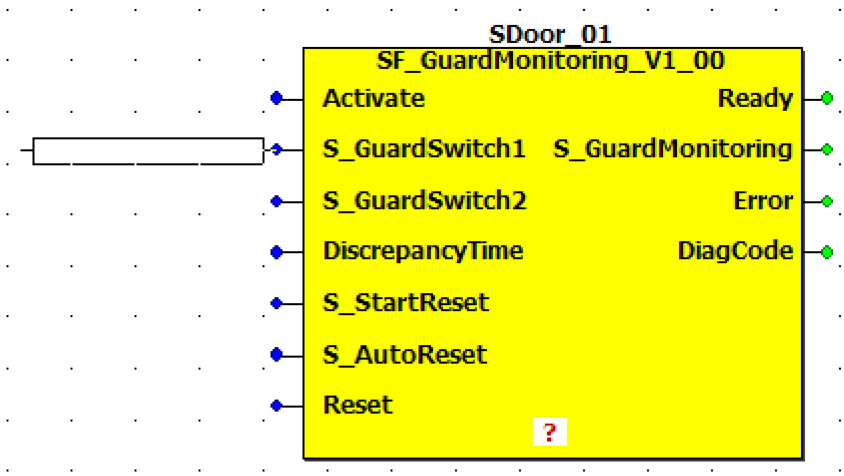


Figure 68: Dragging an object onto a formal parameter

11 Press the left mouse button to deposit the variable and create the connection.

The global variable 'SDoor_01_Input1' will be inserted at input 'SGuardSwitch1' of the function block, as shown in the following illustration. Since it is a safe variable (data type 'SAFEBOOL'), it will be inserted without a dashed border.

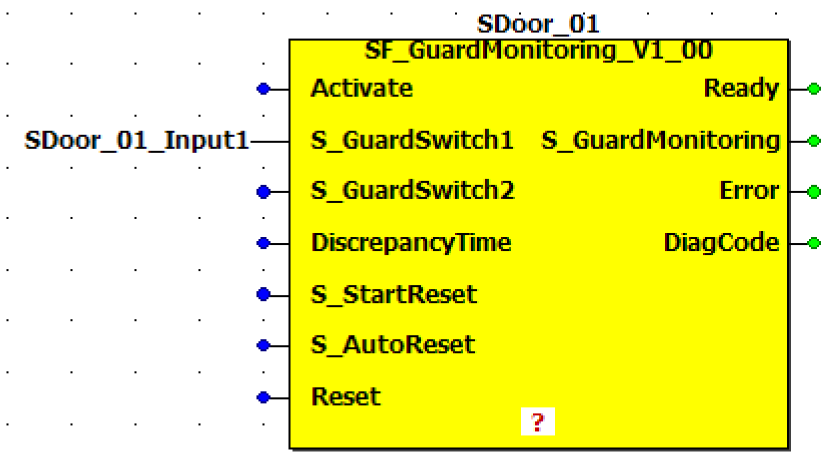


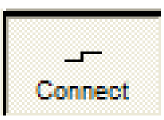
Figure 69: FB SF_GuardMonitoring_V1_00 with variable SDoor_01_Input1

We now want to additionally connect the variable 'SDoor_01_Input1', which has just been inserted, to the 'S_GuardSwitch2' input by using connection mode.

12 Click on the variable 'SDoor_01_Input1' and hold the mouse button down.

13 Reposition the variable a bit to the left to make room for the connection.

14 Click on the ‘Connect’ symbol in the symbol bar:



15 Click on the blue connection point of the 'SGuardSwitch2' input (starting point of the connection line) at the function block for the guard door.

16 Put the cursor on the connection line between the variable 'SDoor_01_Input1' and the 'SGuardSwitch1' input of the function block and click once in order to connect the line. The layout of the line will automatically be conducted by the program and the two objects will be connected.

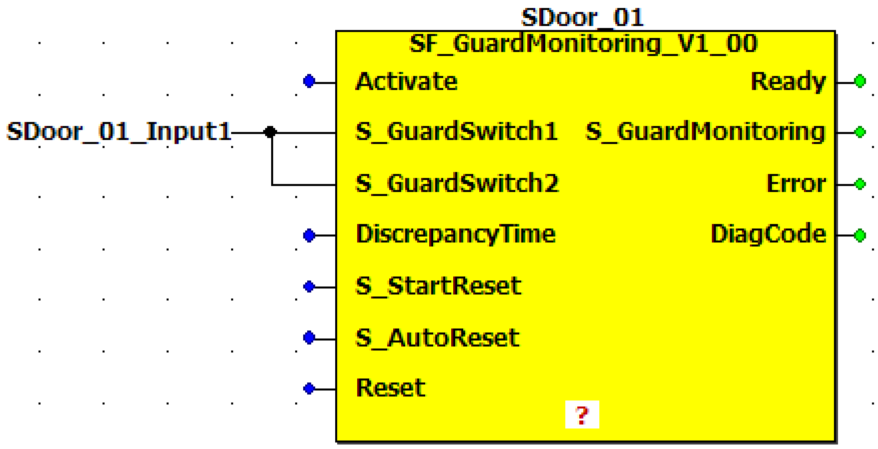


Figure 70: FB SF_GuardMonitoring_V1_00 with variable SDoor_01_Input1 at two inputs

We now want to insert constants and connect them to the inputs of the function block.

17 Double click on the blue connection point of the formal parameter 'Discrepancy Time'.

The 'Variable' dialog will appear.

Enter the following parameters for the new constant 'TIME#0s'. Then click 'OK' to confirm the 'Variable' dialog.

Parameter	**Setting**
Area of validity	Constant
Name	TIME#0s

18 Double click on the blue connection point of the formal parameter 'S_StartReset'.

The 'Variable' dialog will open again. Since we want to insert an additional constant, you will have to activate the 'Constants' option and select the entry 'SAFEFALSE' in the list field 'Name'. Then confirm the 'Variable' dialog with 'OK'.

The constant 'SAFEFALSE' will be inserted at the output 'S_StartReset' of the function block.

19 Repeat step 18 for the input 'S_AutoReset' of the function block in order to insert a 'SAFEFALSE' constant at this input.

We now want to insert a variable which is already being used in our sample project at the 'Reset' input of the function block.

20 Double click on the blue connection point of the input 'Reset'. The 'Variable' will open. Activate the option 'Local' and select the variable 'S0_Reset' from the list field 'Name'. Then confirm the 'Variable' dialog with 'OK'.

The function block should now look like this:

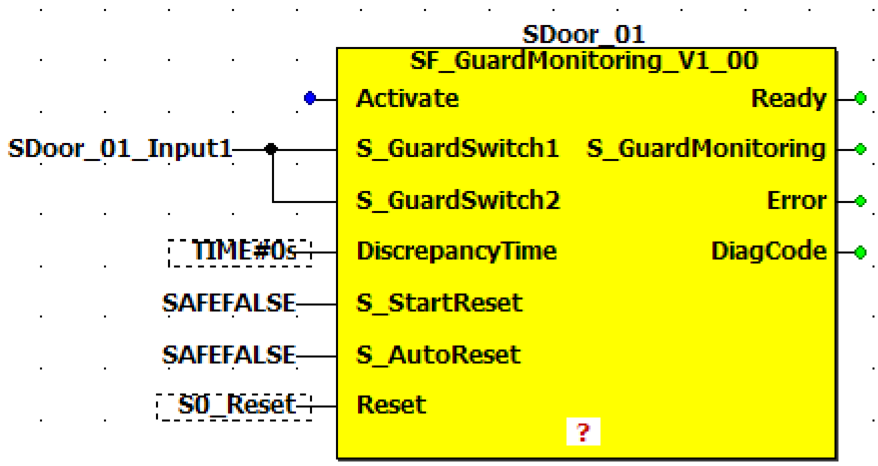


Figure 71: FB SF_GuardMonitoring_V1_00 with inputs set

We now want to insert and connect three more variables to the outputs 'Ready', 'Error' and 'DiagCode' of the function block for the guard door.

21 Double click on the green connection point of the formal parameter 'Ready'.

The 'Variable' dialog will appear.

Use the following settings for the parameters of the new variable (Name: 'Protective_Equipment_OK') in the 'Variable' dialog and then confirm the dialog with ‘OK’.

Parameter	**Setting**
Area of validity	Local
Name	Protective_Equipment_OK
Type	BOOL
Group	NewGroup

22 Double click on the green connection point of the formal parameter 'Error'.

The 'Variable' dialog will appear.

Declare the variable 'Error_SDoor' as listed in the following table and then confirm the dialog with 'OK'.

Parameter	**Setting**
Area of validity	Local
Name	Error_SDoor
Type	BOOL
Group	NewGroup

23 Double click on the green connection point of the formal parameter 'DiagCode'.

The 'Variable' dialog will appear.

Declare the variable 'DiagCode_SDoor' as listed in the following table and then confirm the dialog with 'OK'.

Parameter	**Setting**
Area of validity	Local
Name	DiagCode_SDoor
Type	WORD
Group	NewGroup

Your code should now look like this:

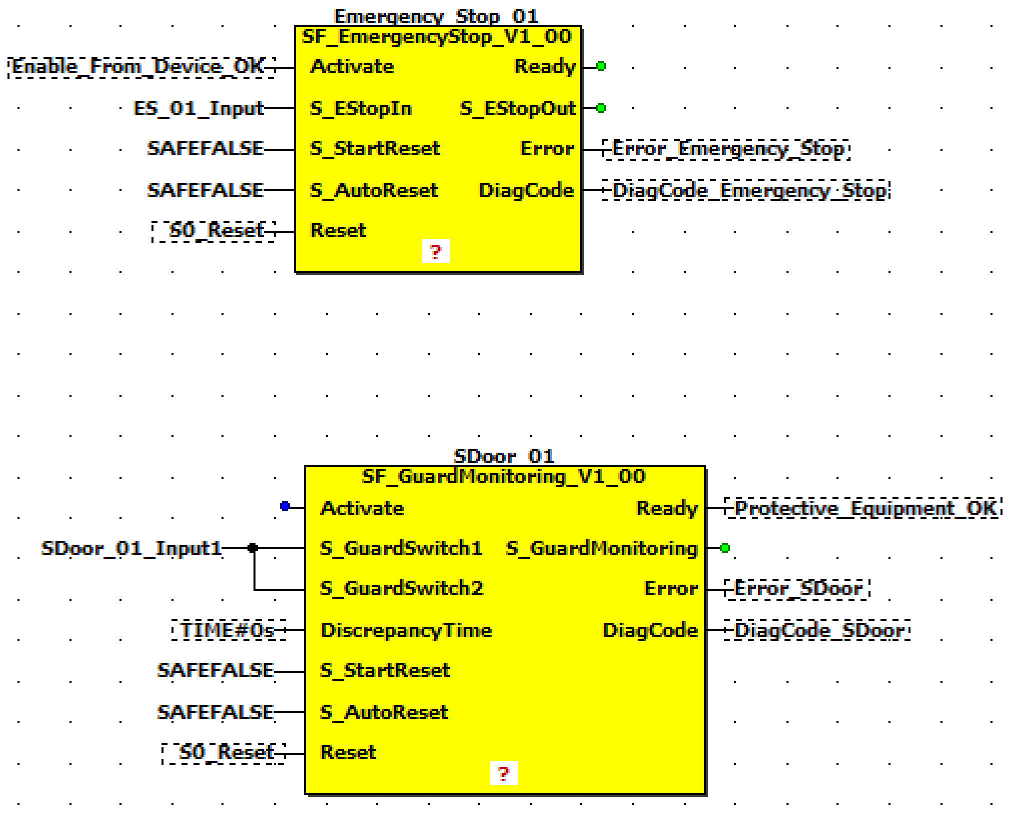


Figure 72: Code spreadsheet

24 Click on 'Save' in the symbol bar to save the code spreadsheet:

Step 3 **INSERTING THE SAFE FUNCTION 'AND_S' AND CONNECTING OBJECTS IN CONNECTION MODE**

We now want to connect the output 'S_EStopOut' of the EMERGENCY STOP function block and the 'S_GuardMonitoring' output of the function block for the guard door to an 'AND_S' function.

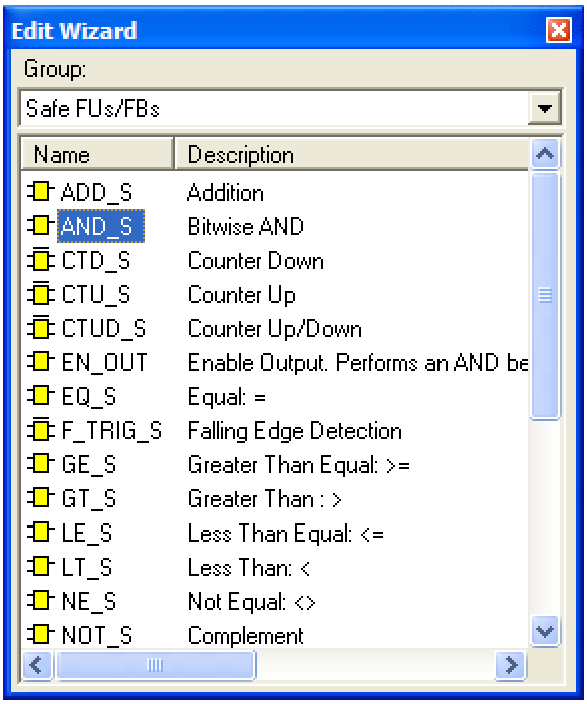
The 'Ready' output of the EMERGENCY STOP function block must then be connected with the 'Activate' input of the function block for the guard door.

CAUTION!

Since we are connecting safe inputs and outputs (data type 'SAFEBOOL'), only the safe function 'AND_S' may be used for the connection.

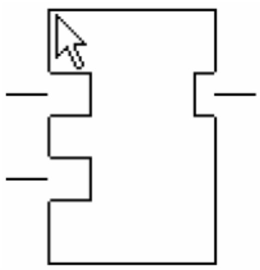
1 Activate the editor assistant by clicking on a free position in the code spreadsheet with the left mouse button. If the editor assistant is not open yet, click on the 'Editor assistant' symbol in the symbol bar:

2 Open the group 'Safe FUs/FBs' in the editor assistant. It contains all accessible safe functions and function blocks.



3 Click with the left mouse button on the entry 'AND_S' and hold the mouse button down. Drag the function into the code.

The schematic outline of the function will appear at the cursor.



4 Push the object to the right next to the EMERGENCY STOP function block (see the following illustration) and press the left mouse button to deposit the function.

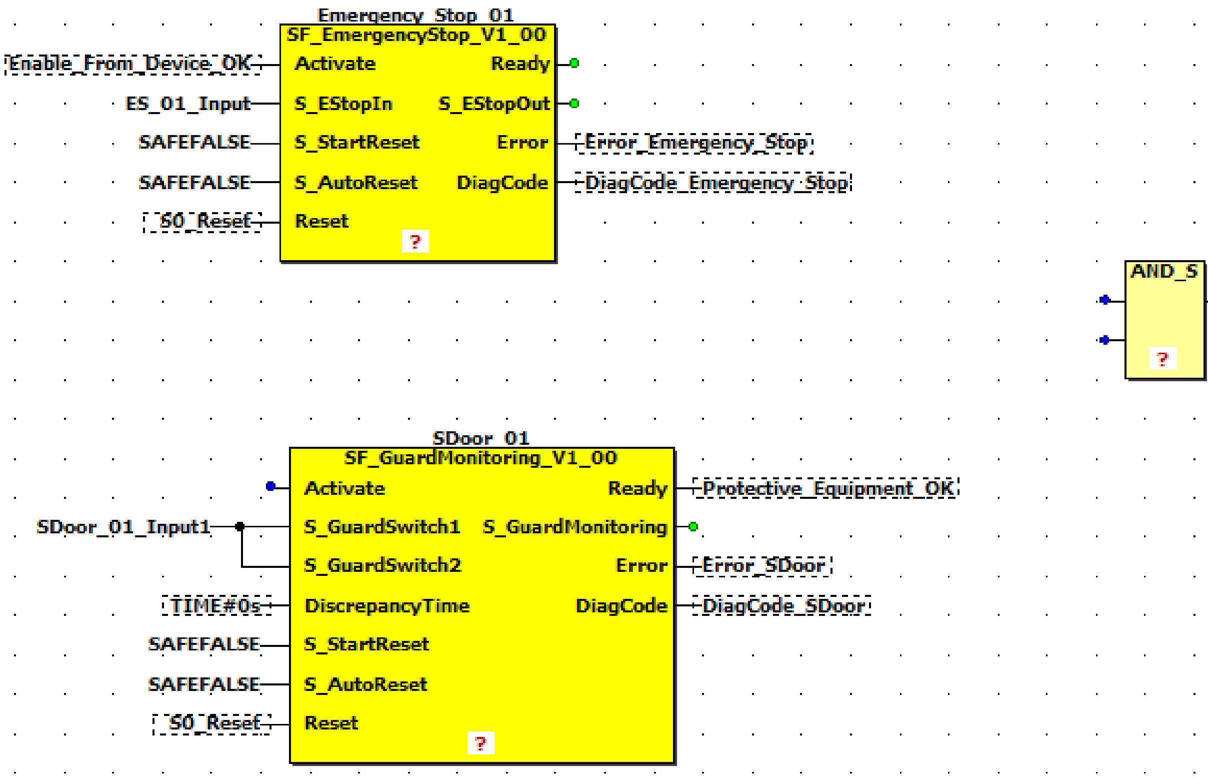
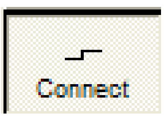


Figure 73: Spreadsheet with FB AND_S

We now want to connect the inputs and outputs by using connection mode.

5 Click on the 'Connect' symbol in the symbol bar:



6 Click on the green connection point of the 'Ready' output at the EMERGENCY STOP function block (starting point of the connection line).

7 Move the mouse to the right and click once to set a corner point:

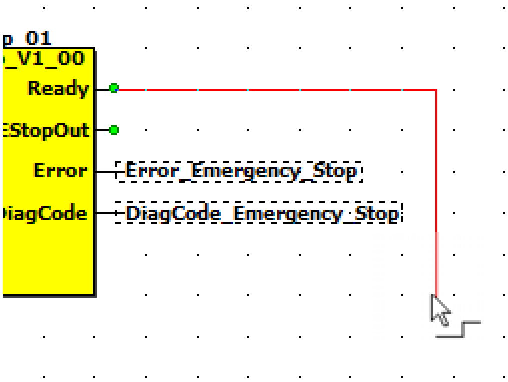


Figure 74: Dragging connection lines

8 Guide the connection line to the blue connection point out the input 'Activate' at the function block for the guard door and click on the formal parameter in order to end the connection line:

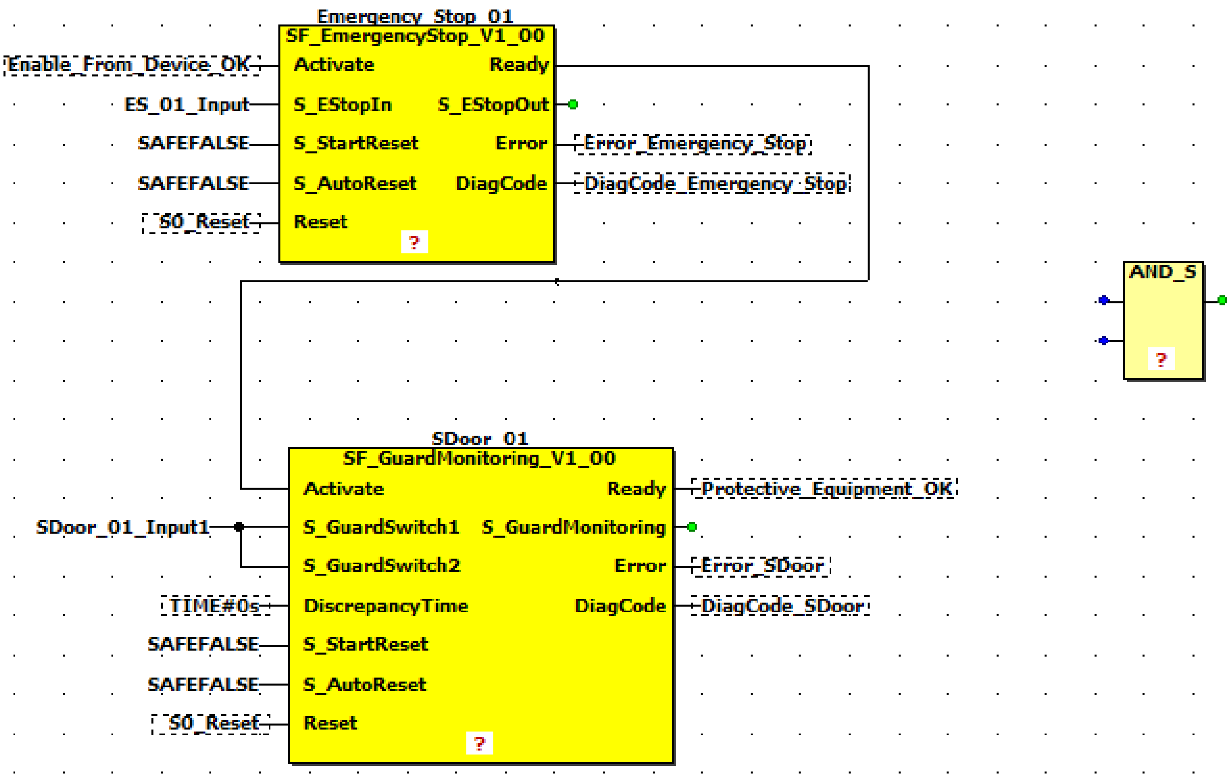


Figure 75: Spreadsheet with connection lines

9 Click on the green connection point on the 'S_GuardMonitoring' connection point of the function block for the guard door, drag the mouse to the lowermost blue connection point of the safe function 'AND_S' and click once to end the line.

10 Click on the green connection point of the output 'S_EStopOut' at the EMERGENCY STOP function block (starting point of the connection line).

11 Drag the mouse to the upper blue connection point of the safe function 'AND_S' and click again in order to end the line. The layout of the line will automatically be conducted by the program and the two objects will be connected.

The connections in your code spreadsheet should now look like this:

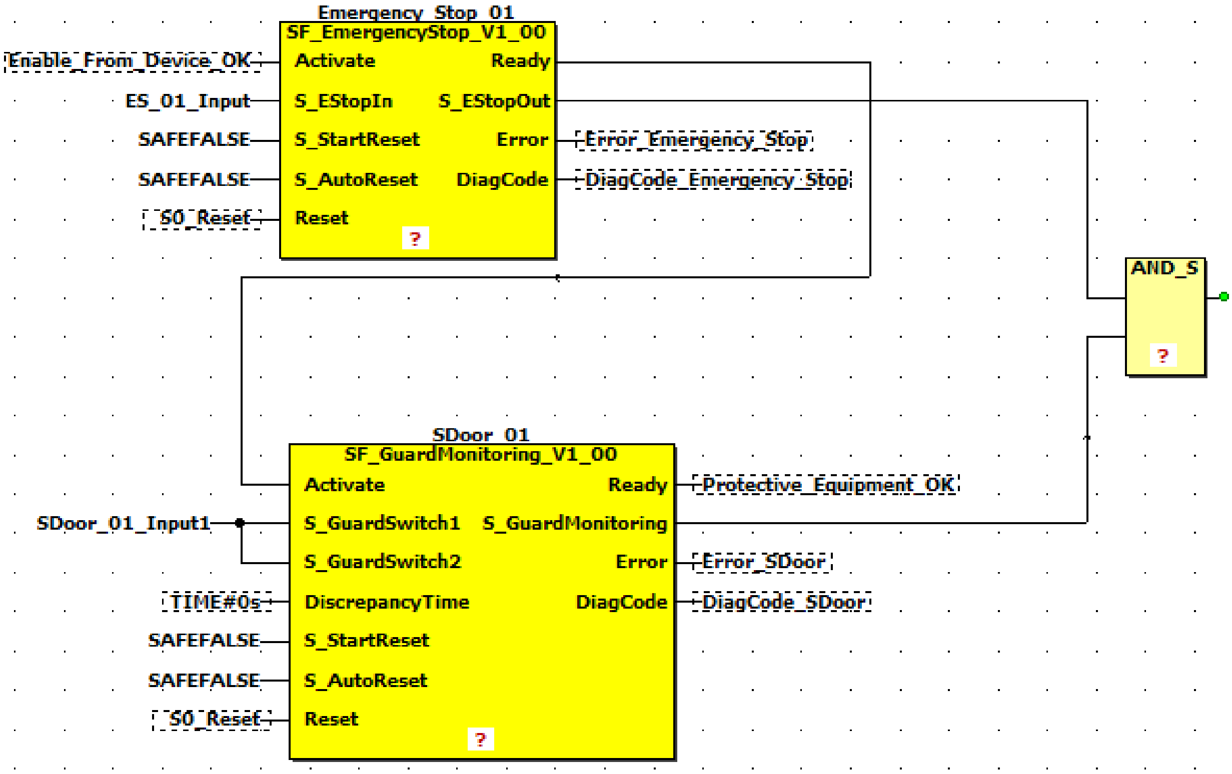


Figure 76: Spreadsheet with connection lines

12 Save the code spreadsheet by clicking on the 'Save' symbol in the symbol bar.

Step 4 **INSERTING AND CONNECTING INDIVIDUAL VARIABLES**

We now want to declare, insert and connect an individual global variable ('Valve_01') in the code spreadsheet.

Click on the 'Mark' symbol in the symbol bar to activate highlighting mode:

Highlight the desired safe output variable in the bus navigator.

1 Highlight the variable in the directory tree.

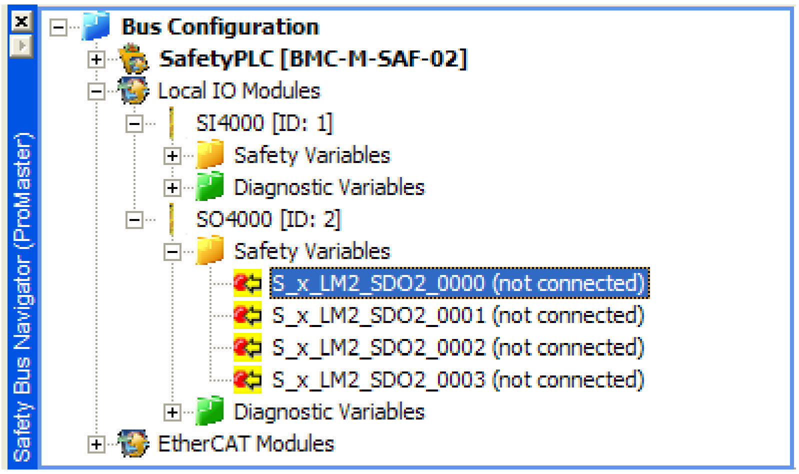


Figure 77: Bus configurator

2 Hold the left mouse button down and drag the variable S_x_LM4_SDO2_0000 into the code.

3 Release the mouse button. The 'Variable' dialog will appear. Here, you will have to declare the new variable which is to be inserted into the code and linked to the input of the safe device.

Use the following settings for the parameters of the new safe variable (Name: 'Valve_01') in the ‘Variable’ dialog:

Parameter	**Setting**
Area of validity	Global
Name	Valve_01
Type	SAFEBOOL
Group	NewGroup
Object type	Variable

4 Click on 'OK' in order to confirm the 'Variable’ dialog.
The variable will appear as a rectangle at the cursor.

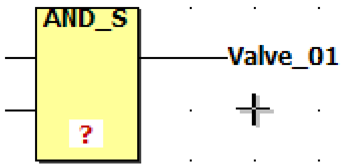
5 Drag the variable onto the green connection point (output) of the 'AND_S' function.

6 Click on the variable 'Valve_01' and hold the mouse button down.

7 Move the variable a bit to the right.

We now want to insert the local variable 'Enable_Contactor_Monitoring' and connect it to the output of the safe function 'AND_S' in connection mode

8 Click on the spreadsheet with the left mouse button to set an insertion mark at the position shown below. The variable 'Enable_Contactor_Monitoring' should be inserted here.

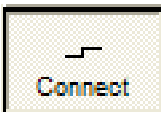


9 Select the menu point 'Objects > Variables' or press <F5> in order to insert a new variable.
The 'Variable' dialog will appear.

Use the following settings for the parameters of the new safe variable (Name: 'Enable_Contactor_Monitoring') in the ‘Variable’ dialog and confirm the ‘Variable’ dialog with ‘OK’.

Parameter	Setting
Area of validity	Local
Name	Enable_Contactor_Monitoring
Type	SAFEBOOL
Group	NewGroup
Use	VAR

10 Click on the 'Connect' symbol in the symbol bar:



11 Click on the blue connection point of the variable 'Enable_Contactor_Monitoring' (starting point of the connection line).

12 Position the cursor on the connection line between the function 'AND_S' and the variable 'Valve_01' and click one in order to end the line. The layout of the line will automatically be conducted by the program and the two variables will be connected.

Your code should now look like this:

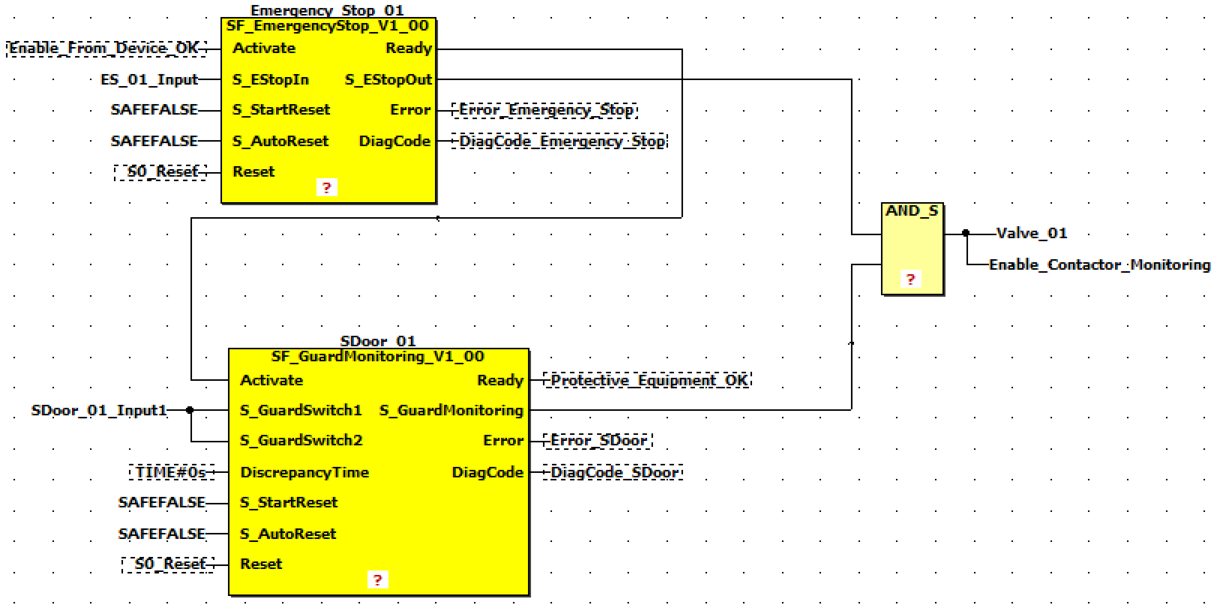


Figure 78: Spreadsheet: The 'Enable_Contactor_Monitoring' variable at the output of safe function 'AND_S'.

Step 5 **DECLARING AND INSERTING THE SAFE function block 'CONTACTOR_MONITORING_01' AND ALL RELATED VARIABLES**

We now want to declare the function block for the guard monitoring 'Contactor_Monitoring' and all input and output variables necessary for it:

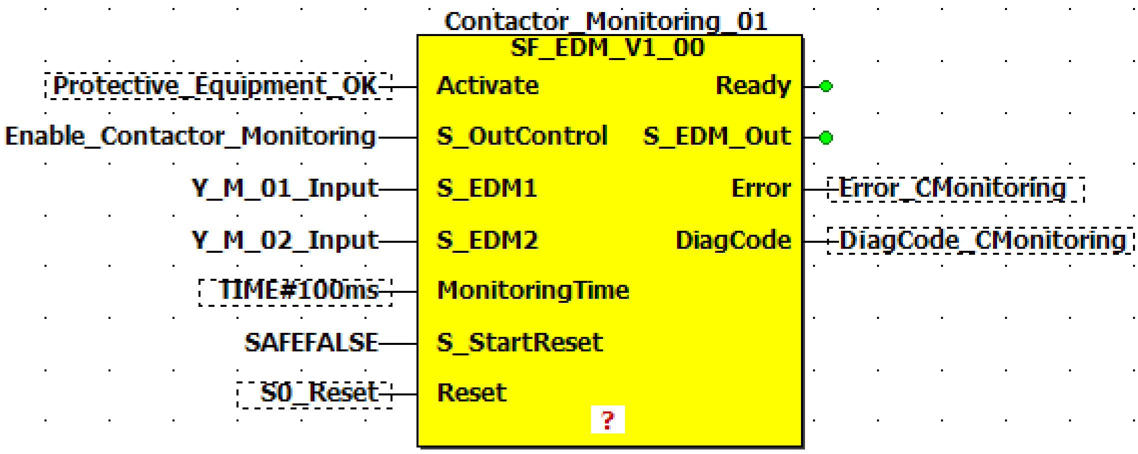
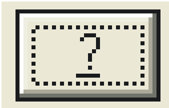


Figure 79: Function block for the guard monitoring 'Contactor_Monitoring_01', which should be declared and inserted.

A detailed description of the safe function block 'SF_EDM_V1_00' can be found in the handbook on the module. You can open this handbook by going to the module with the cursor and clicking at the right. The ‘Function/function block’ window will open:

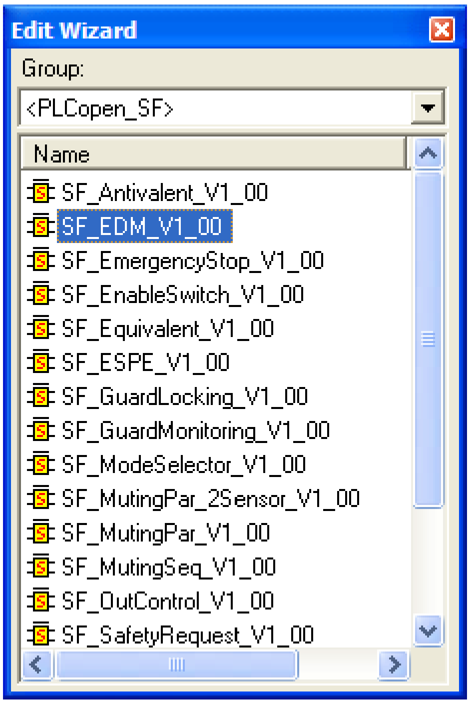
Click on the symbol



and you will receive a detailed description of the function block.

1 Activate the editor assistant by clicking on a free position in the code spreadsheet with the left mouse button.

2 Select the group 'PLCopen_SF' in the editor assistant and click on the entry 'SF_EDM_V1_00' with the left mouse button and hold the mouse button down.



3 Drag the cursor onto the code spreadsheet (which appears as a plus symbol) and release the mouse button.
The 'Variable' dialog will appear.

4 Enter the instance name 'Contactor_Monitoring_01' in the combo-box 'Name', select the group 'NewGroup' and confirm the dialog with 'OK'.

The schematic outline of the function block will appear at the cursor.

5 Drag the object in the code under the function block for the guard door and press the mouse button to deposit it.
Your code should now look like this:

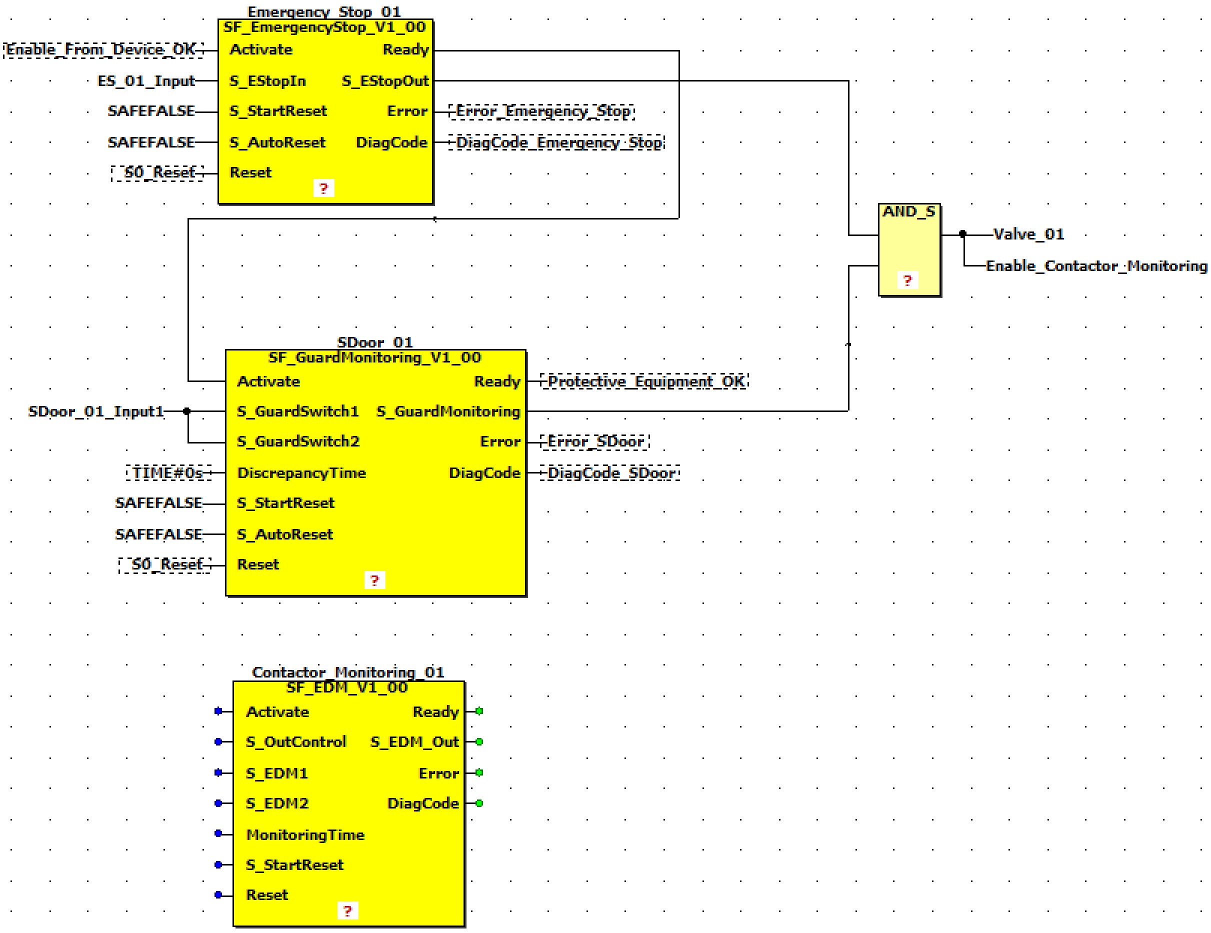


Figure 80: Code spreadsheet 'Main' with inserted function block 'Contactor_Monitoring_01'

We now want to insert local variables, global variables and constants and connect them with the inputs and outputs of the function block.

NOTICE!

A detailed description of how to assign local and global variables to the inputs and outputs of a function block and how terminals are allocated can be found in ▷DECLARING AND INSERTING THE SAFE FUCTION BLOCK 'Emergency_Stop_01' AND ALL RELATED VARIABLES ◁ from page 76 onward.

6 Double click on the blue connection point of the formal parameter 'Activate'.
The 'Variable' dialog will appear.

7 Activate the option 'Local', select the variable 'Protective_Equipment_OK' in list field 'Name' and confirm the dialog with 'OK'.

The variable 'Protective_Equipment_OK' will be inserted at the 'Activate' input of the function block.

8 Double click on the blue connection point of the formal parameter 'S_OutControl'.

The 'Variable' dialog will open again.

9 Activate the option 'Local', select the variable 'Enable_Contactor_Monitoring' in list field 'Name' and confirm the dialog with 'OK'.

The variable 'Enable_Contactor_Monitoring' will be inserted at the 'S_OutControl' input of the function block.

10 Open the directory tree and highlight the variable S_x_LM3_SDI1_0002 in the directory tree.
Hold the left mouse button down and drag the variable S_x_LM3_SDI1_0002 into the code. The 'Variable' dialog will open again.

Use the following settings for the parameters of the new safe variable (Name: 'Y_M_01_Input') in the ‘Variable’ dialog:

Parameter	**Setting**
Area of validity	Global
Name	Y_M_01_Input
Type	SAFEBOOL
Group	NewGroup
Object type	Variable

11 Confirm the 'Variable' dialog and drag the outline of the variables onto the blue connection point of the formal parameter 'S_EDM1' to connect the variable.

12 Open the directory tree and highlight the variable S_x_LM3_SDI1_0003 in the directory tree.
Hold the left mouse button down and drag the variable S_x_LM3_SDI1_0003 into the code. The 'Variable' dialog will open again.

Use the following settings for the parameters of the new safe variable (Name: 'Y_M_02_Input') in the ‘Variable’ dialog:

Parameter	**Einstellung**
Area of validity	Global
Name	Y_M_02_Input
Type	SAFEBOOL
Group	NewGroup
Object type	Variable

13 Confirm the 'Variable' dialog and drag the outline of the variables onto the blue connection point of the formal parameter 'S_EDM2' to connect the variables.

14 Double click on the blue connection point of the formal parameter 'MonitoringTime'. The 'Variable' dialog will open. Highlight the option 'Constants' and enter the 'TIME#100ms'.

15 Confirm the 'Variable' dialog with 'OK'.

16 Double click on the blue connection point of the formal parameter 'S_StartReset'. The 'Variable' dialog will open. Highlight the option 'Constant' and select the constant 'SAFEFALSE' in the list field 'Name'.

17 Confirm the 'Variable' dialog with 'OK'.

18 Double click on the blue connection point of the formal parameter 'Reset'. The dialog 'Variable' will be opened. Highlight the option 'Local' and select the variable 'S0_Reset' in the list field 'Name'.

19 Confirm the 'Variable' dialog with 'OK'.

20 Double click on the green connection point of the formal parameter 'Error'. Use the following settings for the parameters of the new variable (Name: 'Error_CMonitoring') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Local
Name	Error_CMonitoring
Type	BOOL
Group	NewGroup
Use	VAR

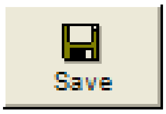
21 Confirm the 'Variable' dialog with 'OK'.

22 Double click on the green connection point of the formal parameter 'DiagCode'. Use the following settings for the parameters of the new variable (Name: 'DiagCode_CMonitoring') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Local
Name	DiagCode_CMonitoring
Type	WORD
Group	NewGroup
Use	VAR

23 Confirm the 'Variable' dialog with 'OK'.

24 Click on 'Save' in the symbol bar to save the code spreadsheet:



The function component 'Contactor_Monitoring_01' should now look as shown in ▹Figure 79◃ on page 99.

The declaration of the function block for guard monitoring is now complete and we can proceed with the development of the code. In the next step, we want to insert the safe function block 'Enable_Contactor_01' and all related variables.

Step 6 **DECLARING AND INSERTING THE SAFE FUNCTION BLOCK 'ENABLE_CONTACTOR_01' AND ALL RELATED VARIABLES**

We now want to declare the function block 'Enable_Contactor_01' for releasing the guard as well as all input and output variables necessary for this:

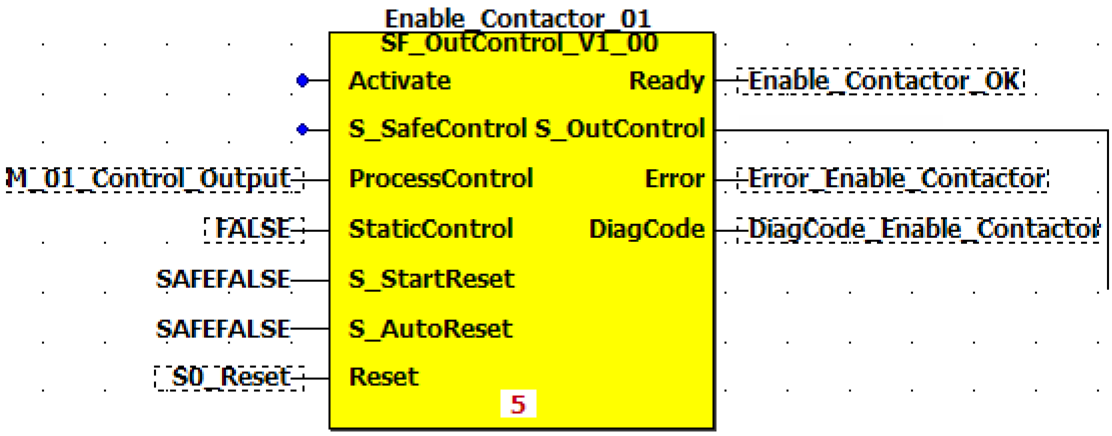
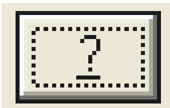


Figure 81: Function block 'Enable_Contactor_01' for releasing the guard, which is to be declared and inserted.

A detailed description of the safe function block 'SF_OutControl_V1_00' can be found in the handbook on the module. You can open this handbook by going to the module with the cursor and clicking at the right. The 'Function/function block' window will open:

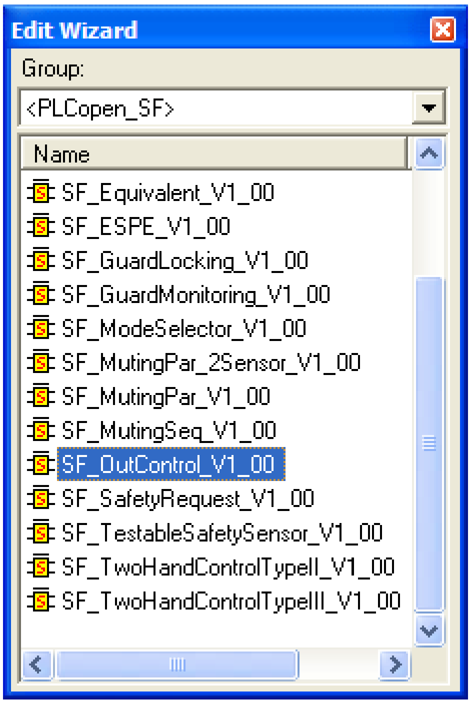
Click on the symbol



and you will receive a detailed description of the function block.

1 Activate the editor assistant by clicking on a free position in the code spreadsheet with the left mouse button.

2 Select the group 'PLCopen_SF' in the editor assistant, click on the entry 'SF_OutControl_V1_00' with the left mouse button and hold the mouse button down.



3 Drag the cursor (which appears as a plus symbol) onto the code spreadsheet and release the mouse button.
The 'Variable' dialog will appear.

4 Enter the instance name 'Enable_Contactor_01' in the combo-box 'Name', select the group 'NewGroup' and confirm the dialog with 'OK'.

The schematic outline of the function block will appear at the cursor.

5 Drag the object into the code next to the function block for guard monitoring and press the left mouse button in order to deposit it.

Your code should now look like this:

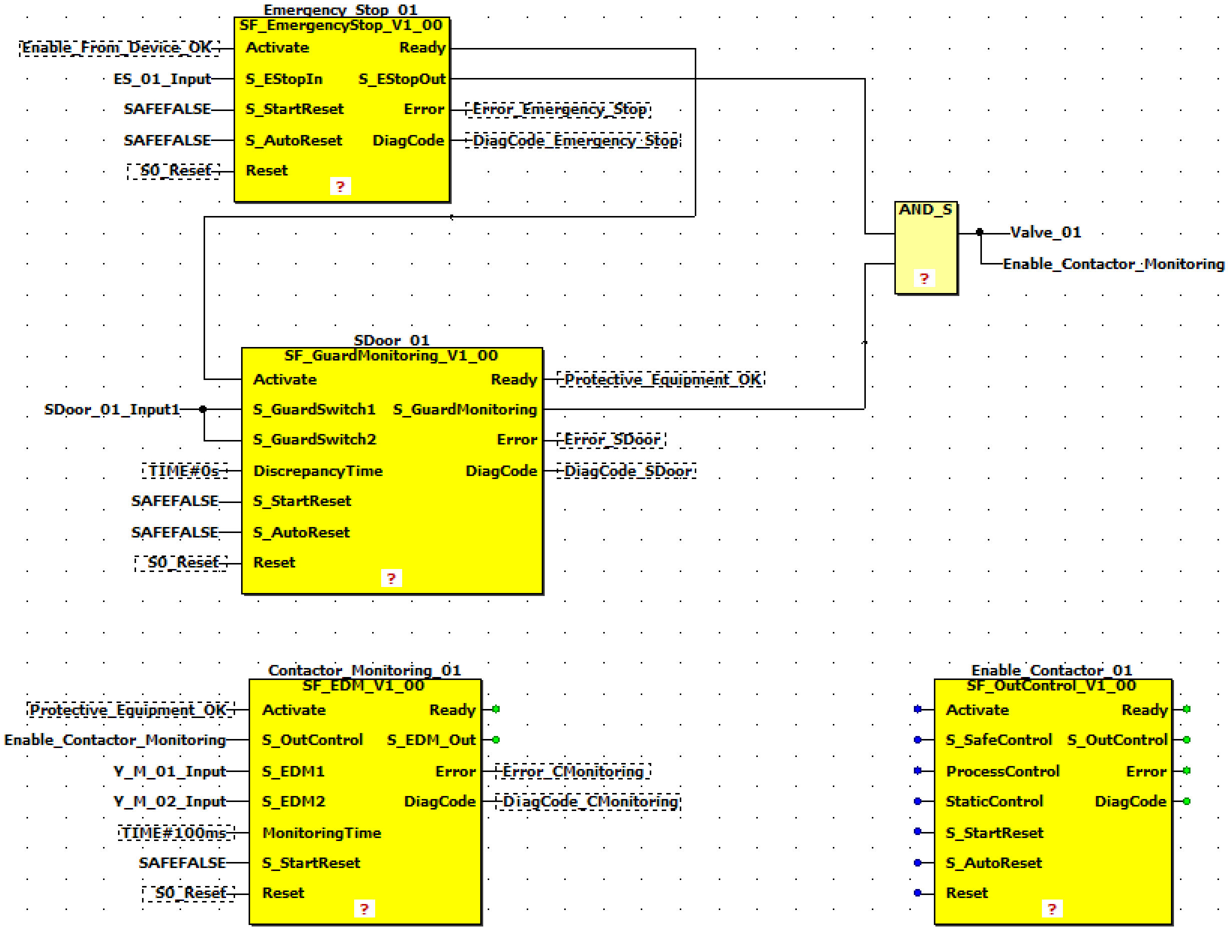


Figure 82: Code spreadsheet 'Main' with inserted function block 'Contactor_Monitoring_01'

We now want to insert local variables, global variables and constants and connect them with the inputs and outputs of the function block.

NOTICE!

A detailed description on how local and global variables are assigned to the inputs and outputs of a function block and how terminals are allocated can be found in ▹DECLARING AND INSERTING THE SAFE FUCTION BLOCK ‘Emergency_Stop_01’ AND ALL RELATED VARIABLES ◃ from page 76 onward.

6 Double click on the blue connection point of the formal parameter 'ProcessControl'. Use the following settings for the parameters of the new safe variable (Name: 'M_01_Control_Output') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Local
Name	M_01_Control_Output
Type	BOOL
Group	NewGroup
Use	VAR

7 Confirm the 'Variable' dialog with 'OK'.

8 Double click on the blue connection point of the formal parameter 'StaticControl'. The 'Variable' dialog will open. Highlight the option 'Constant' and select 'FALSE' in the list field 'Name'.

9 Confirm the 'Variable' dialog with 'OK'.

10 Double click on the blue connection point of the formal parameter 'S_StartReset'. The 'Variable' dialog will open. Select the constant 'SAFEFALSE' in the list field 'Name'.

11 Confirm the 'Variable' dialog with 'OK'.

12 Repeat the last two steps for the formal parameter 'S_AutoReset' in order to assign the constant 'SAFEFALSE' as well.

13 Double click on the blue connection point of the formal parameter 'Reset'. The 'Variable' dialog will open. Highlight the option 'Local and select the 'S0_Reset' in the list field 'Name'.

14 Confirm the 'Variable' dialog with 'OK'.

15 Double click on the green connection point of the formal parameter 'Ready'. Use the following settings for the parameters of the new variable (Name: 'Enable_Contactor_OK') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Local
Name	Enable_Contactor_OK
Type	BOOL
Group	NewGroup
Use	VAR

16 Confirm the 'Variable' dialog with 'OK'.

17 Open the tree directory and highlight the variable S_x_LM4_SDO2_0001 in the directory tree.
Hold the left mouse button down and drag the variable S_x_LM4_SDO2_0001 into the code. The 'Variable' dialog will open again.

Use the following settings for the parameters of the new safe variable (Name: 'M_01_Output') in the 'Variable' dialog:

Parameter	Setting
Area of validity	Global
Name	M_01_Output
Group	NewGroup
Object type	Variable

18 Confirm the 'Variable' dialog and drag the outline of the variables onto the green connection point of the formal parameter 'S_OutControl', to connect the variable.

NOTICE!

In order to connect the variable 'M_01_Output' to a release signal in the sense of the enabling principle (implied AND), set the function block EN_OUT before the assigning it to the safe output and connect it on the input side to the corresponding variable from the standard area. Instructions on inserting the function block EN_OUT and depositing a connecting variable can be found in the following extract.

Extract: Inserting the function block EN_OUT

- Separate the variable 'M_01_Output' from FB 'Enable_Contactor_01' by clicking and removing the connection line between the FB and variable.
- Insert the FB EN_OUT from the library <Safe FUs/FBs> by opening the editor assistant, selecting <Safe FUs/FBs>, clicking FB EN_OUT and dragging it in the spreadsheet 'Main' with the left mouse button held down.
- Position the FB to the right below the FB 'Enable_Contactor_01'.
- Connect the output 'S_OutControl' with input 1 of the FB EN_OUT.
- Couple the variable 'M_01_Output' to the output of FB EN_OUT.
- Now change to ProMaster. Select the tab 'Junction variables input' in the 'ProSafetyPLC' window.

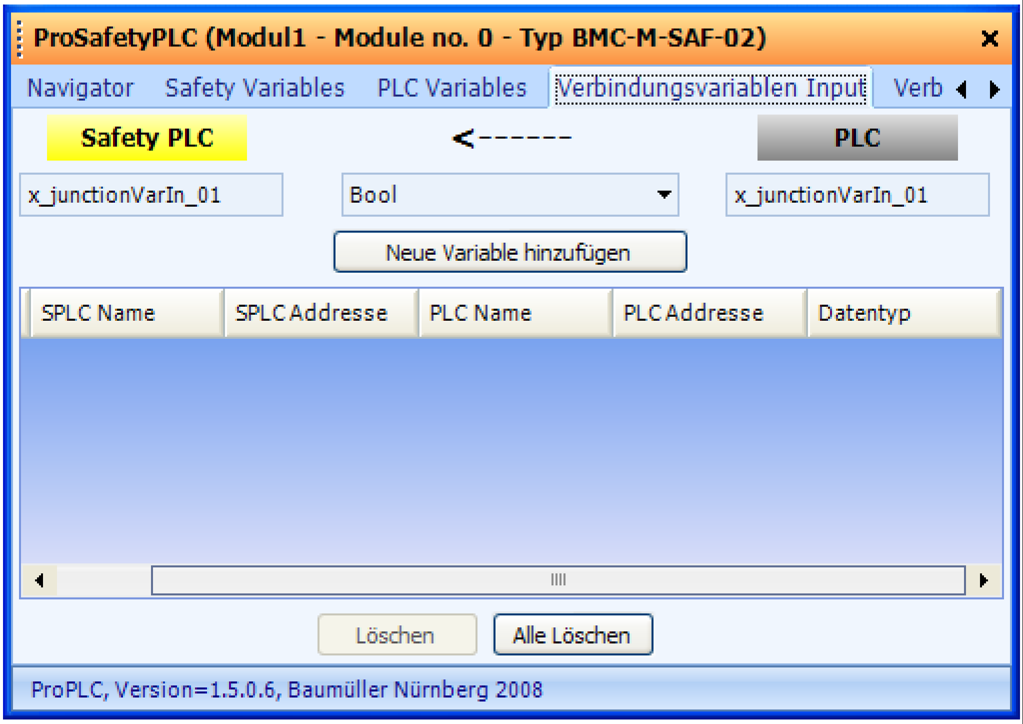


Figure 83: ProMaster - Connection variables input window

- Select 'Bool' in the list field 'Type' and click on 'Insert new variable'. A variable with a default name will appear. This default name can be kept.

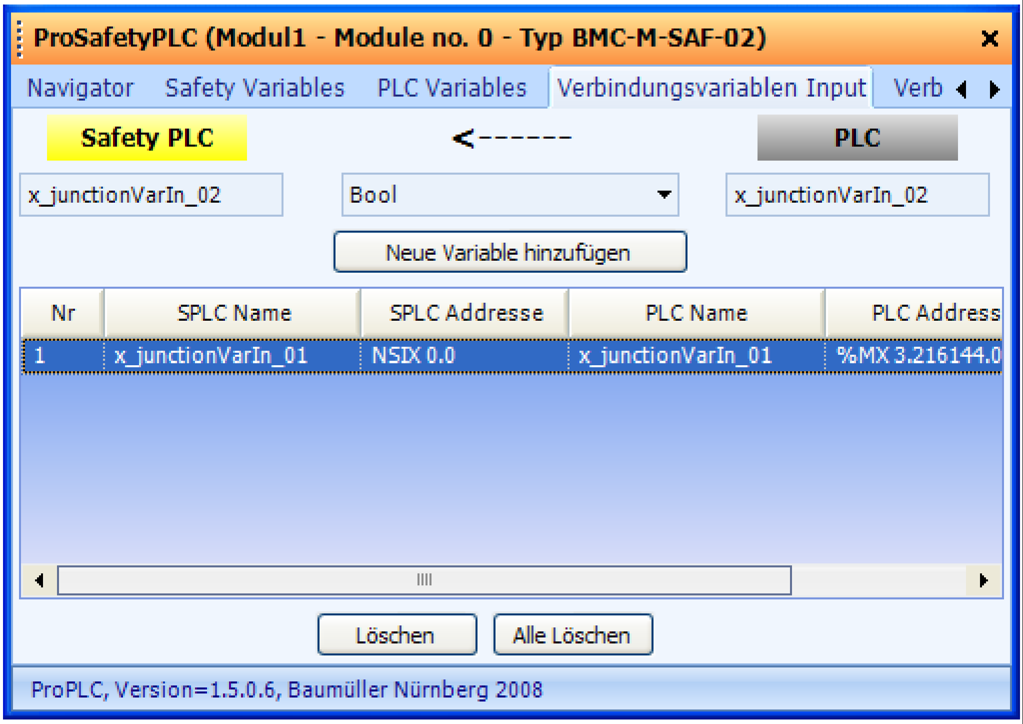


Figure 84: ProMaster - Connection variables input window with a new variable

- Press <F9> in order to update the settings in ProMaster and then change back to ProSafety.
- There, you will find the 'Connection variables' directory in the bus navigator.

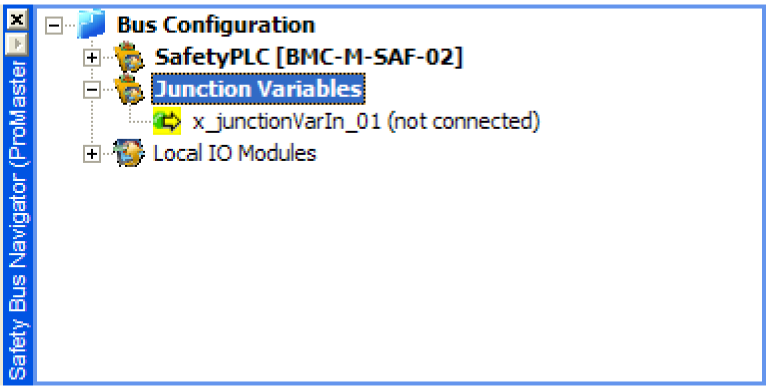


Figure 85: ProSafety bus navigator with 'Connection variables' directory

- Open the directory tree and highlight the variable x_junctionVarIn_01 in the directory tree.
  Hold the left mouse button down and drag the variable x_junctionVarIn_01 into the code. The 'Variable' dialog will open again.
- Highlight the option 'Global' and the dialog with 'OK'. Drag the outline of the variables in the spreadsheet near the function block EN_OUT and click to place the variable. The spreadsheet should now look like this:

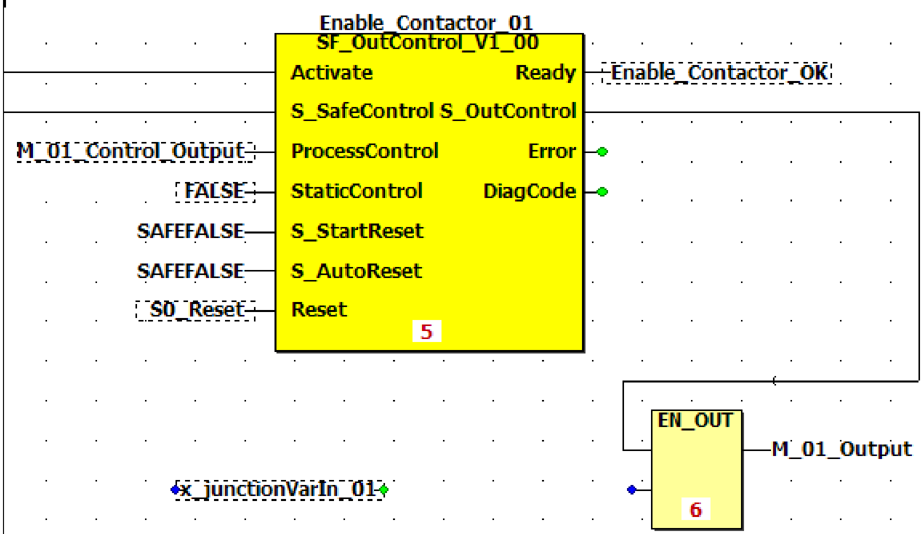


Figure 86: Spreadsheet with connection variable

- Drag the variable x_junctionVarIn_01' onto the blue connection point of the input 2 of the function block 'EN_OUT'. Your spreadsheet should now look like this:

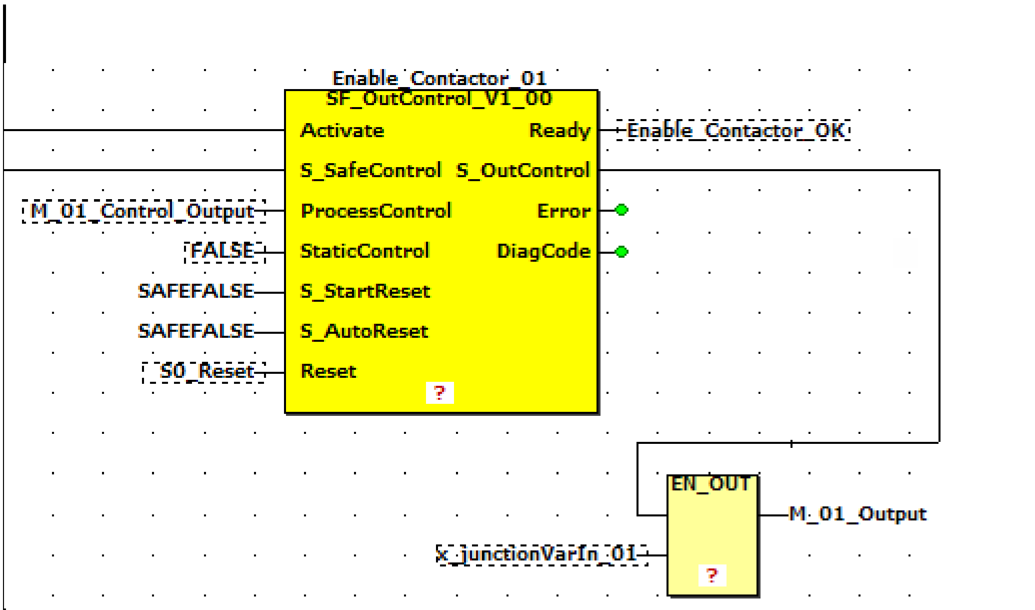


Figure 87: Excerpt from the code spreadsheet with connection variable to function block 'EN_OUT'

19 Double click on the green connection point of the formal parameter 'Error'. Use the following settings for the parameters of the new variable (Name: 'Error_Enable_Contactor') in the 'Variable' dialog:

Parameter	**Setting**
Area of validity	Local
Name	Error_Enable_Contactor
Type	BOOL
Group	NewGroup
Use	VAR

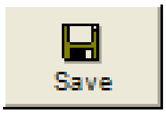
20 Confirm the 'Variable' dialog with 'OK'.

21 Double click on the green connection point of the formal parameter 'DiagCode'. Use the following settings for the parameters of the new variable (Name: 'DiagCode_Enable_Contactor') in the 'Variable' dialog:

Parameter	**Setting**
Area of validity	Local
Name	DiagCode_Enable_Contactor
Type	WORD
Group	NewGroup
Use	VAR

22 Confirm the 'Variable' dialog with 'OK'.

23 Click on 'Save' in the symbol bar in order to save the code spreadsheet:



The function block 'Enable_Contactor_01' should now appear as shown in ▷Figure 81◁ on page 104.

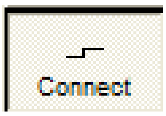
In the next step, we will connect the two previously inserted function blocks with the use of connection mode.

Step 7

CONNECTING TWO FUNCTION BLOCKS IN CONNECTION MODE

We now want to connect the function blocks 'Contactor_Monitoring_01' and 'Enable_Contactor_01' to one another by using connection mode.

1 Click 'Connect' in the symbol bar to switch to connection mode:



2 Click on the green connection point 'Ready' of the function block 'Contactor_Monitoring_01' (starting point of the connection line).

3 Drag the cursor to the blue connection point of the input 'Activate' at the function block 'Enable_Contactor_01' and click once in order to end the line. The layout of the line will automatically be conducted by the program and the two formal parameters will be connected.

4 Repeat the last two steps and draw a connection line between output 'S_EDM_Out' from function block 'Contactor_Monitoring_01' and the input 'S_SafeControl' of the function block 'Enable_Contactor_01'.

The two function blocks should now be connected to one another, as shown in the following illustration:

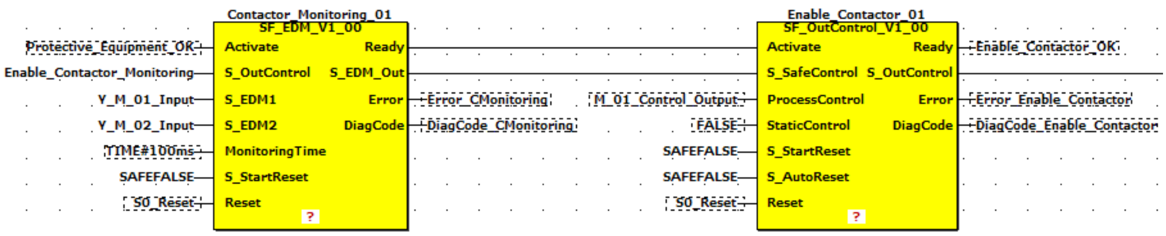


Figure 88: Connecting FB 'Contactor_Monitoring_01' and 'Enable_Contactor_01'

Step 8 **INSERTING DESCRIPTIVE TEXTS (COMMENTARIES) IN THE CODE-SPREADSHEET**

In order to complete our project compilation, we would like to insert descriptive texts (commentaries) in order to name the objects in the code spreadsheet.

1 Click on 'Highlight' in the symbol bar in order to switch to highlighting mode:

2 Click on the spreadsheet with the left mouse button in order to position an insertion mark at the position showed below. The commentary will be inserted at this position.

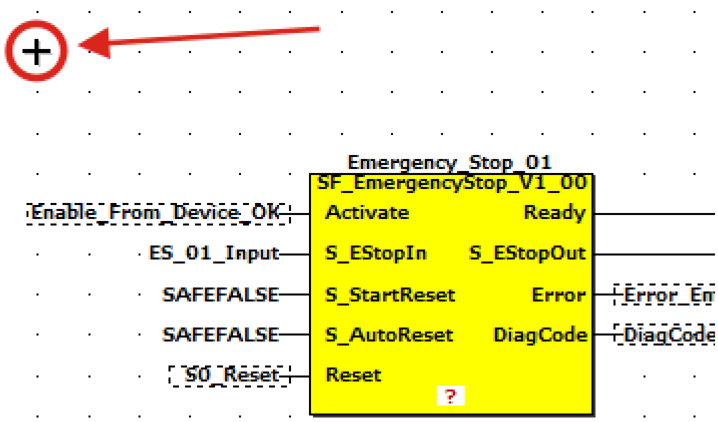


Figure 89: Position for commentary

3 Select the menu item 'Objects > Text (Comment)...'. The dialog 'Comment' will appear.

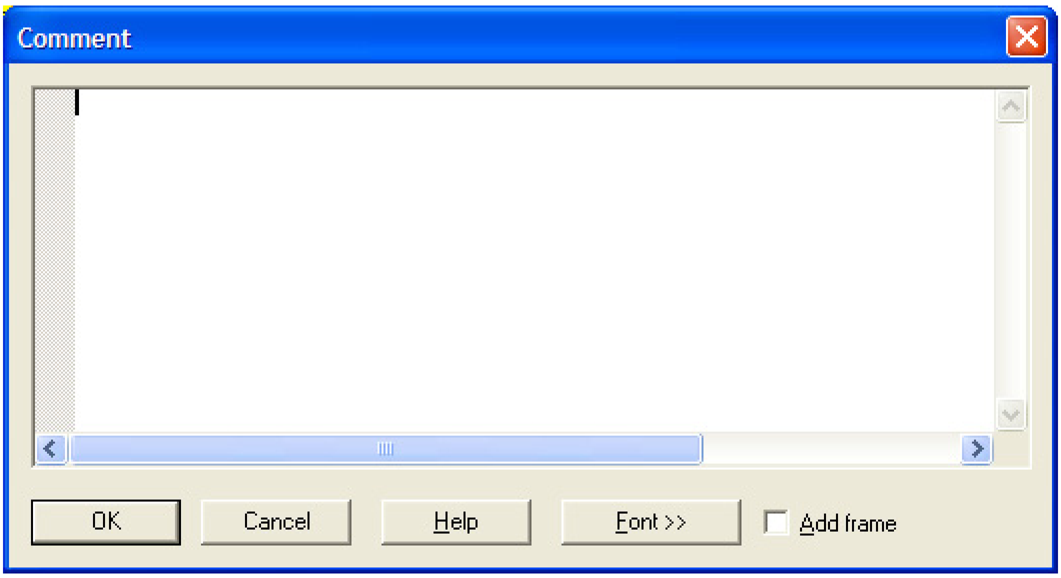


Figure 90: The 'Comment' dialog for entering commentaries in the code spreadsheet

4 Enter the text 'Emergency stop machine' into the entry field.

	NOTICE! Click on the 'Font >>' button to change font properties.

5 Click on 'OK' to insert the commentary in the function block.

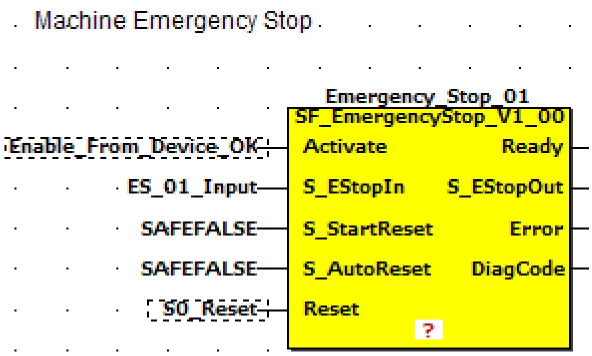


Figure 91: Code spreadsheet 'Main' with commentary

	NOTICE! If desired, you can relocate the commentary in order to align it to the function block. To do this, click on the text, hold the left mouse button down and drag the text to the desired position.

6 Repeat the course of action described above for the other function blocks and insert the following designations (commentaries):

Function block	Commentary
SDoor_01	Guard door machine
Contactor_Monitoring_01	Return monitoring Motor01

7 In addition, we want to insert two further variables to describe special variables in the code spreadsheet. To do this, enter the following descriptions (commentaries):

Variable	Commentary
Release_From_Device_OK	'Release_From_Device_OK' Signal of a device monitoring
M_01_Control_Output	'M_01_Control_Output' Signal from standard PLC (via connection variable)

The position of the commentaries in the code spreadsheet can be found in ▷Figure 40◁ on page 62.

8 Save the code spreadsheet by clicking on the 'Save' symbol in the symbol bar.

This completes the development of the code for our sample project and we now want to proceed with compiling the project and subsequently sending it to the safety control.

10.1.4.6 Phase 5: Compiling the sample projects

Once we have completed the processing of the project, we will have to compile it. During the course of compiling, the contents of the spreadsheets will be converted into special code which can be implemented by the safe control.

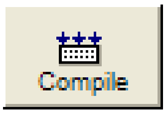
NOTICE!

If a POU in the project tree is marked with an asterisk (*), it means that the POU has not been compiled after the processing of the variable spreadsheets or code spreadsheets. The asterisk is removed upon successful compilation of the project.

You can use this marking as a quick way of determining whether changes have been made in the POU since the last compilation.

Step 1 **COMPILING THE PROJECT**

Click on the 'Compile' symbol in the symbol bar:



If unsaved changes are present, the project spreadsheets will be saved. In this case, a message will appear informing you whether the saving was successful. Click on 'OK' to confirm the message.

In the 'Code' index of the message window, you can see which steps are currently being undertaken by the compiler. If errors and warnings are detected during the compiling process, they will be protocolled in the corresponding message window sheets. After the compiling process, you can open the code spreadsheet directly from the message window in which an error was detected. To do this, simple double click on the corresponding error message in the message window.

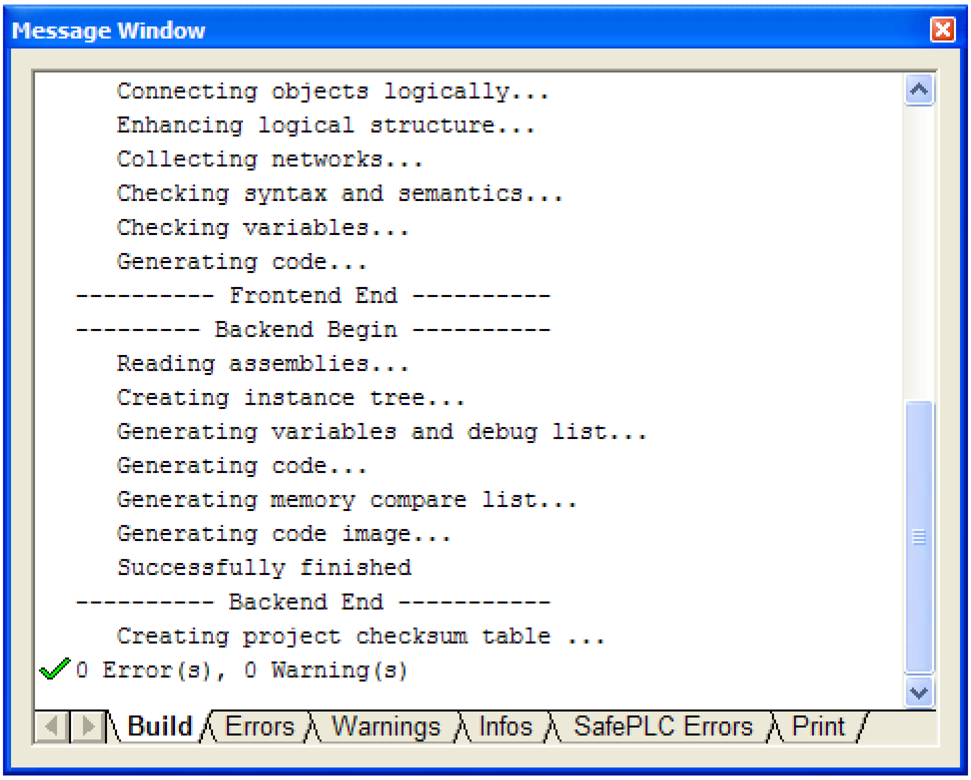


Figure 92: Message window after the project has been compiled using the 'Compile' command.

Step 2 **HANDLING ERRORS AND SYSTEM MESSAGES**

NOTICE!

Errors and warnings may appear during the course of the compilation process.

Errors prevent the complete implementation of the compilation process and appear in the event of syntax errors or structure problems, for instance.

Safety control errors are errors which occur in the safety control. These errors are displayed in the 'Safety control errors' index of the message window. Safety control errors do not halt the compilation process.

Warnings are given in the event of potential problems, such as an unused variable. Warnings do not halt the compilation process.

Take note of the warnings and rectify the errors in order to be able to continue working on the sample project.

- Click on the 'Error' index in the message window to display error messages which have appeared during the compilation process.
- If you want to display warnings which have appeared during the compilation process, click on the 'Warning' index.

- In most cases, you can open an error or warning directly on the spreadsheet in which the programming error or reason for the warning appeared by double clicking on it. The corresponding line or object will be highlighted.

  The system additionally offers you a help page on all errors. There, you can find out why the error has occurred and how to correct it. In order to call up a help page, highlight the error in the message window and press <SHIFT> + <F1>.
- If errors have occurred, rectify them and compile the project again by click on the 'Compile' symbol in the symbol bar.
- The program can only be sent to the safety control once it has been compiled successfully (free of errors).

10.1.4.7 Phase 6: Send the project to the safety control

CAUTION!

In the section, the **debug mode of the safe programming system** will be used. Switching to debug mode means that you will be leaving the safe operation mode. Before switching into debug mode, make sure that no damage can occur as a result of potential unintentional or incorrect operations of the safety control. Keep in mind that the safety control does not stop the operation automatically upon switching into debug mode.

Make certain that no one is located in the danger zone and that no one can enter the danger zone.

Step 1 **SETTING THE COMMUNICATION PARAMETERS**

Before you can send the project, a communication connection between your PC and the connected safety control must be established.

NOTICE!

The parameters used for this communication connection are automatically imported from the ProMaster project.

Normally, these settings do not need to be changed manually.

The communication connection settings have already been carried out in ProMaster (see ▷Figure 30◁ on page 56). They are overtaken in ProSafety can be inspected under 'Online > Communication connections' and changed if necessary.

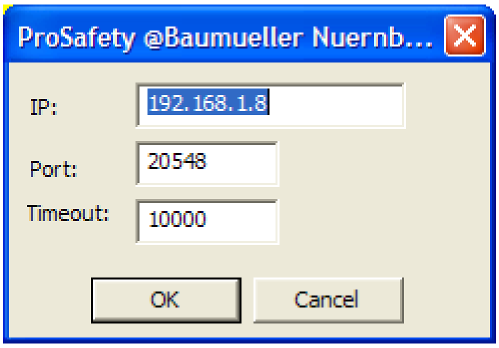


Figure 93: Communication settings

Further information on communication settings can be found in the operation manual b maXX safe PLC.

Step 2 **SENDING THE PROJECT**

The compiled project must now be sent to the safety control. The communication with the safety control takes place in the control dialog 'Safe PLC'.

1 Click on the symbol 'Safe PLC' in order to open the control dialog.

If there is already a project on the safety control other than the one to be sent from ProSafety to the PLC, the following security notice will appear:

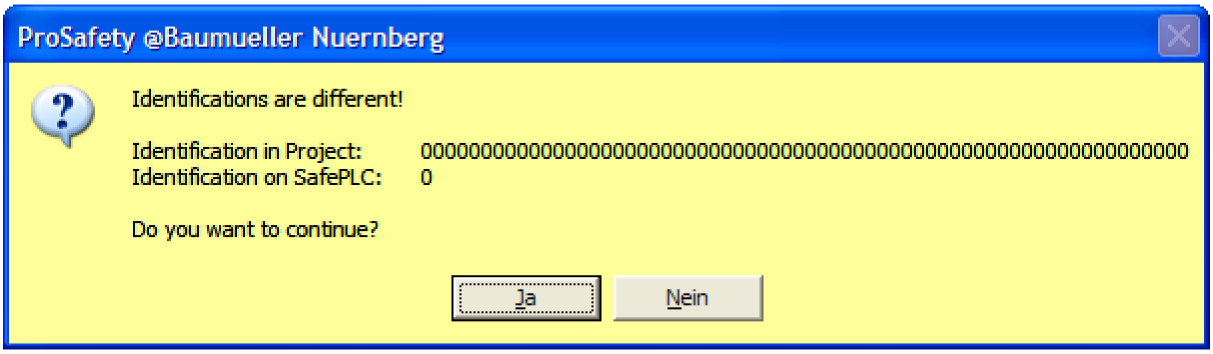


Figure 94: Query which appears in the event that there are different projects

If the time difference between the time on the safety control and the time on the PC is greater than 2 minutes, a dialog for synchronizing the clocks will appear.

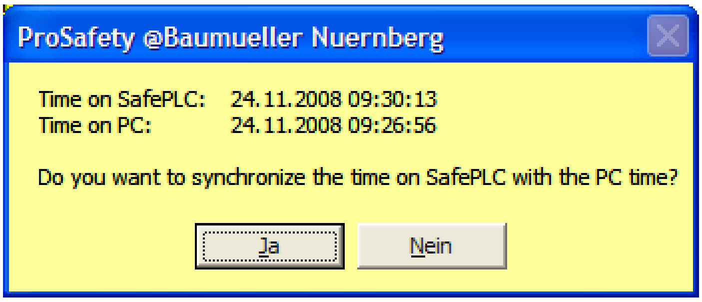


Figure 95: Dialog for synchronizing the time on the safety control

2 Click on 'Yes' to synchronize the time on the safety control.

The control dialog will be opened in safe mode (Status: Run [Safe]). Safe mode prevents unauthorized changes from being carried out on the project or the safety control from being stopped unintentionally.

Figure 96: The' Safety control' dialog in safe mode

3 Press the 'Debug' button in the control dialog to switch the safety control into debug mode.

Read the message which appears and confirm the message dialog within 30 seconds.

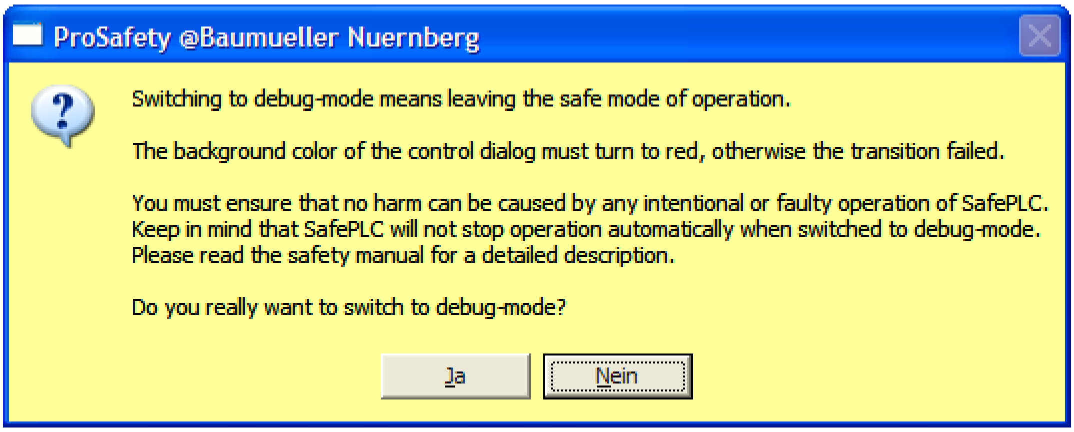


Figure 97: 'Switch to debug mode' message window

The control dialog will be opened in the non-safe debug mode. The safety control will run (Run [Debug]):

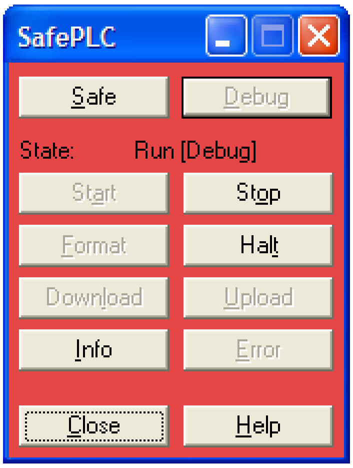


Figure 98: 'Safety control' dialog in non-safe debug mode

The non-safe debug mode is used to send the current project to the safety control, to start and stop execution of the program or execute addition commands via the dialog 'Safety control'.

4 Press the 'Stop' button to stop the safety control. The 'Download' button will become active.

5 Click on 'Download' in the control dialog.
The 'Download Options' dialog will appear:

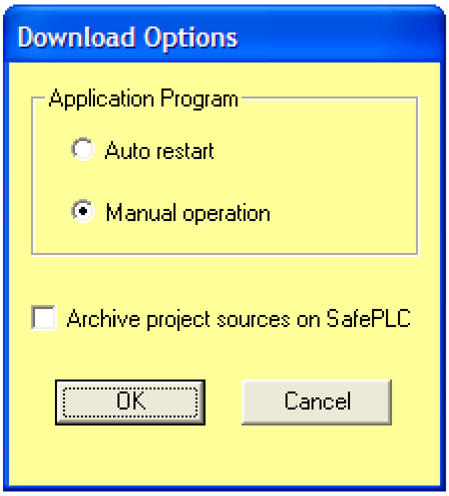


Figure 99: The 'Sending options' dialog for selecting special sending functions

NOTICE!

If you highlight 'Auto restart', the safety control will be started automatically upon resetting. If you highlight the 'Manual operation' option, the safety control will remain in 'Stop' status after resetting, and will have to me started manually.

6 Make sure that the 'Manual operation' option in the 'Download Options' has been selected and click on 'OK' to send the compiled project including the device parameterization data, project check sums, etc., to the safety control. You can abort the sending process with the 'Abort' button.

NOTICE!

A message notifying you of differing project versions between the safety control and PC may potentially appear. In this case, click on 'OK' to confirm the message and start the sending process.

The sending process will be indicated by a blue status bar on the lower edge of the screen. Once the sending process has finished, you will receive a message indicating whether the transmission was successful:

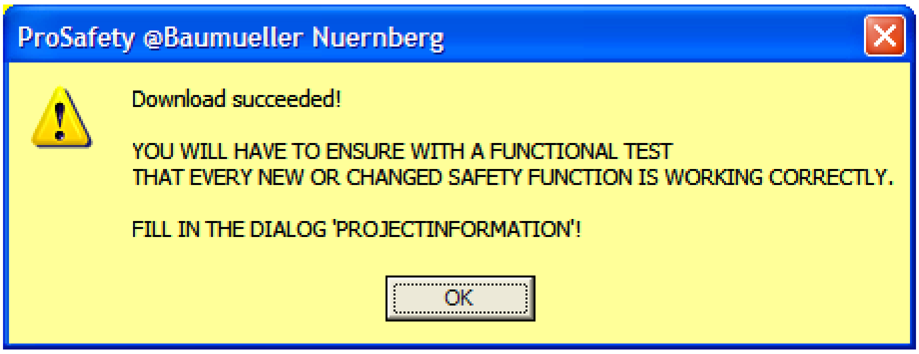


Figure 100: ProSafety message: Sending successful

7 Click on 'OK' in order to confirm the message.

The safety control will be reset automatically and initialized again. This can potentially take several minutes. It is not possible to communicate with the safety control during the resetting process.

8 Click on the 'Safe PLC' button to open the 'Safety control' dialog.

NOTICE!

A timeout will take place here, since the safety control has not yet been reset completely.

In this case, wait until the control dialog appears as specified below.

The control dialog will open in safe mode (Status: Stop [Safe]).

Figure 101: The 'Safety control' dialog in safe mode

9 Press the 'Debug' button in the control dialog to switch the safety control into debug mode.

Read the message which appears and confirm the message within 30 seconds.

The control dialog will be opened in non-safe mode (Stop [Debug]).

Figure 102: The 'Safety control' dialog in non-safe debug mode

10 Press the 'Start' button to start the safety control.

Read the message which appears and confirm the dialog with 'Yes' (the status will switch from 'Stop [Debug]' to 'Run [Debug]').

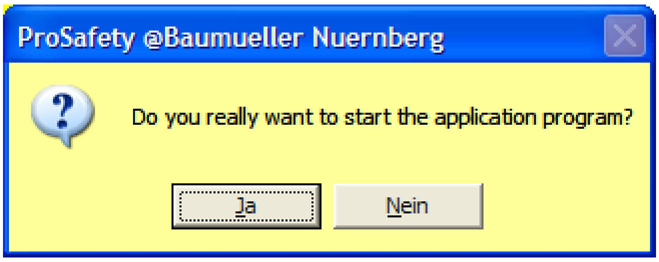


Figure 103: Start application program query

11 Press the 'Safe' button to activate safe mode (the status will switch from 'Run [Debug]' to 'Run [Safe]').

Read the message which appears and confirm the dialog with 'Yes'.

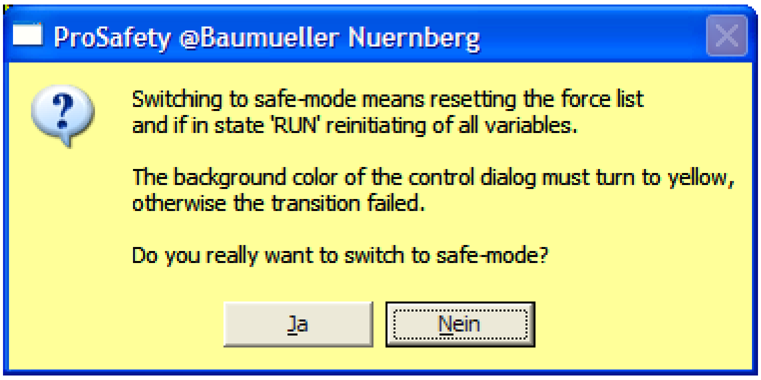


Figure 104: Query “switch to safe mode?”

The safety control will now run in safe mode.

Step 3 **PROJECT INFORMATION DISPLAY**

You may now want to know what is taking place in the safety control at the moment. The 'Safety control info' dialog is available for this purpose. It is opened by means of the 'Info' button in the 'Safety control' dialog.

The dialog displays information on the current project in the safe programming system, on the project which has been saved or is running on the safety control, on the current status of the safety control, debugging information, etc.

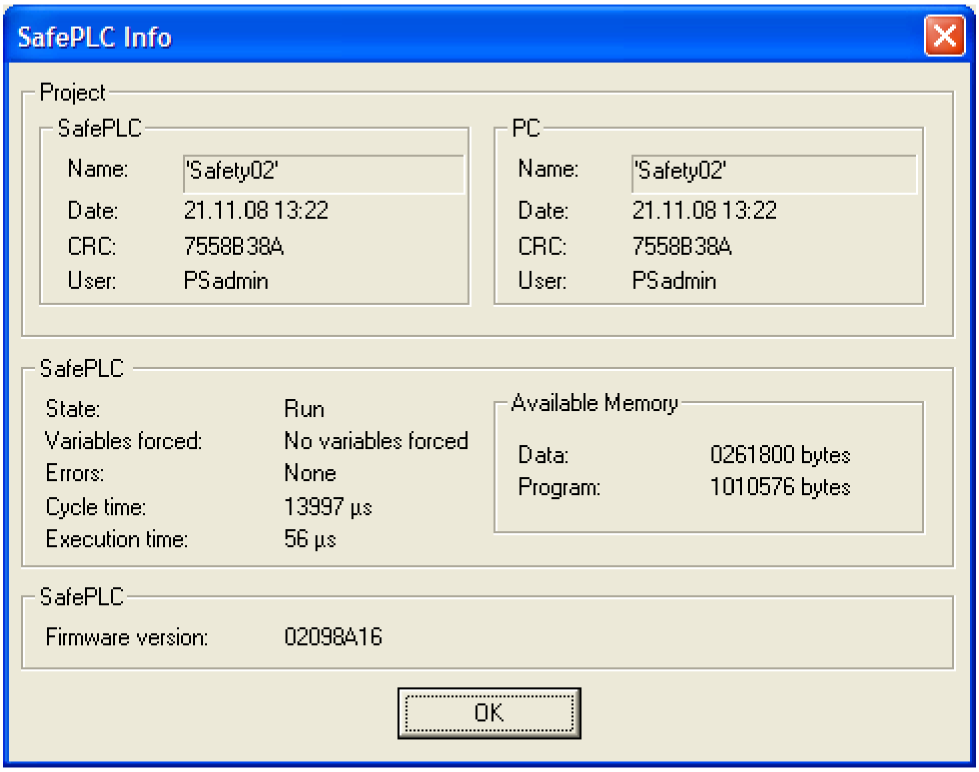


Figure 105: The 'Safety control info' dialog for displaying the project information

10.1.4.8 Phase 7: Debugging the project

CAUTION!

The **debug mode** of the safe programming system will be demonstrated in this section. Switching into debug mode means that you will exit safe operating mode. Before switching into debug mode, make sure that no damage can occur as a result of potential unintentional or incorrect safety control operations. Keep in mind that the safety control does not stop automatically upon switching into debug mode.

Therefore:

- Make sure that no one is in the danger zone and that no one can enter the danger zone.

Once you have sent the project and started the safety control, you can conduct the following operations to debug the code and variable spreadsheets:

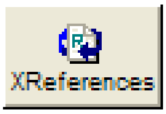
- Display project cross-references in the **cross-reference window** (see ▷CROSS-REFERENCE WINDOW ◁ from page 125 onward).
- Open spreadsheets in **variable status** to display the online values (see ▷VARIABLE STATUS (ONLINE MODE) ◁ from page 126 onward).
- **Force** global variables and **write over** local variables (see ▷FORCE AND OVERWRITE ◁ from page 127 onward).
- Display online values and run debug commands from the **watch window** (see ▷WATCH WINDOW ◁ from page 130 onward).

These functions can be used to check the behavior of POUs and variables in order to find programming errors and ensure proper running of the program for the safety control. If you find a programming error, you can switch the affected spreadsheets offline and work on them. Once you have corrected the error, you will have to compile the changes and send them to the safety control.

CROSS-REFERENCE WINDOW

The cross-reference window contains all variables and function blocks which are being used in the current project. This window is thus an especially helpful tool for locating and correcting errors.

1 Click on the 'Cross-references' symbol in the symbol bar to open the cross-reference window (if it is not already open):



2 Click on the background of the cross-reference window with the right mouse button to open the context menu.

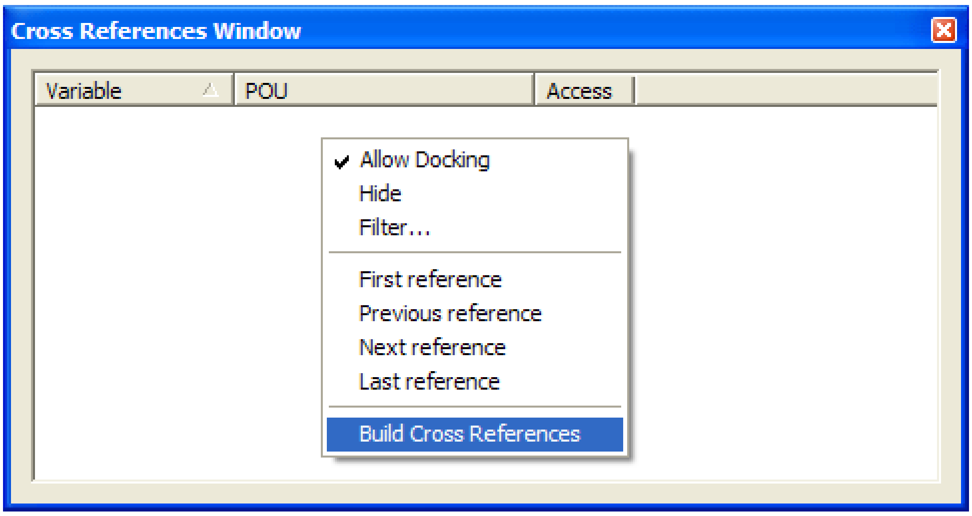


Figure 106: Cross-reference window with context menu for creating cross-references

3 Select the menu item 'Build Cross References'.

The cross-reference list of the current spreadsheet will be generated.

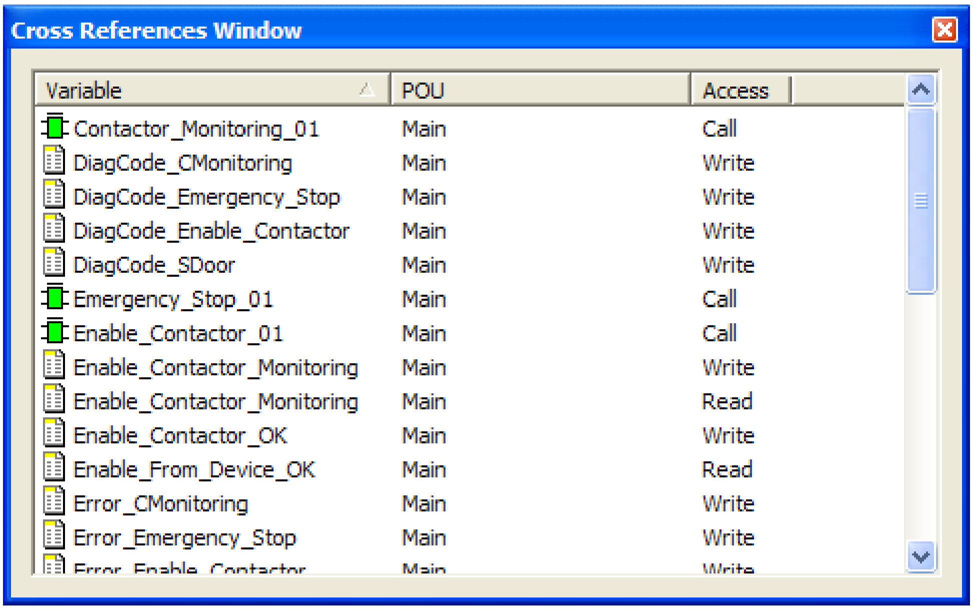


Figure 107: Cross-reference list in the sample project

4 In order to open a spreadsheet in which a particular variable is being used, simply double click on the corresponding variable in the cross-reference window.

When you highlight a variable in the spreadsheet, the corresponding variable will also be highlighted in the cross-reference window.

5 Close the cross-reference window and the message window by clicking on the applicable symbols in the symbol bar.

VARIABLE STATUS (ONLINE MODE)

Spreadsheets can be switched from editing mode (offline) into variable status (online) and vice versa. The variable status is helpful in finding programming errors and ensuring that the safety control program is running properly. The current values and conditions of the variables are shown in variable status.

Variable status can be used both in the safety control's **safe mode** and **debug mode**.

1 Make certain that the safety control is running. In the upper area of the control dialog 'Safety control', you will see which status the safety control is in. If the program is not running, start the program execution by clicking on the 'Start' button in the control dialog.

2 Before switching into variable status, make sure that the POU 'Main' code spreadsheet is open. If this is the case, click on the 'Variable status' symbol in the symbol bar.

The following illustration depicts the function block 'Enable_Contactor_01' from our project in variable status (online mode):

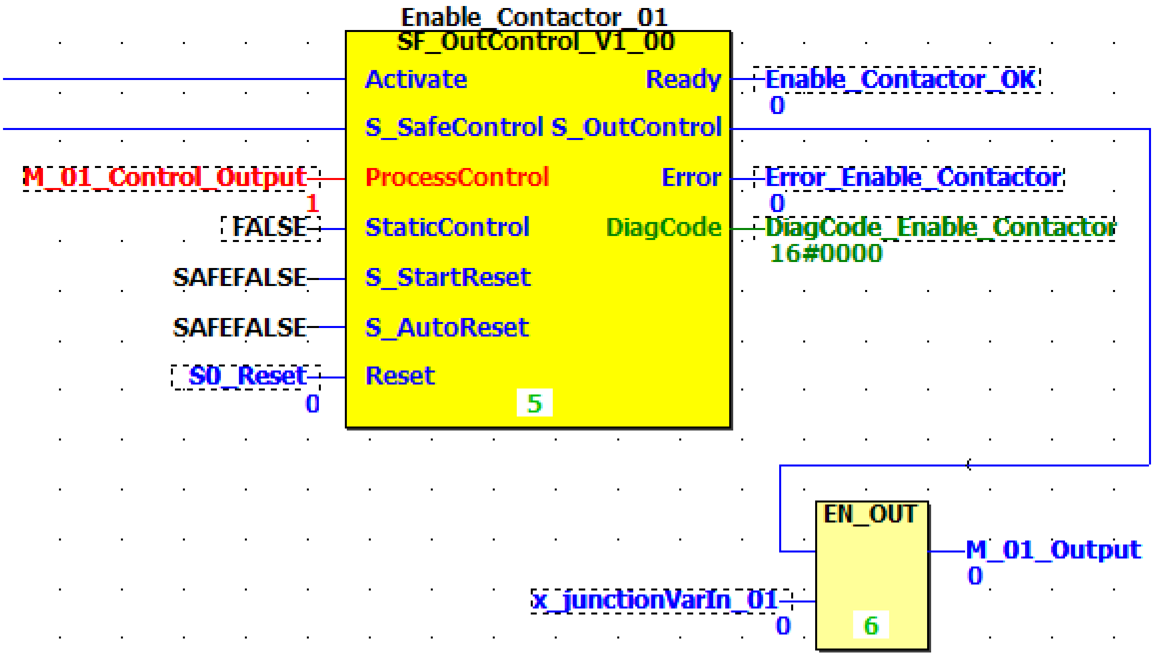


Figure 108: Depiction of a function block in variable status

The variables and current values are displayed in different colors. These colors represent the different conditions and have the following meanings:

- blue = FALSE (Boolean variables)
- red = TRUE (Boolean variables)
- green = Values (non-Boolean variables)

You can switch between online and offline mode by clicking on the 'Variable status' button.

FORCE AND OVERWRITE

Variables can be forced and overwritten in variable status (online mode). The new value of the corresponding variables is assigned in both cases.

NOTE!

Forcing and overwriting are only possible in debug mode while the safety control is running (status 'Operation' or 'Stop').

What is the difference between forcing and overwriting?

When **overwriting** a variable with a value, the new value will only be used for one work cycle. Once the cycle has ended, the variable will be processed normally again. Overwriting is only possible for **local variables**.

With **forcing**, the new value will be used for the global variable until you reset the forced variable back to its normal value. Only **global variables** (that is, physical inputs and outputs) can be forced.

The course of action in forcing and overwriting variables is nearly identical.

CAUTION!

Be extremely careful when forcing or overwriting variables when the safety control is running. Forcing and overwriting means that the program on the safety control is executed with the values of the forced or overwritten variables.

Forcing the global variable 'M_01_Output'

We now want to force the global variable 'M_01_Output' at the input 'S_OutControl' of the function block 'Enable_Contactor_01':

1 Click on the variable 'M_01_Output' with the right mouse button and select the 'Debug dialog...' in the context menu.

The dialog 'Debug: CPU' will appear. Since the variable is currently 'FALSE', the value 'TRUE' will be preselected automatically:

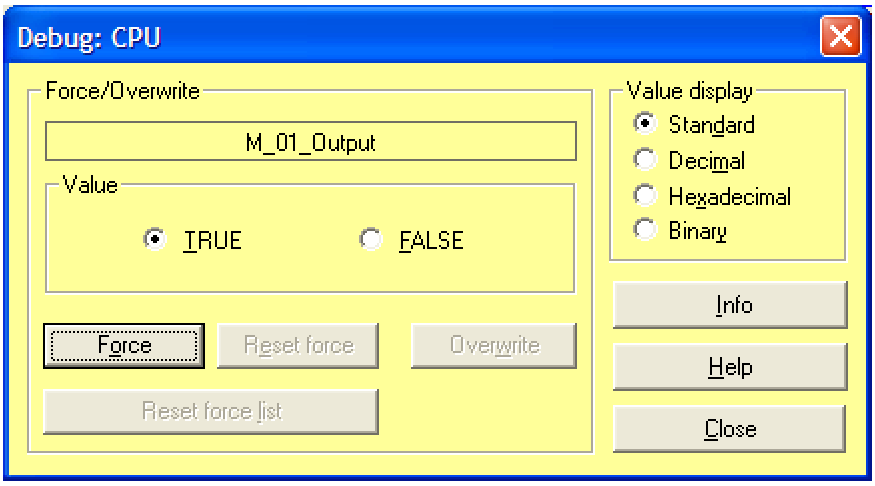


Figure 109: 'Debug: CPU' dialog for forcing variables

2 Click on 'Force'.
A warning message will appear to notify you that forcing a variable can cause machines to start up.

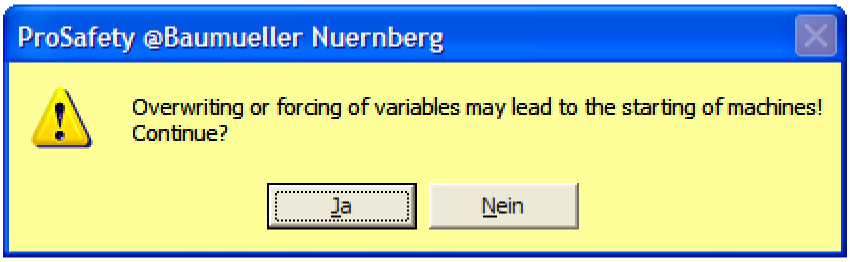
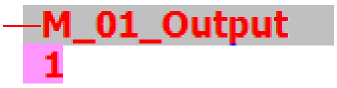


Figure 110: Notification of dangers when 'forcing'

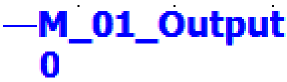
3 Click on 'Yes' in order to confirm the message.

By doing so, the variable 'M_01_Output' will be forced to 'in' and highlighted red in the online spreadsheet.



The pink background indicates that the variable is currently being forced.

4 Double click on the variable 'M_01_Output' again and select 'Force back' in the debug dialog in order to deactivate the forcing. The variable will be reset to its normal value:



Overwriting local variables

Overwriting local variables is identical to forcing global variables. The only difference is that the overwritten values are only valid for a single work cycle.

The following illustration shows the 'Debug: CPU' dialog for the local variable *"My local variable"*. In such case, only the 'Overwrite' button will be active in the dialog:

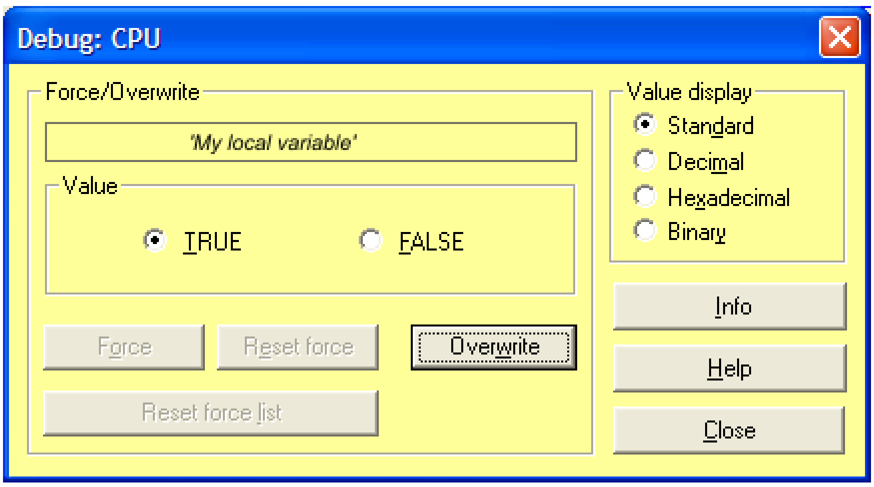


Figure 111: 'Debug: CPU' dialog for overwriting variables

NOTICE!

Forcing and overwriting variables is also possible in variable spreadsheets in online mode. Here, the debug dialog can be called up via the context menu or by double clicking on the desired variable.

The force list will be erased when you close the safe programming system, that is, all forced and overwritten variables will be reset to their original values.

WATCH WINDOW

In the watch window, you can collect variables from various spreadsheets in order to gain an overview of how these variables work together. If you have inserted a variable into the watch window, you will no longer have to open the related spreadsheet to be able to see the current value.

NOTICE!

The watch window can be used in **safe mode** and in **debug mode**.

If the safety control is running in **safe mode** and the spreadsheets have been switched into variable status, you can insert variables in the watch window and display their online values. The number of variables which can be inserted into watch window is not limited.

If the control is in **debug mode**, you can additionally use the watch window to force and overwrite variables, that is, to debug the project (see ▹FORCE AND OVERWRITE ◃ on page 127).

Variables can be inserted into the watch window from the code spreadsheets or variable spreadsheets (in debug mode).

1 If your project is running in safe mode now, switch it into debug mode by pressing the 'Debug' button in the 'Safety control' dialog.

CAUTION!

The **debug mode** of the safe programming system will be demonstrated in this section. Switching into debug mode means that you will exit safe operating mode. Before switching into debug mode, make sure that no damage can occur as a result of potential unintentional or incorrect safety control operations. Keep in mind that the safety control does not stop automatically upon switching into debug mode.

Therefore:

- Make sure that no one is in the danger zone and that no one can enter the danger zone.

2 Select the menu item 'View> Watch window' or click on the spreadsheet with the right mouse button and select 'Open watch window...' in the context menu. The watch window will open.

3 Open POU 'Main' code spreadsheet by double clicking on the corresponding symbol in the project tree.

4 Click on 'M_01_Output' in the spreadsheet with the right mouse button to open the context menu. Here, select the menu item 'Insert in watch window' in order to enter the variable in the watch window.

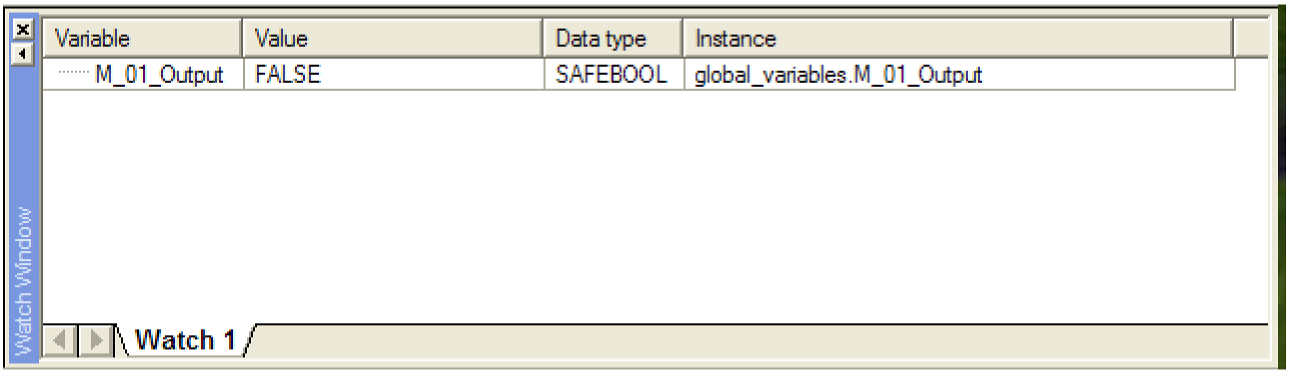


Figure 112: Watch window

5 As a second example, we will insert the global variable 'NA_01_Input' into the watch window. To do this, open the POU 'Main' local variable spreadsheet by clicking on the 'Global decl.' symbol in the symbol bar.

6 Click on 'NA_01_Input' in the variable table with the right mouse button in order to open the context menu. Select the menu item 'Insert in watch window' in order to enter the variable in the watch window.

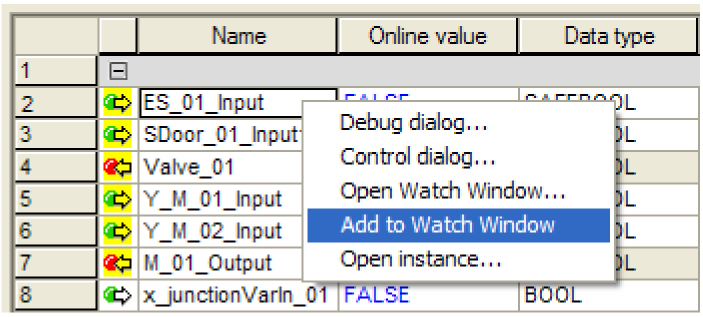


Figure 113: Inserting a variable into the watch window

The watch window should now look like this:

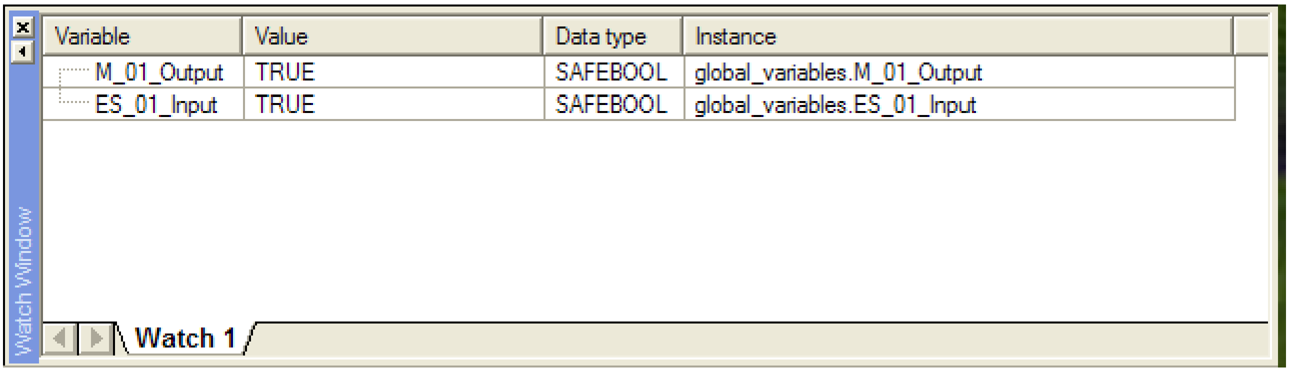


Figure 114: Watch window

NOTICE!

When you force or overwrite variables (see ▹Overwriting local variables ◃ on page 129), the current variable values will also be displayed in the watch window.

10.1.4.9 Phase 8: Entering project information

CAUTION!

Once you have sent a project to the safety control and have found and corrected any errors contained (that is, once the project has been finished), the project information, that is, the most important information on the project, **will have to be entered** for security reasons as well as for purposes of comprehensibility and archiving.

When you print out the project documentation (see ▹Page 134), the project information entered will be printed out together with it and can be archived that way.

The 'Project info' dialog will open when you select the menu item 'Project > Project information...'.

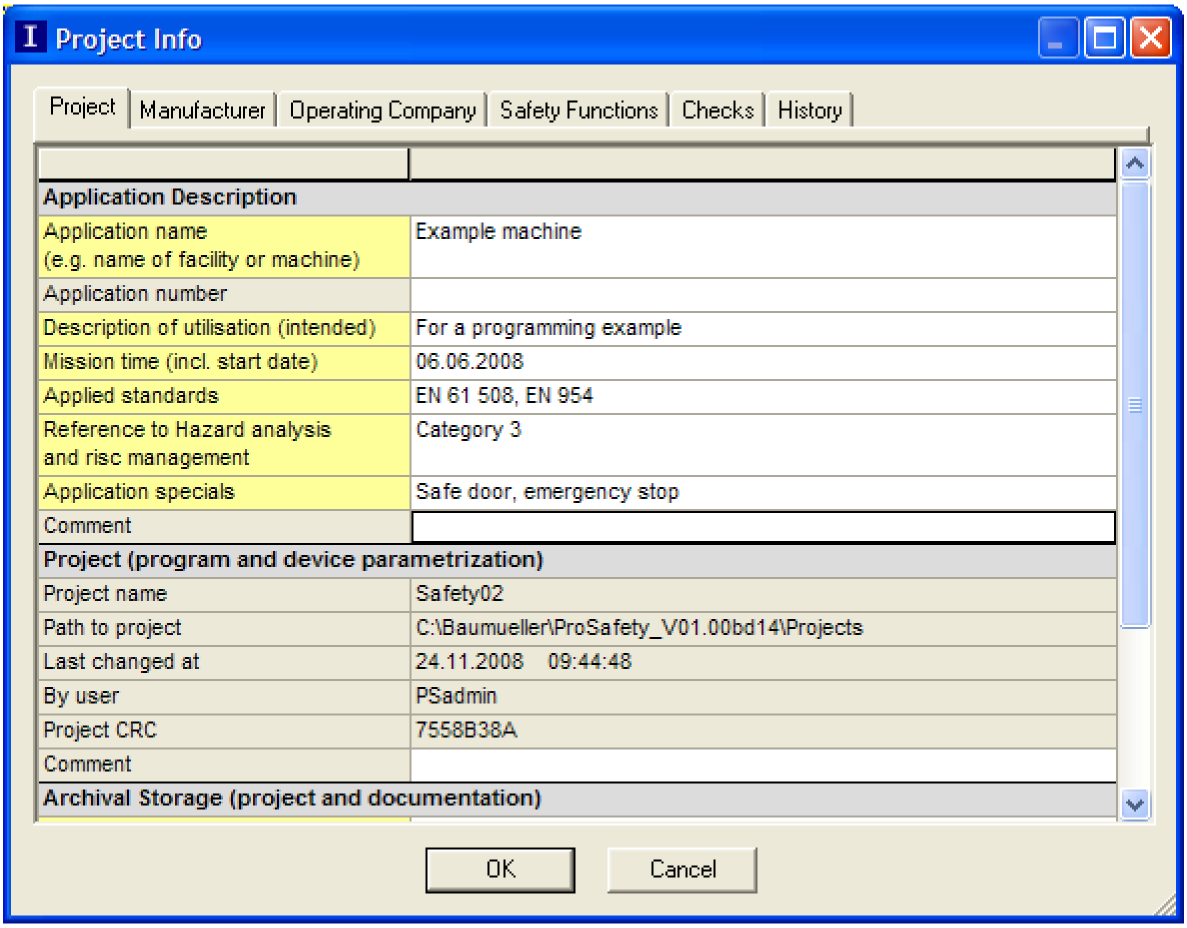


Figure 115: The 'Project info' window

The dialog is used to enter or display the most important project information for the current project. Here, you will receive information on project-related data (description of the application, designation, name of creator/processor, etc.), manufacturer data, operator and installation site as well as data on safety inspections and proof of change for the project.

CAUTION!

Fields with a yellow background are mandatory fields and must be filled in by the user each time a new project version is developed. Fields with a gray background are optional. However, it is **highly recommended** to fill out all entry fields including the optional fields.

The 'Project' area in the 'Project (program and device parameterization)' index is write protected. This data originates from the project administration of the programming system. Project name and directory path, date, time, last user logged in at the time of the last change to the project as well as the CRC check sum and user currently logged in are displayed.

The data in the 'Inspections' index are part of the acceptance test.

NOTICE!

Some of the entry fields are represented as being inactive. These fields are dependent on other fields and only become active once an entry is made in the corresponding related fields. (For example: Index 'Project', area 'Archive': The field 'Storage location of the printout' can only be filled out once 'Yes' has been selected in the field 'Print project').

10.1.4.10 Phase 9: Printing the project documentation

CAUTION!

Once you have sent a project to the safety control and have found and corrected any errors contained (that is, once the project has been finished), the entire project **will have to be printed out** for security reasons and for comprehensibility and archiving purposes.

Printing the entire project means that all code and variable spreadsheets, local and global cross-references, project information and bus navigator settings, as well as (if desired) the parameters of the safe devices will have to be printed out and archived.

STEP 1 SELECTING A PRINTER

Open the Windows standard dialog 'Printer settings' with the command 'Configure printer' in the sub-menu 'File'. Here, you can select the printer desired and conduct all printer settings.

STEP 2 SETTING THE PAGE LAYOUT TEXTS

In the programming system, you can define page layout texts which are printed as standard together with each page. Here, you can enter company data and integrate a company logo (bitmap file). These elements are then printed out on the spreadsheets.

To enter page layout texts:

1 Select the menu item 'File > Print project'.

2 Click the 'Page layout texts...' button in the 'Print project' dialog.

3 Enter the desired texts, select (if desired) a company logo using the 'Search' button and confirm the dialog with 'OK'.

STEP 3 PAGE VIEW

The page view shows you the spreadsheet as it will appear when printed. It makes it easier for you to arrange the elements on the page in a well-organized and structured manner.

NOTICE!

Cross-references are not displayed in the page view.

This is how to open the page view:

1 Make sure that the spreadsheet which you want to view is the active window.

2 Select the menu item 'Page view' from the sub-menu 'File'.
The page view of the active spreadsheet will be displayed.

3 In order to print this individual spreadsheet, click on the 'Print' button.

STEP 4 PRINTING PARTS OF PROJECTS

1 Select the menu item 'Print project...' under the sub-menu 'File'. The dialog 'Print project' will appear.

2 Deactivate the control box of the project parts which you do not want to print in the 'Print' area of the 'Print project' dialog.

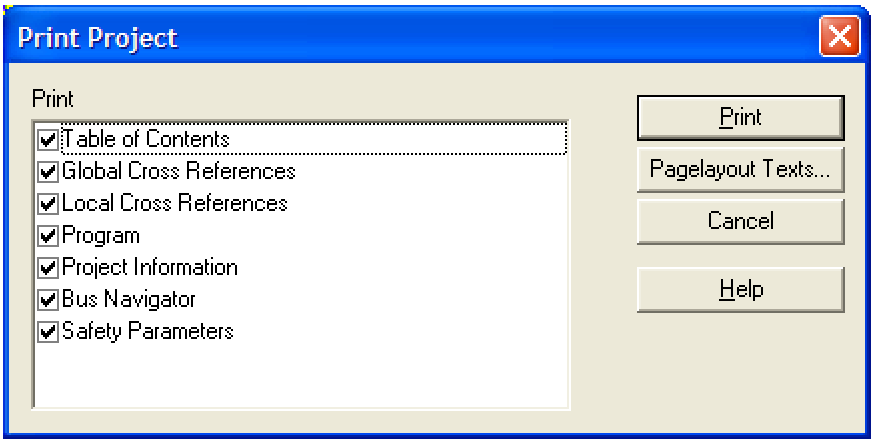


Figure 116: The 'Print project' dialog

3 Click on 'Print' to begin the printing process.

STEP 5 PRINTING A SINGLE SPREADSHEET

Single code spreadsheets or variable spreadsheets which are currently active can be printed out by using the menu item 'File > Print'.

NOTICE!

Cross-references will not be printed when printing a single spreadsheet with the 'Print' option.

Proceed as follows:

1 Make sure that the spreadsheet to be printed is the active window.

2 Select the menu item 'Print...' from the sub-menu 'File'.
The spreadsheet will be printed.

10.1.4.11Phase 10: Project backup

The safe programming system offers the possibility of archiving projects by compressing the entire project into an archive file. The archive file contains all of the files of the current project, that is, the project file 'projectname.swt', the code files, the variable declarations and several internal files which are needed to restore the projects from the archive file.

Libraries are not compressed and must be installed at each respective computer workplace.

CAUTION!

In order to prevent data from being lost, it is recommended to compress the files into archive files regularly and save them on a reliable data medium (CD, net drive, mirrored hard drive, etc.).

Only the safe application including the bus device parametrization will be compressed when you compress the project. The ProMaster project will have to be archived separately!

COMPRESSING PROJECT FILES IN AN ARCHIVE FILE

NOTICE!

Make sure that you have entered the project information in the dialog 'Project info' (see ▹Phase 8: Entering project information ◃ from page 132 onward) before you compress the project

Proceed as follows:

1 Select the menu item 'File > Save project under / Compress project under'.
The dialog 'Save project/compress under' will appear'.

2 Set the entry 'Compressed project files (*.szp)' in the list field 'File type'.

3 Enter a name for your archive file in the text field 'File name'.

4 Click on 'Compress' to begin the process.
Once the project has been compressed, a corresponding message will appear in the status line.

DECOMPRESSING AN ARCHIVE FILE

Proceed as follows:

1 Select the menu item 'File > Open project / Decompress project'.
The dialog 'Open/decompress project' will appear.

2 Set the entry 'Compressed project files (*.szp) in the list field 'File type'.

3 Enter the name and the path for the archive file manually or search for the archive file (*.szp) in the directory structure.

4 Click on 'Decompress' to decompress the archive file.
In the following dialog, you will be asked if the current project should be overwritten by the decompressed project of if the archived project should be decompressed into a new project with the name 'Untitled'.

10.1.5 Communication between control and PC

See ▹Page 117.

10.1.6 Debug functions

See ▹Page 124.

10.1.7 Safe and standard function blocks

Library BM-BMSD01_10bd00

Function block name	Description
SF_BMSD01_PD_Read_V1_00	safe cyclic reading of safety module status words
SF_BMSD01_PD_Write_V1_00	safe cyclic writing of control words to the safety module

Library BM_SafeBitLib_10bd00

Function block name	Description
SAFEBOOLS_TO_SAFEWORD_V1_00	puts a safe word together out of safe bit variables
SAFEWORD_TO_SAFEBOOLS_V1_00	reads out safe bits from safe word

LIbrary PLCopen_SF

Function block name	Description
SF_Antivalent_V1_00	Antivalent signal evaluation
SF_EDM_V1_00	External device monitoring
SF_EmergencyStop_V1_00	EMERGENCY STOP
SF_EnableSwitch_V1_00	Enabling switch

Function block name	Description
SF_Equivalent_V1_00	Equivalent signal evaluation
SF_ESPE_V1_00	Electrosensitive protective equipment (ESPE)
SF_GuardLocking_V1_00	Guard door locking
SF_GuardMonitoring_V1_00	Guard door monitoring
SF_ModeSelector_V1_00	Operation mode selection switch
SF_MutingPar_2Sensor_V1_00	Parallel muting with two sensors
SF_MutingPar_V1_00	Parallel muting
SF_MutingSeq_V1_00	Sequential muting
SF_OutControl_V1_00	Output control
SF_SafetyRequest_V1_00	Safety request
SF_TestableSafetySensor_V1_00	Test safety sensor type 2
SF_TwoHandControlTypeII_V1_00	Two-handed switching type 2
SF_TwoHandControlTypeIII_V1_00	Two-handed switching type 3

10.1.7.1 IEC 61131-3 AND THE SAFE PROGRAMMING SYSTEM

Due to special safety requirements, only part of the features defined in IEC 61131-3 are implemented in the safe programming system.

The following list shows a compilation of the implemented IEC features:

- Variables must be declared (similar to the declaration of variables in higher programming languages).
- Global and local data are distinguished from one another.
- I/O variables (global variables) may be defined.
- Programming means symbolic programming.
- The source code of a safety control program is structured in program organization units. User-defined function blocks may be programmed and instanced.
- The programming languages function block diagram (FBD) and ladder diagram (LD) are available for the development of the program code.
- The use of specially developed specific libraries for the safety control.

LIBRARIES IN THE SAFE PROGRAMMING SYSTEM

In accordance with IEC, you can reuse functions and function blocks from integrated libraries in the project you are working on.

Due to special safety requirements, only specially developed safe libraries may be used in the safe programming system. These libraries contain special reusable functions and FB-POUs. The file extension is *.swl.

The sub-tree 'Libraries' in the project tree is available for the use of libraries (that is, integration and removal). Each integrated library is displayed with its own symbol in this subtree.

Once a library has been integrated, the functions and function blocks it contains can be inserted from the editor assistant into a code spreadsheet by using drag and drop.

PROGRAM ORGANIZATION UNITS (POU)

Program organization units, or POUs for short, are the language elements of an SPS program. They are small independent software units which contain program code. The name of a POU must be unique, that is it may only be assigned once within a single project.

Two types of POUs are available in the safe programming system: a program and the user-defined function blocks (FBs), the number of which is up to the user. Each POU consists of two different parts: the variable declaration part and the code part. Both are referred to as 'spreadsheets'. All **local variables** are declared in the declaration part. The instruction or code part of a POU contains instructions programmed in the programming languages FBD and LD.

Function blocks are POUs with multiple input and output parameters and internal storage. The value which a function block returns as a result depends on the current value of its internal storage. Additional function blocks or functions can be called up in one function mode. Programs cannot be called up. Recursive calls are not permissible. The abbreviation for function block is FB.

In addition to all IEC-defined FBs and specific FBs for a safety control (saved in a library), the user-defined FB-POUs in the editor assistant are also available once the related spreadsheets have been processed, saved and compiled. This way, a call for a user-defined FB can simply be inserted into the code of the POU making the call by using drag and drop.

The act of calling an FB in another POU is referred to as instancing.

Programs contain a logical combination of functions and function blocks corresponding to the requirements of the control process. The behavior of programs is similar to that of functions modules. Programs possess input and output parameters and can have an internal storage.

Only one program per project is allowed in the safe programming system. This program is automatically inserted upon the creation of a new project. The program name 'Main' cannot be changed and the program can neither be copied nor deleted.

10.1.7.2 INSTANTIATION OF FUNCTION BLOCKS

In accordance with IEC 61131-3, the safe programming system offers the possibility of instantiation. Instantiation means that a function block is defined once and can than be used multiple times. This applies to all FBs equally: user-defined FBs (created in a FB-POU), IEC defined FBs and FBs in libraries.

Since function blocks always have an internal storage, their values must be saved each time the function block is called up in other storage areas. Instance names are used to accomplish this. The instance name is declared in the variable declaration of the POU in which the function is used. Each instance has a designator (the instance name) and possesses input and output parameters.

10.1.7.3 VARIABLES AND DATE TYPES

In accordance with IEC 61131-3, the programming is carried out by using variables instead of a directly addressing inputs and outputs or the use of flags. Variables are automatically declared in the variable spreadsheets when they are inserted into the code.

These variable spreadsheets are implemented as variable tables. The declarations are thus not stated in pure text form (as described in the IEC), but in table form, which makes the declarations much easier to manage. Each table contains a declaration of a variable or instance, each column in the table stands for a variable property (i.e. an element of the declaration). This way, the table reflects the complete declaration syntax in accordance with IEC 61131.

A variable's **area of validity** determines which POU a variable can be used in. The possible areas of validity are **local** and **global**. The area of validity of each variable is defined by the place where the variable is declared (local or global variable spreadsheet) and by the keyword used for the declaration.

Both types of variables are supported in the safe programming system: local variables (see below) and global variables (see ▹Page 140).

10.1.7.4 LOCALE VARIABLES

When a variable is only used within a single POU, it is called a local variable (its **area of validity** is local).

Locale variables must be declared by one of the key words VAR, VAR_INPUT or VAR_OUTPUT in the variable spreadsheet of the corresponding POU.

Since local variables cannot be connected to terminals (physical inputs and outputs), they are referred to as symbolic variables. The **symbolic** variables are deposited from the safe programming system in free storage areas of the safety control. The addresses are not known to the user. Symbolic variables may have an optional initial value.

FB instances are treated the same as local variables: their instance numbers must be declared with VAR.

Local variables can be overwritten in debug mode (see ▹FORCE AND OVERWRITE ◃ from page 127 onward).

10.1.7.5 GLOBAL VARIABLES

When a variable can be used in each POU of the projects, it is called a global variable.

Variables with a global **area of validity** must be declared in the global variable spreadsheet. The global variable spreadsheet **does not contain** the 'Use' column for selecting the IEC defined declaration keyword VAR_GLOBAL, since this keyword is automatically assigned to all global variables. (Note for advanced "IEC users": Global variables can be used in the safe programming system without the additional local VAR_EXTERNAL declaration.)

Global variables are only allowed as **I/O variables** in the safe programming system. This means that they must be connected to a terminal (a physical input or output). These variables are termed **addressed** variables in the IEC standard. However, the difference lies in that a logical address need not be manually entered in the safe programming system, rather the terminal is connected to the global variables using drag and drop for reasons of security. Furthermore, it is automatically checked whether the sizes of the global variable and the terminal fit together. If necessary, the table editor will adapt the data type of the global variables to the data type of the connected terminals.

As with the symbolic variables, I/O variables may also have an optional initial value (see next section). I/O variables can be forced in debug mode ▷FORCE AND OVERWRITE ◁ from page 127 onward.

10.1.7.6 INITIALIZING VARIABLES

Variables can be assigned initial values in accordance with IEC 61131-3. This means that a variable which is being used in the SPS program for the first time is called up with its initial value. Initial values can be assigned for local (symbolic) and global output variables. Global variables which have been assigned to a physical input cannot be initialized. The initial value entered must match the data type. For example, it is not possible to assign the initial value '5' to a variable from the data type BOOL. In such case, the system will display an error message during the compiling process. The initialization of variables is optional. If no initial value is used, the variable will be initialized with the standard initial value of the respective data type.

NOTICE!

Initial values must be inserted in the 'Initial value' column of the variable table.

10.1.7.7 KEYORDS FOR DECLARING VARIABLES

Variables are declared by means of keywords in accordance with IEC 61131-3.

In the safe programming system, the keywords for declaring variables are selected in the 'Use' column of the variable table.

The IEC-defined keywords which are relevant to the safe programming system are described in the following table:

Keyword	is used to declare
VAR	• local (symbolic) variables in the local variable spreadsheet of a POU. • FB instances in the local variable spreadsheet of a POU.
VAR_INPUT	variables which are input parameters of function block POUS. A value is transferred to an FB, from another POU for instance, by means of input variables. An input variable can only be a local (symbolic) variable. An input variable may only be read by the FB. Write access to input variables is not allowed in accordance with IEC 61131-3. For reasons of compatibility, though, no inspection is conducted to check whether a write access takes place during the compilation of the project.

Keyword	is used to declare
VAR_OUTPUT	Variables which are output parameters of function component POUS. An FB makes a value available (to another POU for instance) by means of output variables. An output variable can only be a local (symbolic) variable. It can be written and read by the FB.
Not visible in the safe programming system: VAR_GLOBAL	**Note:** Although this keyword is defined in the IEC standard for the declaration of global variables, it need not be manually selected in the global variable spreadsheet. Since all variables declared in the global declaration table are automatically global, the keyword VAR_GLOBAL is also automatically assigned to each variable declared there. For that reason, the 'Use' column is **not displayed** in the global variable spreadsheet. Global variables are always I/O variables (assigned to a physical input or output) in the safe programming system).

DATE TYPES IN THE SAFE PROGRAMMING SYSTEM

Date types establish the properties for the values of variable. They define the initial value, the area of the potential values and the number of bits.

The following elementary data types defined in IEC 61131-3 are available in the safe programming system:

Date type	Description	Size	Area
BOOL	Boolean	1	0...1
INT	Integer	16	-32768... 32767
TIME	Duration	32	0…2.147.483,647s
BYTE	Bit string of length	8	0…255 (0x00...0xFF)
WORD	Bit string of length	16	0...65.535 (16#00...16#FFFF)
DWORD	Bit string of length	32	0...4.294.967.295 (16#00....16#FFFFFFFF)

The following safe data types are also available:

Date type	Description	Size	Area
SAFEBOOL	Boolean	1	0...1
SAFEINT	Integer	16	-32768... 32767
SAFETIME	Duration	32	0…2.147.483,647s
SAFEBYTE	Bit string of length	8	0…255 (0x00...0xFF)

Date type	Description	Size	Area
SAFEWORD	Bit string of length	16	0...65.535 (16#00...16#FFFF)
SAFEDWORD	Bit string of length	32	0...4.294.967.295 (16#00....16#FFFFFFFF)

10.1.7.8 PROBLEMS AND SOLUTIONS

This section contains a list of potential problems which may occur when working with the safe programming system. It describes measures to be carried out for each problem and the user's necessary reactions.

The descriptions are arranged in **categories** corresponding to the various parts of the programming system which can report problems.

- General (applies to the entire programming system)(see below).
- Code editor and variable editor (see ▷Page 144).
- Project tree (see ▷Page 144).
- Device parameterization editor (see ▷Page 145).
- Compiler (see ▷Page 145).
- Online communication between the programming system and the safety control (see ▷Page 146).
- Messages from the safety control (see ▷Page 147).

GENERAL

Problem	Solution
The installation inspection has detected a defective system file while starting the safe programming system. A corresponding message window is displayed.	Deinstall the safe programming system and start the setup program from the installation CD in order to reinstall the software.
The operating system inspecting routine has detected that you wish to start the safe programming system on an operating system which is not supported.	Install an operating system which is supported by the safe programming system or ask our technical support if a newer version of the safe programming system is available which your current operation system supports.
An error has occurred (accompanied by a corresponding message) which cannot be rectified by any measure described here.	Please consult our technical support.
The safe programming system or one of its functionalities does not behave as described in the user documentation or in the online help.	Please consult our technical support.

CODE EDITOR AND VARIABLE EDITOR

Problem	Solution
You have tried to open a spreadsheet, but the spreadsheet cannot be loaded due to a check sum error. The editor displays a message window.	The affected POU is damaged and must be deleted. If a 'Main' POU is involved (which cannot be deleted), the project cannot be used any longer. Revert to your last backup copy of the project (as described in ▹DECOMPRESSING AN ARCHIVE FILE ◃ on page 136).
The editor reacts unexpectedly to an entry in a spreadsheet. For example: you have inserted a coil, but a contact is displayed. This can be attributed to a faulty operation, a sporadic error or a systematic error.	Undo your last entry (press <Ctrl>+<Z> to do so) and repeat the entry. If the result is incorrect again, please consult our technical support.
A message window appears during the editing process in which the editor reports damaged file, a sporadic error or a systematic error.	The spreadsheet will automatically be closed. You will not have the opportunity to save the last changes implemented.
You have tried to open the global variable spreadsheet, but the variable spreadsheet could not be loaded due to a check sum error. The editor displays a corresponding error window.	The project can no longer be used, since the variable spreadsheet for global declarations cannot be deleted. Revert to your last backup copy of the project (as described in ▹DECOMPRESSING AN ARCHIVE FILE ◃ on page 136).

PROJECT TREE

Problem	Solution
After copying a POU in the project tree (via the clipboard), the POU does not contain the expected code or variable spreadsheet.	Delete the defective POU copy and copy it again. If the problem persists, please consult our technical support.

DEVICE PARAMETERIZATION EDITOR

Problem	Solution

COMPILER

Problem	Solution
A spreadsheet cannot be read by the compiler due to a check sum error. A corresponding error message will be displayed in the message window.	The affected POU is damaged and must be deleted. If a 'Main' POU is involved (which cannot be deleted), the project can no longer be used. Revert to your last backup copy of the project (as described in ▹DECOMPRESSING AN ARCHIVE FILE ◃ on page 136).
The second compiler has discovered an error in the project structure and displays a corresponding error message in the message window.	Please consult our technical support.
A compiler has detected a syntactic or semantic error in the user program and displays a corresponding error message in the message window.	Open the affected spreadsheet and correct the error. Spreadsheets containing an error can be open directly from the message window by double clicking on the error message. The location of the error (the object/element) is automatically highlighted in the spreadsheet.
A compiler displays an error message in the message window, the cause of which you are unable to rectify (i.e. 'Internal error').	Try to compile the project again. If the error is reported again, please consult our technical support.

ONLINE COMMUNICATION BETWEEN PROGRAMMING SYSTEM AND SAFETY CONTROL

Problem	Solution
An error occurs while sending the project to the safety control or it is not possible to establish a connection to the safety control at all. A message window is displayed in both cases.	Check the connection settings and the cabling and try it again. If the problem occurs as before: Try to establish a connection to another safety control which is not being used (if available). If this is successful, replace the defective safety control. If the problem occurs as before: Try to establish a connection to the safety control from a different PC. If this is successful, there may be a problem with the network adaptor of your computer. If the problem persists, please consult our technical support.
After successfully sending the project, the programming system discovers that the check sum of the project on the safety control does not correspond to the check sum of the project on the PC. A corresponding message window will be displayed.	Send the project again. If the problem occurs as before: Try to send the project to another safety control which is not being used (if available). If this is successful, replace the defective safety control. If the problem persists, please consult our technical support.
You are trying to switch into debug mode via the control dialog, but the status remains 'Safe'.	If the status is 'Safe [STOP]' or the RUN/STOP switch is set to STOP, you will have to set the RUN/STOP switch to RUN. Please note that the safety program will start running during the process. You will then be able to switch into debug mode.
After a debug timeout (10 min. after disconnecting online communication in debug mode), the safety control can no longer be started.	Reset the hardware or turn the power off and back on again, or send the project to the safety control again. It will then start again.

SAFETY CONTROL MESSAGES

Problem	Solution
The safety control reports an error occurring during the process of translating the first compiler's program (the second compiler's program is already completely translated on the PC).	Please consult our technical support.
The safety control reports an internal error.	Please consult our technical support.

10.2 Configuration (Bus configurator)

10.2.1 Bus navigator and safe parameterization

The bus navigator lists all safe input and output devices which have been configured in the ProMaster project in a tree diagram. These can be the local IOs to the safety control, the remote IOs to bus couplers and safe drive systems.

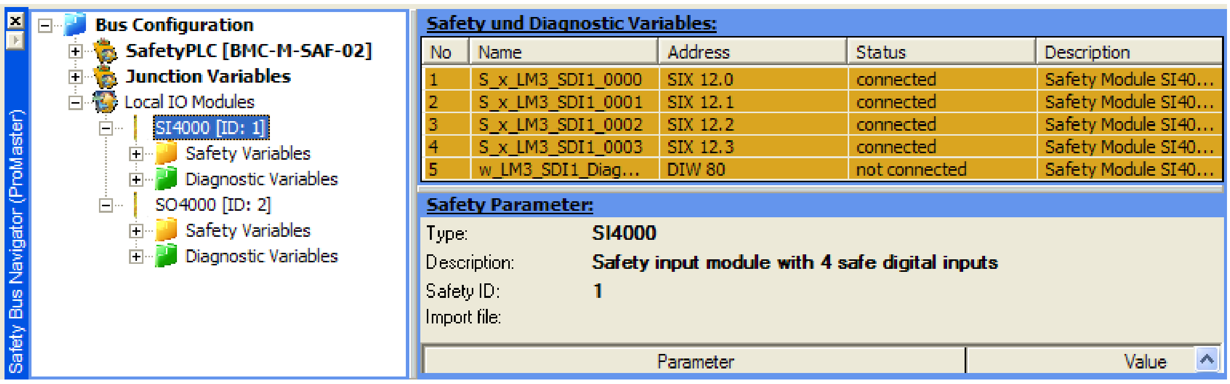


Figure 117: Bus configurator

The system-wide unique safety ID is entered in brackets next to the device designation (see also chapter ▷Address/ID allocation and safe station numbers ◁ from page 156 onward). Devices cannot be added to or deleted from the bus navigator, nor can variables be changed. This is only possible in ProMaster. The bus navigator will then overtake the changes made in ProMaster.

If the user expands the display on a device ("+"), it will display all IO variables and diagnostic variables applied for this device which relate directly to safe input and output data present in the device and the diagnosis information provided for this purpose. The user can now select a variable entry with the mouse and drag it into the ProSafety editor window (drag and drop). If this entry has not yet been connected with any IEC variables in ProSafety, a variable entry dialog will be opened in ProSafety and a variable name from the bus navigator will be suggested for the IEC variable. The user will normally accept this suggestion. The IEC variable will then have the same name as the IO variable or diagnostic variable applied in ProMaster and displayed in the bus navigator. However, the user also has the option of changing the name of the IEC variable name in ProSafety or of connecting the bus navigator variable to an already existing IEC variable. In this case, the variable in ProSafety will have a different name than in ProMaster and the bus navigator. It is not possible to overtake the variable name from ProSafety into the bus navigator. In any case, the reference to the IO variable is given via the specified designation of the connecting terminal in the variable list in ProSafety in the “Connecting terminal” column. With diagnostic variables, a unique designation of the diagnostic information is also entered in this column so as to enable the allocation between the IEC programming system and the bus navigator/ProMaster.

The address of the IEC variables in the safety input map or in the safety output map of the safe OS will also be assigned by the bus navigator when the above-mentioned user action is carried out, and will be invisible to the user.

The safe device parameters can be parameterized using the options presented in the right part of the bus navigator (in the window area under 'Safety parameters').

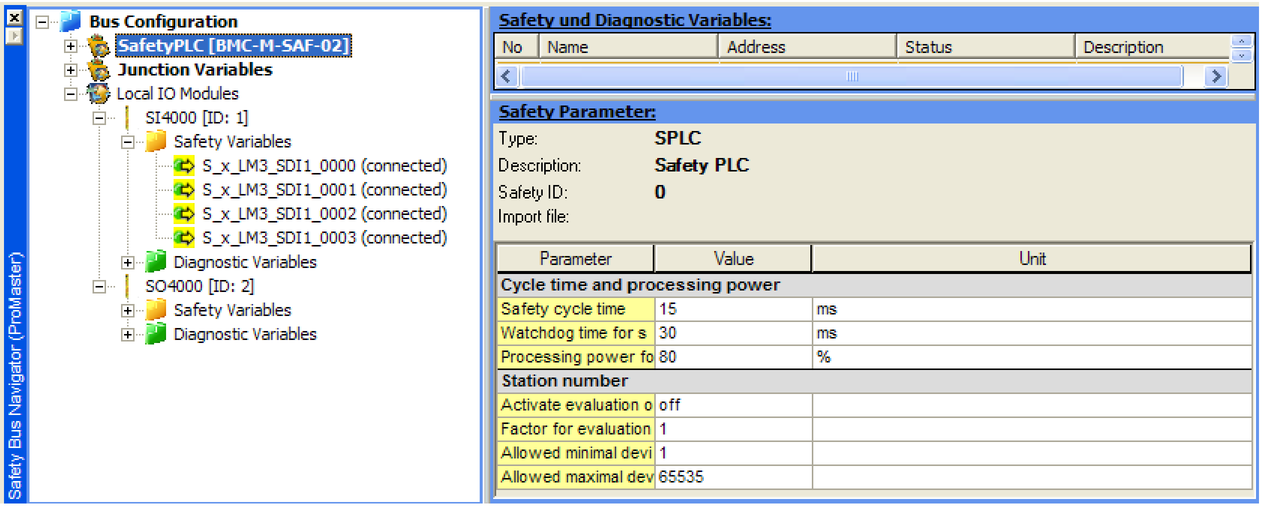


Figure 118: Setting the safety parameters

The user can call up the safe parameterization in the bus navigator by selecting a device in the bus navigator window area. He can then conduct the safe parameterization of this device. General settings which must be parameterized safely, such as the number range for a safe station number (see ▷Address/ID allocation and safe station numbers ◁ from page 156 onward), are conducted on the device "Safety PLC".

As the first step, after a further device has been inserted in the bus configuration in ProMaster, the user confirms the insertion of the new device upon starting the bus navigator. The safety ID and device type for the device created by ProMaster are displayed in a message. Once the user confirms the correctness of the statements, the insertion of the device into the project will be confirmed safely. This will be important later in checking the bus configuration.

The user can then set the existing parameters for the selected device. Parameters which are not changed by the user will retain their default value which is provided in the device file from Baumüller. A safe parameter can be a watchdog time, to name one example.

The entire safe parameterization for all devices is written in total by the bus navigator into the parameterization file, which is transferred to the safety PLC together with the safe IEC application program and the bus configuration during the project download.

The integration of the bus navigator and the safe parameterization is presented summarily in the following ▷Figure 119◁.

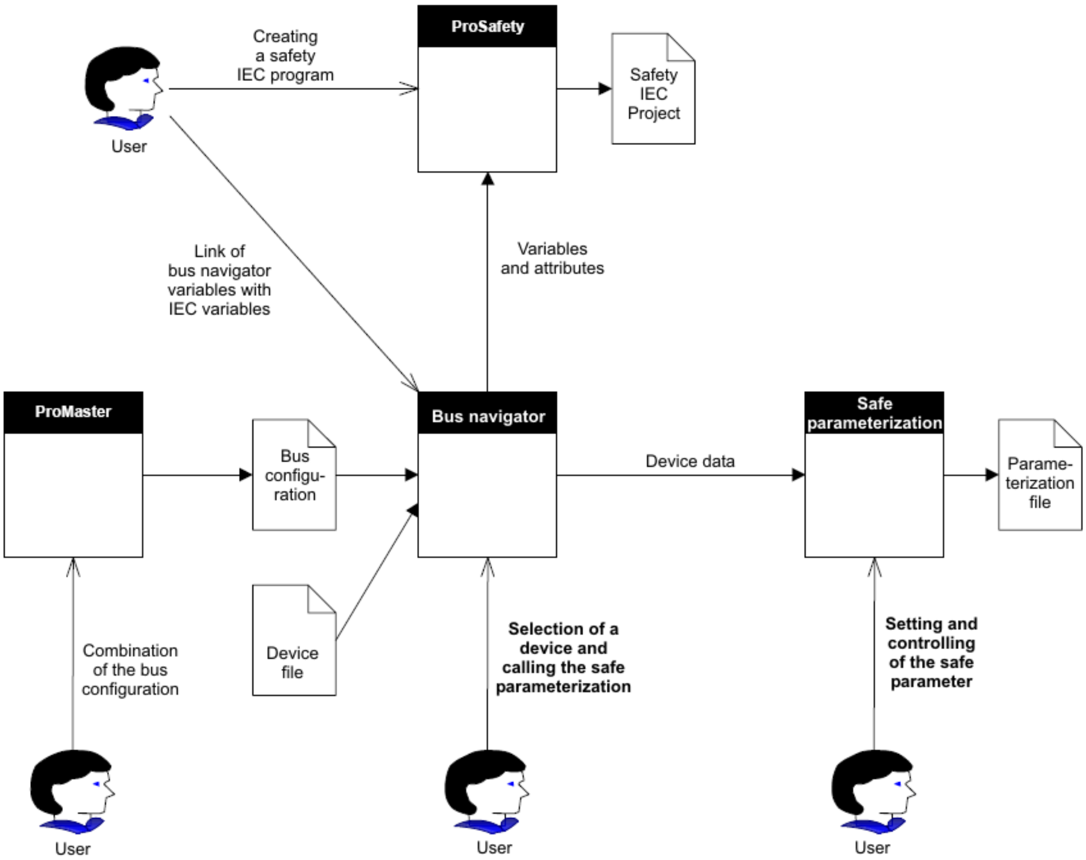


Figure 119: Integration of the bus navigator and safe parameterization

10.3 Parameterization of the safe device parameterization editor)

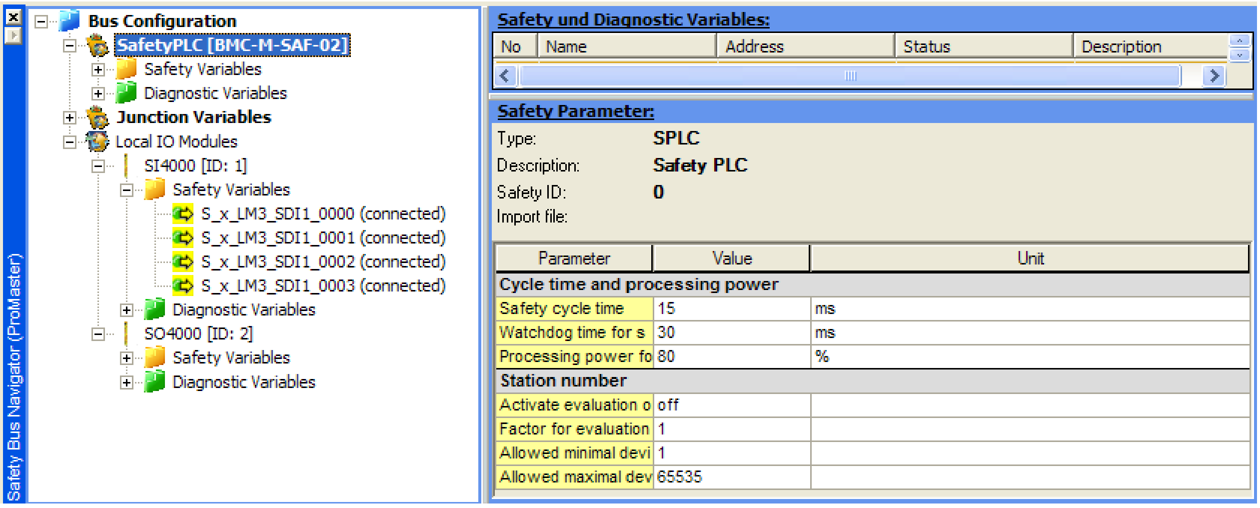


Figure 120: Setting the safe parameters

Parameter	Value range	Unit	Description
Cycle time and processing power (see ▷Setting cycle times ◁ from page 154 onward)			
Safety cycle time	1-65535	ms	The time it takes to completely process a cycle in the OmegaSafe runtime system incl. the safety application program and safe communication. The parameter states after which time the next safety cycle will be started, assuming all program parts in the current safety cycle have been processed. The safety cycle time is the complete mount of time between the first safety time slice which has called up the first safety block up to the last time slice which has run the last part of the last safety block. It is not (!) the added processor execution time for the safe part.
Watchdog time for safety cycle	1-65535	ms	The processing of the safety cycle is monitored temporally. The maximum limit for this is the watchdog time. If the execution takes longer than this time, safe status will be engaged.

Parameter	Value range	Unit	Description
Processing power for safety	10 – 90	%	These parameters can be used to set the proportion of the PLC processing power for the safety part. Higher values mean that more processing power will be available for the safety part. In doing so, take care that the IEC program in the motion control event task on the standard PLC is programmed in such a way that it does not take longer than is shown in the above-mentioned parameters for the standard part. Example: Processing power for safety at 60 % → Performance processing for standard 40 %. In such case, the motion program cannot take longer than 40 % * 2 ms = 0.8 ms during a motion interrupt cycle of 2 ms. It should even take significantly less time than this 0.8 ms, since the remaining time difference for the other tasks remain on the standard PLC. If the motion control program takes longer than 0.8 ms, the time for the safety part will be shortened and the safety cycle time will increase. If the watchdog time is exceeded, the safety control will engage in safe status (see ▷Setting cycle times ◁ from page 154 onward).

Parameter	Value range	Unit	Designation
Station number (see ▷Address/ID allocation and safe station numbers ◁ from page 156 onward)			
Activate station number evaluation	In, out	-	In: The station number set is taken into account during safe communication and influences the device IDs used. Out: The station number set is not taken into account during safe communication. The device IDs are identical to the safety IDs on the project.
Factor in station number evaluation	1 – 32767	-	Factor for the calculation of device IDs. Device ID = Safety ID + station number * factor (see description “Addressing with the safe station number” on page ▷Page 158)
Minimum device ID allowed	1 – 65535	-	The minimum device ID which can result in the use of the station number for the creation of device IDs (minimum safety ID in project + minimum station number * factor).
Maximum device ID allowed	1 – 65535	-	The maximum device ID which can result in the use of the station number for the creation of device IDs (maximum safety ID in project + maximum station number * factor).

10.4 Meaning of the diagnostic variables

EtherCAT master

Variable	Value / Bit	Meaning
w_SPLC_Diag_M*_01	Bit 0 – 3	0: Power on 1: Init 2: Pre-operational 4: Safe-operational 8: Operational E: Fatal error F: Reset
	Bit 4 – 15	Reserved

The normal operating status when the EtherCAT bus is running is “operational”. All other statuses are start-up statuses or error statuses.

Safety IOs

Variable	Value / Bit	Meaning
w_LM*_SDx_Diag_01	Bit 0	0: Configuration data of the safety device is invalid 1: Configuration data of the safety device is valid
	Bit 1	0: Safety communication has not yet started running to the safety device 1: Safety communication has started running to the safety device (display saved)
	Bit 2	0: An error has not occurred 1: An error related to the safety device has occurred (saved display)
	Bit 3	0: An error has not occurred 1: An error related to the safety device has occurred (current display)
	Bit 4	0: Safe parameter download to the safety device not active 1: Safe parameter download to the safety device active
	Bit 5	0: Safety communication in another status 1: Safety communication running to the safety device in “FailSafeData” status

Variable	Value / Bit	Meaning
	Bit 6	0: Safety communication in another status 1: Safety communication is running to the safety device in “ProcessData” status (normal operation status)
	Bit 7	0: No error reported by safety device 1: Safety device is reporting an error (FailSafeData)
	Bit 8 – 15	Reserved

x stands for I with the input module and for O with the output module

Safety drives

Variable	Value / Bit	Meaning
w_SD*_Diag_01	Bit 0 – 15	As with safety IOs

Variable	Value / Bit	Meaning
w_SD*_DiagSMC_01	Bit 0	0: Safety drive has not been initialized by safety motion control 1: Safety drive has been initialized by safety motion control
	Bit 1	0: No safety motion command for the axis active 1: Safety motion command for the axis active
	Bit 2	Reserved
	Bit 3	Reserved
	Bit 4	Reserved
	Bit 5	Reserved
	Bit 6	Reserved
	Bit 7	0: An error has not occurred 1: An error in safety motion control related to the safety drive has occurred (current display)
	Bit 8 – 15	Reserved

10.5 Setting cycle times

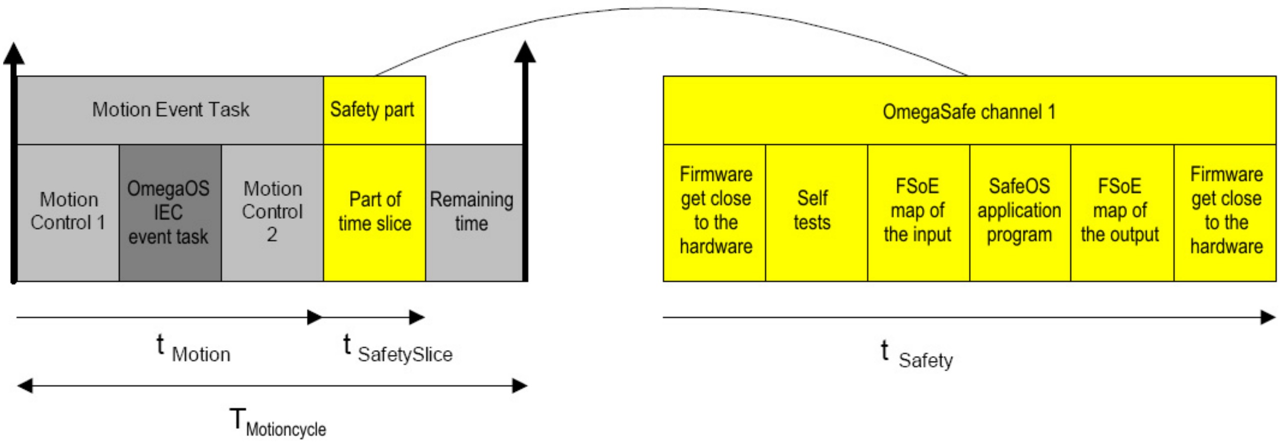


Figure 121: Time schema cycle times

The time schema is arranged as follows: The user configures a cycle time for motion control function (z. B. $T_{Motioncycle}$= 1, 2, 4 or 8 ms) in ProMaster (PLC configuration). This creates cyclic synchronization signals (in ▹Figure 121◃ illustrated as thick vertical arrows) which call up a high-priority event task. The interval created by this is divided into the following blocks in a simplified manner.

The first system component is called up in the standard area of the safety control as the first for the standard motion control, followed by the IEC user program of the motion event task. The second part of the motion control system follows thereafter. After being completed (after t_{Motion}), it is switched to the safe component. In the process, a section from the program area depicted on the right in ▹Figure 121◃ will be executed in the time available for the safe part ($t_{SafetySlice}$). One block of this is the user's Safety IEC program. The time remaining after the safe part has ended will be available for the further tasks and system functions of the standard IEC system (remaining time). It should be noted that the motion execution time (t_{Motion}) is not constant, but rather depends on the activated functions and user program. Only the time left over can be used for safety and the remaining time. The duration of t_{Motion} and $t_{SafetySlice}$ together may never become greater than the motion cycle time $T_{Motioncycle}$.

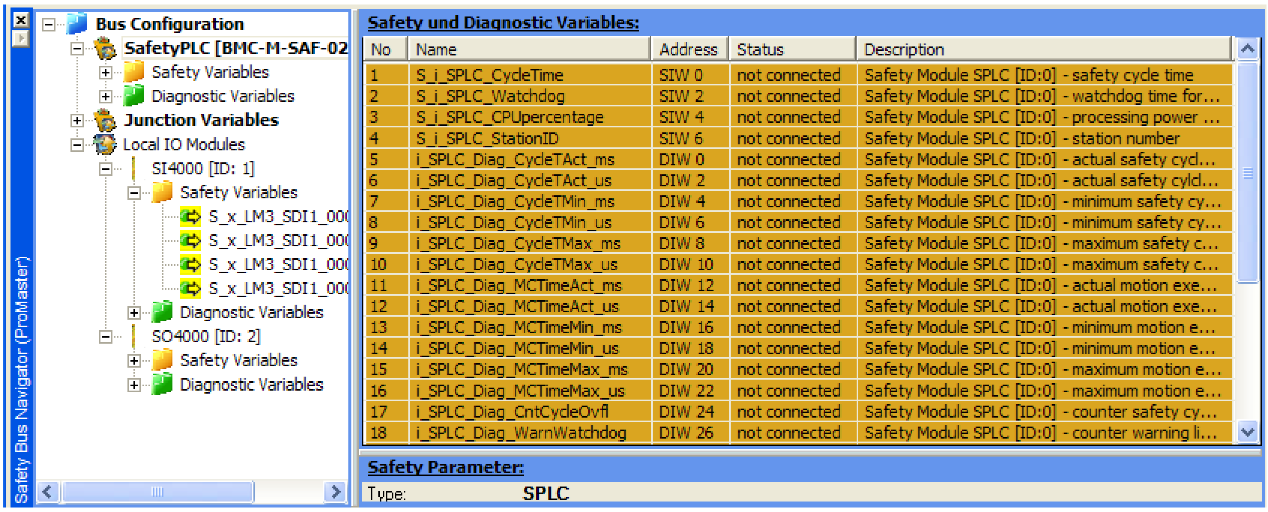


Figure 122: Safety and diagnosis variables

The specified measurement values are displayed for the user in the following table.

Value displayed	Function
Safety cycle time measurement value	After each safety cycle, the time actually needed to process all safety blocks is displayed on diagnostic variables (i_SPLC_Diag_CycleTAct_ms). In addition, a minimum value and maximum value storage of the execution time is displayed on diagnostic variables (i_SPLC_Diag_CycleTMin_ms and i_SPLC_Diag_CycleTMax_ms). This can serve to assist in setting the processing power distribution, the safety cycle time and the safety watchdog time.
Motion execution time measurement value (t_{Motion})	In each motion interrupt, the execution time measured for this interrupt for the entire motion part (i_SPLC_Diag_TimeAct_ms) is displayed on diagnostic variables. In addition, a minimum value and maximum value storage of the execution time is displayed on diagnostic variables (i_SPLC_Diag_TimeMin_ms and i_SPLC_Diag_TimeMax_ms). This can serve to assist in setting the processing power distribution and the safety cycle time.
Exceeding safety cycle time	A counter which states how often the safety cycle has taken longer than the set safety cycle time is displayed as a diagnostic variable (i_SPLC_Diag_CntCycleOvfl). This can serve to assist in setting the processing power distribution, the safety cycle time and the safety watchdog time.
Safety watchdog warning threshold	A counter indicating how often the safety cycle has taken longer than 80 % of the time set as safety watchdog time is displayed as an additional diagnostic variable (i_SPLC_Diag_WarnWatchdog). This can serve to assist in setting the processing power distribution, the safety cycle time and the safety watchdog time.

NOTICE!

These variables are available in both safe programming (ProSafety) and standard programming (ProProg).

The processing power distribution, safety cycle time and safety watchdog time can additionally be read as safe PLC parameters via safe variables in the input image for the safety IEC application.

10.6 Address/ID allocation and safe station numbers

There are two procedures to distinguish from which the user can select for his machine for the unique addressing of the safe field bus participants: with and without safe station number. In compliance with the system-wide unique address allocation, the two procedures can also be implemented in mixed use on a single machine which is made up of multiple safety controls.

10.6.1 Addressing without station number

The procedure without station number means that the user creates a configuration in ProMaster for all safe and non-safe devices of the entire machine and that an individual IEC program with individual configuration is present on each safety control in the machine.

In this case, the safety IDs with which the devices are uniquely designated in ProMaster and ProSafety will be the same as the device IDs with which the actual existing devices are distinguished.

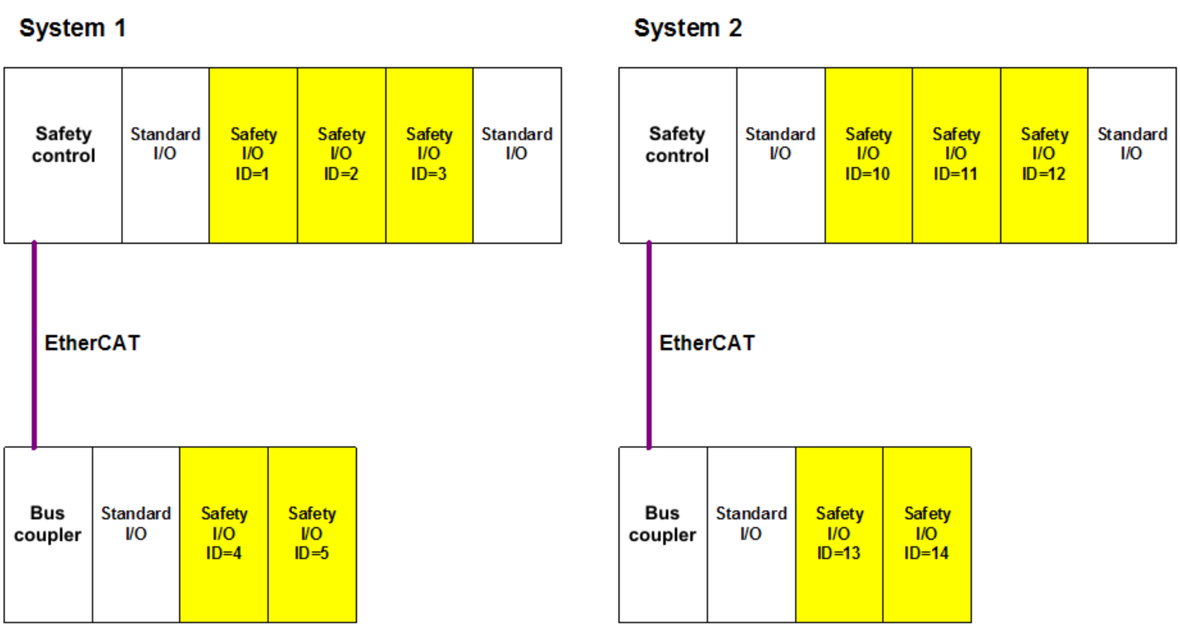


Figure 123: Example ID allocation without station number

A sample configuration

h two safety controls is provided in In ▷Figure 123◁. In the procedure described above, safety IDs 1 through 5 in the ProMaster project are configured for system 1 and safety IDs 10 through 14 are configured for system 2. Precisely these IDs must be set as device IDs with the safety IO modules. This interrelation is displayed in the following table.

ProMaster project safety IDs	Device IDs
System 1	
1	1
2	2
3	3
4	4
5	5
System 2	
10	10
11	11
12	12
13	13
14	14

Setting the IDs:

The device IDs can be set by means of either dip switches or safe parameterization in the devices. The position of the dip switches can be found in the operation manual of the respective devices.

With the safety protocol FSoE, the protocol parameter "Connection ID" is the same as the safety ID.

Uniqueness range

CAUTION!

The device ID must be in the unique in the range which can be reached by the communication network used for the safety protocol. Networking multiple machines in a standard ethernet network should also be taken into consideration.

NOTICE!

The addressing procedure without safe station number can be used for safe IO modules as well as for safe drives. In the case of the safe drive, the term axis ID is introduced for the safety motion control function block for ProSafety. This axis ID is identical to the safety ID used above and also the same as the device ID.

10.6.2 Addressing with safe station number

The procedure with the station number means that the user creates a configuration in ProMaster for all safe and non-safe devices of the entire machine, but that the identical safe (and non-safe) IEC program is present on multiple safety controls in the machine. The machine modules which should receive the identical safe IEC program must also be configured identically (at least as far as the safe components and safe functionality are concerned).

In this case, the device ID, which are used to distinguish the devices which are actually present, are created by an established algorithm from the safe station number and the safety IDs which are used in ProMaster and in ProSafety.

The device ID thus computes as:

Device ID = Safety ID + station number * factor.

The "factor" is a safe parameter of the safety PLC which is safely parameterized for the applicable project via the bus navigator.

The stations number is set on the safety PLC via a safe course of operation by means of buttons, rotary switches and seven segment displays and saved remanently. See the b maXX safe PLC operation manual for more on this.

As before, an overall configuration will be compiled in ProMaster. With the safety controls, a station number will additionally be configured (The value 0 is not allowed in the process! This value shuts off the use of the station number). In ProSafety, the same safety IDs are used for each of the safety controls with station numbers and the same safety IDs are also used in the configuration file for the safe stack. This means that both the user’s IEC program and the configuration file for the safe stack can be identical on all safety controls for which a station number has been set.

In addition, an entry which activates the conversion of the safety IDs to actual device IDs by means of the safety numbers is carried out in the configuration file for the safe stack.

For this purpose, there are the following statements, which the user parameterizes via the bus navigator:

- Factor for the multiplication of the station number
- With the FSoE, it is also specified which minimum and maximum connection ID (=device ID) may result for the permissible station numbers. (So that the stack can also implicitly check permissible setting range for the safe station numbers.)

The stack the communications outwardly to the devices with the calculated device ID (=connection ID with FSoE). The device IDs are uniquely allocated to the safety IDs used in ProSafety by means of the conversion rule above. The safe stack conducts data storage in the safety input image or safety output image of the safe OS identically on each safety control with station number.

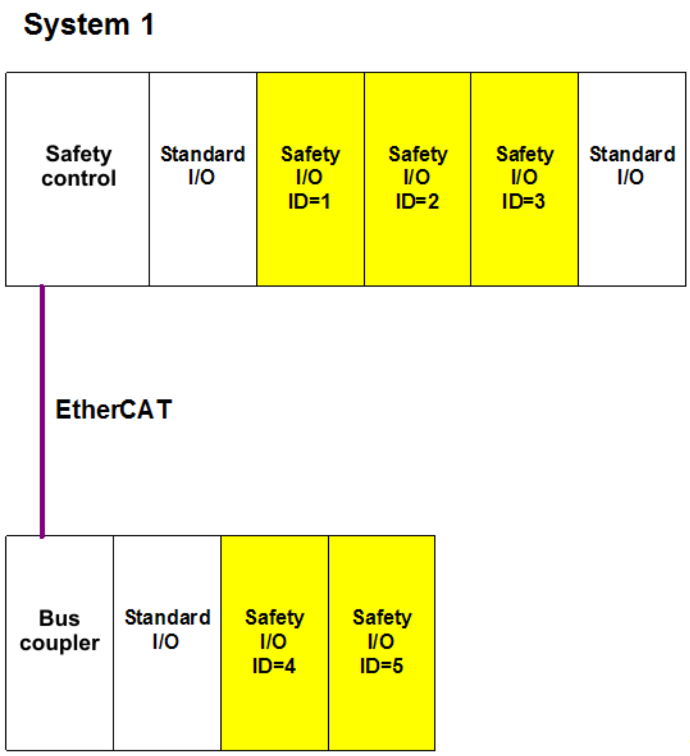


Figure 124: Sample ID allocation in ProMaster

A sample configuration is provided in ▷Figure 124◁. Only the safety IDs in the range of 1 through 5 are configured in ProMaster.

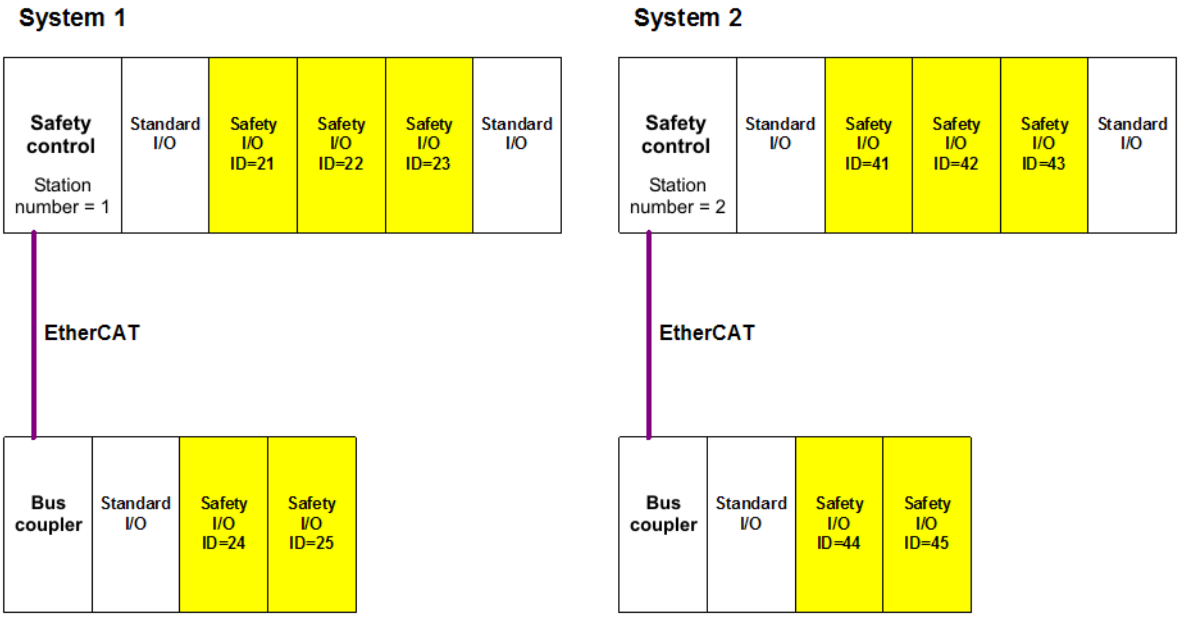


Figure 125: Sample ID allocation with safe station number

A corresponding example with two safety controls which should receive the same safety IEC program and the same configuration files is provided In ▹Figure 125◃. The station number 1 is set on the safety control for system 1. The station number 2 is set on the safety control for the second system. In the example, it is accepted that the factor for the calculation of the device IDs is set to 20. This then results in the specified device IDs which are shown in the following table for comparison (calculation device ID = safety ID + station number * factor).

ProMaster Safety IDs	Device IDs for system 1	Device IDs for system 2
1	21	41
2	22	42
3	23	43
4	24	44
5	25	45

Setting the device IDs and uniqueness range:

CAUTION!

Setting the devices IDs and their range of uniqueness is specified without station number, just as in the in the procedure above.

NOTICE!

The addressing procedure with safe station number can be used for both safe IO modules and safe drives. In the case of the safe drive, the term axis ID is introduced for the safety motion control function block for ProSafety. This axis ID, which is connected to the function block, is identical to the safety ID used above. The resulting device ID is to be set on the drive as with the IO modules.

10.7 Calculation of the maximum reaction time for a safety function

The warranty for the observance of the maximum reaction time of a safety function happens essentially via considering the adjustable watchdog times of the used safety components.

Both the safety control and the safe input and output terminals possess a safety-related parameter to adjust the watchdog time respectively. The watchdog time of the input and output terminals defines the maximum period of a total communication cycle. The watchdog time of the safety control watches the compliance of the maximum allowable period from reading the input map via processing the application program up to writing the output map.

If the watchdog time is exceeded in one component it takes the safety state.

The watchdog time of the safe PLC is set via the safe device parameterization editor according to ▷Chapter 10.3◁ of this manual

The setting of the watchdog times of the communication participants (input and output terminals) you will see in the according documentations.

The maximum reaction time for a safety function which is requested by a sensor and triggered by an actuator (e.g. contactor) results from the following diagram:

Figure 126: Chain to determine the maximum reaction time

T_S	Delay time from the sensor
$T_{I\text{-terminal}}$	Delay time up to the input data will be in the output map of the communication memory of the input terminal
$WDT_{I\text{-terminal}}$	Watchdog time of the input terminal
WDT_{PLC}	Watchdog time of the safe PLC
$WDT_{O\text{-terminal}}$	Watchdog time of the output terminal
$T_{O\text{-terminal}}$	Delay time up to the output data from the communication memory will be assumed and lie at the output of the output terminal
T_A	Delay time from the actuator

10.7 Calculation of the maximum reaction time for a safety function

For the maximum reaction time applies:

$$T_{Reaction(WC)} = T_S + T_{I\text{-}terminal} + 2 \times WDT_{I\text{-}terminal} + 2 \times WDT_{PLC} + 2 \times WDT_{O\text{-}terminal} + T_{O\text{-}terminal} + T_A$$

10.7.1 Example for setting the watchdog times

Within the risk analysis for the safety function of an example application a maximum reation time of 100 ms is determined.

With the safety system must be ensured, that the determind reaction time will be adhered to always. This requirement must be fulfilled in a fault also.

The following delay times muss be known in addition to the watchdog times:

T_S	= 5 ms (only exemplary value)
$T_{I\text{-}terminal}$	= 3 ms (only exemplary value)
$T_{O\text{-}terminal}$	= 2 ms (only exemplary value)
T_A	= 10 ms (only exemplary value)
Sum	= 20 ms

You will find this values in the according documentations of the connected components. Exemplary values are assumed for evaluation the watchdog times to be set.

Now remain for the watchdog times to be set:

$$100\text{ ms} - 20\text{ ms} = 2 \times WDT_{I\text{-}terminal} + 2 \times WDT_{PLC} + 2 \times WDT_{O\text{-}terminal}$$

$$80\text{ ms} / 2 = WDT_{I\text{-}terminal} + WDT_{PLC} + WDT_{O\text{-}terminal}$$

$$40\text{ ms} = WDT_{I\text{-}terminal} + WDT_{PLC} + WDT_{O\text{-}terminal}$$

If an watchdog time for the safe PLC of e.g.

$$WDT_{PLC} = 10\text{ ms}$$

is set due to the timing of the processing power, the motor cycle and the set safety cycle time (see ▷Chapter 10.5◁ of this manual), for the terminals remain with equal distribution

$$WDT_{I\text{-}terminal} = WDT_{O\text{-}terminal} = 15\text{ ms}$$

With the setting of this values it is secured, that the calculated maximum reaction time of 100 ms values of this example will be never exceeded.

NOTICE!

The values in this example are virtual. you will find the real values in the according user documentations of the used terminals, sensors and actuators.

10.8 Error Codes

usError Code (hex)	usAdd Error Code	ulExtErrorCode	Meaning
A000	3	-	Wrong stack type (<> 0x53414630 = 0x"S""A""F""0")
A000	20	10	Conflict of versions required incompatible version of OmegaSafe code > incompatible version of OmegaSafe code
A000	20	11	Conflict of versions required compatible version of OmegaSafe code > compatible version of OmegaSafe code
A000	20	20	Conflict of versions required incompatible version of FSoE stack code (SFM) > incompatible version of FSoE Stack Codes (FSoE Stack Code)
A000	20	21	Conflict of versions required compatible version of FSoE stack code (SFM) > compatible version of FSoE Stack Codes (FSoE Stack Code)
A000	20	22	Conflict of versions the FSoE Stack Code does not support the version (incompatible version) of the FSoE configuration file
A000	20	23	Conflict of versions the FSoE Stack Code does not support the version (compatible version) of the FSoE configuration file
A000	24	-	OmegaSafe FSoE memory: Number of safe devices = 0
A000	25	-	OmegaSafe FSoE memory: Number of safe devices > 255
A000	30	Index of the loop	Safety ID = 0
A000	32	Index of the loop	ConnectionID outside of the valid area (in SAFEGRID parameterized)
A000	33	Index of the loop	Overflow at calculation the ConnectionID.
A000	34	Index of the loop	Double ConnectionID
A000	37	Index of the loop	Safety ID assigned several times

usError Code (hex)	usAdd Error Code	ulExtErrorCode	Meaning
A000	80	requested Safety ID	Error at reading of SDevPara.saf - general
A000	81	requested Safety ID	Error at reading of SDevPara.saf - Safety ID (not existing)
A000	82	requested Safety ID	Error at reading of SDevPara.saf - Object (not existing)

APPENDIX A - ABBREVIATIONS

CAN Controller Area Network

CBPB Controller Based Parallel Bus

CPU Central Processing Unit

DC Diagnostic Coverage

DPRAM Dual Ported RAM

DRAM Dynamic RAM

EMV Electromagnetic compatibility

EN European standard

ESD Electrostatic sensitive device

EXT, ext External

I/O Input/Output

I/O-Bus Bus for input and output module; **I**/**O**

LED Light emitting diode

MTTF$_d$ Mean Time To Failure

NOVRAM **No**n-**v**olatile RAM

OPC OLE for Process Control (OLE: Object Linking and Embedding)

PFD Probability of Failure on Demand (mean residual error for a dangerous error upon request))

PFH Probability of Failure per Hour (Residual error rate for a dangerous error per hour)

PLC Process loop control, storage programmable control, SPS

PROPROG wt II Tool for programming b maXX PLC (BMC-M-PLC-01)

ProProg wt III Tool for programming b maXX PLC (BMC-M-PLC-01 and BMC-M-PLC-02)

RAM Random access memory

RISC Reduced Instruction Set Computers

SDRAM Synchronized Dynamic RAM

SFF Safe Failure Fraction (Fraction of failures leading to safe status)

SIL Safety integrity level

SW Software

USS® Siemens trademark, universal serial interface

Index

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**