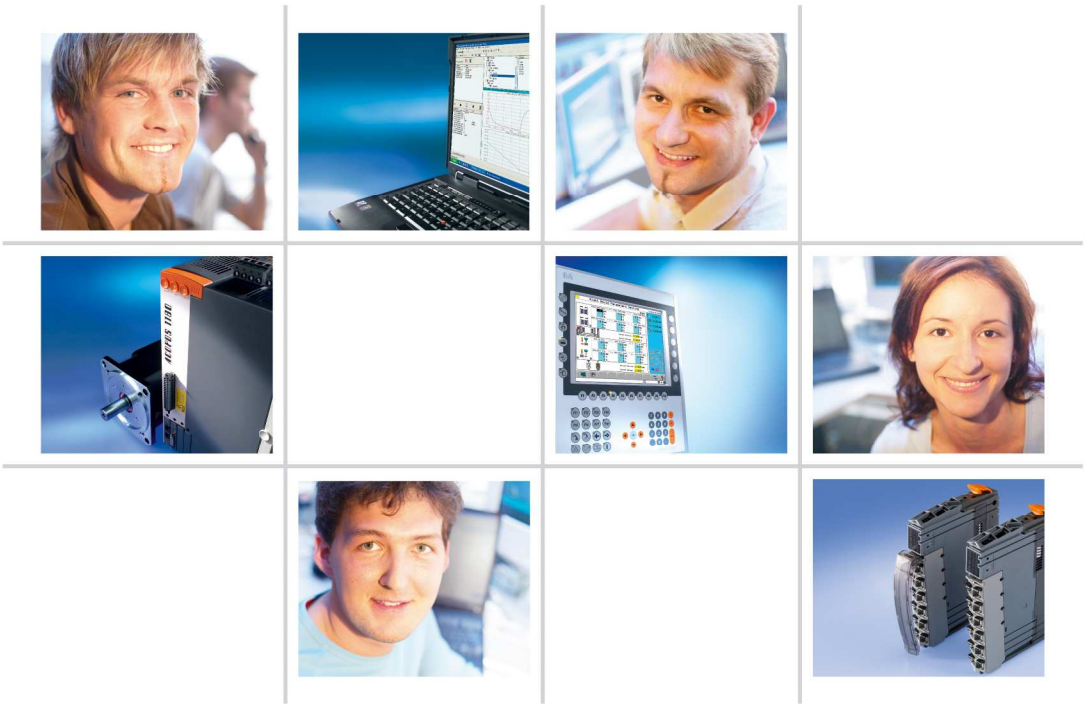
ASiST SafeDESIGNER

TM510



Perfection in Automation

www.br-automation.com

Requirements

Training modules:	TM210, TM500
Software:	Automation Studio 3 with SafeDESIGNER Toolset
Hardware:	CPU with POWERLINK V2 Interface / Interface card

Table of contents

1. INTRODUCTION

The training module "ASiST SafeDESIGNER" will help to familiarize you with the SafeDESIGNER.

Throughout this training course, we will deal with the aspects of safety-oriented application development and explain the features available in Automation Studio and in SafeDESIGNER.

We will also go over the documentation requirements of the safety application and at the end we will implement a realistic sample project with a standard and safety application.

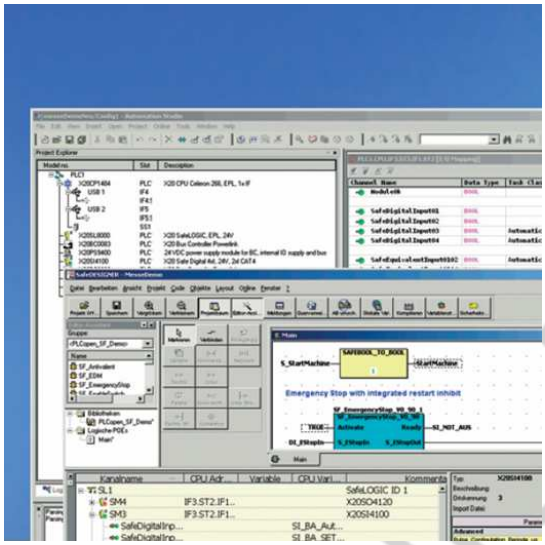


Fig. 1: SafeDESIGNER

Using a programmable safety solution opens up new possibilities for handling errors. SafeDESIGNER enhances Automation Studio by including all of the functions needed for developing a simple, efficient and cutting-edge safety application.

1.1 Objective

The goal of this training module is to become familiar with using SafeDESIGNER as well as the procedure and aspects involved in developing a safety-oriented application.

You will receive an overview of the features available in Automation Studio and SafeDESIGNER.

You will learn about the documentation requirements of a safety application.

You will also learn how to create a safety application and how to perfectly configure the application to meet your own requirements.

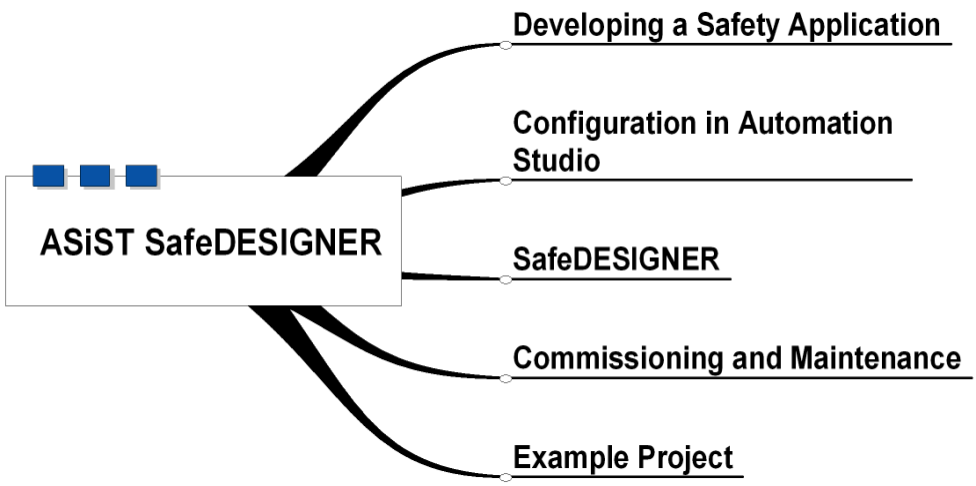


Fig. 2: Overview

Caution:

In order to correctly implement a safety application, the applicable regulations and standards absolutely must be adhered to in all phases of the safety application's life cycle. This training module only covers the use of SafeDESIGNER. Therefore, this training module can in no way replace a sound safety education.

2. DEVELOPING A SAFETY APPLICATION

A V-model is very useful for viewing the development of a safety application.

The model represents the verification as well as validation.

The verification provides evidence for important questions, e.g.: Are we creating the safety application correctly?

Validation tests the question, e.g.: Are we creating the right safety application?

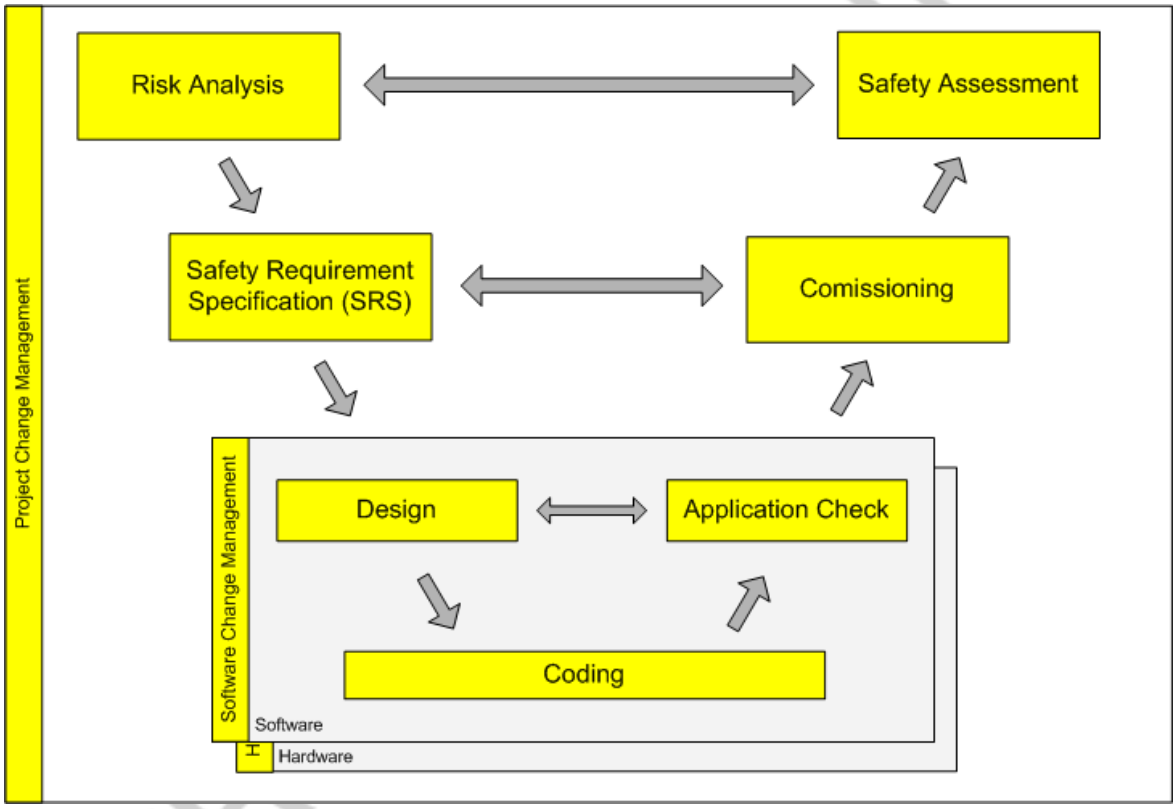


Fig. 3: V-model

The **risk analysis** of the machine is carried out in the first step of the V-model. The initial dangers are analyzed and the attainable level of safety is determined.

This is followed by the **Safety Requirement Specification (SRS)**, which identifies detailed methods for preventing and reducing the risks.

The SRS can be divided into the following points:

- Determining the operating modes
- List of safety functions
- Specification of the safety functions and response times
- List of hardware components
- Specification of hardware components

Furthermore, the SRS also contains the intended safety-relevant components such as light curtain, E-stop, mode selector switch, etc.

Another important part is the division of the system into different operating states.

The next section covers the specification of the previously determined safety functions. These are held to predetermined guidelines and standards which pertain to the response time, interface and priority of the sections to be performed. The function of operating modes is also determined at the same time.

Example:

A light curtain serves as a safety measure. This should only be enabled in the "PRODUCTION" operating mode because there is no danger in the "SERVICE MODE". In certain operating modes, the light curtain is not the only device triggered. Others may include an E-stop button. Therefore, the respective priority of these devices must be clearly defined at all times – how and when which procedure should operate. The response times and frequency with which the triggered function occurs is also specified. However, their error responses and their limitations for certain states must also be adhered to as closely as possible.

The SRS also contains the list of required hardware components. The final point in the SRS deals with the exact specification of the selected hardware components.

The next step is to draw up the **design of the safety application** and the design of the test specification. Other project steps will also be performed on the side at the same time.

The design serves the programmer as guideline both for programming as well as the for the tests that must be performed on the safety application. This is an implementation of parameters, function blocks, and cycle times in a clean and understandable form.

The **test specification** is a collection of specifications designed for testing the individual safety components. Keep in mind that the safety function states described in the SRS must be accordingly tested with these specifications and for there ability to be executed. The listed points represent the expectations and desired reaction when performing the tests.

After completing the safety application design and the test specification described earlier, it is then time to begin **configuring** the hardware and **programming** the software.

In the next step, the settings and functions that have been established will be **verified** in the SafeDESIGNER against the prepared **design**.

> **Note:**
>
> Are we creating the safety application correctly?

Any errors discovered in the configuration of the functionality must be documented in writing and the steps must be repeated starting from the design up to verification.

The next step in the V-model involves **commissioning**, which represents yet another verification process. At this point, the functions specified in the SRS are verified for correct processing.

> **Note:**
>
> Are we creating the safety application correctly?

The final section of the V-model, the **safety evaluation** has been reached once the steps mentioned up to now have been completed. This evaluation determines whether the safety application fulfils the functions set forth in the risk analysis and meets the desired safety level.

> **Note:**
>
> Are we creating the right safety application?

If any errors are found in the project or planning at this point, the affected parts of the V-model must be repeated.

After successful validation, the customer has the possibility to obtain a certificate for this machine from a third party.

3. CONFIGURATION IN AUTOMATION STUDIO

From the safety standpoint, Automation Studio manages all of the modules for the safety-relevant components.

From the Automation Studio standpoint, the SafeIO modules act like functional input and output modules with hidden complexity. Variables can be connected to the available I/O channels, which are used like normal I/Os in the standard application.

Administration involves the following points:

- Adding the SafeLOGIC
- Safety application name
- Data exchange between standard CPU and SafeLOGIC via communication channels
- Adding the SafeIO modules
- Assigning the SafeIO modules to the SafeLOGIC
- Access to available I/O data from the safety components

Note:

Automation Studio Version 3.x with the SafeDESIGNER toolset is required for developing.

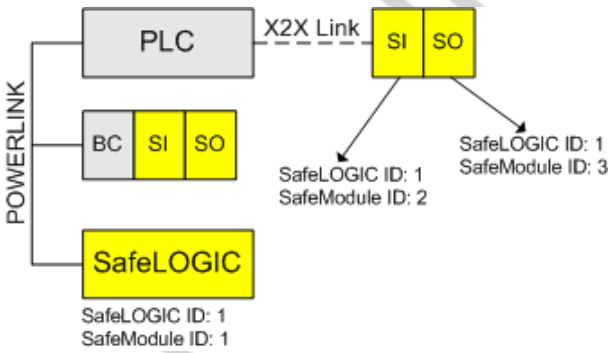


Fig. 4: Possible configuration

The **SafeLOGIC** builds a **virtual network** around itself. A **unique SafeLOGIC ID** must be assigned for each SafeLOGIC. The SafeMODULE ID represents the number of individual modules in the network. The SafeLOGIC always has the SafeMODULE ID 1.

Each inserted **SafeIO module** must be **assigned a SafeLOGIC**. The SafeLOGIC ID of the corresponding SafeLOGIC is specified. A **unique SafeMODULE ID** for the network is then assigned for each module.

3.1 Adding the SafeLOGIC

The SafeLOGIC is the central collection point for all safety data.

It is added to a POWERLINK interface in Automation Studio. This can be an interface right on the CPU (e.g. X20 CPU) or on an interface card (e.g. IF789).

The defined node number for the POWERLINK network must be configured using the wizards. Additionally, the version of the SafeDESIGNER to be used is also determined when adding the name of the safety application.

The SafeLOGIC ID is used later to assign the SafeIO modules to a SafeLOGIC. This number is automatically assigned by the system, but can be changed if necessary.

Note:

Multiple SafeLOGICs can be configured and used in an Automation Studio project.

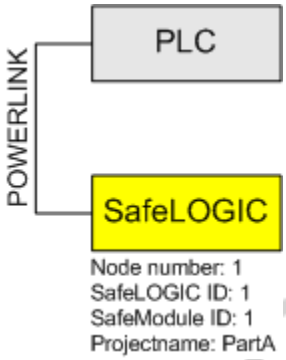


Fig. 5: Possible SafeLOGIC configuration

Open the settings for the POWERLINK interface by **right-clicking on CPU:Open Powerlink**.

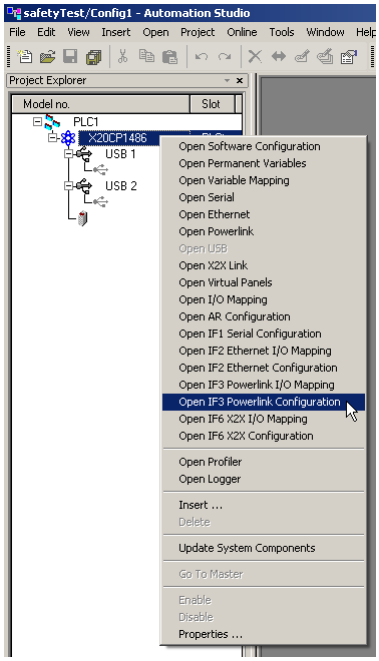


Fig. 6: Opening the POWERLINK configuration

Set the **operating mode** to **POWERLINK V2** and save the settings.

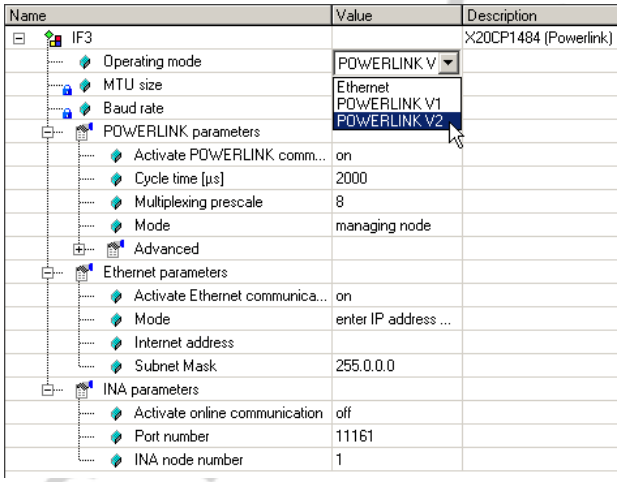


Fig. 7: Powerlink configuration

The SafeLOGIC can now be added to the POWERLINK interface.

Open the POWERLINK interface by **right-clicking on CPU:Open Powerlink**.

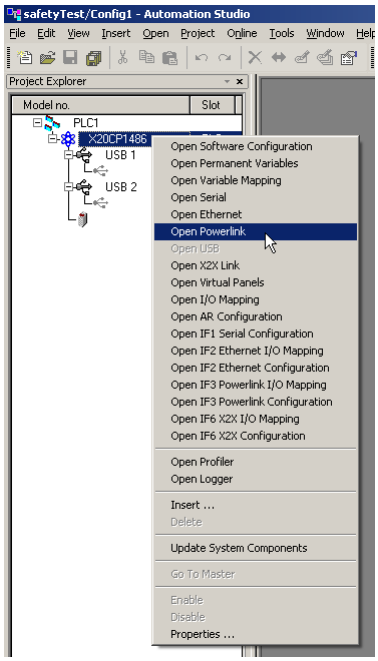


Fig. 8: Opening the POWERLINK interface

Open the wizard for adding a new module by **right-clicking:Insert**.

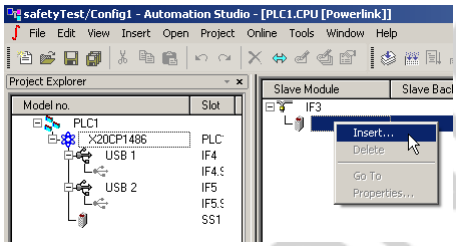


Fig. 9: Inserting a New Module

Select SafeLOGIC Plus (X20SL8001) from the following dialog box.

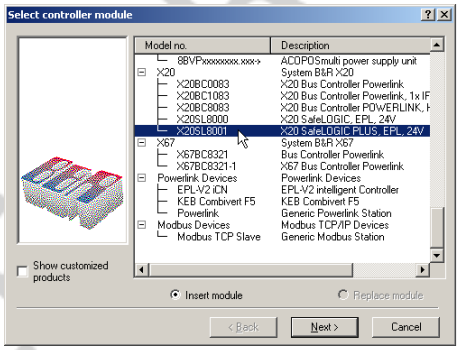


Fig. 10: Select SafeLOGIC

Continue with the wizard by clicking on **Next**.

The following view is then shown, which is used to add the node number for the SafeLOGIC.

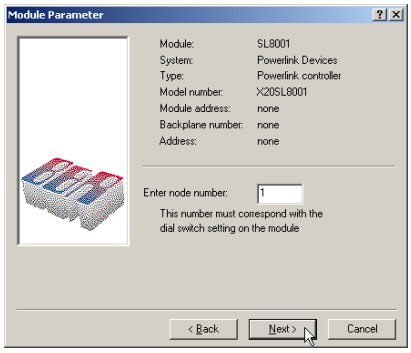


Fig. 11: Setting node numbers

Continue with the wizard by clicking on **Next**.

The wizard can be used to enter the name of the safety application, the SafeDESIGNER version to be used and the SafeLOGIC ID.

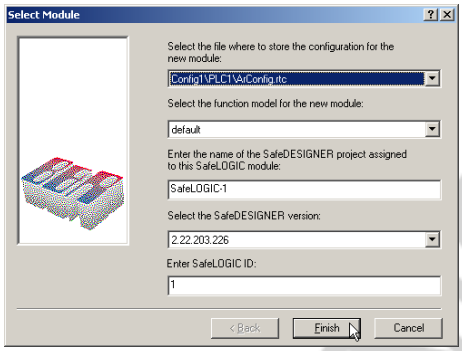


Fig. 12: SafeLOGIC settings

Note:

The SafeLOGIC ID is automatically assigned by the system, but can be changed if necessary.

Close the wizard by clicking on **Finish**.

The Automation Studio configuration for SafeLOGIC can be opened by **right-clicking on SafeLOGIC:Open I/O Configuration**.

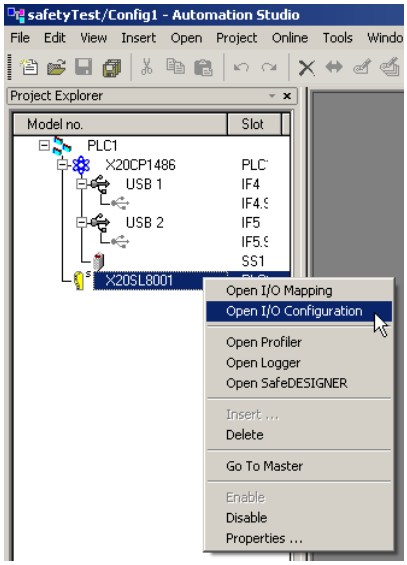


Fig. 13: Opening the SafeLOGIC configuration

The following view appears for configuring the SafeLOGIC.

Name	Value	Description
IF3.ST1		X20SL8001
POWERLINK parameters		
Mode	controlled node	
Response timeout [us]	25	
Multiplexed station	off	
Function model	default	Module´s operating mode
General		
Module supervised	on	Service mode if there is no hardware module
Node used as IP gateway	240	
SafeLOGIC ID	1	
SafeMODULE ID	1	
SafeDESIGNER project	SafeLOGIC-1	
Safe Runtime version		
SafeDESIGNER version	2.53.203.249	
CPU to SafeLOGIC communication		Consumed by this SafeLOGIC
Number of BOOL channels	8	
Number of UINT channels	0	
Number of UDINT channels	0	
SafeLOGIC to CPU communication		Produced by this SafeLOGIC for other CPUs
Number of BOOL channels	8	
Number of UINT channels	0	
Number of UDINT channels	0	

Fig. 14: SafeLOGIC configuration

The configuration editor is used to configure communication channels between standard CPU and SafeLOGIC. There are data points consumed by the SafeLOGIC (input) and data points produced by the SafeLOGIC (output).

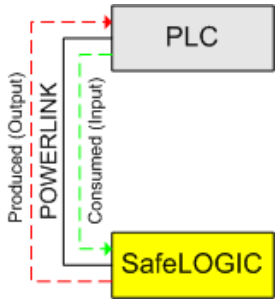


Fig. 15: Communication channels CPU – SafeLOGIC

Among other things, this makes it easy to transfer the signal from an unsafe acknowledge button to the SafeLOGIC.

The SafeLOGIC I/O Mapping can be opened by **right-clicking on SafeLOGIC:Open I/O Mapping**.

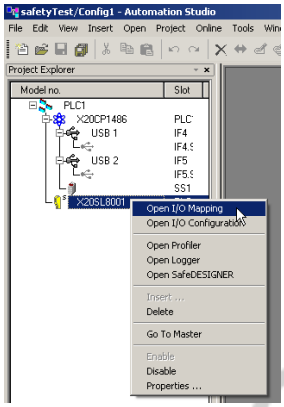


Fig. 16: Opening the SafeLOGIC I/O Mapping

The following view appears for linking the individual I/O channels with variables.

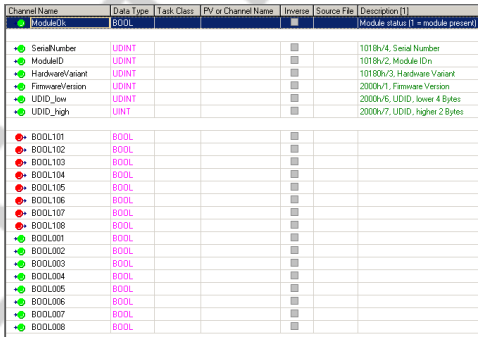


Fig. 17: SafeLOGIC I/O Mapping

The I/O Mapping allows you to connect variables to the individual configured communication channels.

The number of communication channels between the standard CPU and SafeLOGIC can be set as desired using the SafeLOGIC configuration.

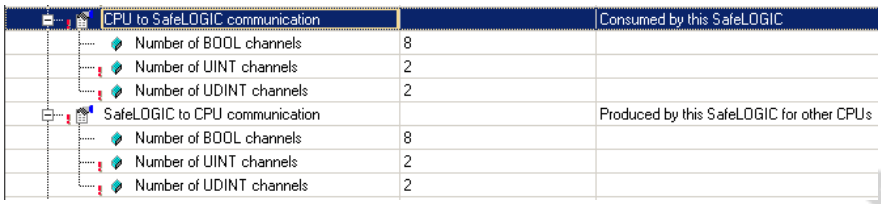


Fig. 18: SafeLOGIC communication channels

Note:

The communication channels between the standard CPU and SafeLOGIC are not transferred securely.

Example: Adding SafeLOGIC

Create a new Automation Studio project with an X20 CPU.

Add a SafeLOGIC standard (X20SL8000) to a POWERLINK interface.

3.2 Adding the SafeIO modules

The SafeIO modules can be added right to an X2X Link interface or behind a POWERLINK bus connector.

> **Note:**
>
> Safety modules can be mixed with normal modules as needed.

The SafeLOGIC ID is used to assign the individual SafeIO modules to a SafeLOGIC.

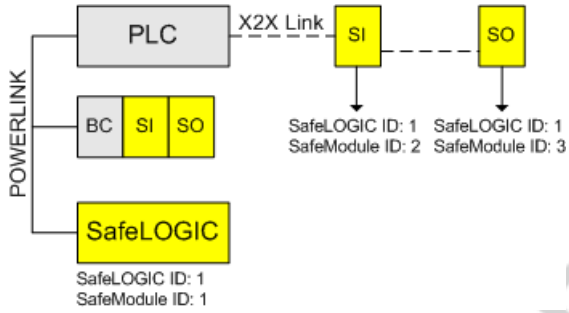


Fig. 19: Possible configuration of SafeIO modules

Open the X2X Link interface by **right-clicking on CPU:Open X2X Link**.

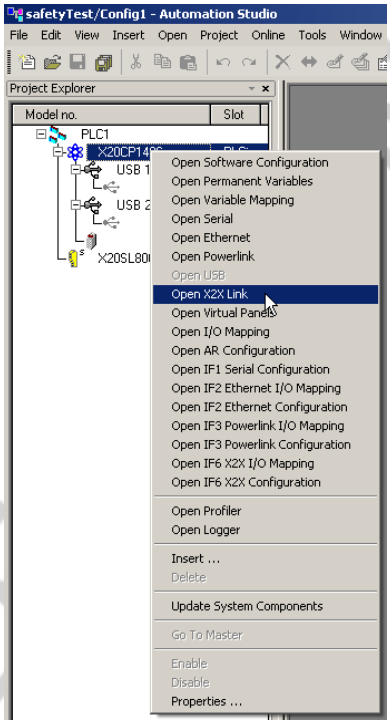


Fig. 20: Opening the X2X Link interface

Open the wizard for adding a new module by **right-clicking**:**Insert**.

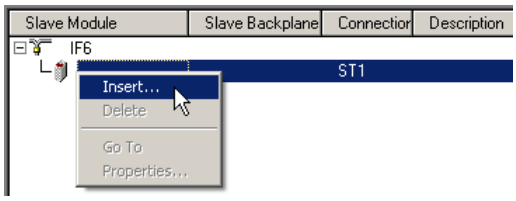


Fig. 21: Inserting a New Module

Select an SI module (e.g. X20SI2100) from the following dialog box.

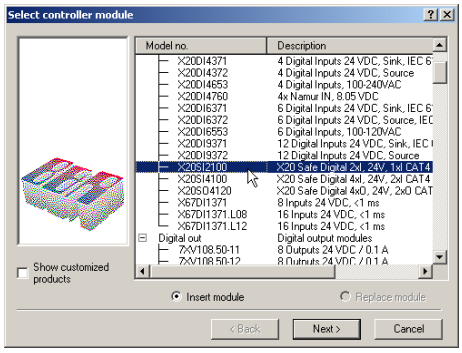


Fig. 22: Selecting an SI module

Continue with the wizard by clicking on **Next**.

The wizard can be used to assign the SafeIO module to a specific SafeLOGIC. This is done using the SafeLOGIC ID. The SafeMODULE ID can be changed if necessary.

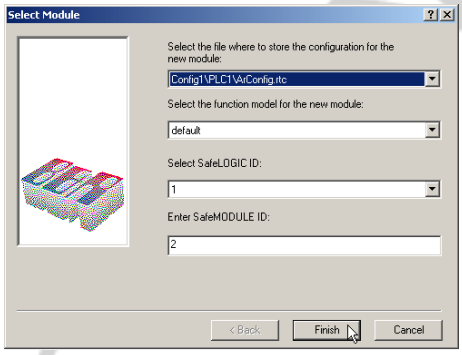


Fig. 23: SafeIO settings

Note:

The SafeMODULE ID is always assigned automatically by the system, but can be changed if necessary.

Close the wizard by clicking on **Finish**.

The SafeMODULE ID can be changed manually if a module is mapped, but should contain the old SafeMODULE ID.

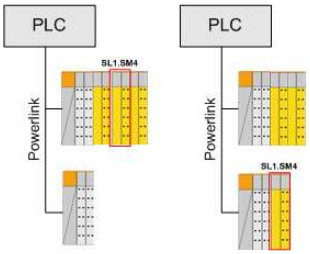


Fig. 24: SafeMODULE ID

A SafeIO module (SafeMODULE ID 4) should be mapped from one bus connector to another.

The system automatically assigns a consecutive SafeMODULE ID number if the module is deleted and a new module is added. This must be changed to 4 manually.

This makes it possible to exchange a SafeIO module without affecting the safety application because the SafeMODULE ID does not change.

The Automation Studio configuration for SafeIO module can be opened by **right-clicking on SafeIO:Open I/O Configuration**.

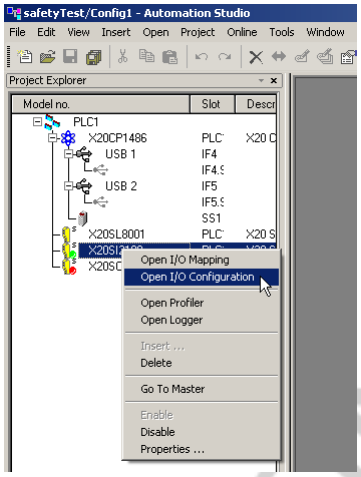


Fig. 25: Opening the configuration of the SafeIO module

The following view appears for configuring an SI module.

Name	Value	Description
IF6.ST1		X20SI2100
Function model	default	Module's operating mode
General		
Module supervised	on	Service mode if there is no hardware module
Module information	off	Additional module information
Error information	on	Provide error codes
SafeLOGIC ID	1	
SafeMODULE ID	2	

Fig. 26: Safe Digital In configuration

The following view appears for configuring an SO module.

Name	Value	Description
IF6.ST2		X20SO2120
Function model	default	Module's operating mode
General		
Module supervised	on	Service mode if there is no hardware module
Module information	off	Additional module information
Error information	off	Provide error codes
Physical state information	off	Provide read back values for all channels
SafeLOGIC ID	1	
SafeMODULE ID	3	
max switching frequenc...	1 Hz	max switching frequency
max switching frequenc...	1 Hz	max switching frequency
Output signal path		
DigitalOutput01	direct	
DigitalOutput02	direct	

Fig. 27: Safe Digital Out configuration

Note:

The parameter "Error information" must be set to "on" to enable access to the SafeIO module error data via the I/O Mapping.

The required switching frequency of the output can be defined for each SO module channel. An internal check is made, after a specific amount of time has elapsed (depending on the switching frequency), to determine if the output was really switched off. Incorrect configuration of the switching frequency can cause unwanted errors to occur.

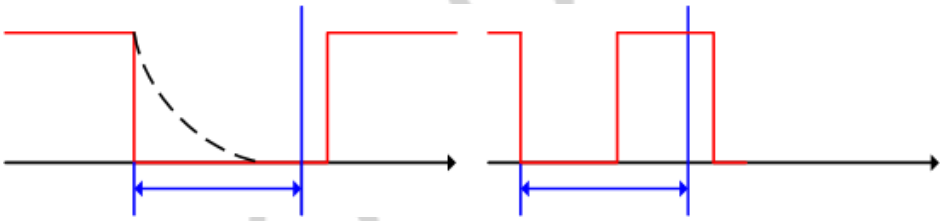


Fig. 28: Switching frequency

Note:

Make sure that this switching frequency is supported by the connected actuator.

On an output module, the approval principle can be defined additionally for each individual channel. The output channel is either shown or hidden in the I/O Mapping depending on the set approval principle.

Name	Value	Description
IF6.ST2		X20SO2120
Function model	default	Module´s operating mode
General		
Module supervised	on	Service mode if there is no hardware module
Module information	off	Additional module information
Error information	off	Provide error codes
Physical state information	off	Provide read back values for all channels
SafeLOGIC ID	1	
SafeMODULE ID	3	
max switching frequenc...	1 Hz	max switching frequency
max switching frequenc...	1 Hz	max switching frequency
Output signal path		
DigitalOutput01	direct	
DigitalOutput02	direct / via SafeLOGIC	

Fig. 29: Approval settings in Automation Studio

Two different settings can be made for the approval principle:

- "direct": Output channel is visible in the I/O Mapping
- "via SafeLOGIC": Output channel is hidden in the I/O Mapping

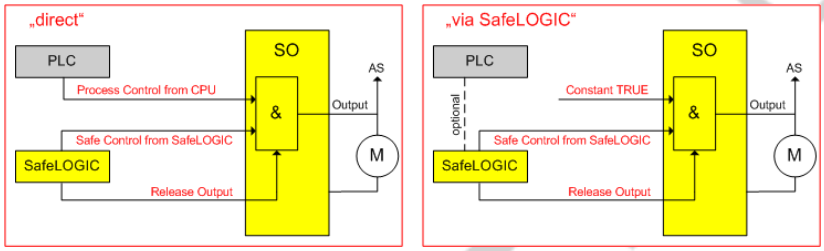


Fig. 30: Approval principle

The I/O Mapping for a SafeIO module can be opened by **right-clicking on SafeIO Module:Open I/O Mapping**.

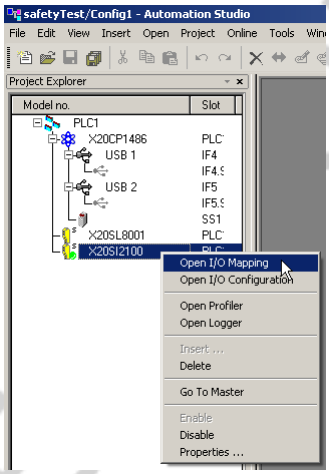


Fig. 31: Opening the I/O Mapping for a SafeIO module

The following view for an SI module appears for accessing the individual I/O channels with variables.

Channel Name	Data Type	Task Class	PV or Channel Name	Inverse	Source File	Description [1]
ModuleOk	BOOL					Module status (1 = module present)
SafeDigitalInput01	BOOL					24 VDC, <50 us switching delay, sink
SafeDigitalInput02	BOOL					24 VDC, <50 us switching delay, sink
SafeEquivalentInput0102	BOOL					Equivalent FUB channel 1/2
SafeAntivalentInput0102	BOOL					Antivalent FUB channel 1/2

Fig. 32: I/O Mapping of an SI module

The I/O Mapping can be used to connect variables to the individual inputs or to the automatic multi-channel analysis provided.

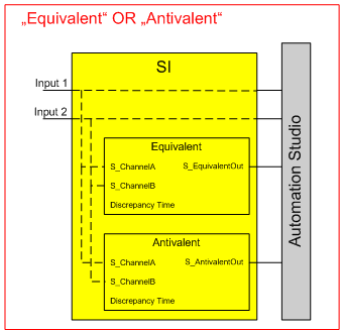


Fig. 33: Multi-channel analysis

Note:

The multi-channel analysis is configured automatically in SafeDESIGNER.

The following view for an SO module appears for accessing the individual I/O channels with variables.

Channel Name	Data Type	Task Class	PV or Channel Name	Inverse	Source File	Description [1]
ModuleOk	BOOL					Module status (1 = module present)
DigitalOutput01	BOOL					
DigitalOutput02	BOOL					

Fig. 34: I/O Mapping of an SO module

The I/O Mapping can be used to connect variables to the individual outputs, depending on the set approval principle.

Example: Adding SafeIO modules

Add an SI module and an SO module on the X2X Link interface of the X20 CPU in your project.

3.3 Cross Link Task

The Cross Link Task is used to copy data from the local X2X-Link or from another POWERLINK network to the POWERLINK network which contains the SafeLOGIC. A cycle time can be defined in Automation Studio for this copy procedure.

> **Note:**
>
> The Cross Link Task runs on the CPU.

Right-click on the CPU, select Properties and go to the **Timing** tab to reach the mask for defining the cycle time.

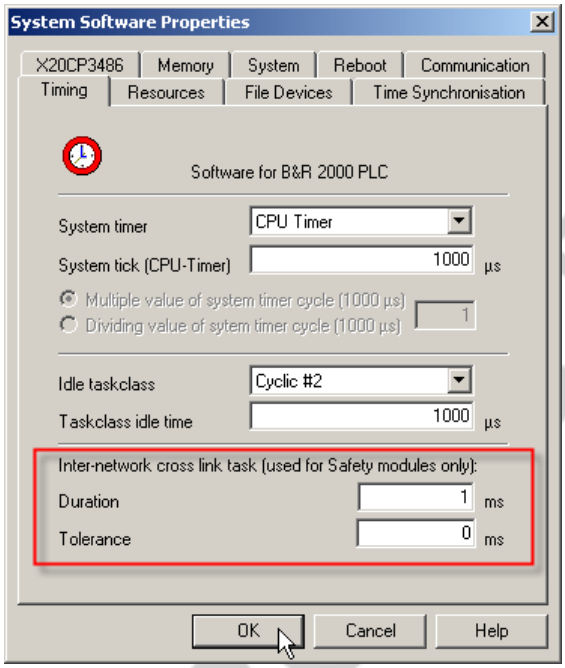


Fig. 35: Cross Link Task cycle time

4. SAFEDESIGNER

SafeDESIGNER represents the core of the safety programming.

The SafeDESIGNER is used to create the safety application, which will run on the SafeLOGIC, and to configure the individual modules. To do this, all of the safety-relevant components that are assigned to the corresponding SafeLOGIC are automatically taken from the Automation Studio configuration.

The safety application (i.e. SafeDESIGNER) can be opened for the selected SafeLOGIC by **right-clicking on SafeLOGIC:Open SafeDESIGNER**.

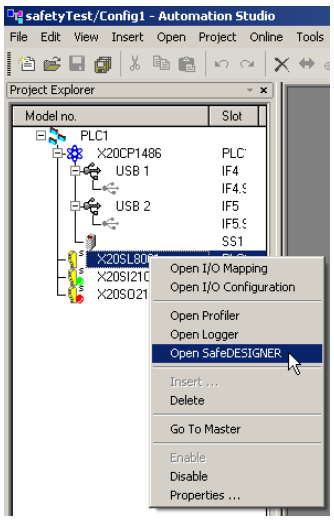


Fig. 36: Opening SafeDESIGNER

The SafeDESIGNER is started and a login window is displayed.

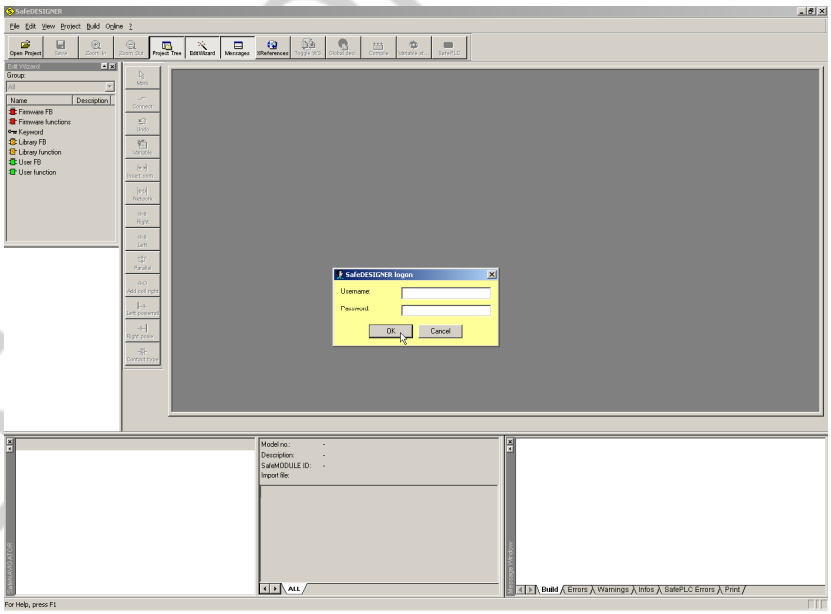


Fig. 37: SafeDESIGNER – Login window

Note:

You will be requested to create a user ("Administrator" group) the first time the SafeDESIGNER is started.

The SafeDESIGNER functions are available after logging in successfully (username and password), depending on the user's rights.

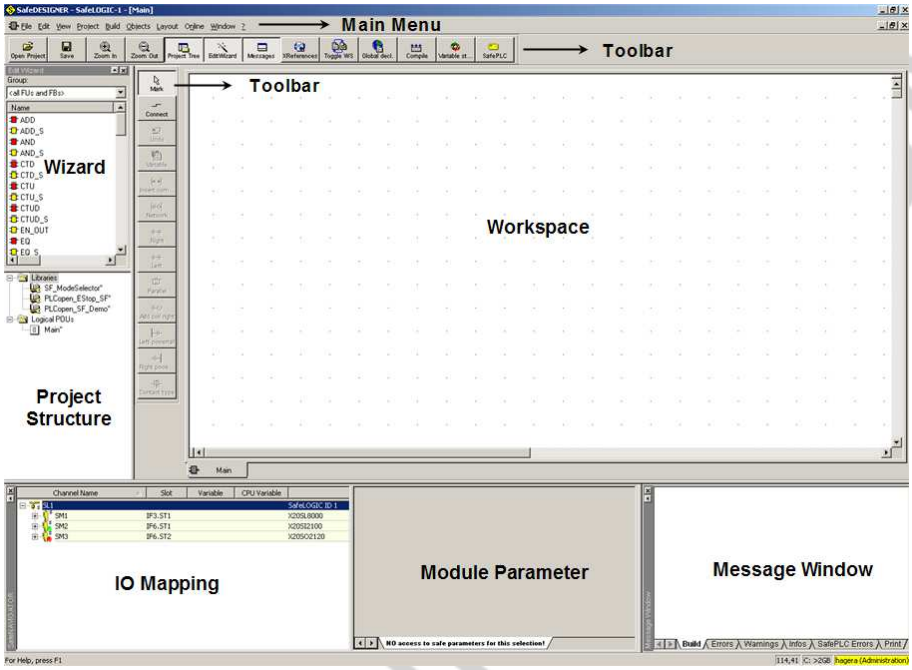


Fig. 38: SafeDESIGNER

Various basic functions such as compiling, establish online communication, etc., can be performed using the horizontal toolbar in SafeDESIGNER.

Fig. 39: Horizontal toolbar

- Zoom In, Zoom Out: Zooming
- Project Tree: Project view window shown / hidden
- Edit Wizard: Edit Wizard shown / hidden
- Messages: Message window shown / hidden
- XReferences: Window for cross-references
- Toggle WS: Toggle between worksheet and local declaration
- Global declarations: Show global declaration
- Compile: Compile safety application
- Variable status: Show variable status (monitor mode)
- SafePLC: Open communication window for SafeLOGIC

The vertical toolbar is only enabled when the focus is in the worksheet. This toolbar can be used to perform various basic functions for the graphic editor.

- Mark: Set mark
- Connect: Draw connection between function blocks
- Undo: Undo last action
- Variable: Create new variable
- Insert comment: Insert comment to worksheet
- Network: Insert a LD network
- Right, Left, Parallel, Add coil right, Left powerrail, Right powerrail, Contact type: Commands in a LD network

Fig. 40: Vertical toolbar

Note:

Help for the SafeDESIGNER can be called up using the **F1** key or via **?:Contents**.

A safety application is developed in SafeDESIGNER by following these steps:

- Program safety functions
- Link the I/O channels
- Configure the modules
- Establish online communication and perform a download
- Application diagnostics
- Create documentation

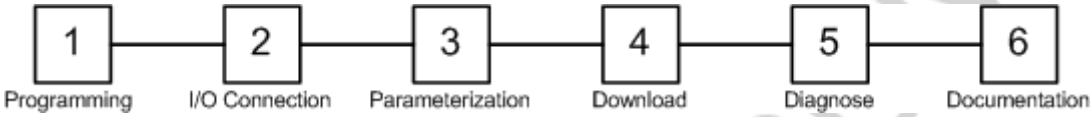


Fig. 41: Creating a safety application

4.1 Program safety functions

The programming interface in SafeDESIGNER is a graphic editor, which supports the programming languages LD and FUB. The programming languages can be used together as needed.

In addition to the graphic editor, the programming interface also includes an Edit Wizard, for managing the safety application's function blocks and a window showing the current project structure.

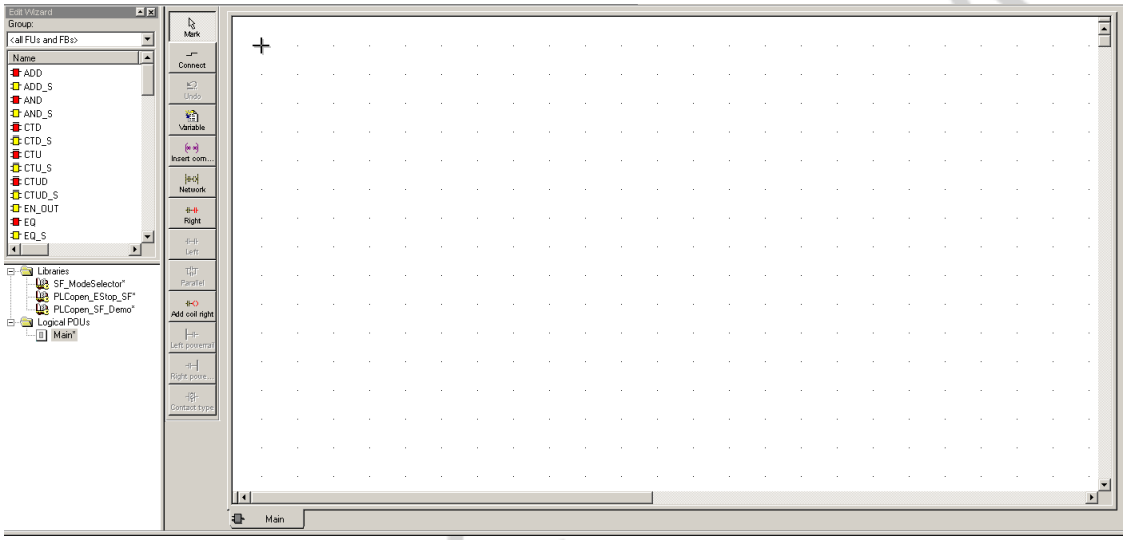


Fig. 42: Programming interface

Note:

The safety application is a Single Task System. That means that everything is programmed in one worksheet (Main).
You can create your own function blocks. This will be explained in greater detail later in the training module.

Strict data type separation is used when programming in SafeDESIGNER. There are normal and safe data types.

Normal data types	Bit	Safe data types	Bit
BOOL	1	SAFEBOOL	1
BYTE	8	SAFEBYTE	8
WORD	16	SAFEWORD	16
UINT	16	SAFEINT	16
TIME	32	SAFETIME	32
DWORD	32	SAFEDWORD	32

Note:

It is possible to convert safe data types into normal data types.

4.1.1 Edit Wizard

The Edit Wizard can be used to add the available function blocks to the worksheet.

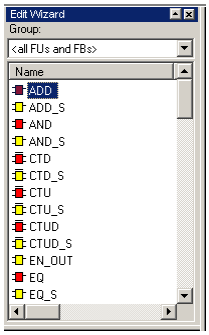


Fig. 43: Edit Wizard

The function block view can be grouped (e.g. show all safe FUBs, show all normal FUBs, show all FUBs from a certain library).

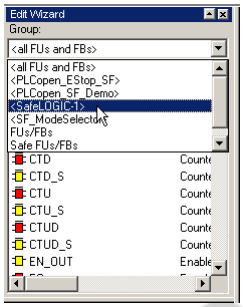


Fig. 44: Group Edit Wizard

Note:

Normal function blocks are displayed with red in accordance to IEC 61131-3, safe function blocks are displayed with yellow in accordance to IEC 61131-3.
Function blocks from the PLCopen Safety Library are marked with a red S and yellow background.

4.1.2 Adding a FUB to the worksheet

A drag & drop function is provided for adding a function block to the worksheet.

To do this, you must first select the corresponding function block in the Edit Wizard.

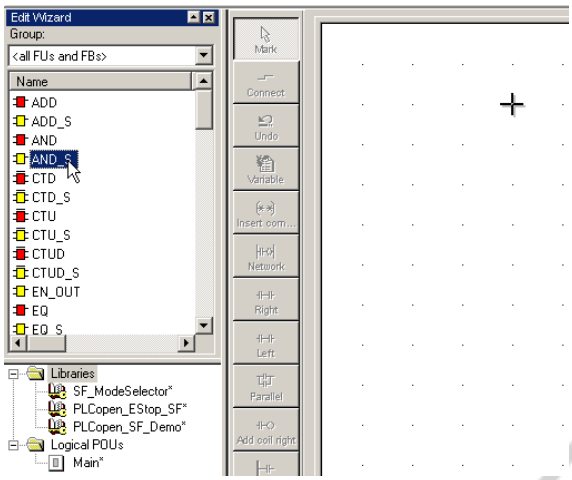


Fig. 45: Selecting a function block in the Edit Wizard

This function block can now be dragged into the worksheet by holding down the left mouse button and dropped by releasing the mouse button.

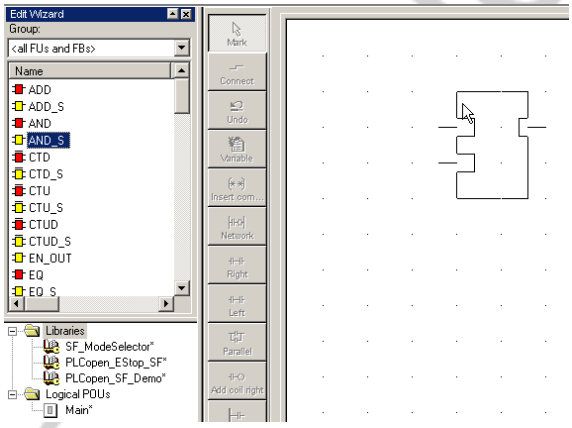


Fig. 46: Dragging a function block into the worksheet

The function block is inserted at the desired position by left-clicking.

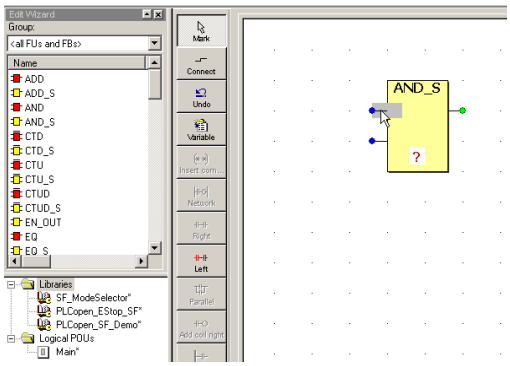


Fig. 47: Function block added to the worksheet

If a more complex function block (e.g. emergency stop from the PLCopen Safety library) is added, then a dialog box appears for creating a respective instance variable with the respective data type.

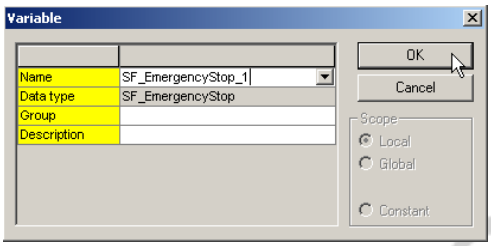


Fig. 48: Instance variable for a function block

4.1.3 Insert a New Variable

The **Variable** button is provided in the toolbar for adding a new variable.

Fig. 49: Button for adding a new variable

A wizard is opened for setting the variable name, data type and scope for the variable.

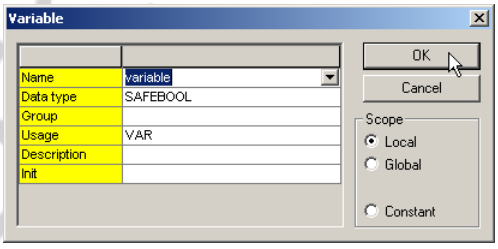


Fig. 50: Settings for the variable

The Scope parameter can be used to choose between 3 different settings:

- Local: for the current worksheet
- Global: for an I/O channel
- Constant: Constants

Note:

The wizard can also be opened via **Object:Variable**.

Local variables are only used in the current worksheet. To do this, the Scope parameter is set to "Local" and the name and data type of the variable are defined.

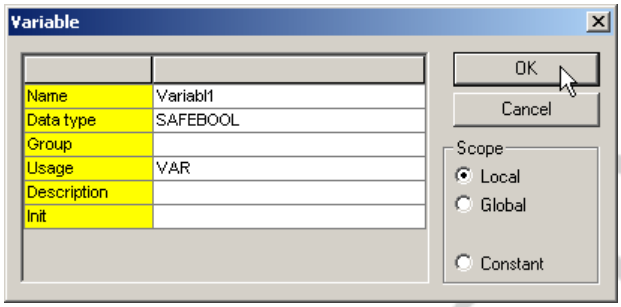


Fig. 51: Adding local variables

The variable is added at the current cursor position after the dialog box has been confirmed.

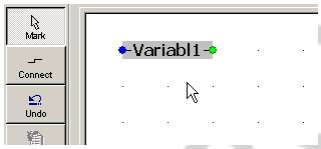


Fig. 52: Variable in the worksheet

The **Toggle WS** button is used to toggle between the Code view and the Declaration view of the current worksheet.

Fig. 53: Buttons for toggling the view

All of the declared local variables from the worksheet are displayed in the declaration view.

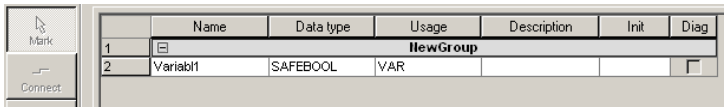


Fig. 54: Local variable declaration

Global variables are only used for I/O channels. To do this, the Scope parameter is set to "Global" and the name and data type of the variable are defined.

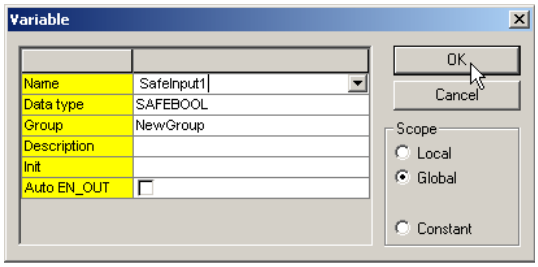


Fig. 55: Adding global variables

The variable is added at the current cursor position after the dialog box has been confirmed.

Fig. 56: Variable in the worksheet

Note:

A global variable must always be assigned an I/O channel. Global variables that are not connected to an I/O channel will cause an error message when compiling.

The Wizard can also be used to add constants to the worksheet. To do this, the Scope parameter is set to "Constant" and a pre-defined constant is selected.

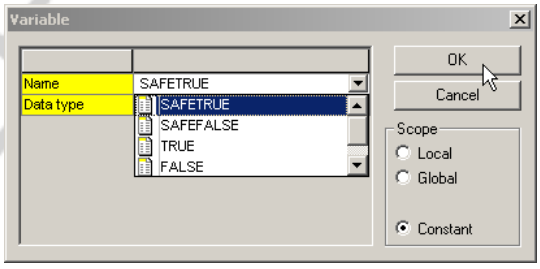


Fig. 57: Adding constants

The variable is added at the current cursor position after the dialog box has been confirmed.

Fig. 58: Constants in the worksheet

Note:

Constants can be added for different data types using DATA#TYPE#VALUE (e.g. SAFETIME#duration for a time constant).

4.1.4 Linking a function block

Click on and mark a variable to link it to an input or output.

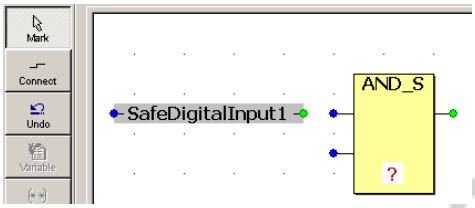


Fig. 59: Marking a variable

The variable can now be moved in the worksheet by holding down the left mouse button.

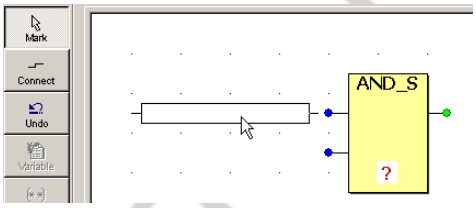


Fig. 60: Moving a variable

Release the left mouse button to link the variable with an input or output.

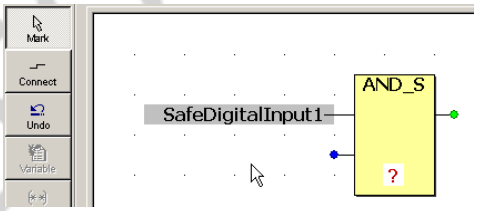


Fig. 61: Variable linked with input

A function block's output is often linked with the input of the next function block. These type of connections can be made using the **Connect** button.

Fig. 62: Button for drawing connection lines

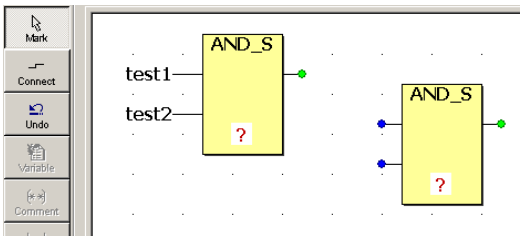


Fig. 63: Example for connection lines

The **Connect** button must be pressed before drawing a connection line from the output to the input of the next AND function block.

The cursor changes in the worksheet after the button has been pressed.

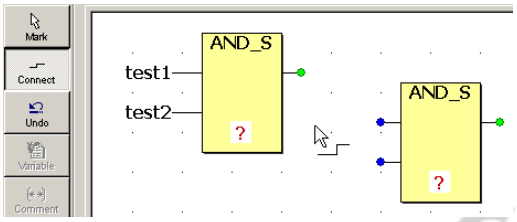


Fig. 64: Changed cursor

Now move the cursor to the output, left-click and drag to the input. A wizard helps draw the line.

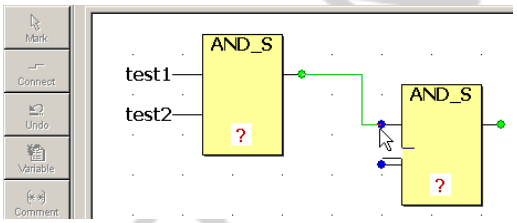


Fig. 65: Drawing a connection line

Release the left mouse button to mark the final connection line in the worksheet.

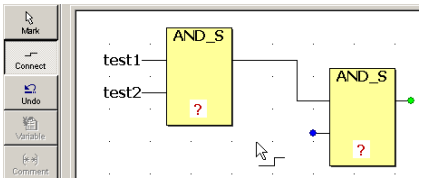


Fig. 66: Finished connection line

Example: Adding a function block

Add an AND function block to the worksheet. Link an input with a local variable.

4.1.5 Insert comment to worksheet

The **Insert Comment** button can be used to add a comment at the current cursor position.

Fig. 67: Button for adding a comment

A window is opened for entering text including various text options.

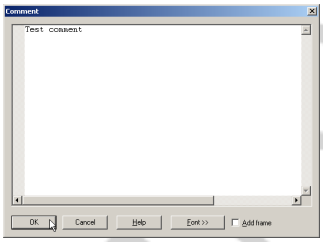


Fig. 68: Adding a comment

A separate window is opened for making various text settings.

The comment is added at the current cursor position after the window has been confirmed.

Fig. 69: Adding a comment in the worksheet

4.2 Linking I/O channels

The IO Mapping window is used to establish an interface between the safety-relevant components in Automation Studio and SafeDESIGNER. This window can be reduced to the configured safety components.

The IO Mapping window can be used to link variables that are used in the safety application with the I/O channels of the modules.

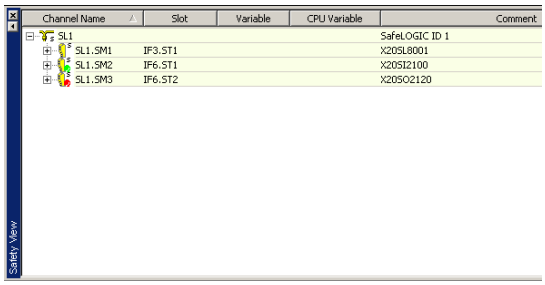


Fig. 70: IO Mapping window

Note:

The IO Mapping window mirrors the SafeLOGIC virtual network explained earlier.
All channels that have a yellow arrow are safety-relevant and must reference to safe data types.

The data is sorted according to the different buttons selected.

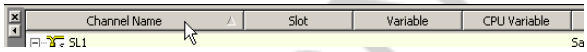


Fig. 71: Sort IO Mapping window

The safety components are sorted in the "Channel Name" column according to their SafeMODULE ID.

The physical position of the safety components is listed in the "Slot" column. This display is for information purposes and cannot be changed in SafeDESIGNER.

The "Variable" column contains the name of the I/O data point in the safety application.

The "CPU Variable" column contains the name configured in Automation Studio for the I/O data point.

4.2.1 SafeLOGIC (SL)

The configurable communication channels are available for the SafeLOGIC. Additionally, access to the machine option parameters is available in a SafeLOGIC Plus variation.

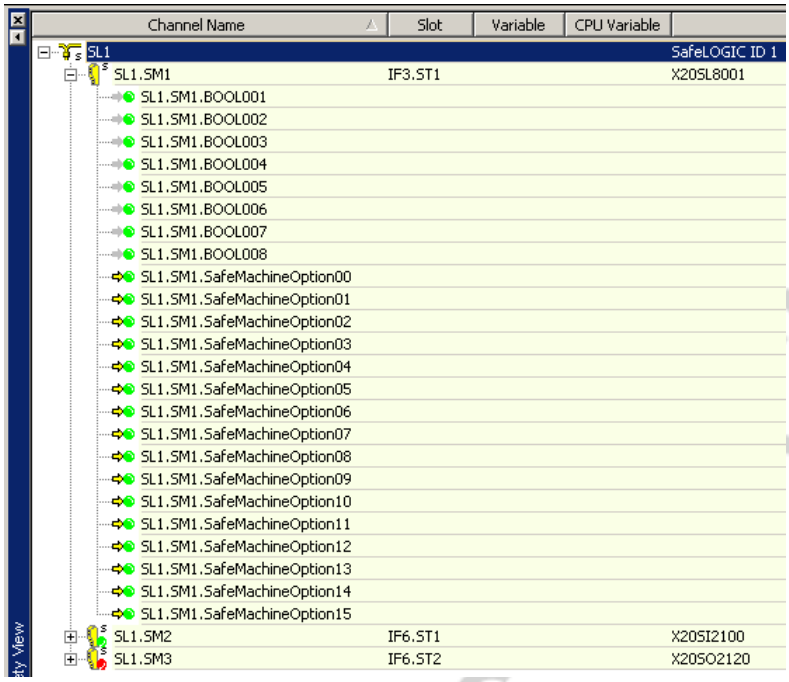


Fig. 72: IO Mapping - SafeLOGIC

Note:

Machine option parameters will be explained in greater detail when covering the SafeLOGIC parameters.

4.2.2 Safe Digital In (SI)

The individual input channels are available for a Safe Digital In (SI) module. Additionally, the multi-channel analysis can also be accessed as well as the module status information.

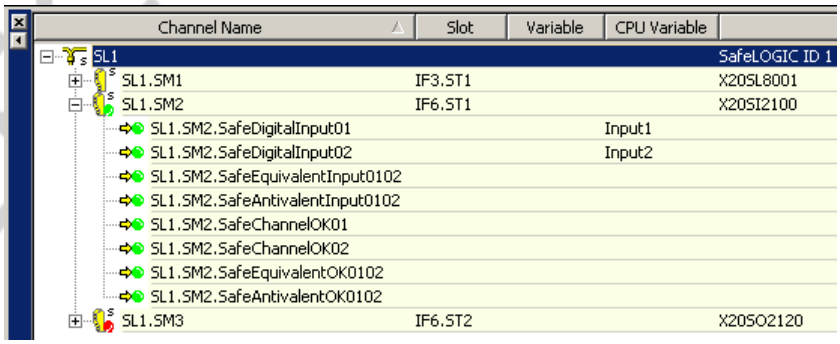


Fig. 73: IO Mapping - Safe Digital In

4.2.3 Safe Digital Out (SO)

The individual output channels are available for a Safe Digital Out (SO) module as well as acknowledgement of the automatic restart inhibit. In addition, status information about the module can also be accessed.

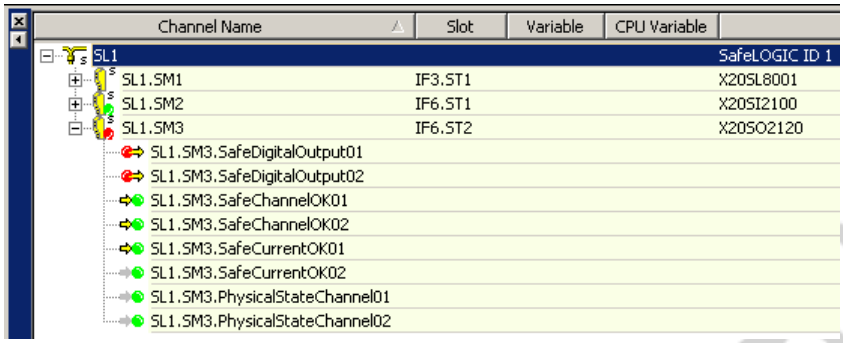


Fig. 74: IO Mapping - Safe Digital Out

4.2.4 Linking an I/O channel with a new variable

An I/O channel must first be selected in the IO Mapping window before it can be linked with a variable.

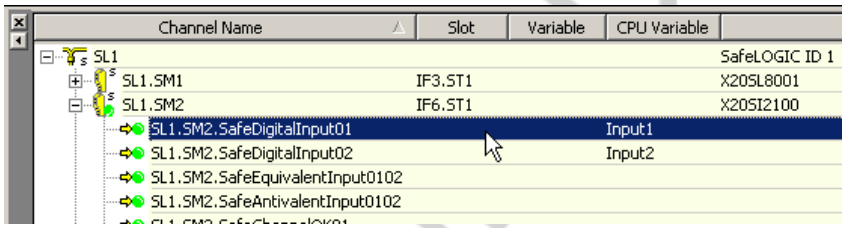


Fig. 75: Marking an I/O channel

This I/O channel can now be dragged to the worksheet by holding down the left mouse button.

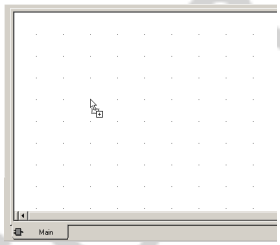


Fig. 76: Dragging the I/O channel into the worksheet

After releasing the mouse button, the familiar dialog box is opened for declaring a variable for this I/O channel for the safety application.

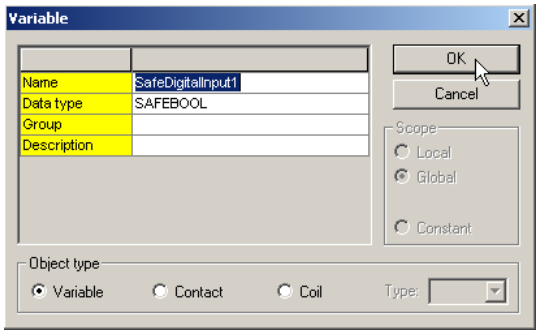


Fig. 77: Declaring a variable for the I/O channel

Note:

In the event that a variable was already specified for an input in Automation Studio, the Wizard automatically suggests this as the variable name. Changing the variable name in Automation Studio later on does not affect the safety application in any way.

The variable can be placed in the worksheet after the dialog box has been confirmed.

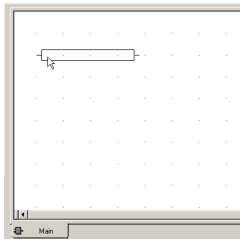


Fig. 78: Placing a variable

The variable position is set by left-clicking the mouse.

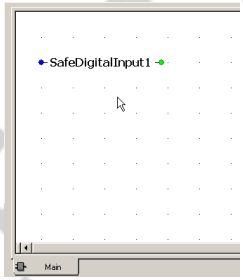


Fig. 79: Variable in the worksheet

In the IO Mapping window, the variable name of the safety application is added to the corresponding I/O channel.

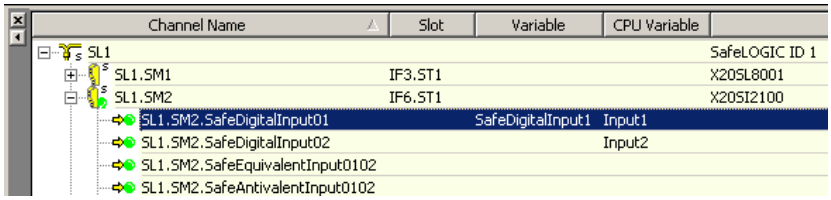


Fig. 80: I/O channel with the name of the variable

The **Global declaration** button is used to change to the declaration view for global variables.

Fig. 81: Button for global declaration view

All global variables are displayed in the declaration view. The "Terminal" field shows the link to the I/O channel.

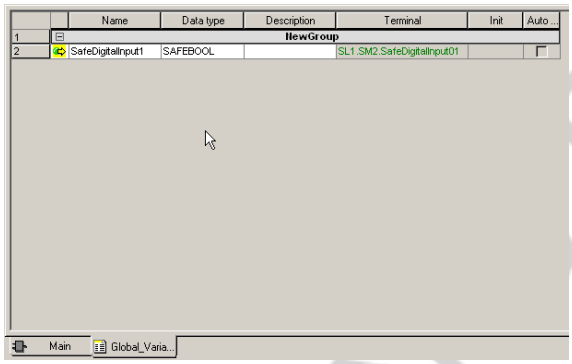


Fig. 82: Global declarations

4.2.5 Linking an I/O channel with an existing variable

As explained earlier, global variables that you have created must be linked with an I/O channel.

The **Global declaration** button is used to change to the declaration view for global variables.

Fig. 83: Button for global declaration view

A global variable that is not connected with an I/O channel is displayed with a red terminal field.

Fig. 84: Global declarations

An I/O channel must first be selected in the IO Mapping window before a link can be established with a variable.

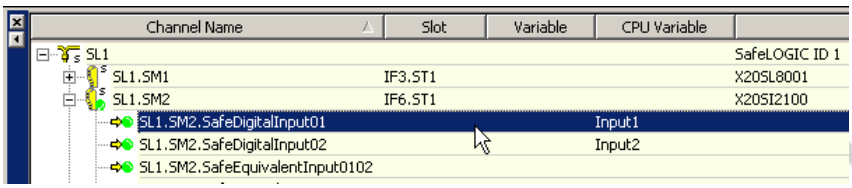


Fig. 85: Selecting an I/O channel

The I/O channel is clicked and dragged to the variable's terminal field.

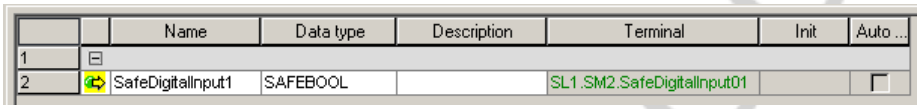


Fig. 86: Variable linked with I/O channel

In the IO Mapping window, the variable name of the safety application is added to the corresponding I/O channel.

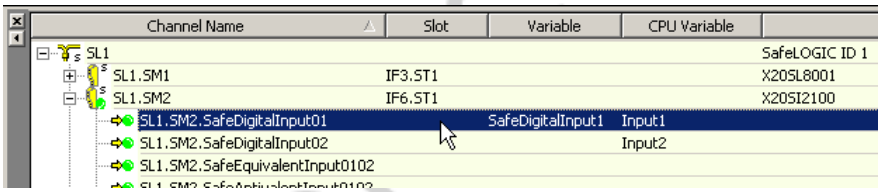


Fig. 87: I/O channel with the name of the variable

Example: I/O link

Link the input of the function block created earlier with an input channel and the output with an output channel.

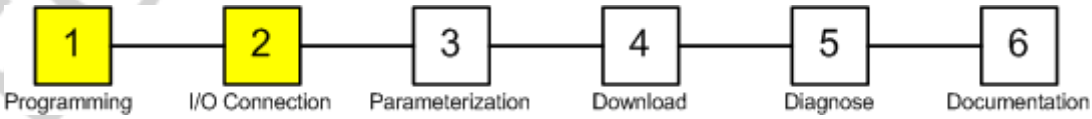


Fig. 88: Creating a safety application

4.3 Configure the modules

The parameters for a component can be defined using the parameter editor by selecting a safety component in the IO Mapping window.

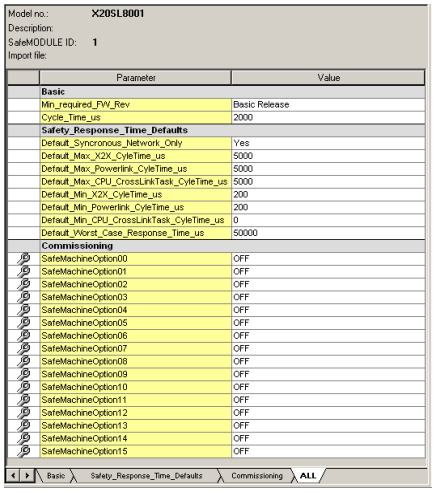


Fig. 89: Parameter editor

> **Note:**
>
> The "ALL" tab should always be enabled in the parameter editor in order to see all of the required parameters.

4.3.1 General parameters

Min_required_FW_Rev – Minimal required Firmware Revision
Specifies the minimum firmware version required.

Optional
Specifies whether or not module is optional.

External UDID
External machine configuration

Synchronous_Network_Only– Synchronous network
Specifies whether the network (cycle times) is synchronous

Max_X2X_CycleTime_µs – Maximum X2X cycle time
Maximum X2X cycle time that can be set in the standard application

Max_Powerlink_CycleTime_µs – Maximum POWERLINK cycle time
Maximum POWERLINK cycle time that can be set in the standard application

Max_CPU_CrossLinkTask_CycleTime_µs – Maximum crosslink task cycle time
The crosslink task runs on the standard CPU and is used to copy data from the X2X Link and POWERLINK interface.
Maximum crosslink task cycle time that can be set in the standard application

Min_X2X_CycleTime_µs – Minimum X2X cycle time
Minimum X2X cycle time that can be set in the standard application

Min_Powerlink_CycleTime_µs – Minimum POWERLINK cycle time
Minimum POWERLINK cycle time that can be set in the standard application

Min_CPU_CrossLinkTask_CycleTime_µs – Minimum crosslink task cycle time
The crosslink task runs on the standard CPU and is used to copy data from the X2X Link and POWERLINK interface.
Minimum crosslink task cycle time that can be set in the standard application

Worst_Case_Response_Time_µs – Worst case response time

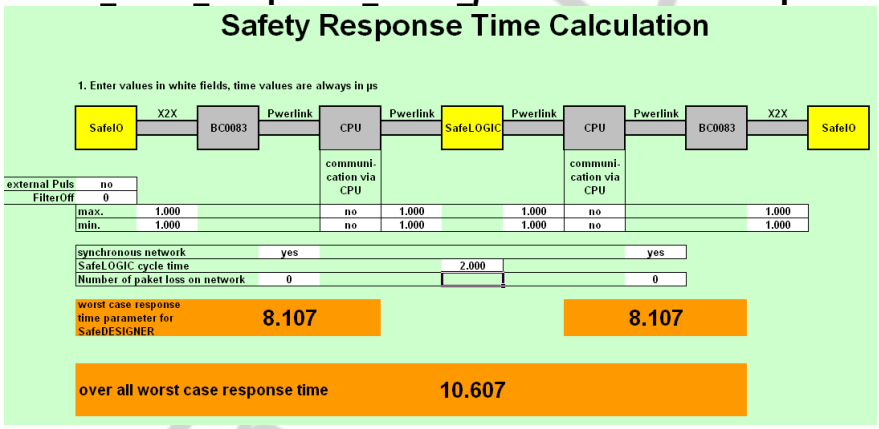


Fig. 90: Response time

4.3.2 Specific SafeLOGIC parameters

Cycle_Time_µs - Cycle time
This parameter is used to define the cycle time (in µs) of the safety application.

Commissioning - Option parameter
These parameters can be used to implement variable machine options. They can be enabled or disabled during commissioning. The individual option parameter channels are located in the IO Mapping window by SafeLOGIC, and can therefore be integrated right into the safety application.

4.3.3 Specific SI parameters

Puls_Mode – Pulse mode

- "internal": This setting is used to apply the module's own clock generation. It is used in one-channel and two-channel switches.

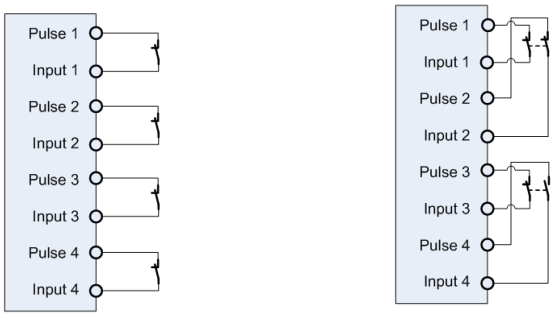


Fig. 91: Pulse mode "internal"

- "external": This setting is used for multi-channel switches and very long cable lengths.

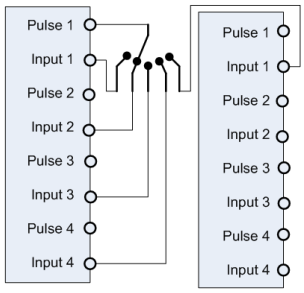


Fig. 92: Pulse mode "external"

Note:

All of the switch's channels that are being used must be configured to "external".

- "none": This setting is used for active sensors (light curtain, laser scanner). The module's clock generation is disabled with this setting. Any gaps in the test for the connected OSSD outputs must be masked out with the filter parameters.

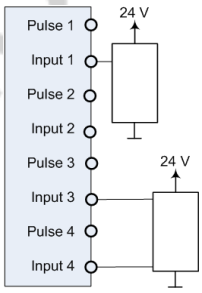


Fig. 93: Pulse mode "none"

Filter_Off_µs – Filter for Low time
This parameter is used for the High-Low transition. For example, it can be used for OSSD signals to bypass the Low time.

Filter_On_µs – Filter for High time
This parameter is used for the Low-High transition. These parameters can be used to extend High-signals that are too short so that they can be detected by the system.

Discrepancy_Time_µs - Discrepancy time
Used for multi-channel analysis and determines with which time interval the two switching elements should change their state.

> **Note:**
>
> Multi-channel analysis is handled automatically by the system. A check is made to determine which channel of the multi-channel analysis will be used in the safety application.

4.3.4 Specific SO parameters

Disable OSSD – Disable OSSD (Output Switching Sensor Device)
This parameter can be used to disable the internal automatic switch-off test for the channels.

4.4 Online communication and download

With the **Compile** button, the safety application is compiled, assigned a unique CRC number and can then be transferred to the SafeLOGIC.

Fig. 94: Button for compiling

A message window is used to show the compiling progress.

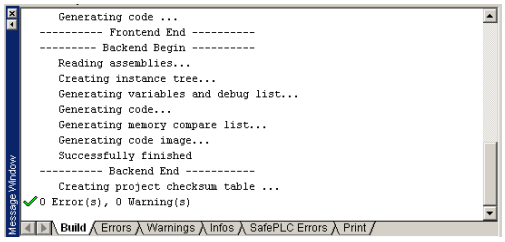


Fig. 95: Message window

The safety application can now be transferred to the SafeLOGIC via online communication.

There are a few ways to establish online communication with the SafeLOGIC:

- Direct connection (point-to-point)
- Via the standard CPU

Direct connection

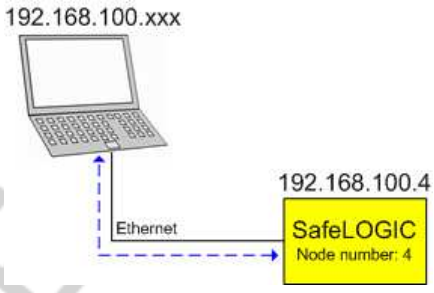


Fig. 96: Online via direct connection

An IP address is set on the PC from the address area of the POWERLINK network (192.168.100.xxx). The SafeLOGIC always has the defined node number as the last digit in the IP address.

Via the standard CPU

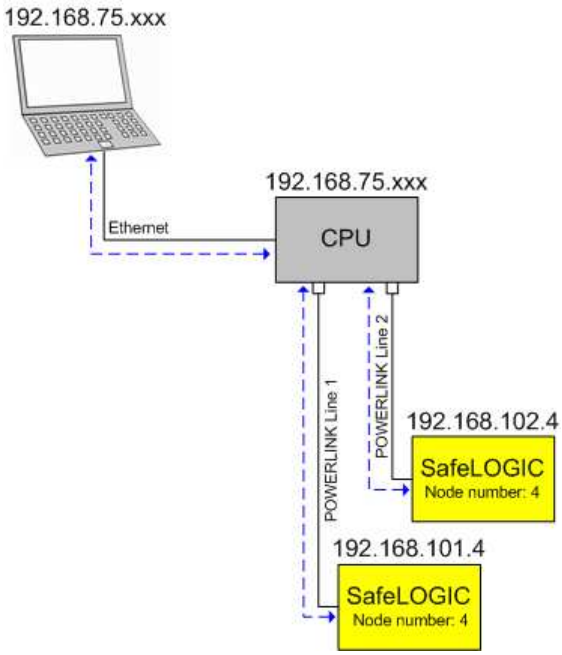
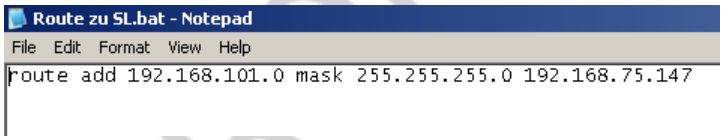


Fig. 97: Online via the standard CPU

If available, a direct connection (point-to-point) to the standard CPU can be used to establish online communication to the SafeLOGIC. To do this, the standard CPU automatically routes the data.

The route to the SafeLOGIC via the standard CPU must be specified on the PC. A Windows command is provided for this purpose.



- route add: Windows command for adding a route
- 192.168.101.0: Target network (in this case the POWERLINK network on the first line)
- mask: Subnet mask
- 255.255.255.0: Subnet mask
- 192.168.75.147: Gateway (in this case, the IP address of the standard CPU)

Click on **Online:TCPIP communication settings** to open the online communication settings.

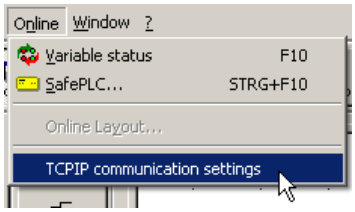


Fig. 98: Open the online communication settings

A dialog box is opened for configuring the online communication settings.

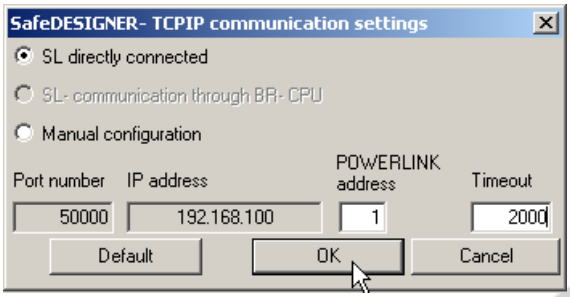


Fig. 99: Direct connection

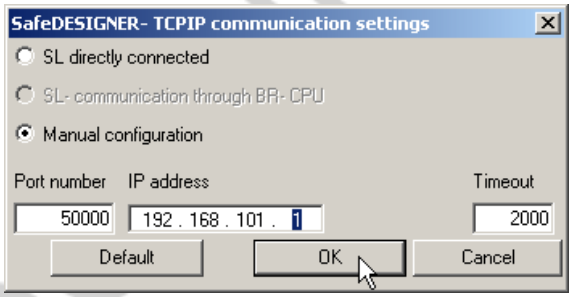


Fig. 100: Via the standard CPU

The **SafePLC** button is used to open the communication window for the SafeLOGIC.

Fig. 101: Button for opening the communication window

The SafeLOGIC states are shown in color after the communication window has been opened. A yellow background indicates that the SafeLOGIC is in Run[Safe] state. A red background means that the SafeLOGIC is in Run[Debug] state.

Fig. 102: Run[Safe] state

Fig. 103: Run[Debug] state

The **Debug** and **Safe** buttons can be used to switch between states.

The following actions are possible in the Run[Safe] state:

- Debug: Changes to the Run[Debug] state
- Info: Displays information
- Upload: Uploads project sources to the SafeDESIGNER
- Error: Displays error information

Change to the Run[Debug] state to perform a download.

A notification appears when changing from the Run[Safe] state to the Run[Debug] state, indicating that you are leaving the safe state.

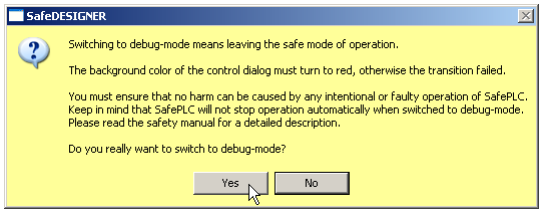


Fig. 104: Output window

The following actions are possible in the Run[Debug] state:

- Safe: Changes to the Run[Safe] state
- Stop: Stops the program execution → Stop[Debug] state
- Halt: Pauses the program → Halt[Debug] state
- Upload: Uploads project sources to the SafeDESIGNER
- Info: Displays information
- Error: Displays error information

The SafeLOGIC must be stopped (in Stop[Debug] state) to perform a download.

Fig. 105: Stop[Debug] state

The following actions are possible in the Stop[Debug] state:

- Safe: Changes to the Run[Safe] state
- Start: Starts the safety application → Run[Debug] state
- Halt: Pauses the program → Halt[Debug] state
- Download: Downloads the safety application
- Upload: Uploads project sources to the SafeDESIGNER
- Info: Displays information
- Error: Displays error information

The safety application can now be downloaded. A dialog box appears for defining the execution routines after the download and for storing the project source.

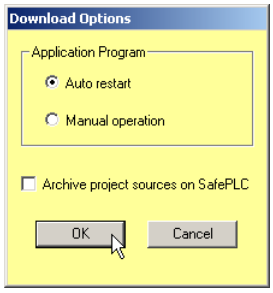


Fig. 106: Download options

- Auto restart: The safety application is automatically started after restart
- Manual operation: The safety application must be started manually via the communication window
- Archive project sources on SafePLC: The sources are saved on the SafeKEY

A notification appears after a successful download.

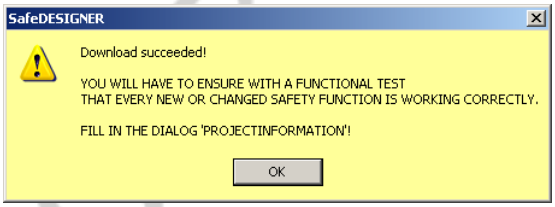


Fig. 107: Successful download

The safety application runs automatically when the SafeLOGIC is restarted or must be started manually via the communication window and the Run[Debug] state.

Example: Compiling and downloading

Compile your project, establish a direct connection to the SafeLOGIC and perform a download.

Fig. 108: Creating a safety application

4.5 Application diagnostics

SafeDESIGNER provides a few possibilities for performing diagnostics in the safety application:

- Checking the I/O status bits
- Checking the variable status (monitor mode)
- Forcing variables
- Executing the safety application in cycles
- Logbook

4.5.1 Checking the I/O status bits

Each channel contains one bit of information about the state of the channel or the multi-channel analysis.

Additional information about the current and the physical current is provided for SO modules.

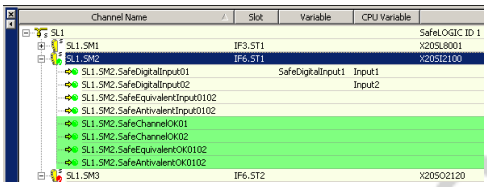


Fig. 109: SI status information

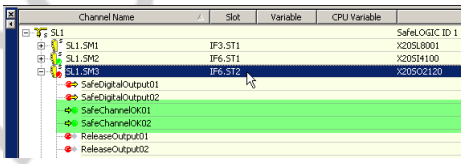


Fig. 110: SO status information

Note:

The status information channels must be used in the worksheet to receive data.

4.5.2 Checking the variable status

The SafeDESIGNER has a sort of monitor mode, similar to Automation Studio. It can be enabled using the **Variable status** button.

Fig. 111: Button for enabling variable status

The current value of each variable is now displayed in the graphic editor.

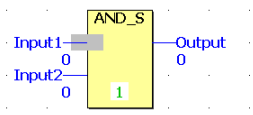


Fig. 112: Variable status enabled

4.5.3 Forcing variables

It is possible to force variables, as in Automation Studio. To do this, variable status (monitor mode) must be enabled and you must change to the Run[Debug] state.

Caution:

Forced variables can cause dangerous circumstances on the machine. Therefore, always make sure that the machine is appropriately secured.

Double-clicking on the variable will open a wizard for activating the force procedure.

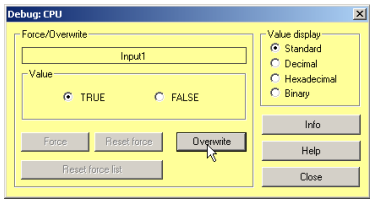


Fig. 113: Forcing variables

This action must be confirmed by the user. A notification window warns of the potential risks.

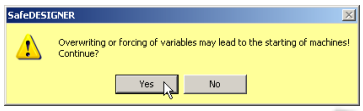


Fig. 114: Activate forcing

The forced variables are represented by a different color in the graphic editor.

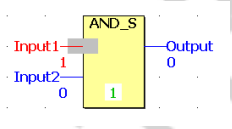


Fig. 115: Forced variables I

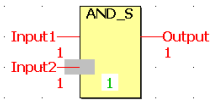


Fig. 116: Forced variables II

If you switch to the Run[Safe] state via the communication window, then all forced variables are reset and a corresponding notification window appears.

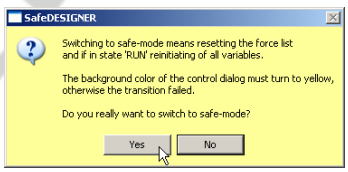


Fig. 117: Message window indicating that forced variables will be reset

4.5.4 Executing the safety application in cycles

SafeDESIGNER allows running the safety application in cycles (single cycle) for testing purposes.

To do this, you must first change to the Run[Debug] state and then to the Halt[Debug] state using the **Halt** button.

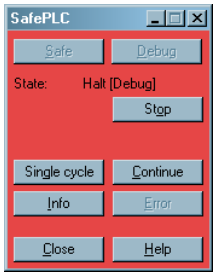


Fig. 118: Halt[Debug] state

The following actions are possible in the Halt[Debug] state:

- Stop: Stops the program execution, Stop[Debug] state
- Single cycle: Safety program runs 1 time
- Continue: Safety program runs cyclically
- Info: Displays information
- Error: Displays error information

The safety application runs one time by pressing the **Single cycle** button.

Note:

Press **Continue** to change back to the Run[Debug] state.

Example: Application diagnostics

Test and diagnose your safety application using the methods explained.

4.5.5 Logbook

The system keeps a logbook of the safety-related components which can be accessed via Automation Studio.

The following entries are made in this logbook:

- Exchanging the modules
- Configure the modules
- Application download
- Firmware download
- Changes to the Fail Safe state

Note:

The logbook can be accessed by selecting **Open:Logger** and then selecting the module **$arlogsaf**.

4.6 Create documentation

SafeDESIGNER can be used to create documentation for the safety application.

Click on **Project:Project Information** to open the documentation interface.

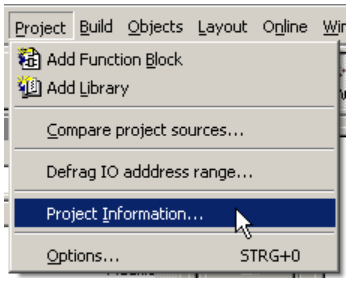


Fig. 119: Opening the documentation interface

An interface is opened for entering data relevant to the documentation.

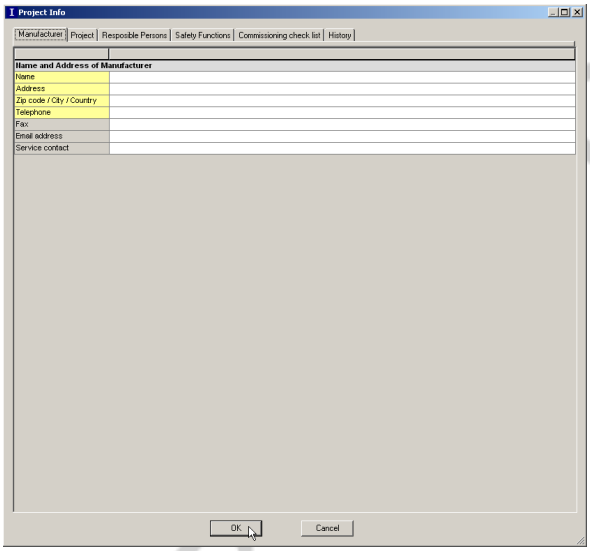


Fig. 120: Project documentation

The documentation options in SafeDESIGNER consist of the following aspects:

- Manufacturer: Details about the machine manufacturer
- Project: Information about the safety application
- Responsible Persons: Persons responsible
- Safety Functions: List of safety functions
- Commissioning check list: List to help during commissioning
- History: Safety application history

Note:

It is recommended to always fill out the yellow fields.
Gray fields can be filled out as desired.

4.6.1 Manufacturer

Data about the machine manufacturer such as name, address and contact information is entered here.

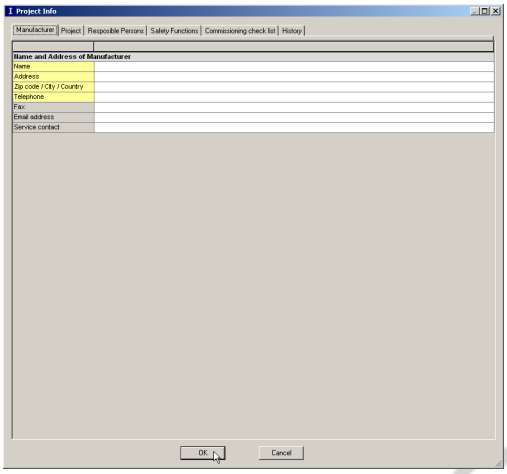


Fig. 121: Manufacturer

4.6.2 Project

Data about the machine and the safety application, such as the machine name and machine number, is entered here.

The project data is automatically maintained by the system.

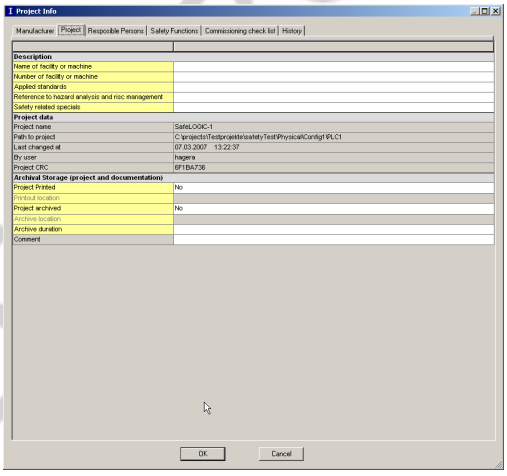


Fig. 122: Project

Note:

The unique CRC number of the safety application is also stored here.

4.6.3 Responsible Persons

This tab contains a list of the persons responsible for the project, such as the project manager, safety application, safety application testers, etc.

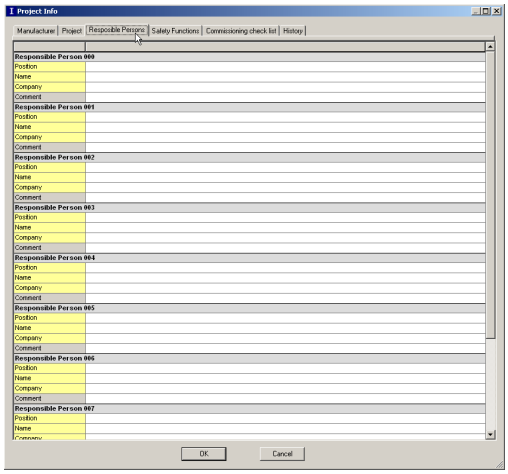


Fig. 123: Responsible Persons

4.6.4 Safety Functions

The safety functions configured for the machine are entered here.

You can see here if a safety function was successfully tested during integration (see V-model).

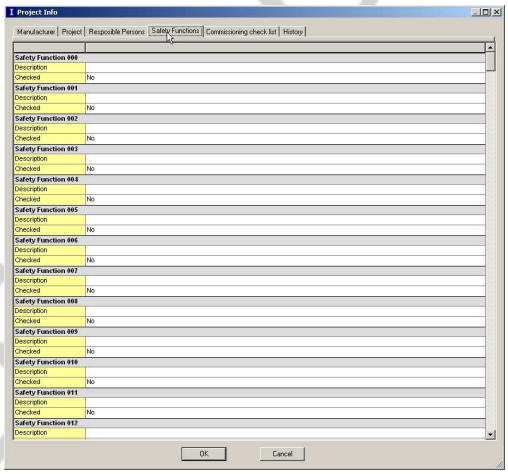


Fig. 124: Safety Functions

4.6.5 Commissioning check list

This list should provide some help for the validation.

This is where the safety application programmer can enter the data, which must be taken into consideration during commissioning such as network connections, cabling, etc.

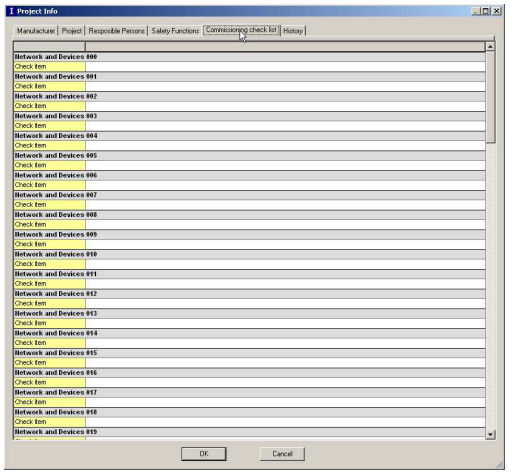


Fig. 125: Commissioning check list

4.6.6 History

The history text for the safety application is managed on the History tab. The respective CRC number, and other information regarding changes is entered for each revision.

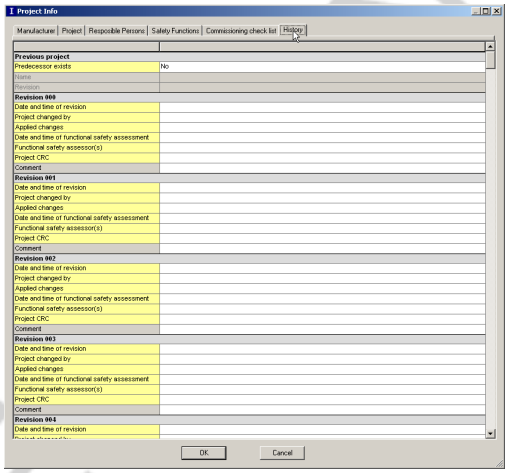


Fig. 126: History

The project documentation can be printed once the project information has been filled out.

Click on **File:Print Project**.

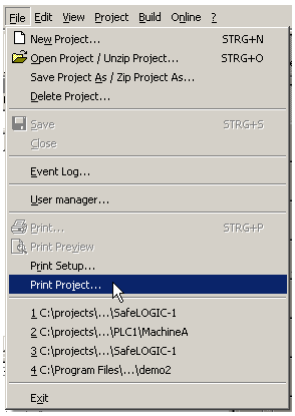


Fig. 127: Printing documentation

A new window is opened where individual sections of the documentation can be selected.

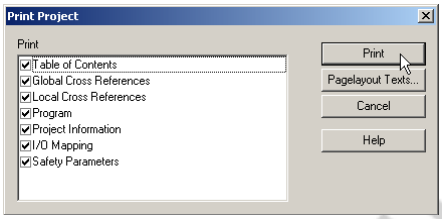


Fig. 128: Selecting documentation sections

These include:

- Table of contents: Table of contents
- Global cross references: Global cross references
- Local cross references: Local cross references
- Program: Main worksheet and self-made function blocks
- Project information: Project information
- I/O mapping
- Safety parameters: Module settings and parameters

The **Pagelayout Texts** button can be used to make additional settings for the documentation.

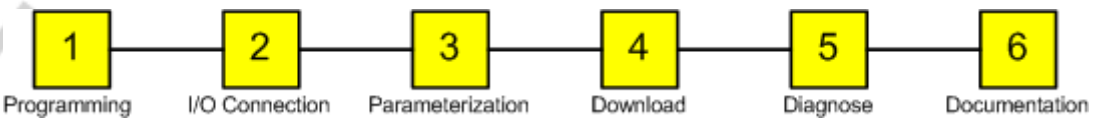


Fig. 129: Creating a safety application

4.7 Event Logger

SafeDESIGNER has an integrated Event Logger for automatically recording specific user actions.

The Event Logger can be accessed via **File:Event Log**.

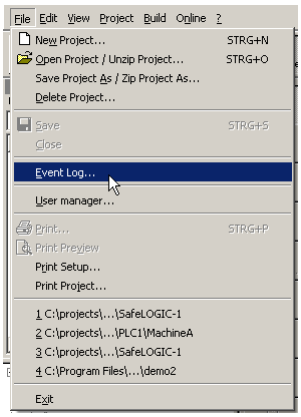


Fig. 130: Opening the Event Logger

A new window is opened where the Event Logger entries are displayed.

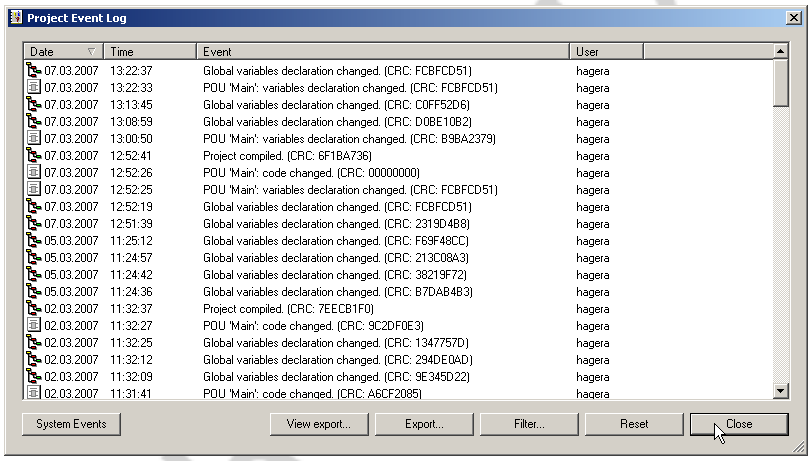


Fig. 131: Event Logger records

4.8 Adding a LD network

The **Network** button is provided in the toolbar for adding a network.

Fig. 132: Inserting a LD network

A network is inserted to the worksheet at the current cursor position.

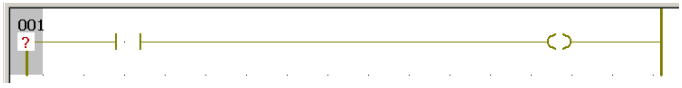


Fig. 133: Network added to worksheet

Variables must now be connected to the network elements. A wizard is opened by double-clicking on the element.

The scope must be set to "Local" to link a local variable. A new variable can be created or a previously declared variable can be defined using the drop-down menu.

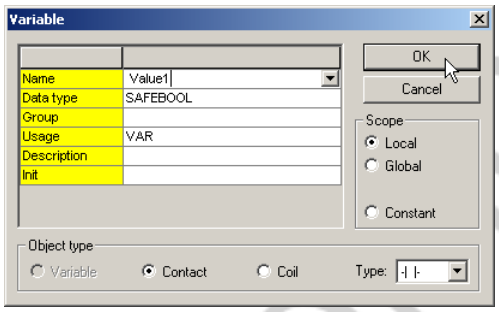


Fig. 134: Wizard for linking local variables

The variable is connected to the element after the dialog box has been confirmed.

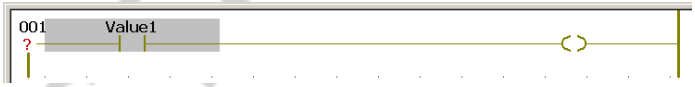


Fig. 135: Variable linked with element

The scope must be set to "Global" to link an I/O variable. A new variable can be created or a previously declared variable can be defined using the drop-down menu.

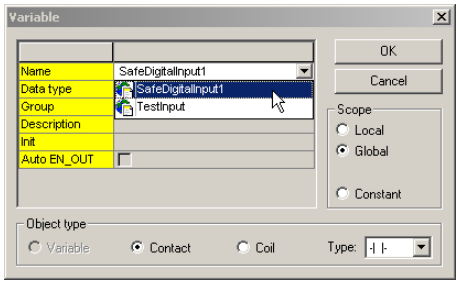


Fig. 136: Wizard for linking global variables

Note:

The variable must have been declared first or linked later with an I/O channel.

The variable is connected to the element after the dialog box has been confirmed.

Fig. 137: Variable linked with element

4.8.1 Network with function block

It is also possible to group more complex networks with a function block

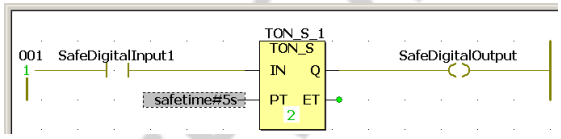


Fig. 138: Complex network with function block

To do this, a network must first be added to the worksheet.

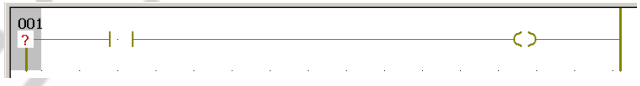


Fig. 139: Inserting a network

Left-click to select the middle part of the network.

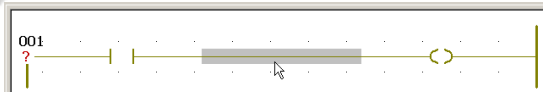


Fig. 140: Select network

This part of the network is deleted using the Delete key.

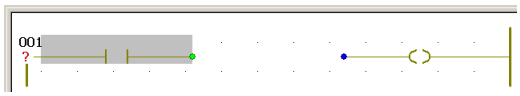


Fig. 141: Part of network deleted

A function block can now be added to the middle of the network.

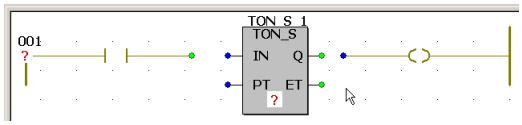


Fig. 142: Function block added

The respective connection lines must now be drawn using the **Connect** button.

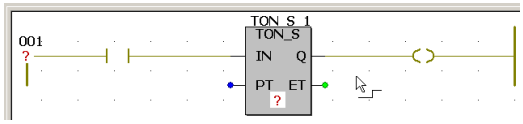


Fig. 143: Drawing connection lines

A time constant must still be connected to the PT input. Double-click on the input to open a wizard. The scope to set to constant. A time constant is defined with "safetime#duration".

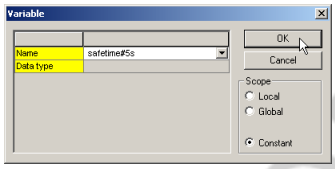


Fig. 144: Creating time constants

The respective network elements are then linked with variables.

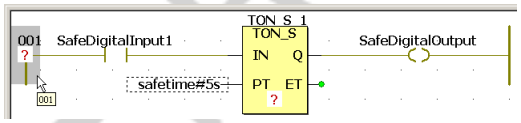


Fig. 145: Finished network

Example: Inserting a LD network

Add a network to your project, add a basic function block to the network and link local variables or I/O channels.

4.9 Creating your own function block

You can create your own function blocks in SafeDESIGNER to hide functionalities or to use the same code repeatedly.

Right-click in the project view to open the menu for creating function blocks.

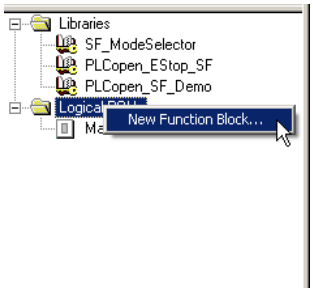


Fig. 146: Creating a new function block

A dialog box is opened for defining the name of the function block.

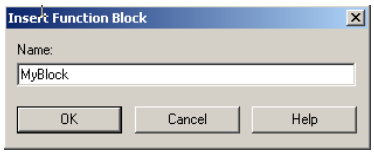


Fig. 147: Defining the function block name

The function block appears in the project view after confirming this dialog box.

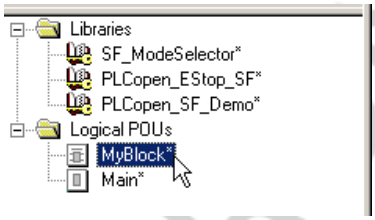


Fig. 148: Own function block in the project view

Note:

The function block is programmed in a separate worksheet.
It can directly access I/O variables.

Double-click on the function block in the project view to open the worksheet for programming. The programming languages LD and FUB are once again provided.

The type of usage can be defined for the corresponding variable in the variable declaration. You can choose between a local variable, an input or an output.

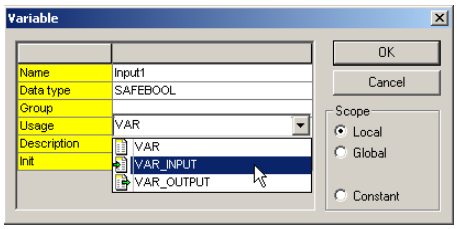


Fig. 149: Variable declaration in the function block

Function blocks can be added directly using the Edit Wizard.

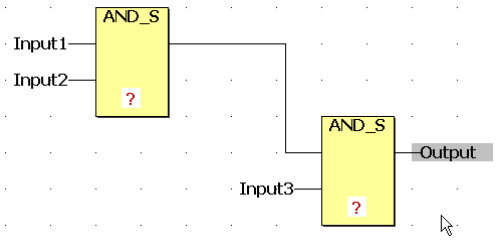


Fig. 150: Own function block

The created function block can be added using the Edit Wizard. To do this, the project name must be selected in the Edit Wizard for the group.

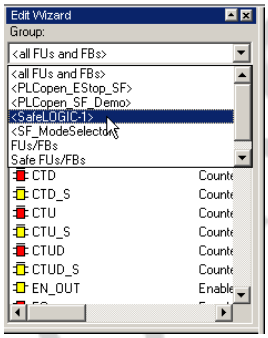


Fig. 151: Group Edit Wizard

Now, all of the function blocks that you have created are shown in the Edit Wizard and can be added directly to the main worksheet.

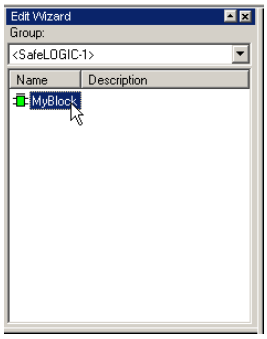


Fig. 152: Own function blocks

An instance variable is required.

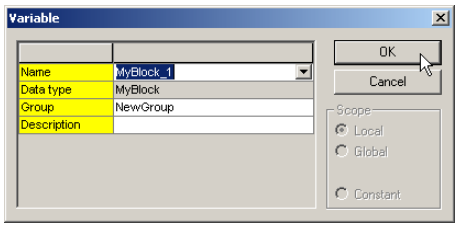


Fig. 153: Creating an instance variable

The inputs and outputs can be linked as shown earlier.

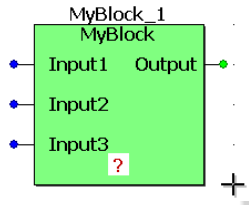


Fig. 154: Own function block in the work sheet

Example: Creating your own function block

Create a basic function block, use it in your worksheet and test its functionality.

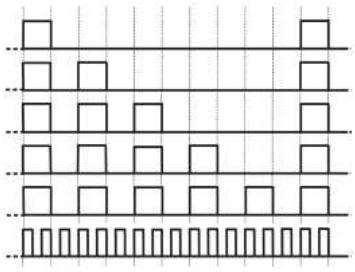
5. COMMISSIONING AND SERVICE

This section will cover a few different scenarios involving commissioning and maintenance.

The SafeLOGIC has a user interface that consists of status LEDs and a operating mode switch. This interface is used to display states and to perform various actions.

Fig. 155: SafeLOGIC user interface

5.1 Status LEDs

LED	Color	Status	Description
R/E		Off	Boot phase
	Green	On	Application found and executed
		Blinking	Application found, but CPU is stopped
	Yellow	On	SafeDESIGNER in DEBUG mode
		Blinking	SafeDESIGNER in debug mode, application stopped
		Blinking quickly	No application found on the SafeKEY
ENTER	Green	On	Authorization missing
		Blinks for 0.8 s	Confirmation of correct entry
		Blinks for 5 s	Faulty operation
MXCHG	Yellow	Off	Module configuration OK
			Exchange of 1 module detected
			Exchange of 2 modules detected
			Exchange of 3 modules detected
			Exchange of 4 modules detected
			Exchange of more than 4 modules
			Missing module detected
FW-ACKN	Yellow	Off	Firmware configuration OK
		Blinking	Firmware update executed
		On	SafeKEY was exchanged

FAIL	Red	**F**	**A**	**I**	**L**	
		X	X		X	Boot phase, loading of the firmware, status when SafeKEY is missing
		X	X	X	X	Complete HW test (max. duration approx. 5 s)
		X	**X**	X	**X**	Initialization and start up of the firmware
					X	Preoperational state
						Operational state
		X	X	X	X	Fail-safe status of the entire module
SKEY	Yellow	Off				No access to the SafeKEY
		Blinking				Access to the SafeKEY

5.2 Operating mode switch

The operating mode switch is used to set the operating mode.

Switch position	Operating mode	Description
FW-ACKN	Acknowledge Firmware	Exchange Firmware
SK-XCNG	SafeKEY exchange	Exchange SafeKEY
SK-COPY	SafeKEY copy	Copy configuration file from the SafeKEY
SCAN	Scan	Perform module scan
Test	Test	Perform LED test
1,2,3,4,n	Exchange module	Confirms a module exchange with 1, 2, 3, 4 or more than 4 modules

Note:

More information can be found in the User's Manual.

5.3 Starting up the system

- Structure hardware
- Create Compact Flash or perform download (standard CPU)
- Starting up the entire system
- Wait for firmware update (lasts approximately 3x as long because there is 3x more data)
- SafeKEY is empty (no application) – the F A I L LEDs on the SafeLOGIC are lit red
  The r LED on the module is lit green, the channel LEDs red and SE is off

Note:

A different reaction/blinking can be expected if the SafeKEY is not empty.

- Get online on the SafeLOGIC via the standard CPU
- Download safety application
- LED FW-ACKN lit (SafeKEY must be acknowledged)
- Acknowledge SafeKEY (set SK-XCNG and confirm with ENTER)
- Automatic SafeLOGIC reboot
- MXCHG LED blinking (indicates the number of new modules)
- Acknowledge new modules, (set 1, 2, 3, 4, or n and confirm with ENTER)
- Module data is written to the SafeKEY
- LED FW-ACKN blinking
- Acknowledge firmware, set FW-ACKN and confirm with ENTER
- Confirmed firmware version is saved to the SafeKEY
- Automatic SafeLOGIC reboot
- Modules are lifted
- Standard application and safety application running
- Perform comprehensive test

5.4 Module missing

The system checks the safety-relevant hardware configuration in a time interval set by the system. Any faulty modules found are indicated by a quickly blinking MXCHG LED.

5.5 Module exchange

The system checks the safety-relevant hardware configuration in a time interval set by the system. Any new modules found are signaled by the SafeLOGIC.

- MXCHG LED blinking (indicates the number of new modules)

Note:

If more modules were replaced than were signaled, then you can start a manual run-through using the **SCAN** button.
This could occur on larger machines where the automatic scan takes a long time

- Acknowledge new modules, set 1, 2, 3, 4, or n and confirm with ENTER
- Run test on the affected machine part

5.6 Replace SafeLOGIC

- Replace SafeLOGIC
- Insert old SafeKEY
- MXCHG LED blinking 1x slowly
- Acknowledge new SafeLOGIC, set 1 and confirm with ENTER
- No test required

Note:

A SafeLOGIC firmware update might take place.

5.7 Safety application update

- Old SafeKEY is inserted
- Set SK-COPY and confirm with ENTER
- The configuration data is copied from the SafeKEY to RAM
- Remove old SafeKEY
- Insert new SafeKEY
- Confirm with ENTER
- System restarts
- Run test on the affected machine part

5.8 Firmware update

- The Compact Flash contains a new firmware version
- Wait for firmware update
- FW-ACKN blinking
- Acknowledge firmware, set FW-ACKN and confirm with ENTER
- Modules are lifted
- Perform comprehensive test

6. EXAMPLE PROJECT

Control of a circular saw is an example for a small safety application.

The following safety components are available for the machine:

- E-stop switch (second SI module, channel 3 & 4)
- Operating mode switch (first SI module, channel 1 & 2)
- Start/Stop button (acknowledge button) (first SI module, channel 3)
- Light curtain (second SI module, channel 1 & 2)

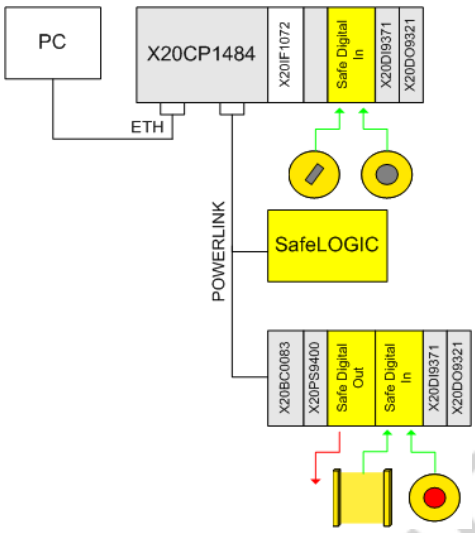


Fig. 156: Machine structure

The enable output for a motor is controlled by the safety application.

The following hardware is available for the machine:

- 1x X20 CPU
- 1x SafeLOGIC
- 2x Safe Digital Input
- 1x Safe Digital Output
- X20 slice(s)

Task: Creating the project

Create a new project using the provided hardware.

6.1 Safety Requirements Specification (SRS)

The following safety functions must be implemented:

- E-stop switch function
- Operating mode switch function
- Light curtain function

6.1.1 E-stop switch function

The following function must be implemented:

- Acknowledge not necessary after startup
- Acknowledge required via the acknowledge button after unlocking the E-stop
- Simultaneousness within 200ms
- Maximum response time: 500ms

6.1.2 Operating mode switch function

The following function must be implemented:

- Acknowledge not necessary after startup
- 2 possible operating modes are used (manual, automatic)
- The operating modes can be switched directly without acknowledge
- The maximum time for an invalid state can be 500ms
- Maximum response time: 500ms

6.1.3 Light curtain function

The following function must be implemented:

- Acknowledge not necessary after startup
- Acknowledge required via the acknowledge button after the light curtain has been passed-through
- Simultaneousness within 300ms
- Maximum response time: 500ms

6.1.4 Interaction of functions

Two operating modes are implemented:

- Manual operation
- Automatic operation

> **Note:**
>
> Make sure that an E-stop will shut off the output in all operating modes.

Manual operation

The following applies if manual operation is selected using the operating mode switch:

- The output is only enabled by the safety application as long as the Start/Stop button is pressed and the safety features have not been violated
- The light curtain is not enabled in this operating mode
- The motor is turned on in the standard application as long as the Start/Stop button is pressed
- A safety feature violation must be acknowledged via a positive edge of the Start/Stop button

Automatic operation

The following applies if automatic operation is selected using the operating mode switch:

- The output is always enabled by the safety application as long as none of the safety features has been violated
- The light curtain is enabled in this operating mode
- The motor is started in the standard application via a positive edge on the Start/Stop button, and stopped if another positive edge is reached
- A safety feature violation must be acknowledged via a positive edge of the Start/Stop button

> **Note:**
>
> The channel for the approval principle can be used for control in the standard application.

6.2 Programming the safety application

Task: Programming the safety application

Program the function specified in the SRS using the respective functions blocks from the PLCopen Safety Library.
Use the function blocks SF_EmergencyStop, SF_ModeSelector and SF_ESPE.
Be sure to take the automatic restart inhibit of the SO module into consideration.

Note:

A more detailed description of the function blocks can be found by **right-clicking on the FUB:FB/FU Help**.

6.3 Configure the modules

Task: Configure the modules

Define the module parameters according to the connected safety components (E-stop, light curtain).

6.4 Download and commissioning

Task: Download and commissioning

Establish an online connection with the SafeLOGIC (via the standard CPU), format your SafeKEY and then download your safety application.

Startup your safety application.

6.5 Diagnostics and module replacement

Task: Diagnostics and module replacement

Test the functionality of your safety application.

Replace a module (module or SafeLOGIC from your neighbor) and test your application.

7. SUMMARY

You are now familiar with the B&R Integrated Safety Technology concept and can apply it to your machines.

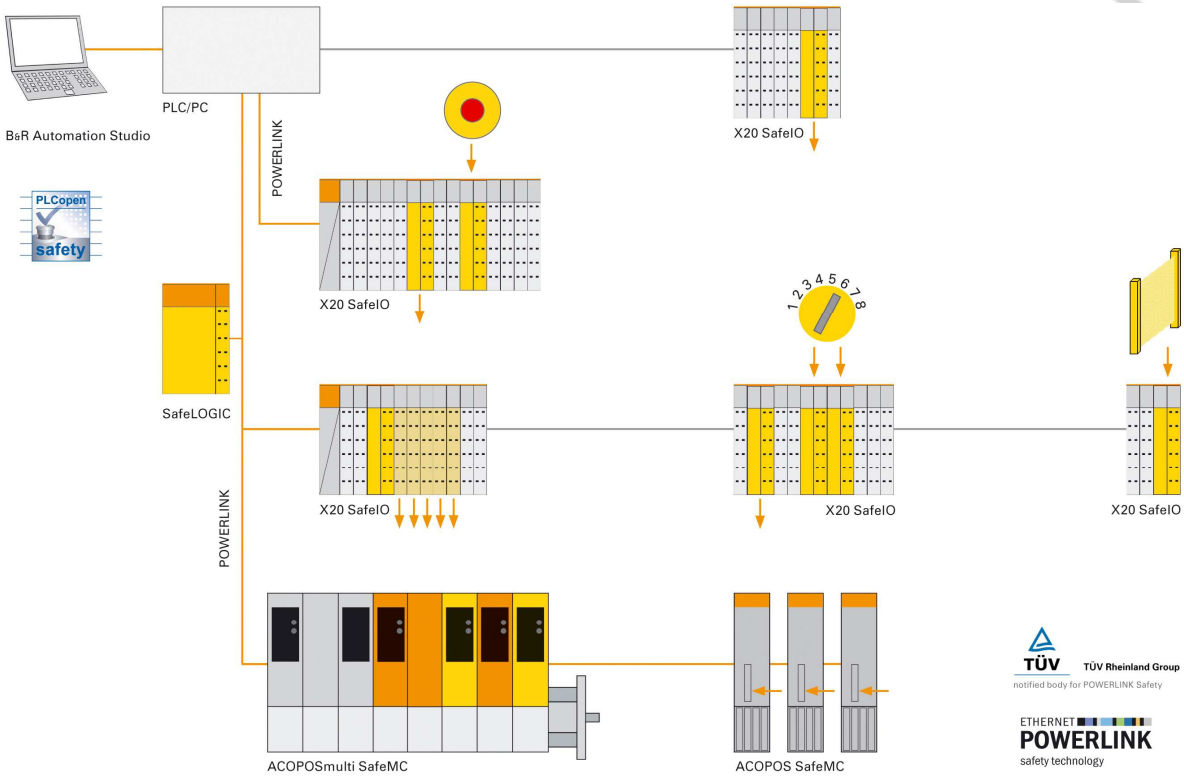


Fig. 157: B&R integrated safety technology

You know the steps between Automation Studio and SafeDESIGNER for creating a simple or complex safety application.

You understand the procedure for creating a safety application.

You are able to easily create documentation for your safety application.

You have programmed your own safety functions.

You are familiar with the procedure for commissioning and maintaining the safety technology.

8. APPENDIX

8.1 Function blocks from the PLCopen Safety Library

SF_Equivalent

SF_Antivalent

SF_ModeSelector

SF_EmergencyStop

SF_ESPE

SF_SafeStop1

SF_SafeStop2

SF_GuardMonitoring

SF_SafelyLimitedSpeed

SF_TwoHandControlTypeII

SF_TwoHandControlTypeIII

SF_GuardLocking

SF_TestableSafetySensor

SF_MutingSeq

SF_MutingPar

SF_MutingPar_2Sensor

SF_EnableSwitch

SF_SafetyRequest

SF_OutControl

SF_EDM

8.2 Possible solution to the sample project

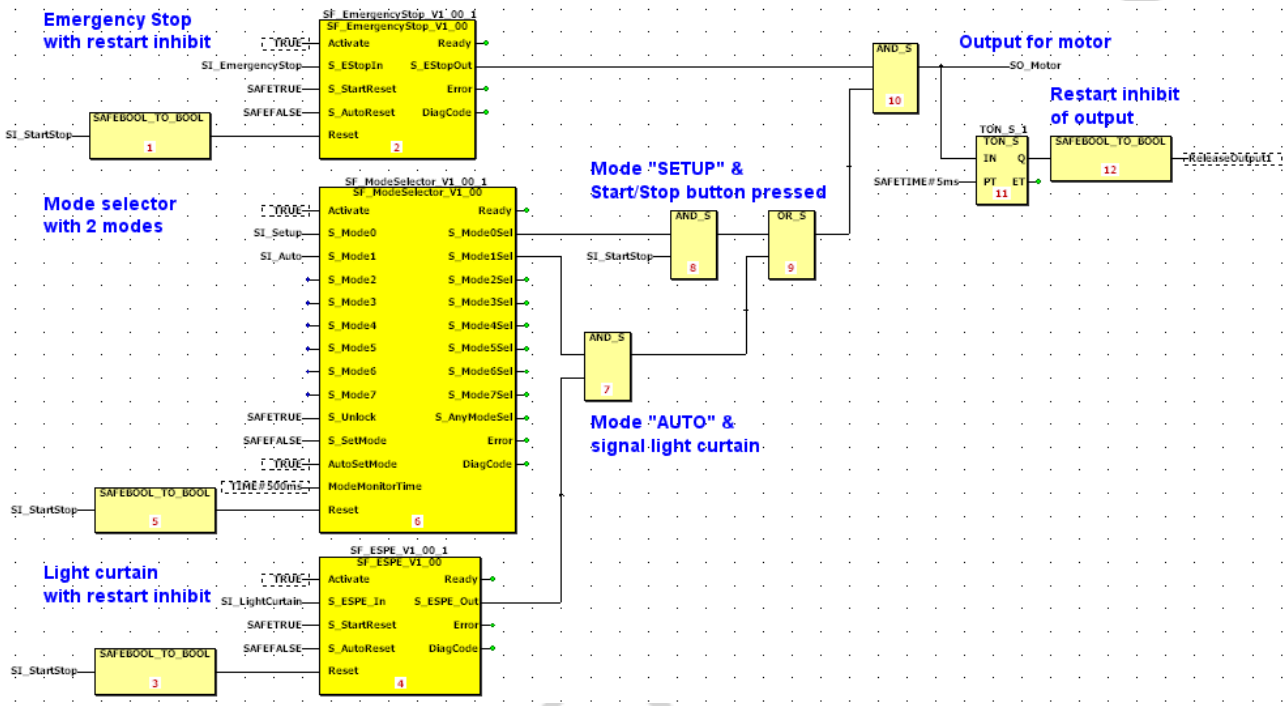


Fig. 158: Possible solution to the sample project

Overview of training modules

TM200 – B&R Company Presentation **
TM201 – B&R Product Spectrum **
TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM212 – Automation Target **
TM213 – Automation Runtime
TM220 – The Service Technician on the Job
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Generation
TM240 – Ladder Diagram (LAD)
TM241 – Function Block Diagram (FBD)
TM246 – Structured Text (ST)
TM247 – Automation Basic (AB)
TM248 – ANSI C
TM250 – Memory Management and Data Storage
TM260 – Automation Studio Libraries I
TM261 – Closed Loop Control with LOOPCONR

TM400 – The Basics of Motion Control
TM410 – The Basics of ASiM
TM440 – ASiM Basic Functions
TM441 – ASiM Multi-Axis Functions
TM445 – ACOPOS ACP10 Software
TM446 - ACOPOS Smart Process Technology
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors

TM500 – The Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER

TM600 – The Basics of Visualization
TM610 – The Basics of ASiV
TM630 – Visualization Programming Guide
TM640 – ASiV Alarm System
TM650 – ASiV Internationalization
TM660 – ASiV Remote
TM670 – ASiV Advanced

TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM730 – PVI OPC

TM800 – APROL System Concept
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM840 – APROL Parameter Management and Recipes
TM850 – APROL Controller Configuration and INA
TM860 – APROL Library Engineering
TM865 – APROL Library Guide Book
TM870 – APROL Python Programming
TM890 – The Basics of LINUX

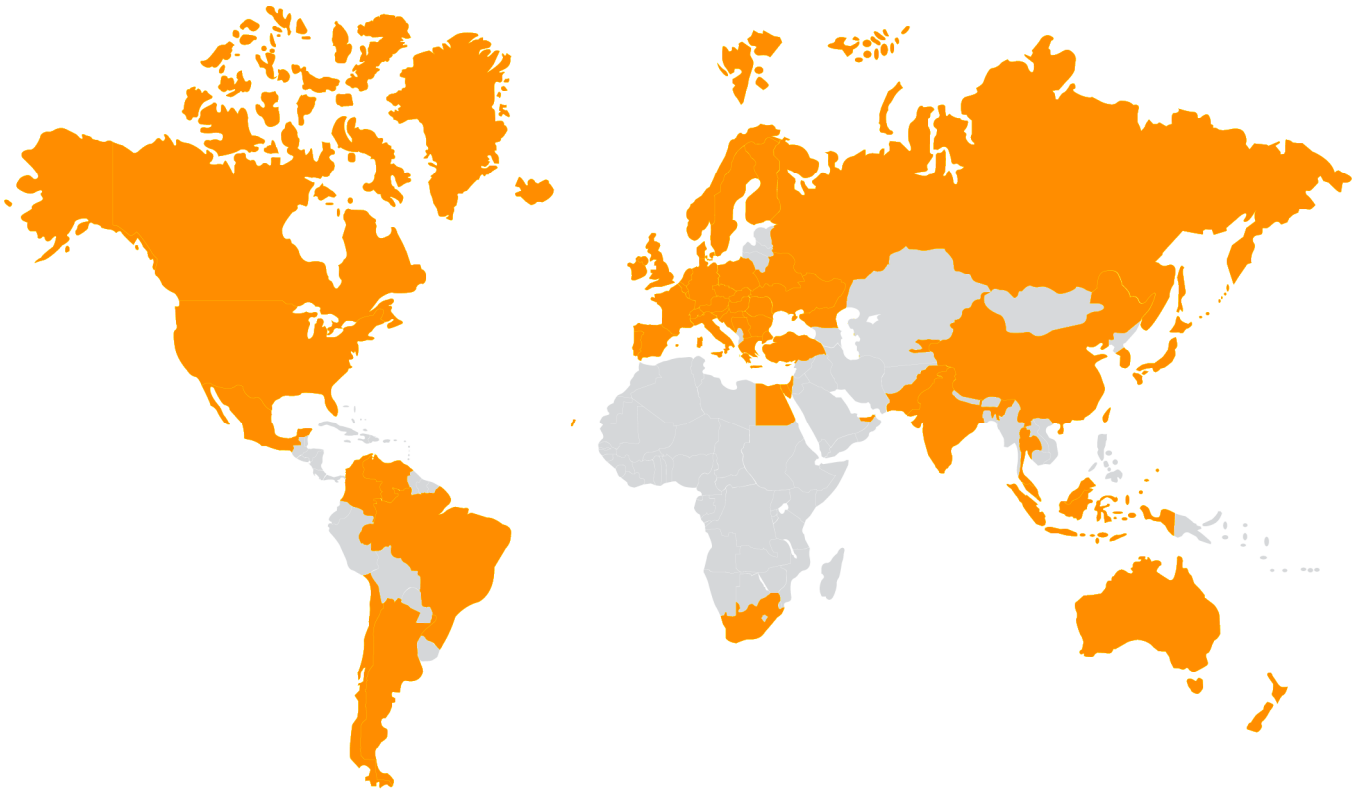
**) see Product Catalog

CORPORATE HEADQUARTERS

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
B&R Strasse 1
5142 Eggelsberg
Austria
Tel.: +43 (0) 77 48 / 65 86 - 0
Fax: +43 (0) 77 48 / 65 86 - 26
info@br-automation.com
www.br-automation.com

TM510TRE.30-ENG 0208

140 offices in more than 55 countries - **www.br-automation.com/contact**



Australia • Argentina • Austria • Belarus • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Croatia • Cyprus
Czech Republic • Denmark • Egypt • Emirates • Finland • France • Germany • Greece • Hungary • India • Indonesia
Ireland • Israel • Italy • Japan • Korea • Luxemburg • Kyrgyzstan • Malaysia • Mexico • The Netherlands • New Zealand
Norway • Pakistan • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa
Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam