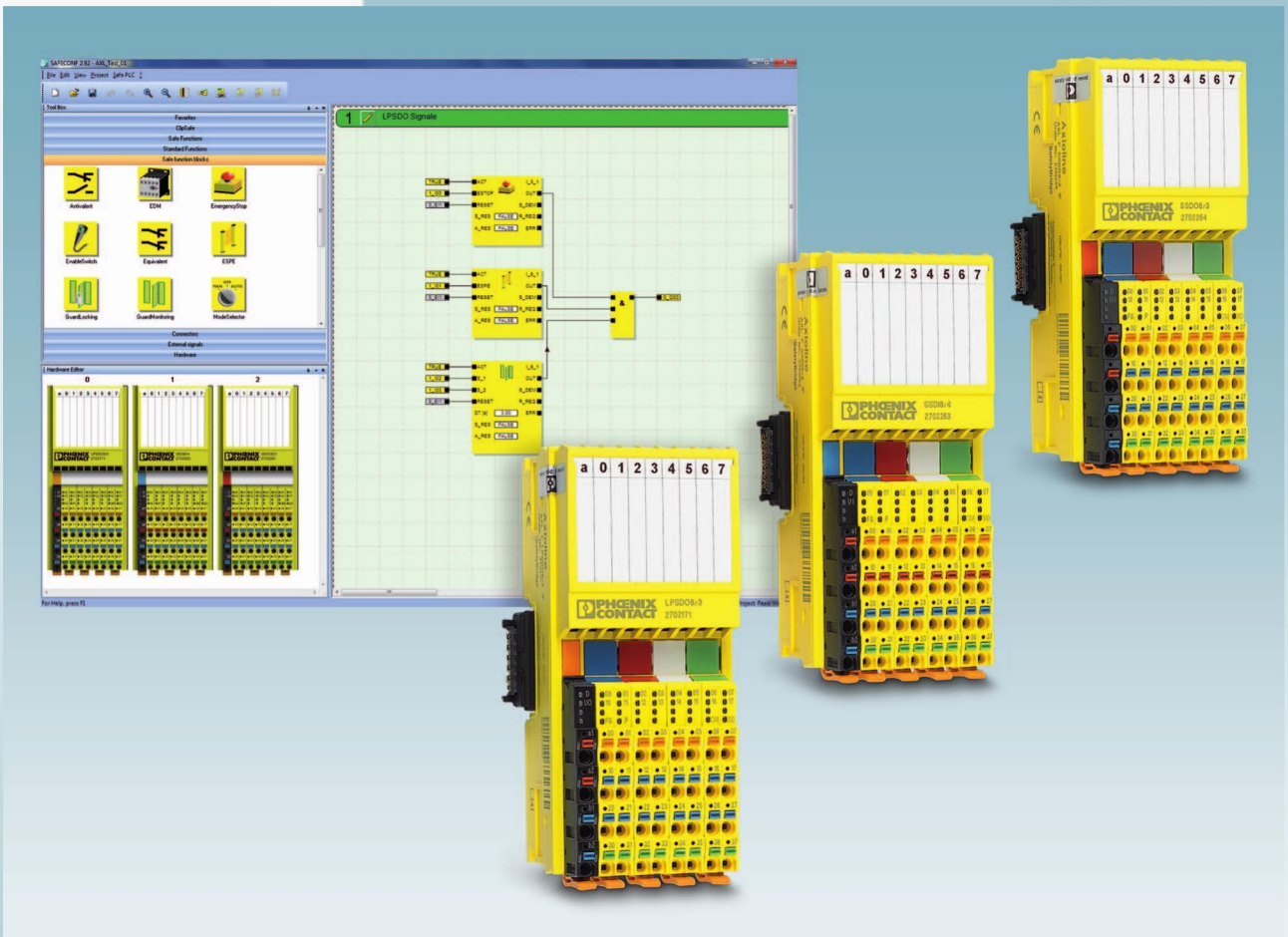


Axioline F SBT V3 configuration on a CODESYS-based controller

Quick start guide

PHOENIX CONTACT
INSPIRING INNOVATIONS

Quick start guide

Axioline F SBT V3 configuration on a CODESYS-based controller

2016-06-08

Designation: UM QS EN AXL F SBT V3 CODESYS

Revision: 00

Order No.: —

This user manual is valid for:

Designation	From HW/FW/FW version	Order No.
AXL F LPSDO8/3 1F	00/100	2702171
AXL F SSDI8/4 1F	01/200	2702263
AXL F SSDO8/3 1F	01/200	2702264
IB IL 24 LPSDO 8 V3-PAC	00/100/100	2701625
IB IL 24 PSDI 8-PAC	00/201	2985688
IB IL 24 PSDI 16-PAC	00/100	2700994
IB IL 24 PSDO 8-PAC	01/200/100	2985631
IB IL 24 PSDOR 4-PAC	01/200/100	2985864
IB IL 24 PSDO 4/4-PAC	01/201/100	2916493

Please observe the following notes

User group of this manual

The use of products described in this manual is oriented exclusively to qualified application programmers and software engineers, who are familiar with the safety concepts of automation technology and applicable standards.

Explanation of symbols used and signal words

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety measures that follow this symbol to avoid possible injury or death.

There are three different categories of personal injury that are indicated with a signal word.

DANGER This indicates a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING This indicates a hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION This indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

This symbol together with the signal word **NOTE** and the accompanying text alert the reader to a situation which may cause damage or malfunction to the device, hardware/software, or surrounding property.

This symbol and the accompanying text provide the reader with additional information or refer to detailed sources of information.

How to contact us

Internet

Up-to-date information on Phoenix Contact products and our Terms and Conditions can be found on the Internet at:
phoenixcontact.com

Make sure you always use the latest documentation.
It can be downloaded at:
phoenixcontact.net/products

Subsidiaries

If there are any problems that cannot be solved using the documentation, please contact your Phoenix Contact subsidiary.
Subsidiary contact information is available at phoenixcontact.com.

Published by

PHOENIX CONTACT GmbH & Co. KG
Flachsmarktstraße 8
32825 Blomberg
GERMANY

Should you have any suggestions or recommendations for improvement of the contents and layout of our manuals, please send your comments to:
tecdoc@phoenixcontact.com

General terms and conditions of use for technical documentation

Phoenix Contact reserves the right to alter, correct, and/or improve the technical documentation and the products described in the technical documentation at its own discretion and without giving prior notice, insofar as this is reasonable for the user. The same applies to any technical changes that serve the purpose of technical progress.

The receipt of technical documentation (in particular user documentation) does not constitute any further duty on the part of Phoenix Contact to furnish information on modifications to products and/or technical documentation. You are responsible to verify the suitability and intended use of the products in your specific application, in particular with regard to observing the applicable standards and regulations. All information made available in the technical data is supplied without any accompanying guarantee, whether expressly mentioned, implied or tacitly assumed.

In general, the provisions of the current standard Terms and Conditions of Phoenix Contact apply exclusively, in particular as concerns any warranty liability.

Phoenix Contact reserves the right to register its own intellectual property rights for the product identifications of Phoenix Contact products that are used here. Registration of such intellectual property rights by third parties is prohibited.

Other product identifications may be afforded legal protection, even where they may not be indicated as such.

Table of contents

1 Introduction

1.1 Purpose of this user manual

This quick start guide uses an example project to describe how to integrate SafetyBridge Technology V3 modules into a CODESYS-based controller.

An EtherCAT® network is used as an example. The procedure described in this quick start guide relates to CODESYS Versions V 2.3 and V 3.5.

1.2 Requirements

Knowledge

Knowledge of the following is required:

- The components used in the application
- The CODESYS software used
- The Microsoft Windows operating system

Hardware

The following hardware is required in order to start up the example system:

Designation	As of HW/FW	Order No.
AXL F LPSDO8/3 1F (logic module)	00/100	2702171
AXL F SSDI8/4 1F (input module)	00/200	2702263
AXL F DI8/1 DO8/1 1H (I/O module)	-	2701916
AXL F BK EC (EtherCAT® bus coupler)	-	2688899
CODESYS-based controller (a Raspberry board in the example)	-	-

- Programming device/PC
- Other components: emergency stop button, external reset button, signal lamp, contactor (optional)

Software

The Phoenix Contact software can be found in the download area for the specified product at phoenixcontact.net/products.

The following software is required in order to start up the example system:

Designation	Order No.
SAFECONF V2.92 or later	2986119
Integration package for SafetyBridge Technology V3	2702171
CODESYS as of V 2.3 or V 3.5	-

- Other software: Microsoft Windows

1.3 Additional documentation

Please refer to the documentation for the software used, the components used in the application, and the function blocks used.

The documentation for the SafetyBridge Technology V3 modules used must be strictly observed.

Description	Type	Order No.
User manual: Axioline F module with integrated safety logic and safe digital outputs	UM EN AXL F LPSDO8/3 1F	2702171
User manual: Axioline F module with safe digital inputs	UM EN AXL F SSDI8/4 1F	2702263

The documentation for Phoenix Contact devices can be found in the download area for the specified product at phoenixcontact.net/products.

1.4 Safety hotline

Should you have any technical questions, please contact our 24-hour hotline.

Phone: + 49 5281 9-462777

E-mail: safety-service@phoenixcontact.com

2 Overview of the integration of the SafetyBridge Technology V3 system

Safety with the SafetyBridge Technology V3 system

Within a SafetyBridge Technology V3 system, safety can only be ensured by using the modules of this system (AXL F LPSDO8/3 1F and 1 to 16 satellites). None of the other components in the overall system are safety-related components. Errors at non-safety-related components or errors during integration of the SafetyBridge Technology V3 system are reliably detected by the SafetyBridge Technology V3 system components. These errors only reduce the system availability but not the system safety.

No safety controllers are required for the implementation of safety functions.

Table 2-1 Integration of a SafetyBridge Technology V3 island

Step	Process	See...
1	**Download and installation of the Phoenix Contact software** (not safety-related)	
	– Download and install SAFECONF configuration software	page 11
	– Download device description file for bus coupler	page 11
	– Download and install integration package for SafetyBridge Technology V3	page 11
2	**Hardware installation** (not safety-related)	
	– Set device DIP switches	page 12
	– Mount and wire bus configuration	page 13 User documentation for the devices
3	**Configure safety logic in SAFECONF** (safety-related)	
	– Create new project, assign safety island number	page 14, page 16
	– Configure the hardware structure	page 16
	– Parameterize I/O channels	page 17
	– Configure safety function	page 14
	– Export configuration and parameter data record	page 21 SAFECONF online help
4	**Configuring a CODESYS project for the controller** (not safety-related)	
	– Create project and import device description file	page 22
	– Configure the hardware structure	page 22
	– Integrate function blocks for SafetyBridge Technology V3	page 23
	– Create SBT program in CODESYS	page 24
	– Import configuration and parameter data record	page 28

Table 2-1 Integration of a SafetyBridge Technology V3 island

Step	Process	See...
	– Process data assignment of the devices	page 33
	– Compile project and download it to the controller	page 35
5	**Startup and overall safety validation** (safety-related)	page 35

System overview:

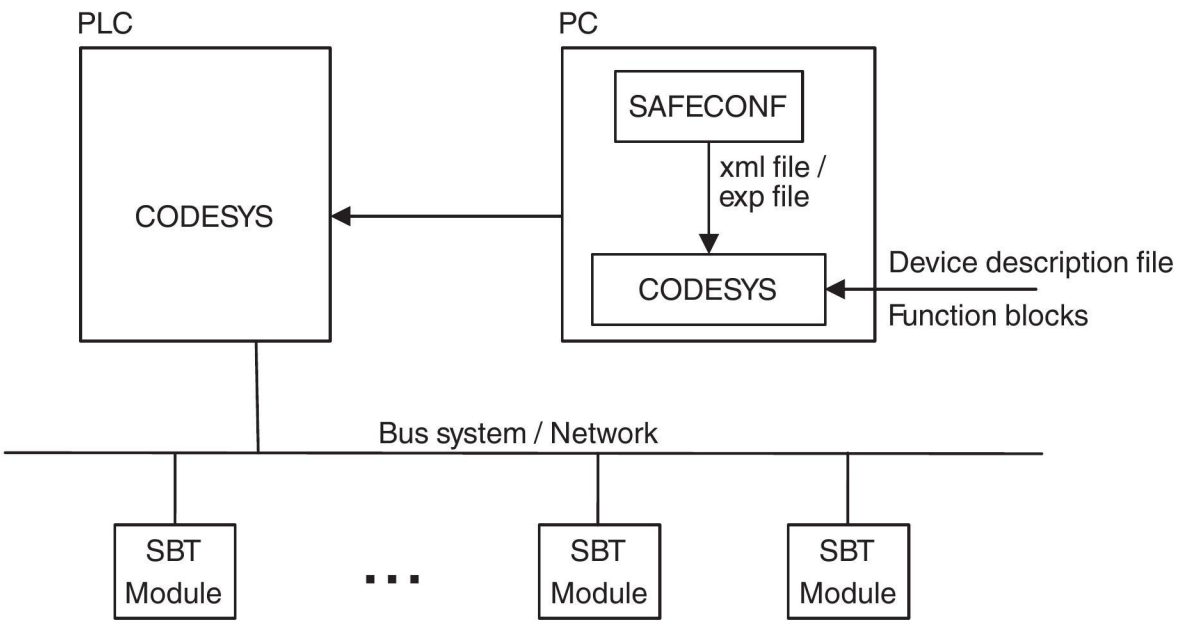


Figure 2-1 System overview of SafetyBridge Technology V3

3 Example project: two-channel emergency stop monitoring

3.1 Download and installation of the Phoenix Contact software

The Phoenix Contact software can be found in the download area for the specified product at phoenixcontact.net/products.

Make sure that you always use the latest version of the integration package and the function blocks for the CODESYS software version. See “Software” on page 7.

SAFECONF

1. Download the SAFECONF configuration software and install the software (Order No. 2986119).

Device description file (ESI)

2. Download the ESI file for the bus coupler (Order No. 2688899).

The ESI file contains all devices that can be connected to the bus coupler.

SBT V3 integration package

3. Download the integration package for SafetyBridge Technology V3 (Order No. 2702171).

Software

	Description	Language	Revision
☑	[exe, 123 MB] Software SafetyBridge technology integration package for controllers from Phoenix Contact, Rockwell and Siemens (S7-1200 from CPU 1214C, S7-1500, S7-300), Schneider as well as CODESYS-based controllers. SBT_V3_PLC_Integration_Packages_2.0.exe	International	2.0

Figure 3-1 Integration package in the download area for Order No. 2702171

4. Install the integration package as follows:
   - Run setup and select “CODESYS” during installation.

Please make a note of where the library files are installed, as you will need this information later when you open the library in CODESYS.

3.2 Hardware installation

3.2.1 Setting the device DIP switches

To implement the example project, make the following settings at the DIP switches of the SafetyBridge Technology V3 modules before installing the bus configuration.

Table 3-1 Setting the DIP switches

	CM		Island number					Satellite number				
	Operating mode	Reserved	SafetyBridge Technology V3 address: 32_{dec} (20_{hex})									
DIP switch	11	10	9	8	7	6	5	4	3	2	1	0
Setting for AXL F LPSDO8/3 1F	**off**	**on**	0	0	0	0	**1**	0	0	0	0	0
			1_{dec}					**0_{dec}**				

	CM		Island number					Satellite number				
	Operating mode	Reserved	SafetyBridge Technology V3 address: 33_{dec} (21_{hex})									
DIP switch	11	10	9	8	7	6	5	4	3	2	1	0
Setting for AXL F SSDI8/4 1F	**off**	**on**	0	0	0	0	**1**	0	0	0	0	**1**
			1_{dec}					**1_{dec}**				

You can display the complete DIP switch setting in the SAFECONF configuration software by right-clicking on the module and selecting “Display address switch”. See Figure 3-10 on page 17.

3.2.2 Mounting and wiring the bus configuration

Refer to the user documentation for the devices.
See “Additional documentation” on page 8.

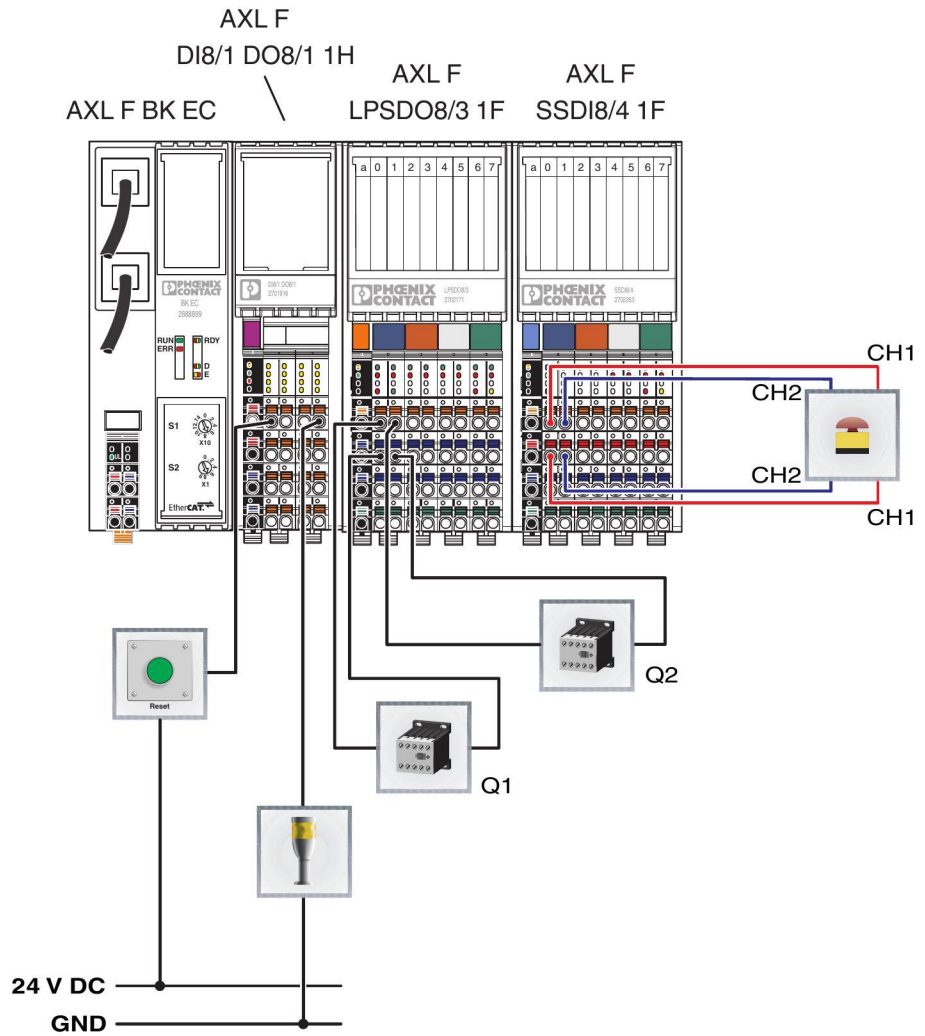


Figure 3-2 Bus configuration for example project

1. After setting the DIP switches on the modules, mount the bus configuration as illustrated.
2. Connect the power supply for the bus coupler and I/O modules in accordance with the corresponding user documentation.
3. Connect channel 1 of the emergency stop button to terminal points 00 and 10 and channel 2 to terminal points 01 and 11 of the SSDI8/4 module.
4. Connect the external reset button to the AXL F DI8/1 DO8/1 1H module.
5. Connect a signal lamp to the AXL F DI8/1 DO8/1 1H module.
6. Optional: connect contactors to terminal points 00 and 10, as well as to 01 and 11 of the LPSDO8/3 module.

3.3 Configuring the safety logic in SAFECONF

If you have any questions about SAFECONF, please refer to the online help for the software.

3.3.1 Creating a new project

- Open the SAFECONF software.
- Create a new project with the Project Wizard.
  To do this, select “File, New Project”.
- Specify the name and storage location for the project.

Do not use spaces, dashes or special characters. Note the name and storage location for the project as you will need this information later.

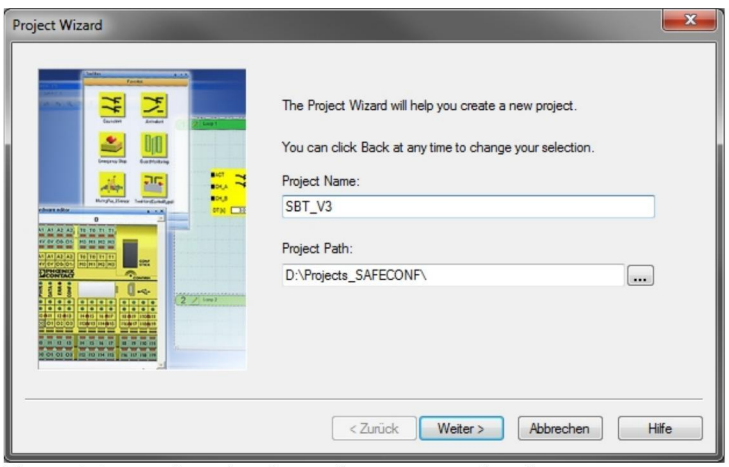


Figure 3-3 Creating the project name and path

Select master device

- Select the AXL F LPSDO8/3 1F master device.

Figure 3-4 Selecting AXL F LPSDO8/3 1F

Select file format

- Select “CODESYS V3” as the file format in which the configuration and parameter data record is to be output.

Alternatively, select **“CODESYS V 2.3”** to output the configuration and parameter data record as an **exp file**.

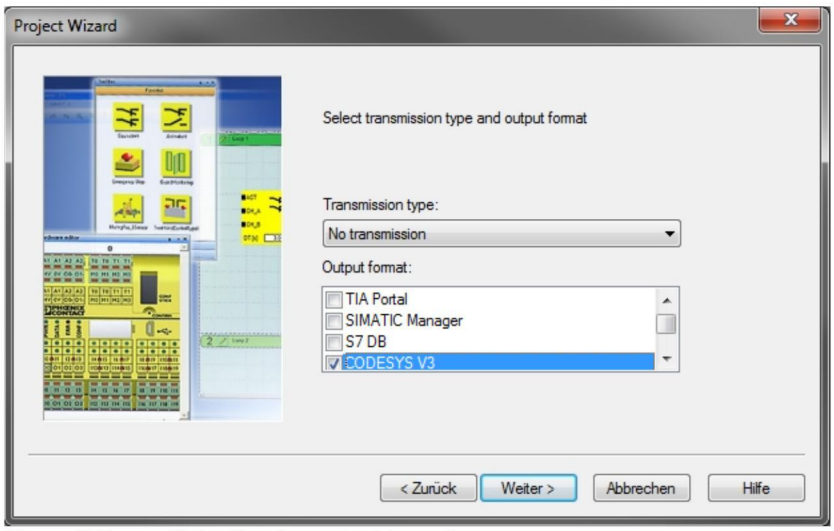


Figure 3-5 Selecting the output format

Enter project description

- Enter a description of the project.

Use a maximum of four characters for the description and version.

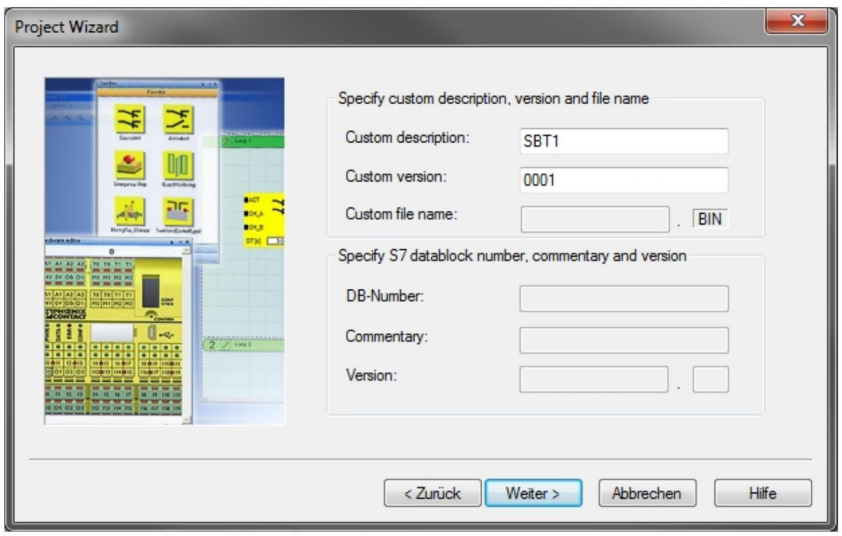


Figure 3-6 Describing the project

- Click on “Finish” to complete the project creation process.

Assign safety island number

When the project is completed, a window opens prompting you to enter the number for the safety island you are configuring.

- Enter an island number (1 in the example).

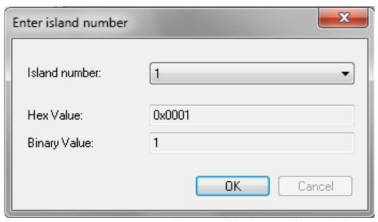


Figure 3-7 Specifying the island number

Specify password

- Specify a password of at least six characters for the project (123456 in the example).

Figure 3-8 Specifying a password

3.3.2 Configuring and parameterizing the hardware structure

- Configure the hardware structure.
  To do this, use drag and drop to move the AXL F SSDI8/4 1F module from the “Hardware” toolbox to the hardware editor.

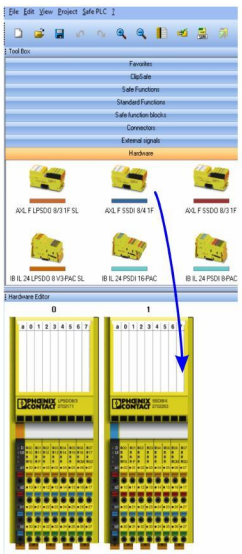


Figure 3-9 Hardware configuration

The corresponding satellite number is displayed via the module.
You can display the complete DIP switch setting by right-clicking on the module and selecting “Display address switch”.

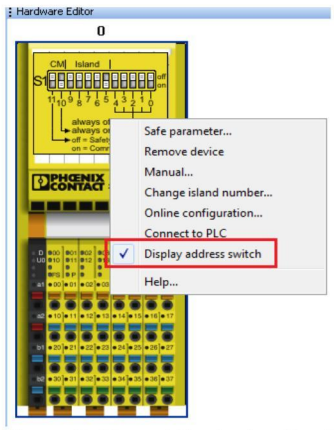


Figure 3-10 Displaying the DIP switch

Parameterize I/O channels

There are two options for parameterizing the input and output channels of the modules:

1 In the hardware editor, double-click on the module. This opens the window for parameterizing the entire module.

2 In the hardware editor, double-click on a terminal point. This opens the window for parameterizing the selected terminal point.

- Parameterize the output channels of the LPSDO8/3 module as illustrated (double-click on the module to parameterize).

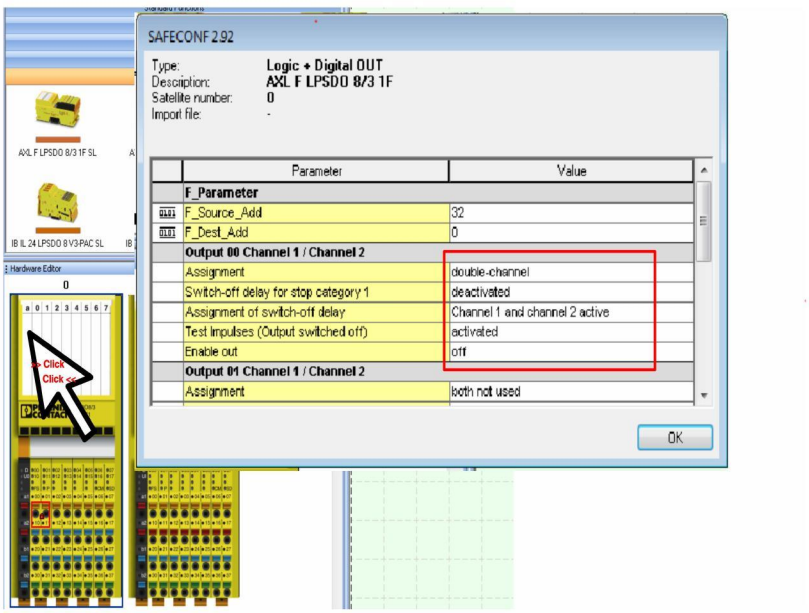


Figure 3-11 Parameterization of the LPSDO8/3

- Parameterize the input channels of the SSDI8/4 module (double-click on the module to parameterize).

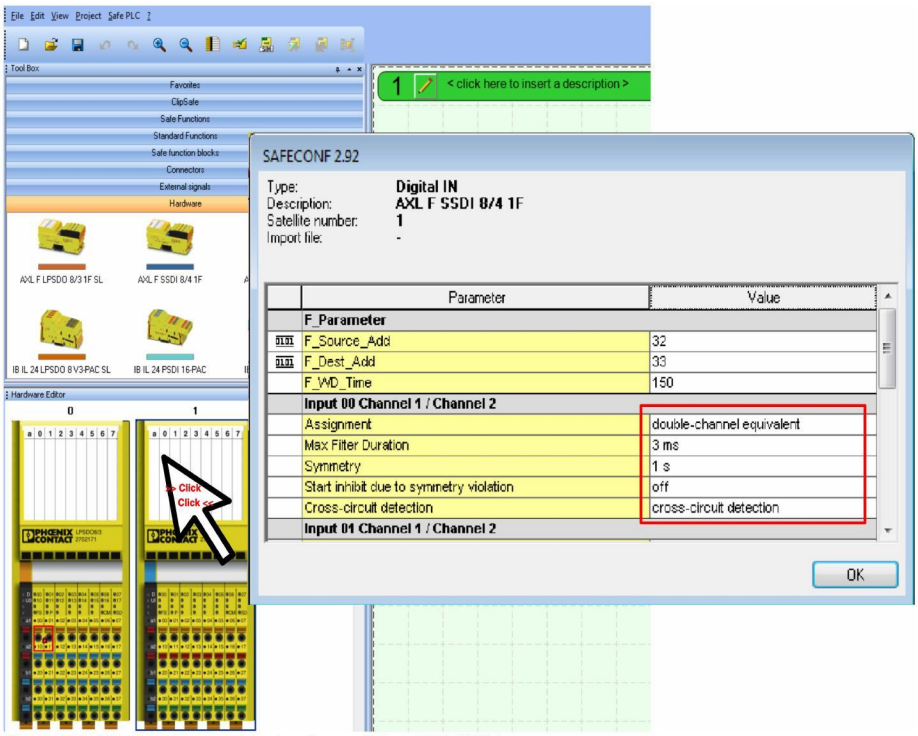


Figure 3-12 Parameterization of the SSDI8/4

Inputs or outputs parameterized for two-channel operation are indicated by a padlock symbol on the module in the hardware editor.

However, the input and output signals are only displayed in single-channel form in the connection editor, even if they are parameterized for two-channel operation. See Figure 3-14 on page 19.

3.3.3 Configuring the safety function

Comment function

You can add comments to both function blocks and signals in SAFECONF. Please refer to the online help for the software.

Insert function blocks

- Configure the safety function. To do this, use drag and drop to move the blocks and signals from the corresponding toolboxes to the connection editor.

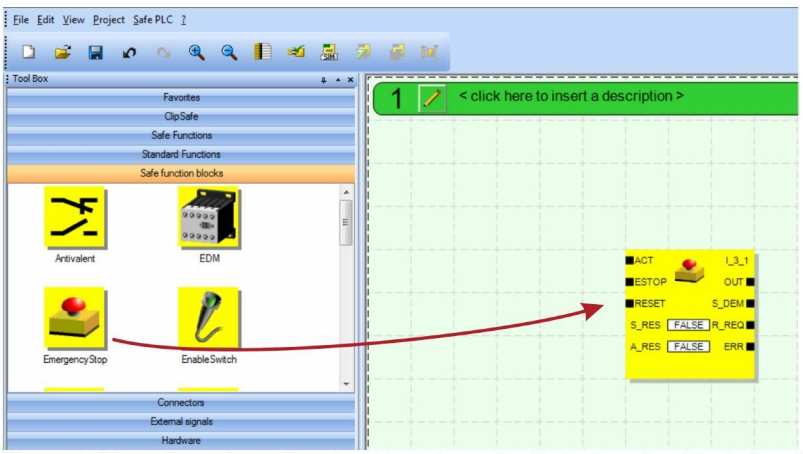


Figure 3-13 Inserting a function block from the “Safe function blocks” toolbox

Insert safe inputs and outputs

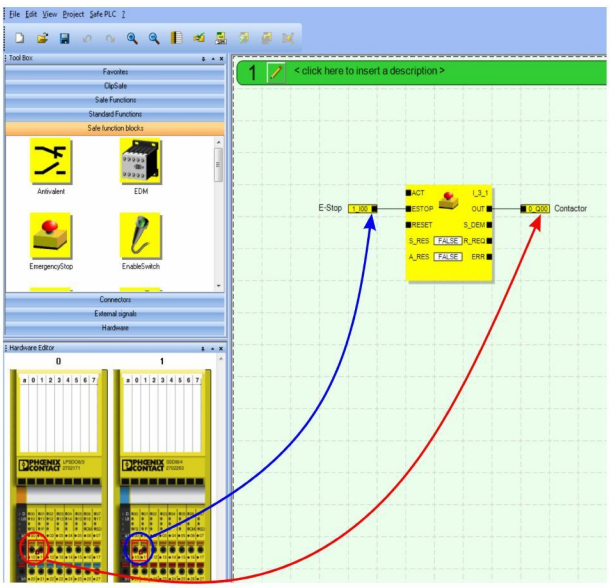


Figure 3-14 Inserting safe inputs and outputs from the hardware editor

When you use drag and drop to place the safety module terminal point directly onto a function block input or output, the connecting line is created automatically.

Insert external signals

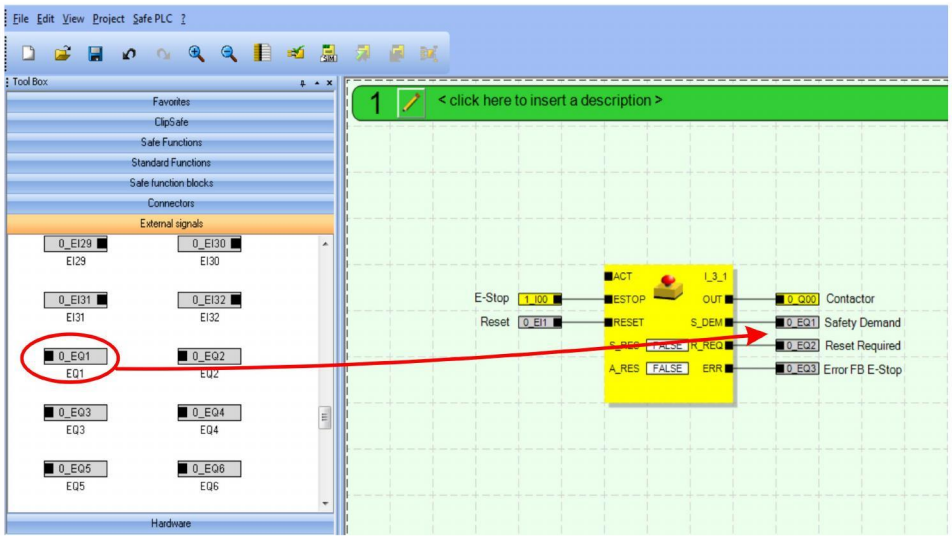
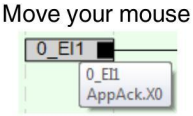


Figure 3-15 Inserting external signals from the “External signals” toolbox

Move your mouse over an external signal to display the corresponding tool tip.



Insert safe functions

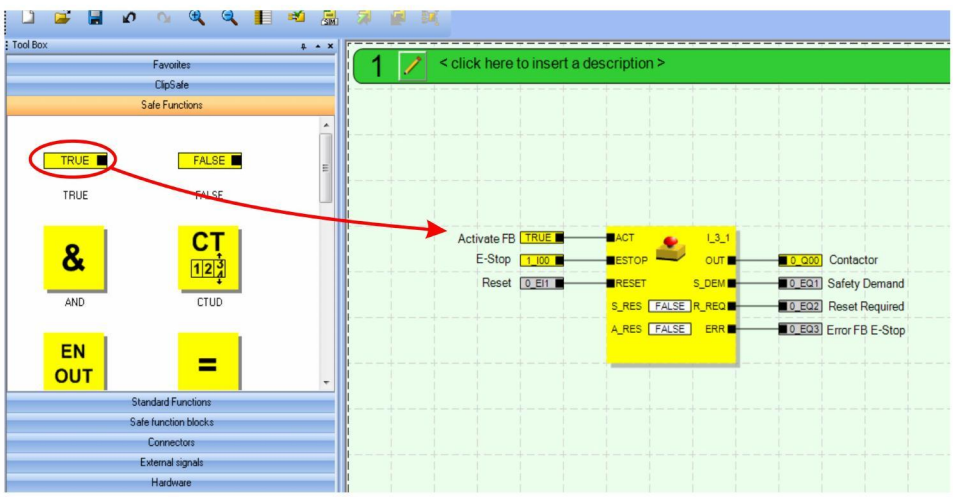


Figure 3-16 Inserting a safe function from the “Safe Functions” toolbox

3.3.4 Exporting the configuration and parameter data record

Check project

- Check the project.
  To do this, select the “Project, Check Project” command or confirm by clicking on the corresponding button.

A message window opens displaying the progress of the check. Once the check is complete, the amount of program memory used by the program is displayed.

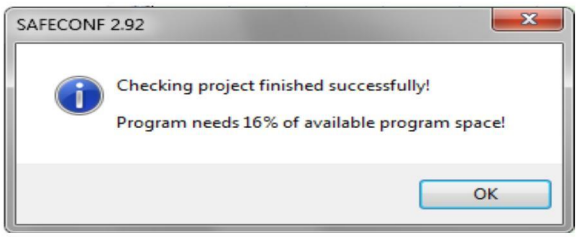


Figure 3-17 Program memory used

xml file for CODESYS V 3.5

If the check is completed without errors, the configuration and parameter data record is created as an xml file. This is saved in the path that you have entered for the project (see Figure 3-3 on page 14) in the “FileOutput” folder.

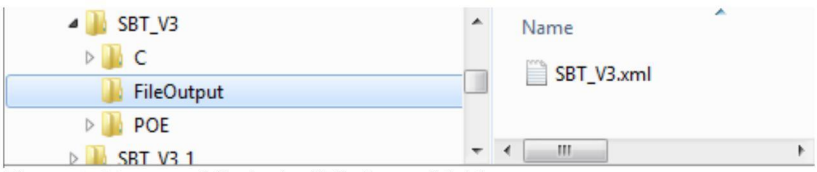


Figure 3-18 xml file in the “FileOutput” folder

exp file for CODESYS V 2.3 (alternative)

If you have selected **“CODESYS V 2.3”** as the output format for the configuration and parameter data record (see Figure 3-5 on page 15), the **exp file** is saved in the “FileOutput” folder. For additional information on this process, please refer to “Importing the configuration and parameter data record as a function block” on page 28.

3.4 Configuring a CODESYS project

If you have any questions about CODESYS, please refer to the online help for the software.

3.4.1 Creating a project and importing the device description file

Import ESI file

- Create a new project in CODESYS.
- Import the device description file for the bus coupler (AXL_F_BK_EC.xml in the example) into the device repository of CODESYS.

3.4.2 Configuring the hardware structure

Insert modules

- Select the following modules in the device repository and use drag and drop to insert them under the EtherCAT® master:
  - AXL F BK EC (Order No. 2688899)
  - AXL F DI8/1 DO8/1 1H (Order No. 2701916)
  - AXL F LPSDO8/3 1F (Order No. 2702171)
  - AXL F SSDI8/4 1F (Order No. 2702263)

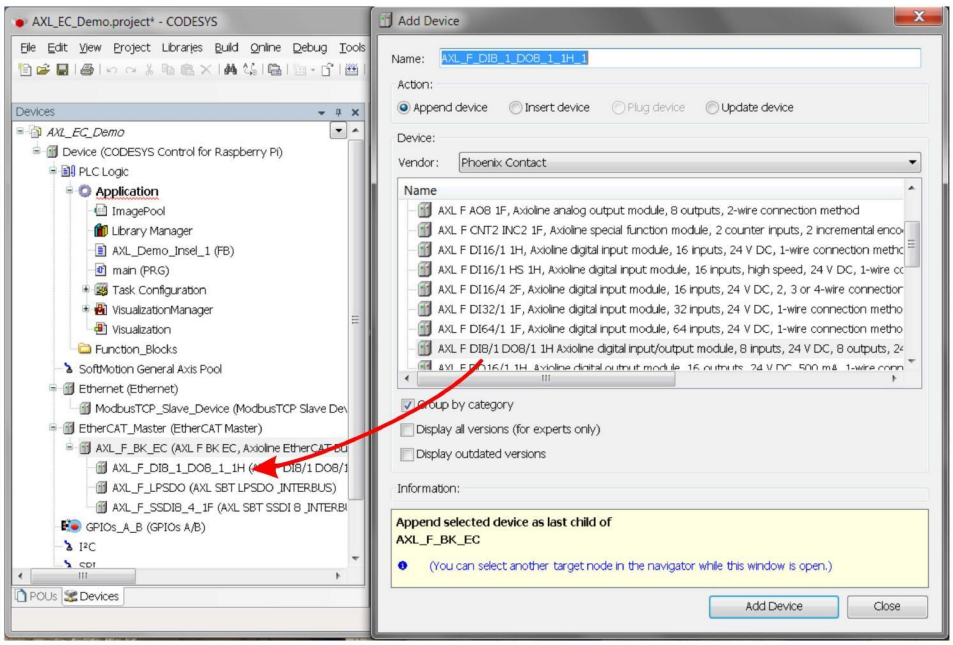


Figure 3-19 Inserting SBT modules in the device overview

3.4.3 Integrating function blocks for SafetyBridge Technology V3

Integrate library from integration package

- Select Library Manager view.

For CODESYS V 3.5:

- Install the “SBT_V3_Lib.compiled-library” library in your library repository.

For CODESYS V 2.3:

- Install the “SBT_V3_LIB_V1_10” library in your library repository.
- Add the library to your project.

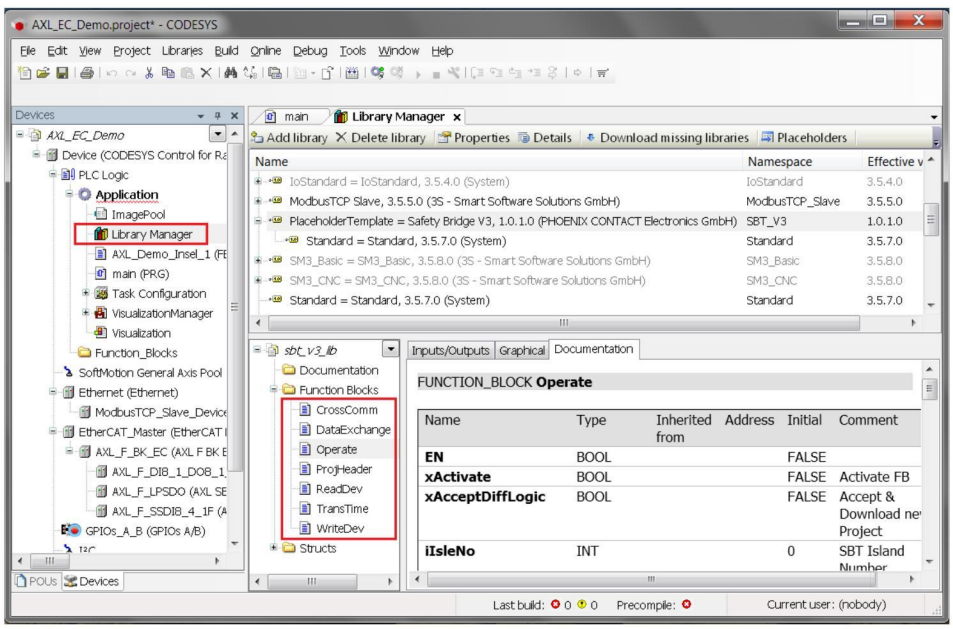


Figure 3-20 Integrating the library

You can now see the open “SafetyBridge V3, 1.0.1.0” library with the corresponding function blocks for SafetyBridge Technology V3:

– SBT_V3.Operate
– SBT_V3.ReadDev
– SBT_V3.WriteDev
– SBT_V3.ProjHeader
– SBT_V3.TransTime
– SBT_V3.CrossComm
– SBT_V3.DataExchange

A description of the function blocks and data types is provided in Appendix B “Description of the function blocks for SafetyBridge Technology V3”.

Please also use the online help in CODESYS.

3.4.4 Creating the SBT program in CODESYS

Proceed as follows to extend your standard application program using the SafetyBridge function blocks from the “sbt_v3_lib” library:

To insert and connect the required SafetyBridge function blocks in your project, proceed as follows:

Function block “SBT_V3.Operate”

- Switch to the “main (PRG)” program.
- Insert the “SBT_V3.Operate” function block in the program.
- Connect the function block as shown in Figure 3-21.

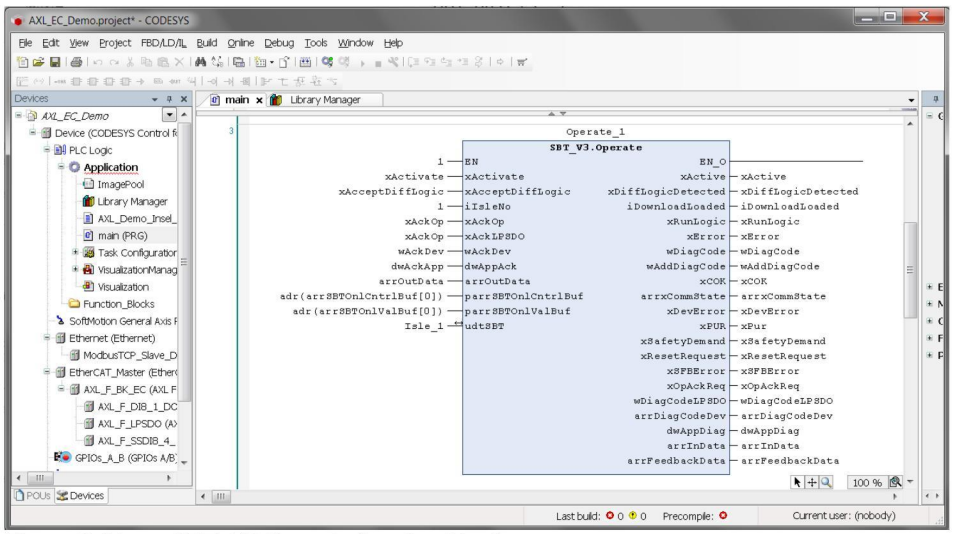


Figure 3-21 SBT_V3.Operate function block

Function block “SBT_V3.ReadDev”

An instance of the “SBT_V3.ReadDev” function block is required for each SafetyBridge module.

Call the instances of the “SBT_V3.ReadDev” function block **before** the “SBT_V3.Operate” function block.

- Insert an instance of the “SBT_V3.ReadDev” function block in the program for the LPSDO8/3 module.
- Insert an instance of the “SBT_V3.ReadDev” function block in the program for the SSDI8/4 module.
- Connect the function blocks as shown in Figure 3-22.

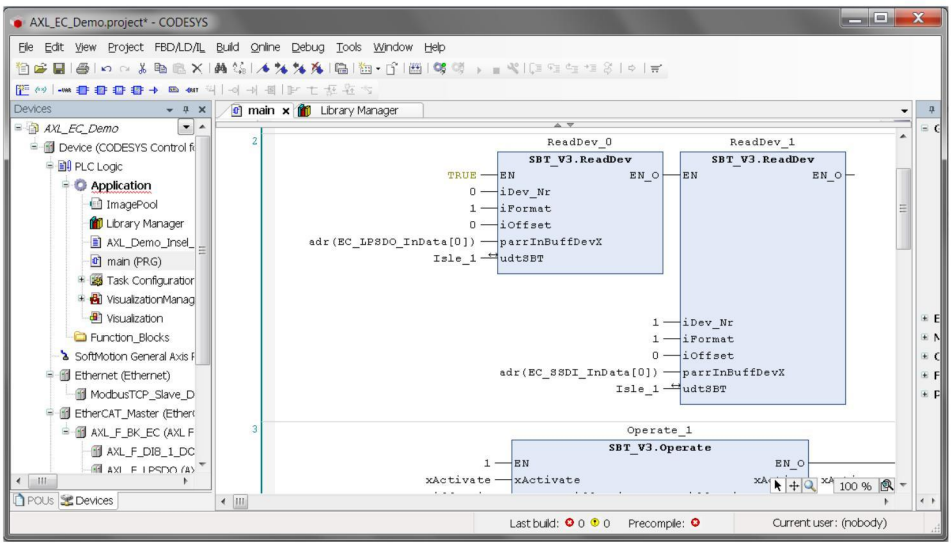


Figure 3-22 SBT_V3.ReadDev function blocks

Function block “SBT_V3.WriteDev”

An instance of the “SBT_V3.WriteDev” function block is required for each SafetyBridge module.

Call the instances of the “SBT_V3.WriteDev” function block **after** the “SBT_V3.Operate” function block.

- Insert an instance of the “SBT_V3.WriteDev” function block in the program for the LPSDO8/3 module.
- Insert an instance of the “SBT_V3.WriteDev” function block in the program for the SSDI8/4 module.
- Connect the function blocks as shown in Figure 3-23.

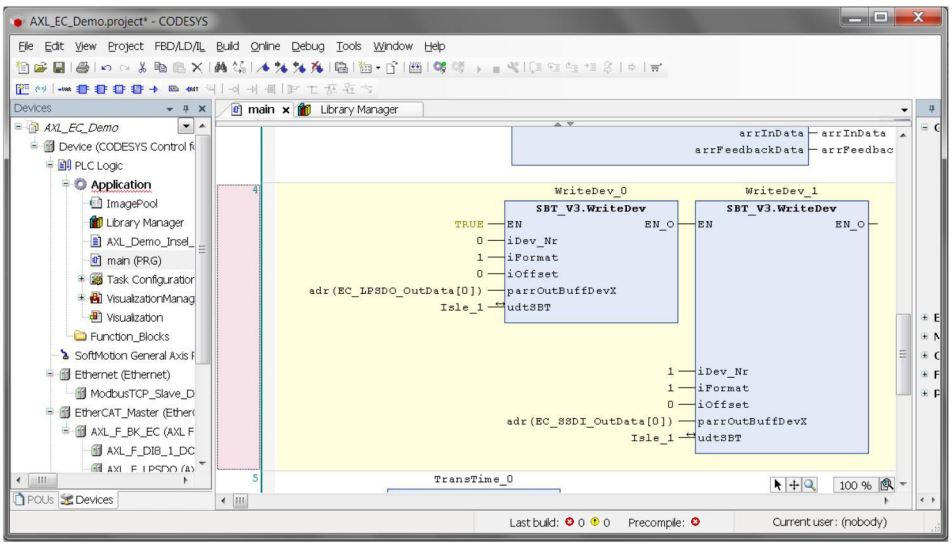


Figure 3-23 SBT_V3.WriteDev function blocks

Function block “SBT_V3.ProjHeader”

- Insert the “SBT_V3.ProjHeader” function block in the program.
- Connect the function blocks as shown in Figure 3-24.
- Link the “udtSBT” structure variable to the same variable as for the “SBT_V3.Operate” function block.

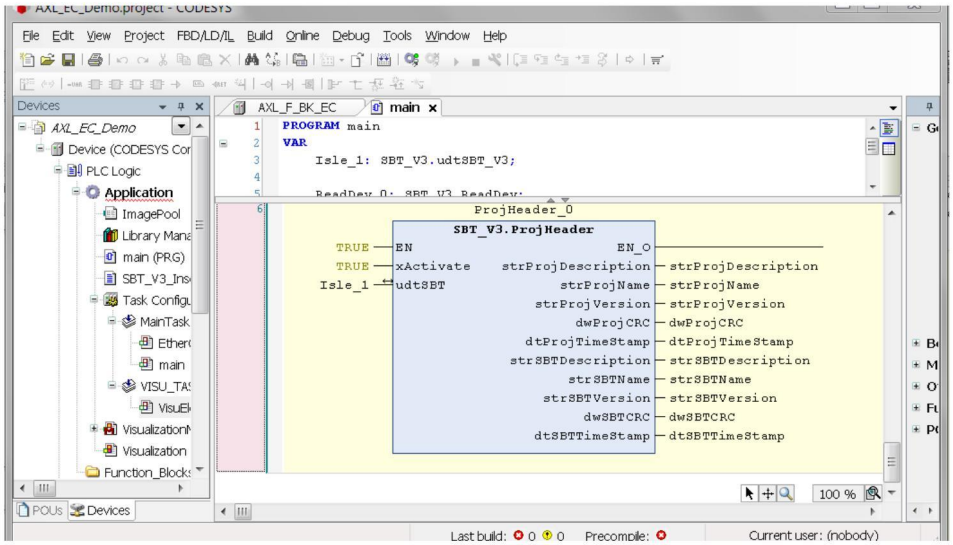


Figure 3-24 SBT_V3.ProjHeader function blocks

Function block “SBT_V3.TransTime”

- Insert the “SBT_V3.TransTime” function block in the program.
- Connect the function blocks as shown in Figure 3-25.
- Link the “udtSBT” structure variable to the same variable as for the “SBT_V3.Operate” function block.

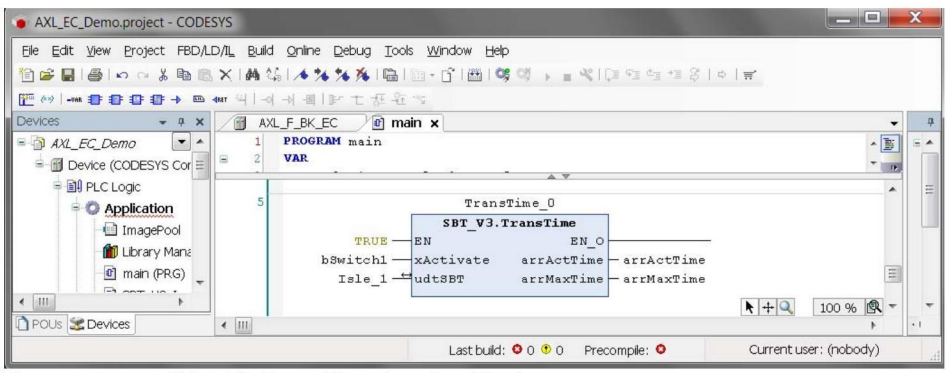


Figure 3-25 SBT_V3.TransTime function blocks

3.4.5 Importing the configuration and parameter data record as a function block

Procedure for CODESYS V 3.5

CODESYS V 3.5

If using **CODESYS V 3.5**, proceed as follows to import the safety logic created in SAFECONF as a function block:

Import xml file

In “Exporting the configuration and parameter data record” on page 21, the **xml file** was created and saved in the “FileOutput” folder under the previously specified project path (see Figure 3-18 on page 21).

- Open the CODESYS project.
- Select “Project, Import PLCopenXML...”.

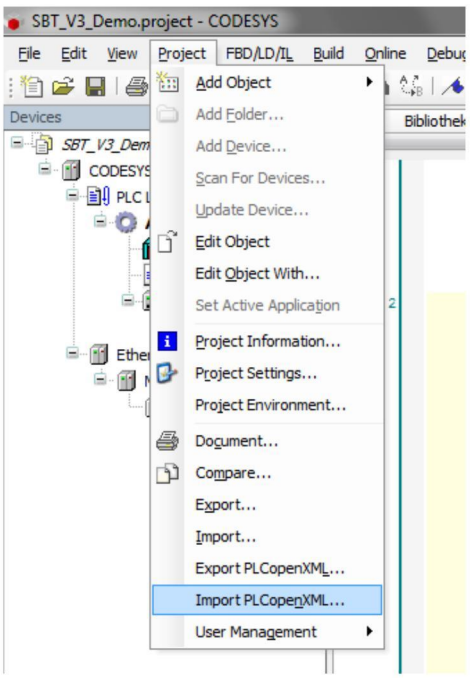


Figure 3-26 Importing the xml file I

- Select the xml file.
- Confirm your selection with OK.

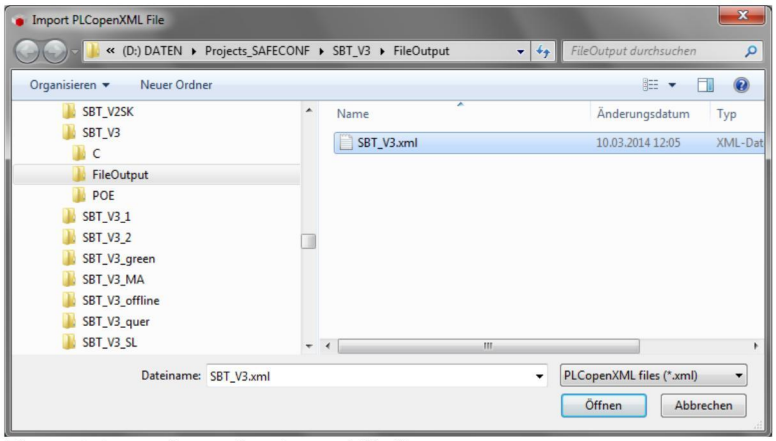


Figure 3-27 Importing the xml file II

- Select the block to be imported.
- Confirm your selection with OK.

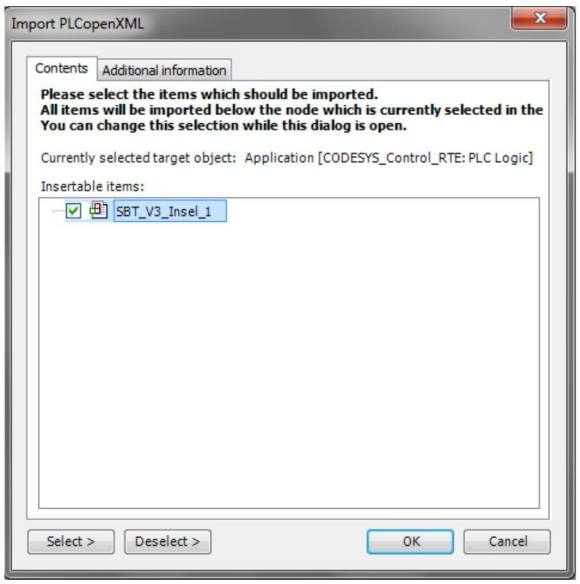


Figure 3-28 Selecting the function block

The function block has been successfully imported if it appears as shown in Figure 3-29.

- Insert the imported function block in the program.
- Connect the function block to other function blocks via the udt structure.

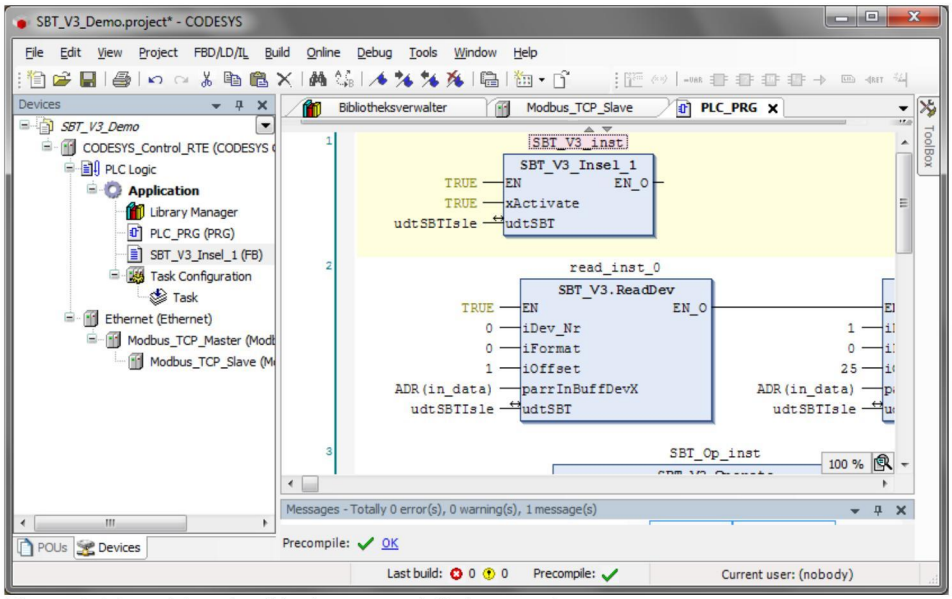


Figure 3-29 Function block successfully imported

Procedure for CODESYS V 2.3

CODESYS V 2.3

If using **CODESYS V 2.3**, proceed as follows to import and manage the safety logic created in SAFECONF as a function block:

Import exp file

In “Exporting the configuration and parameter data record” on page 21, the **exp file** was created and saved in the “FileOutput” folder under the previously specified project path (see Figure 3-18 on page 21).

- Open the CODESYS project.
- Select “Project, Import...”.

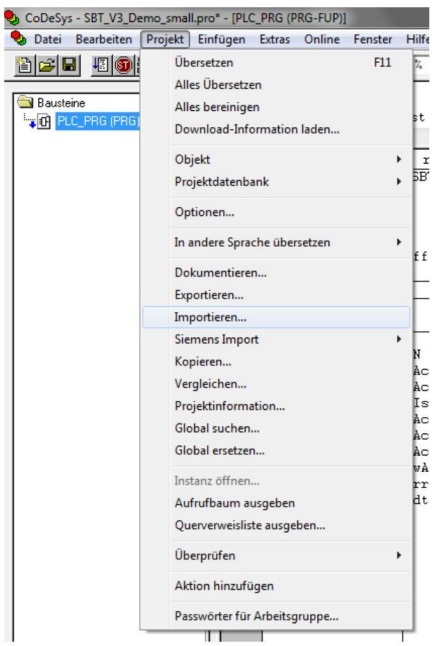


Figure 3-30 Importing the exp file I

- Select the exp file.
- Confirm your selection with OK.

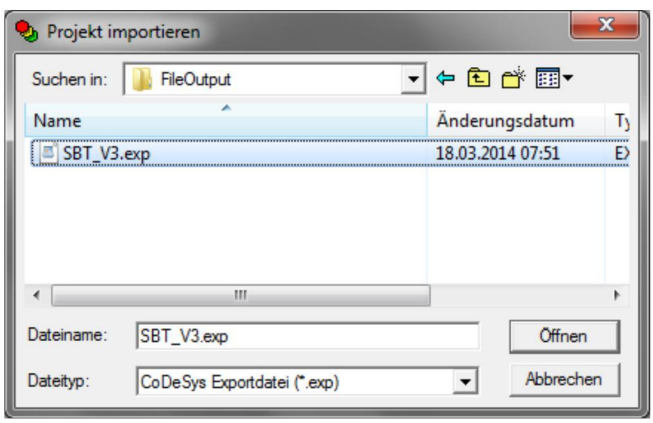


Figure 3-31 Importing the exp file II

The function block has been successfully imported if it appears as shown in Figure 3-32.

- Insert the imported function block in the program.
- Connect the function block to other function blocks via the udt structure.

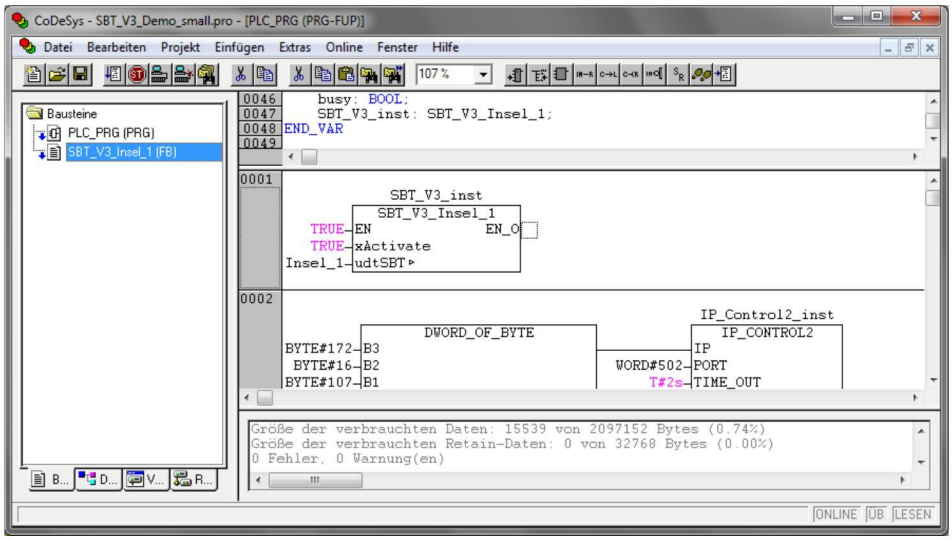


Figure 3-32 Function block successfully imported

3.4.6 Process data assignment of the devices

In this example, the SafetyBridge modules are connected to the EtherCAT® master and the controller via the EtherCAT® bus coupler.

- Open the device view for the AXL F BK EC bus coupler.
- Select “EtherCAT I/O Mapping”.

You can see the mapped addresses of the SafetyBridge modules (“%I ...” and “%Q ...”).

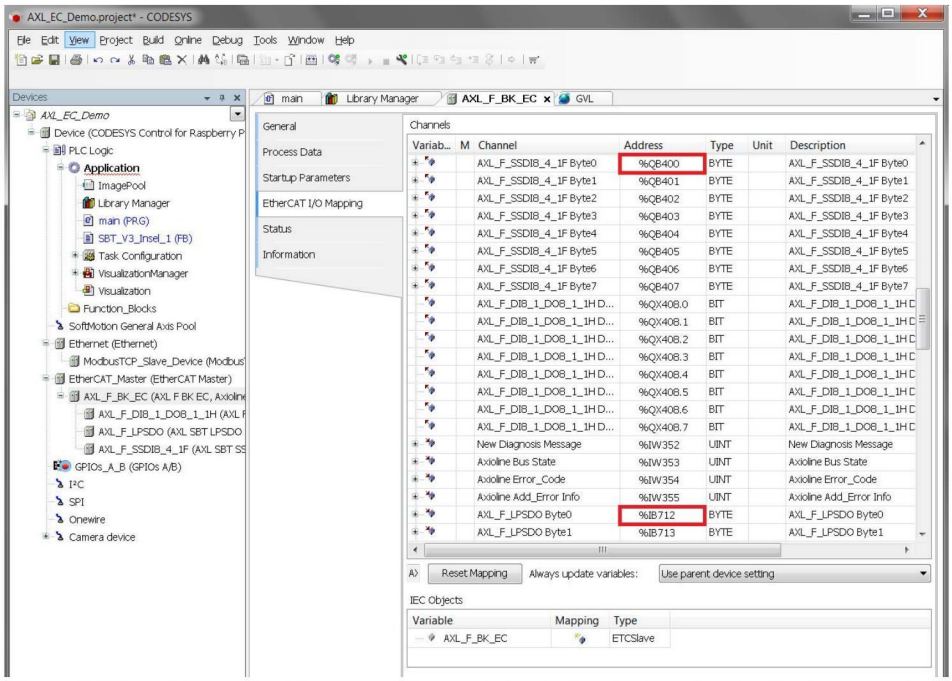


Figure 3-33 Mapped addresses of the modules in the device view for the bus coupler

- Switch to the list of global variables under “POUs, GVL”.
- Assign the mapped addresses of the SafetyBridge modules (“%I ...” and “%Q ...”) to the allocated I/O buffers in your control program.

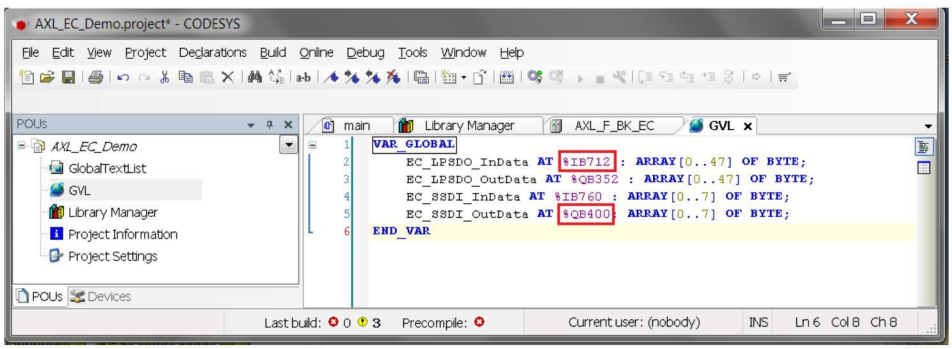


Figure 3-34 Assigning addresses

3.4.7 Complete example project in CODESYS

Once all the necessary function blocks have been imported and connected, the CODESYS project has the following structure:

– Configuration and parameter data record as the “SBT_V3_Insel_1” function block
– 2 instances of the “SBT_V3.ReadDev” function block
– “SBT_V3.Operate” function block
– 2 instances of the “SBT_V3.WriteDev” function block

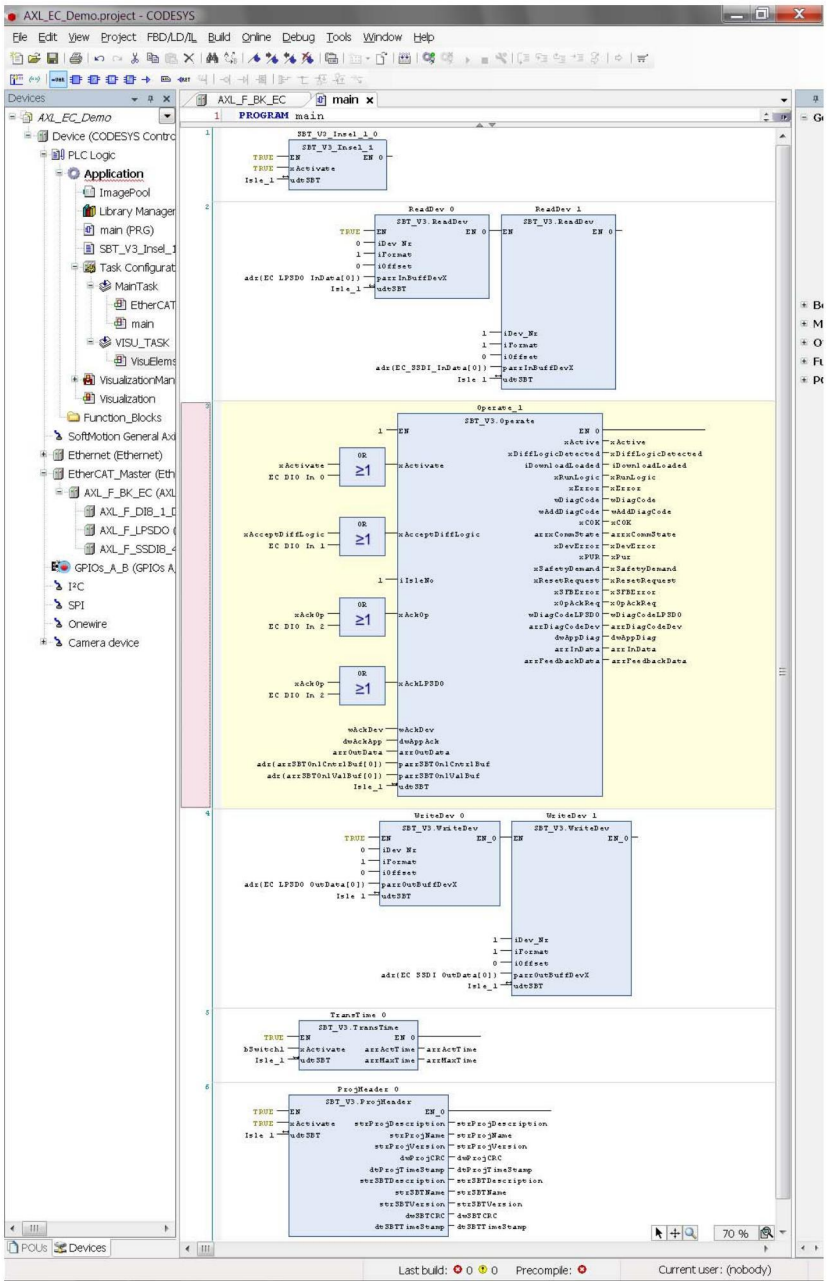


Figure 3-35 Complete example project in CODESYS

3.4.8 Compiling the project and downloading it to the controller

- Compile the created project and download it to the controller.

The controller switches to the “Run” state.
The LPSDO8/3 module indicates that it has not been parameterized yet by flashing the FS LED.

3.5 Startup

A flowchart for starting up and testing the application can be found in Appendix A on page 42.

1. Switch to CODESYS.
2. Open the “main (PRG)” program.
3. Activate the online values in CODESYS.
4. Set the “xActivate” input parameter to “1”.
5. Check the “xDiffLogicDetected” output parameter.

If the value is set to “1”, a new SAFECONF project has been detected.

6. Set the “xAcceptDiffLogic” input parameter to “1”.

You can check the download progress at the “iDownloadLoaded” output parameter.
Download time: approximately 40 seconds (depending on the project size, CPU, and bus speed).

7. Check whether the “xRunLogic” output parameter outputs “TRUE” and the “wDiagCode” output parameter outputs the value 16#8000.

If a different code is output by the “wDiagCode” output parameter, see “Function block diagnostics” on page 49.

Once successfully downloaded, the safe application is ready.
The diagnostics LEDs of the modules now have the following status:

LEDs **off**: FS, CM, SD
LEDs **on**: P, U_O, D

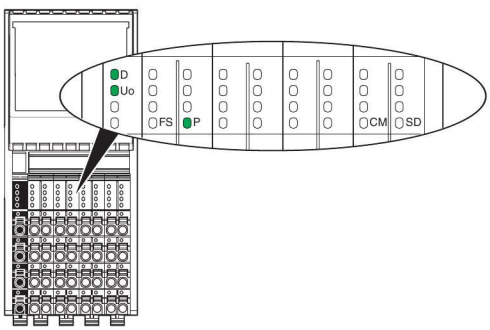


Figure 3-36 Diagnostics LEDs

Perform an overall safety validation after you start up your system.

3.6 Online configuration and connection establishment

In order to monitor the online status of the safety logic of the LPSDO8/3 module during startup or maintenance, two configuration steps are required:

1. Configuration of a Modbus TCP slave in the CODESYS-based controller.
2. Configuration of a Modbus TCP connection in SAFECONF.

3.6.1 Configuration of a Modbus TCP slave in CODESYS

- Integrate a Modbus TCP slave in your CODESYS-based controller using the parameters shown in Figure 3-37.

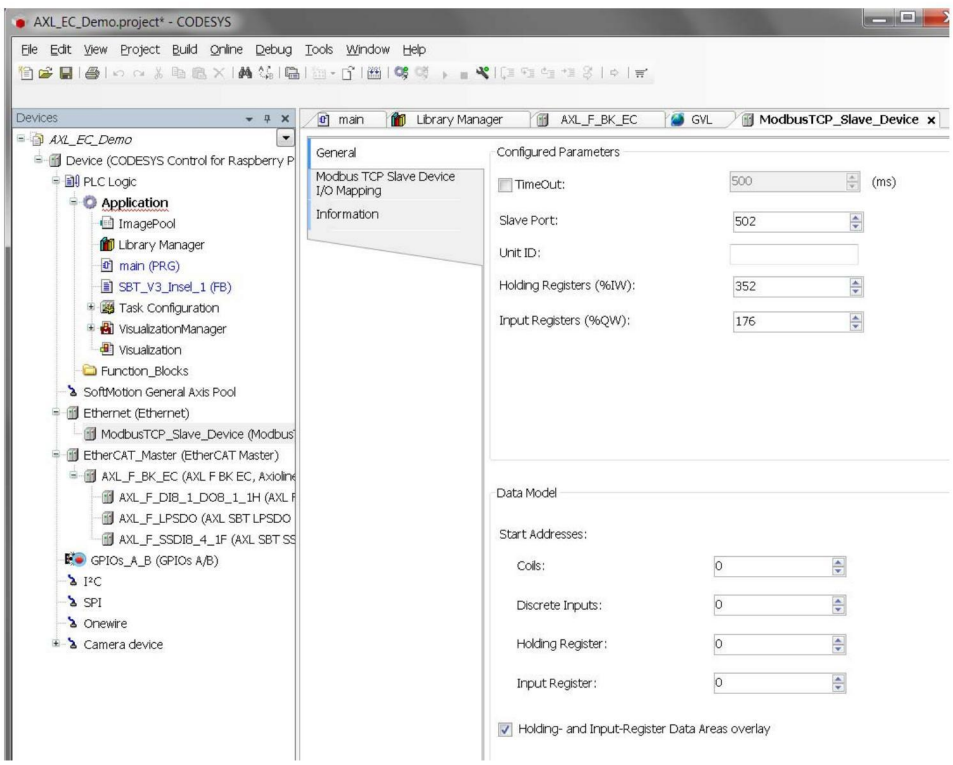


Figure 3-37 Integrating Modbus TCP

- Select “Modbus TCP Slave Device I/O Mapping”.
- Connect the “arrSBTOnlCntrBuf” and “arrSBTOnlValBuf” arrays to your standard application.

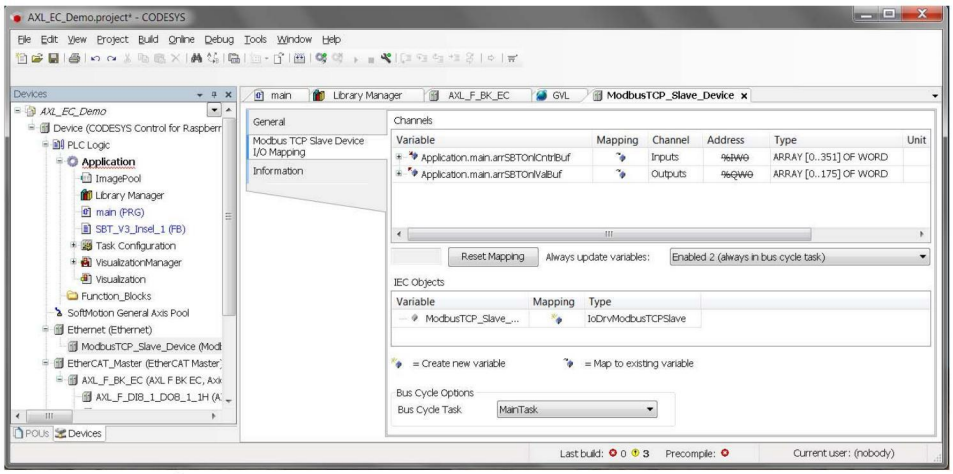


Figure 3-38 Connecting variables

The “arrSBTOnlCntrlBuf” and “arrSBTOnlValBuf” I/O parameters enable communication between SAFECONF and the “SBT_V3.Operate” function block.

3.6.2 Configuration of a Modbus TCP connection in SAFECONF

- Open SAFECONF.
- In the hardware editor, right-click on the LPSDO8/3 module and select “Online configuration...”.

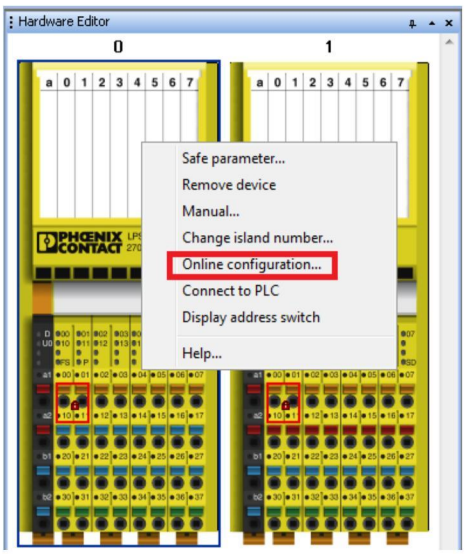


Figure 3-39 Selecting online configuration

- Select the “Generic Modbus/TCP Device” interface and click on “Next”.

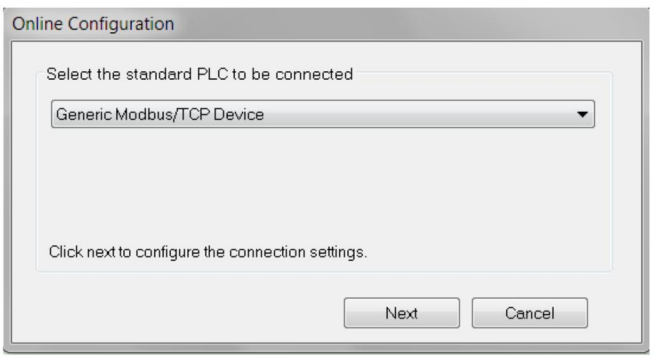


Figure 3-40 Selecting the interface

- Enter the IP address of the CODESYS-based controller.
- Set the “Expert Settings” as shown in Figure 3-41.
- Once this test has been completed successfully, click on “Finish”.

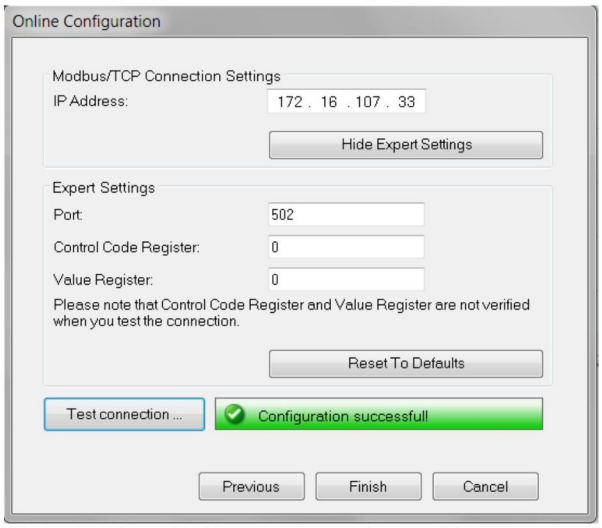


Figure 3-41 Making the connection settings

- After a few seconds, the following message appears in the SAFECONF status bar.

Figure 3-42 SAFECONF status message

3.6.3 Displaying online values in SAFECONF

Communication between SAFECONF and the “SBT_V3.Operate” function block is achieved by the “arrSBTOnlCntrlBuf” and “arrSBTOnlValBuf” I/O parameters.

To view the online values in SAFECONF, proceed as follows:

- Open SAFECONF.
- In the hardware editor, right-click on the LPSDO8/3 module.
- If another online connection is active, disconnect the connection by selecting “Disconnect from PLC”.

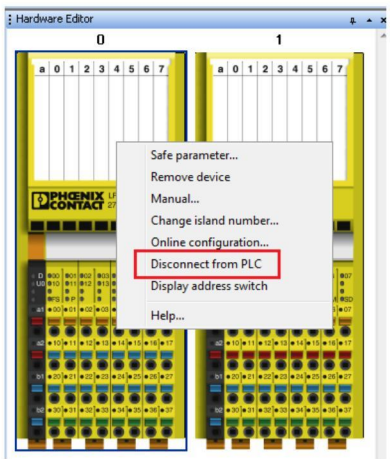


Figure 3-43 Disconnecting an online connection

- In the hardware editor, right-click on the LPSDO8/3 module.
- Establish an online connection by selecting “Connect to PLC”.

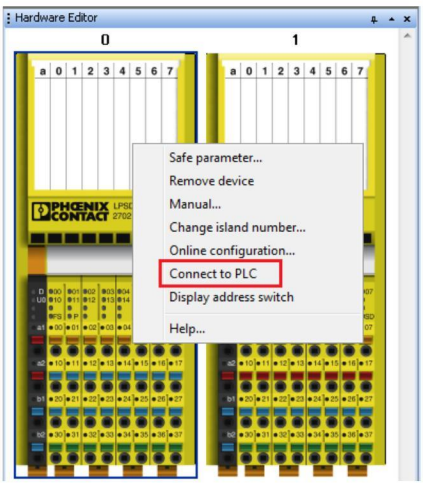


Figure 3-44 Establishing an online connection

- In the SAFECONF tool bar, click on the “Show Online Values” button.

The online values are now displayed in the SAFECONF project.

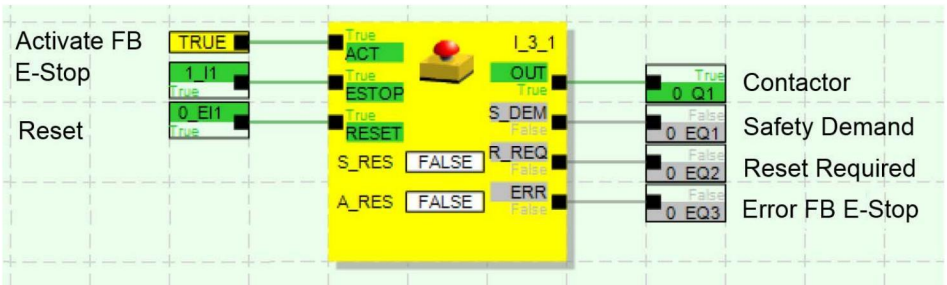


Figure 3-45 Online values in the SAFECONF project

A Flowchart for starting up and testing the application

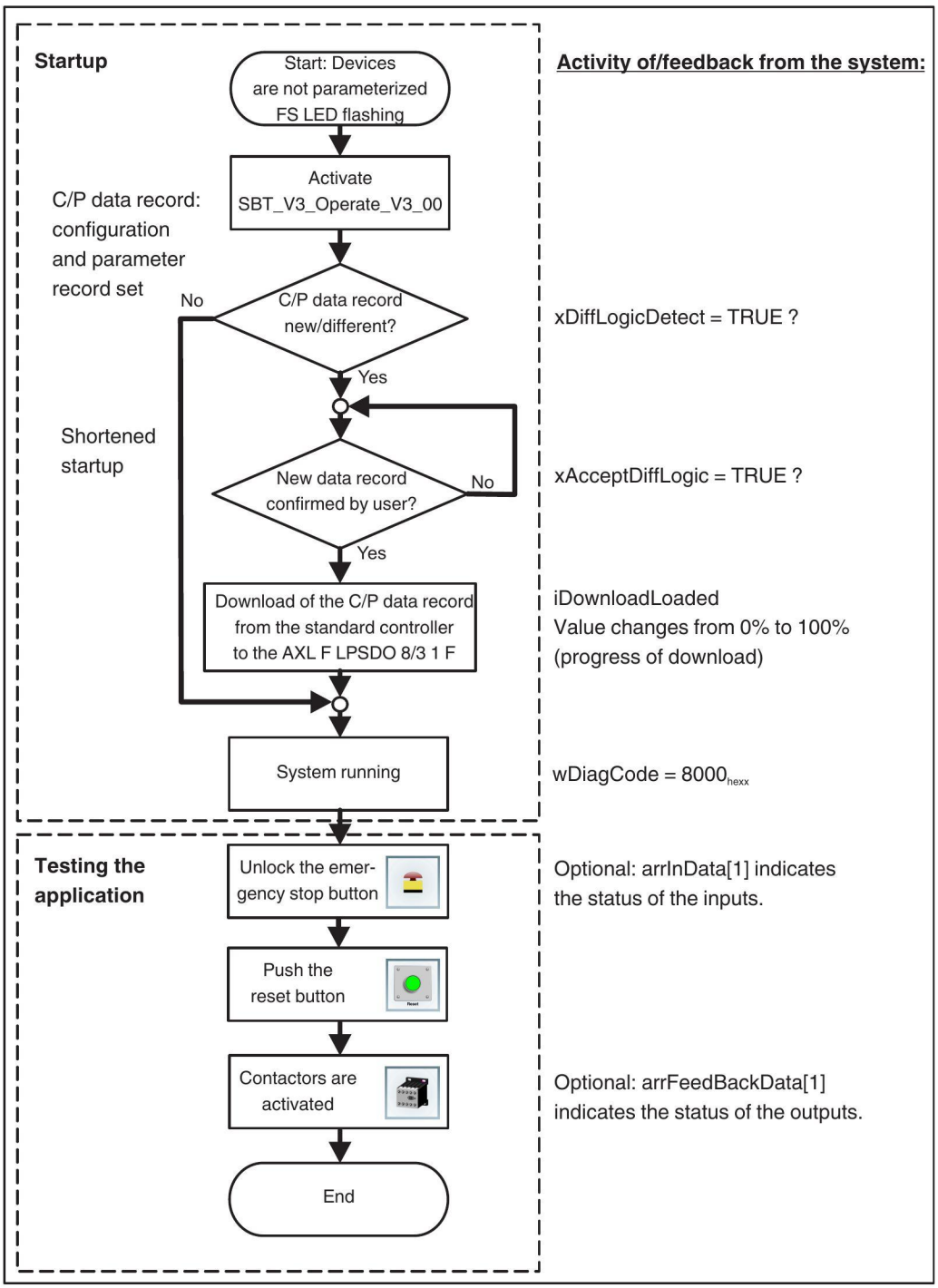


Figure A-1 Flowchart for starting up and testing the application

B Description of the function blocks for SafetyBridge Technology V3

Validity

The function blocks described are valid for the following controller/network combinations.

Table B-1 Validity of the function blocks

Controller	CODESYS Control SoftPLC and all other CODESYS-based controllers					
Bus system	Modbus TCP/UDP	PROFIBUS	PROFINET	EtherNet/IP™	Sercos III	EtherCAT®
Software	CODESYS as of V 2.3 or V 3.5					

Function block overview

Table B-2 Function block overview

Function block	Description	Programming language
SBT_V3.Operate	Function block used to operate an SBT island with a maximum of 16 satellites: – Download the SAFECONF project – Read the input, diagnostic, and feedback signals of all connected SBT modules – Makes the “enableOut” principle available See B 1 on page 45	ST
SBT_V3.ReadDev	Reads input data from the I/O (process image) to the SBT_V3.Operate function block via udtSBT See B 2 on page 50	ST
SBT_V3.WriteDev	Writes output data from the SBT_V3.Operate function block to the I/O (process image) via udtSBT See B 3 on page 51	ST
SBT_V3.ProjHeader	Displays the project information for the connected LPSDO module and the imported SAFECONF project See B 4 on page 52	ST

Function block	Description	Programming language
SBT_V3.TransTime	Measures the runtime of the current and maximum transmission times between the LPSDO module and each of the connected satellites See B 5 on page 54	ST
SBT_V3.CrossComm	Cross communication between SBT islands See B 6 on page 56	ST
SBT_V3.DataExchange	Data exchange if two SBT islands are integrated in different controllers See B 7 on page 57	ST

Data types

Table B-3 Data types

Designation	Description
udtSBT_V3	– Internal user-defined data type – Structures of this data type connect the function blocks of an island – No access required to these structures – Access to all SBT data via the input and output parameters of the function blocks

B 1 SBT_V3.Operate function block

The function block performs the following functions:

- Download of the configuration and parameter data record from the standard controller to the AXL F LPSDO8/3 1F
- Cyclical routing of the SafetyBridge data flow
- Display of diagnostic information for all the SafetyBridge modules of an island
- Acknowledgment of error messages
- Reading the status information for all inputs and outputs

Use the function block once per island.

The figure shows the division of the function block into its individual function areas. The parameters used are described in the tables below.

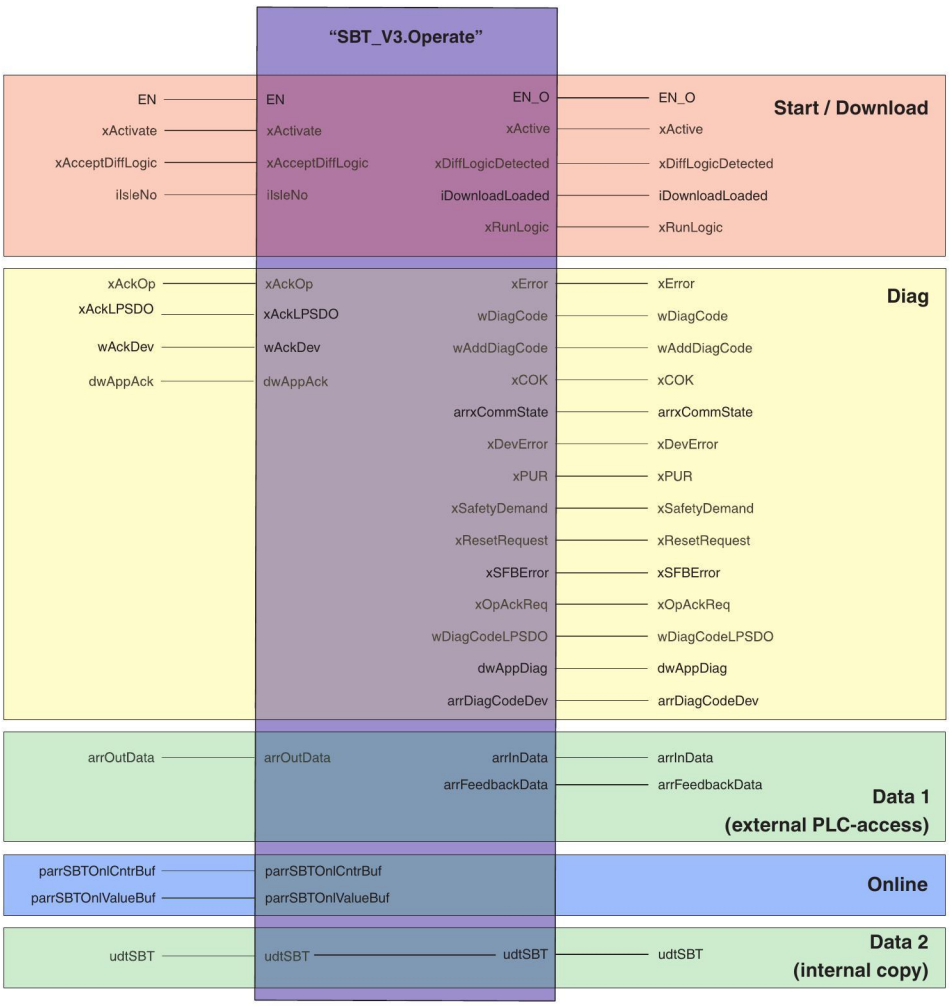


Figure B-1 SBT_V3.Operate function block

B 1.1 Input parameters

Table B-4 Input parameters for the SBT_V3.Operate function block

Name	Type	Description
EN	BOOL	-
xActivate	BOOL	Activation/deactivation of the function block
xAcceptDiffLogic	BOOL	User confirmation signal that the safety logic differs for the controller and the LPSDO module True: user has accepted and confirmed the difference (the download then starts)
iIsleNo	INT	SBT island number Valid range: 1 ... 31 (must be the same as the DIP switch position on the LPSDO module)
xAckOp	BOOL	User confirmation of an “xOpAckReq” signal
xAckLPSDO	BOOL	User confirmation of an LPSDO error
wAckDev	WORD	User confirmation of module errors: wAckDev.0: for module 1 wAckDev.15: for module 16
dwAppAck	DWORD	32 freely configurable acknowledgment signals (application data) from the standard controller to the LPSDO module
arrOutData	Array [0...16] of WORD	Array of “enableOutputs” from all output modules (satellites) of the island; the “enableOut” principle is optional and must be parameterized in SAFECONF The index represents the module number (0 stands for the LPSDO module): arrOutData [0] - enableOut data LPSDO arrOutData [1] - enableOut data module 1 ... arrOutData [16] - enableOut data module 16
parrSBTOnlCtrlBuf	POINTER to array [0...151] of WORD	Pointer to ONLINE-Control array
parrSBTOnlValueBuf	POINTER to array [0...175] of WORD	Pointer to ONLINE-Value array

B 1.2 Output parameters

Table B-5 Output parameters for the SBT_V3.Operate function block

Name	Type	Description
EN_O	BOOL	-
xActive	BOOL	True: FB initialized successfully False: FB not initialized
xDiffLogicDetected	BOOL	True: detection of different safety logic for the controller and the LPSDO module
iDownloadLoaded	INT	Percentage of SBT logic that has been downloaded
xRunLogic	BOOL	Safety logic (SAFECONF logic) running on the LPSDO module
xError	BOOL	Function block error
wDiagCode	INT	Diagnostic code for the function block (status or error)
wAddDiagCode	INT	Additional diagnostic code for the function block (status or error)
xCOK	BOOL	True: communication status of the SBT island is OK False: loss of communication for one or more modules
arrxCommState	Array [0...16] of BOOL	Communication status of each module; each bit represents the status of a module True: communication established False: communication interrupted The index represents the module number (0 is not used): arrxCommState [1] - communication module 1 ... arrxCommState [16] - communication module 16
xDevError	BOOL	Indicates an error in one or more SBT modules
xPUR	BOOL	Error state that cannot be acknowledged; restart is required
xSafetyDemand	BOOL	True: one or more safety demands in the SAFECONF project (LPSDO) False: no safety demand
xResetRequest	BOOL	True: one or more safety function blocks indicate an error False: no error message from a safety function block
xSFBError	BOOL	True: one or more reset requests in the SAFECONF project (LPSDO) False: no reset request
xOpAckReq	BOOL	Acknowledgment by the user is required (one or more communication errors have occurred)
wDiagCodeLPSDO	WORD	Diagnostic code of the LPSDO module (device-specific; see module user documentation)
arrDiagCodeDev	Array [0...16] of WORD	Array of SBT module diagnostic codes (device-specific; see module user documentation)

Name	Type	Description
dwAppDiag	DWORD	32 freely configurable feedback signals from the LPSDO module to the standard controller
arrInData	Array [0...16] of WORD	Array of input data from all input modules (satellites) of the island The index represents the module number (0 stands for the LPSDO module): arrInData [0] - InData LPSDO (no input data) arrInData [1] - InData module 1 ... arrInData [16] - InData module 16
arrFeedbackData	Array [0...16] of WORD	Array of feedback output data from the LPSDO module and the satellites (automatic call) The index represents the module number (0 stands for the LPSDO module): arrFeedbackData [0] - feedback data LPSDO arrFeedbackData [1] - feedback data module 1 ... arrFeedbackData [16] - feedback data module 16

B 1.3 I/O parameters

Table B-6 I/O parameters for the SBT_V3.Operate function block

Name	Type	Description
udtSBT	udtSBT_V3	Structure for data exchange between function blocks

B 1.3.1 Function block diagnostics

Diagnostics

Table B-7 Diagnostic codes

DiagCode	Meaning	
0000_{hex}	Function block is not active	
0100_{hex}	Initialization	
$C100_{hex}$	Initialization error	
	AddDiagCode	**Meaning**
	0004_{hex}	Wrong or invalid island number
8000_{hex}	Function block is active and operating without errors	
8100_{hex}	Reading data from data block	
8200_{hex}	Reading “Project Header” of LPSDO module	
	AddDiagCode	**Meaning**
	0001_{hex}	Initializing reading of “Project Header”
	0002_{hex}	Reading of “Project Header” is complete
8300_{hex}	Comparing “Header” of LPSDO module and the loaded program	
8400_{hex}	Downloading SAFECONF program to the LPSDO module	
	AddDiagCode	**Meaning**
	0000_{hex}	Removing old “Header” and writing a new one
	0001_{hex}	Downloading logic block
	0002_{hex}	Downloading address block
	0003_{hex}	Downloading new “Project Header”
8500_{hex}	Download completed successfully	
$C400_{hex}$	Error during download	
	AddDiagCode	**Meaning**
	$xxyy_{hex}$	Diagnostic code from the LPSDO module (see module user documentation)

B 2 SBT_V3.ReadDev function block

The function block reads the IN process data of a SafetyBridge module and writes the data to the transfer structure for the “SBT_V3.Operate” function block.

1 instance per SBT module

Use one instance of the function block per SafetyBridge module in the island.

Observe the sequence

Arrange all the SBT_V3.ReadDev instances **before** the “SBT_V3.Operate” function block in order to enable the best possible response time in the SafetyBridge system.

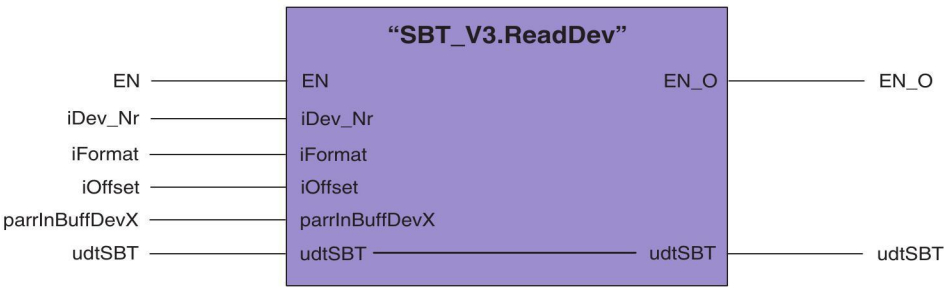


Figure B-2 SBT_V3.ReadDev function block

B 2.1 Input parameters

Table B-8 Input parameters for the SBT_V3.ReadDev function block

Name	Type	Description
EN	BOOL	-
iDev_Nr	INT	Device number 0: LPSDO module 1 ... 16: module 1 ... 16
iFormat	INT	Format of connected “arrInBuffDevX” 0: Motorola 1: Intel Other: for further developments
iOffset	INT	Word offset of SBT module X within “arrInBuffDevX”
parrInBuffDevX	Pointer	Exchange data (inputs) from SBT module X

B 2.2 Output parameters

Table B-9 Output parameters for the SBT_V3.ReadDev function block

Name	Type	Description
EN_O	BOOL	-

B 2.3 I/O parameters

Table B-10 I/O parameters for the SBT_V3.ReadDev function block

Name	Type	Description
udtSBT	udtSBT_V3	Structure for data exchange between function blocks

B 3 SBT_V3.WriteDev function block

The function block writes the OUT process data of a SafetyBridge module from the transfer structure of the “SBT_V3.Operate” function block to the output process image.

1 instance per SBT module

Use one instance of the function block per SafetyBridge module in the island.

Observe the sequence

Arrange all the SBT_V3.WriteDev instances **after** the “SBT_V3.Operate” function block in order to enable the best possible response time in the SafetyBridge system.

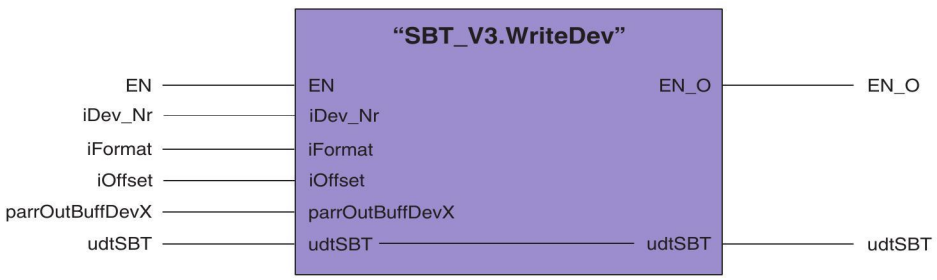


Figure B-3 SBT_V3.WriteDev function block

B 3.1 Input parameters

Table B-11 Input parameters for the SBT_V3.WriteDev function block

Name	Type	Description
EN	BOOL	-
iDev_Nr	INT	Device number 0: LPSDO module 1 ... 16: module 1 ... 16
iFormat	INT	Format of connected “arrInBuffDevX” 0: Motorola 1: Intel Other: for further developments
iOffset	INT	Word offset of SBT module X within “arrInBuffDevX”
parrOutBuffDevX	Pointer	Exchange data (outputs) from SBT module X

B 3.2 Output parameters

Table B-12 Output parameters for the SBT_V3.WriteDev function block

Name	Type	Description
EN_O	BOOL	-

B 3.2.1 I/O parameters

Table B-13 I/O parameters for the SBT_V3.WriteDev function block

Name	Type	Description
udtSBT	udtSBT_V3	Structure for data exchange between function blocks

B 4 SBT_V3.ProjHeader function block

Optional function block

This function block is optional. It is not essential in order to operate a SafetyBridge island.

The function block contains all information about the SAFECONF project and the project stored on the LPSDO module.

1 function block per island

Use the function block once per island.

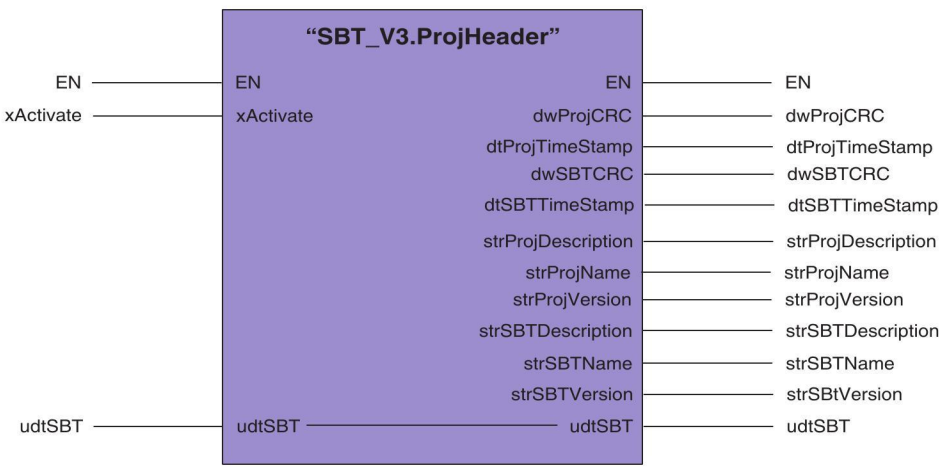


Figure B-4 SBT_V3.ProjHeader function block

B 4.1 Input parameters

Table B-14 Input parameters for the SBT_V3.ProjHeader function block

Name	Type	Description
EN	BOOL	-
xActivate	BOOL	Activation/deactivation of the function block

B 4.2 Output parameters

Table B-15 Output parameters for the SBT_V3.ProjHeader function block

Name	Type	Description
EN_O	BOOL	-
dwProjCRC	DINT	Checksum of the new project (FB)
dtProjTimeStamp	DATE_AND_TIME	Time stamp of the new project (FB)
dwSBTCRC	DINT	Checksum of the active project (LPSDO)
dtSBTTimeStamp	DATE_AND_TIME	Time stamp of the active project (LPSDO)
strProjDescription	STRING	Description of the new project (FB); see SAFECONF
strProjName	STRING	Name of the new project (FB); see SAFECONF
strProjVersion	STRING	Version of the new project (FB); see SAFECONF
strSBTDescription	STRING	Description of the active project (LPSDO)
strSBTName	STRING	Name of the active project (LPSDO)
strSBTVersion	STRING	Version of the active project (LPSDO)

B 4.3 I/O parameters

Table B-16 I/O parameters for the SBT_V3.ProjHeader function block

Name	Type	Description
udtSBT	udtSBT_V3	Structure for data exchange between function blocks

B 5 SBT_V3.TransTime function block

Optional function block

This function block is optional. It is not essential in order to operate a SafetyBridge island.

The function block measures the data transmission times between the logic module and satellite modules 1 to 16 as well as the maximum transmission time of a module.

1 function block per island

Use the function block once per island.

If the data transmission time of a module is greater than the F_WD_Time, a communication error is output at the LPSDO module. In this case, the data transmission time and F_WD_Time should be checked.

Please note that increasing the F_WD_Time has a direct effect on the safety function because it increases response times and therefore delay times and/or safety distances.

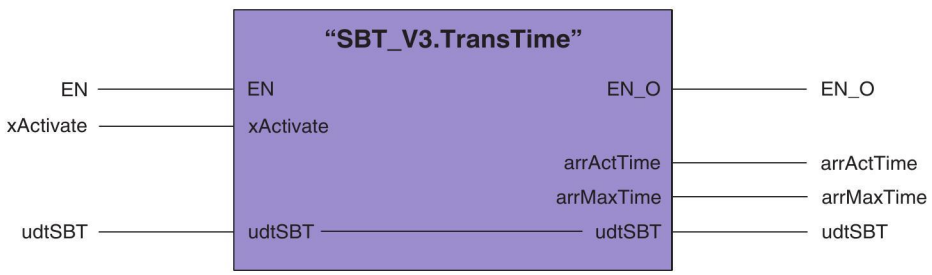


Figure B-5 SBT_V3.TransTime function block

B 5.1 Input parameters

Table B-17 Input parameters for the SBT_V3.TransTime function block

Name	Type	Description
EN	BOOL	-
xActivate	BOOL	Activation/deactivation of the function block

B 5.2 Output parameters

Table B-18 Output parameters for the SBT_V3.TransTime function block

Name	Type	Description
EN_O	BOOL	-
arrActTime	DINT [17]	Current data transmission time between LPSDO module and satellite modules The index represents the module number (0 is not used): arrActTime [1] - current transmission time for module 1 ... arrActTime [16] - current transmission time for module 16
arrMaxTime	DINT [17]	Maximum data transmission time between LPSDO module and satellite modules since the last FB activation The index represents the module number (0 is not used): arrMaxTime [1] - max. transmission time for module 1 ... arrMaxTime [16] - max. transmission time for module 16

B 5.3 I/O parameters

Table B-19 I/O parameters for the SBT_V3.TransTime function block

Name	Type	Description
udtSBT	udtSBT_V3	Structure for data exchange between function blocks

B 6 SBT_V3.CrossComm function block

Optional function block

This function block is optional. It is not essential in order to operate a SafetyBridge island.

In SafetyBridge Technology V3, islands can communicate with one another. Cross communication takes place via a master/slave model, where one or more islands can act as slaves for other master modules. Each island has the “udtSBT” data structure. The “SBT_V3.CrossComm” function block combines the data structures of the individual islands in an array, thereby enabling cross communication between the islands.

1 function block per standard controller

Use the function block once per standard controller.

If another island is in another standard controller, use an additional “SBT_V3.DataExch” function block. See “SBT_V3.DataExchange function block” on page 57.

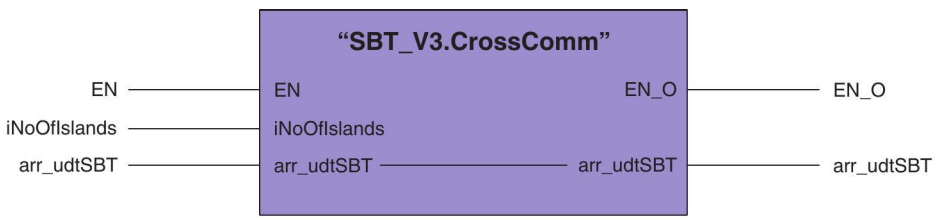


Figure B-6 SBT_V3.CrossComm function block

B 6.1 Input parameters

Table B-20 Input parameters for the SBT_V3.CrossComm function block

Name	Type	Description
EN	BOOL	-
iNoOfIslands	DINT	Maximum index that is used in the “arr_udtSBT” array

To increase performance, the “MaxIslandIndex” parameter limits the cycle for cross communication. If the value is 0, all 31 possible islands are checked by the function block.

B 6.2 Output parameters

Table B-21 Output parameters for the SBT_V3.CrossComm function block

Name	Type	Description
EN_O	BOOL	-

B 6.3 I/O parameters

Table B-22 I/O parameters for the SBT_V3.CrossComm function block

Name	Type	Description
arr_udtSBT	Array [32] of udtSBT_V3	Array of data structures for data exchange between function blocks

The “arr_udtSBT” array should contain all “udtSBT” structures of the islands.
The index of the “arr_udtSBT” array is independent of the index of the island.

B 7 SBT_V3.DataExchange function block

Optional function block

This function block is optional. It is not essential in order to operate a SafetyBridge island.

If two SBT islands are connected to different controllers but cross communication is required between the two islands, the “SBT_V3.DataExchange” function block supports data exchange between the master and slave island. The function block uses input and output buffers, which buffer the data that is received from one controller and sent to the other controller. The data transmission method between the two controllers is not specified and can be any of various options, for example Modbus TCP. However, the data must be transmitted consistently.

1 instance per slave island

Use one instance of the function block per slave island.

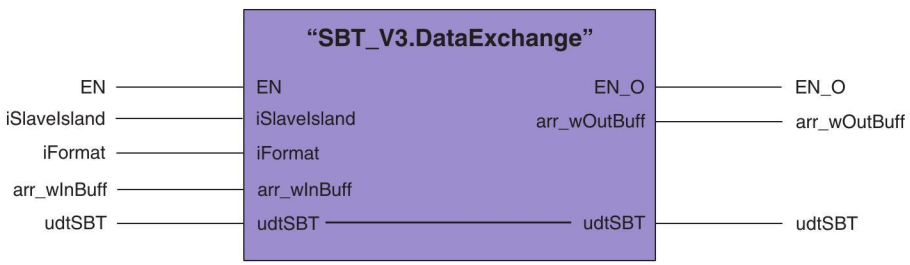


Figure B-7 SBT_V3.DataExchange function block

B 7.1 Input parameters

Table B-23 Input parameters for the SBT_V3.DataExchange function block

Name	Type	Description
EN	BOOL	-
iSlaveIsland	INT	In the master program: number of the slave island with which the master island is exchanging data In the slave program: 0 because the island itself is a slave
iFormat	INT	Specification of the format for the exchanged data 0: Big Endian (e.g., Siemens PLC) 1: Little Endian (e.g., ILC3xx)
arr_wInBuff	Array [0...7] of WORD	Exchange data (input) from the other island

B 7.2 Output parameters

Table B-24 Output parameters for the SBT_V3.DataExchange function block

Name	Type	Description
EN_O	BOOL	-
arr_wOutBuff	Array [0...7] of WORD	Exchange data (output) sent to the other island

B 7.3 I/O parameters

Table B-25 I/O parameters for the SBT_V3.DataExchange function block

Name	Type	Description
udtSBT	SBT_V3_Data	Data structure of the master island In master program: same data structure of the SBT_V3.Operate function block In slave program: not the same data structure of the SBT_V3.Operate function block

C Revision history

Revision	Date	Contents
00	2016-06-08	First publication