06/25 MN050028

XN Safety Designer

Powering Business Worldwide

Company information

All brand and product names are trademarks or registered trademarks of their respective owners.

Service

For service and support, please contact your local sales team.

Contact info.	Eaton.com/contact
Service page:	Eaton.com/aftersales

Original documentation

is the German-language edition of this document

Publication date

06/25 MN050028, 1th edition

Copyright

© 2025 Eaton Industries GmbH, 53105 Bonn

All rights, including those of translation, reserved.

No part of this manual may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, micro-filming, recording, or otherwise, without the prior written permission of Eaton.

Subject to alteration.

Table of Contents

	XN Safety Designer	**1**
	Company information	2
	Table of Contents	3
0.1	About this documentation	17
0.1.1	List of revisions	17
0.1.2	Target group	18
0.1.3	Legal disclaimer	19
0.1.4	Short designations	20
0.1.5	Writing conventions	20
0.1.5.1	Warning labels	21
0.1.5.2	Additional information for use	22
1.	**Using the XN Safety Designer programming software**	**23**
1.1	Definition: Project	23
1.2	General diagram showing an example of a project, hardware topology, and program networks	24
1.3	Flowchart for creating a project	24
1.4	Loading projects with a newer version of XN Safety Designer	26
2.	**Conditions and requirements for users**	**27**
3.	**Creating the hardware configuration**	**28**
3.1	Determining the hardware configuration through an online connection	28
3.2	Manually editing an XS-102-C00-000 in standalone mode	29
3.3	Easily finding out specific information for a Safety CPU	31
4.	**Graphic editor**	**32**
4.1	User Interface	32
4.2	Menu bar	32
4.2.1	File	33
4.2.2	Edit	34
4.2.3	View	35
4.2.4	Build	36

4.2.5 Debug ... 37
4.2.6 Help ... 37
4.3 Toolbars ... 38
4.4 Tree views ... 40
4.4.1 Project Tree ... 41
4.4.2 Hardware Tree ... 43
4.4.3 Function Block Tree ... 46
4.4.4 Variable Tree ... 48
4.4.4.1 Bit tags ... 49
4.4.5 Library Tree ... 51
4.4.6 Function Block Macro Tree ... 52
4.4.7 Subfolders in the Tree view ... 55
4.5 Property Browser ... 57
4.6 Editor ... 58
4.7 Column for inputs ... 59
4.8 Column for outputs ... 62
4.9 Network editor ... 63
4.9.1 Network elements ... 64
4.9.1.1 Bit tags ... 65
4.10 General instructions for using the editor ... 67
4.10.1 Filling the input column ... 67
4.10.2 Filling the output column ... 67
4.11 Working in the network editor ... 68
4.11.1 Adding network elements ... 68
4.11.2 Selecting network elements ... 68
4.11.3 Moving network elements ... 69
4.11.4 Deleting network elements ... 69
4.11.5 Configuring network elements ... 69
4.11.6 Drawing connections ... 70
4.11.7 Selecting connections ... 71
4.11.8 Deleting connections ... 71
4.11.9 Global connections ... 72
4.11.10 Adding comments ... 73
4.11.11 Selecting comments ... 73

4.11.12 Moving comments ... 73
4.11.13 Deleting comments ... 73
4.11.14 Resizing comments ... 74
4.11.15 Entering comment texts ... 74
4.11.16 Copying, cutting, and pasting elements ... 74
4.11.17 Replacing placeholders ... 75
4.11.18 Replacing existing elements ... 75
4.12 Directly jumping to the definition of an element ... 75
4.13 Directly jumping to the start points / end points of a connection 76
4.14 Directly jumping to temporary variables ... 77
4.15 Pan and zoom ... 78
4.16 Changing the column width ... 78
4.17 Error messages when trying to add an element to a network ... 79
4.18 Settings in the project ... 81
4.18.1 Temporary Operation Time ... 81
4.18.2 Max. Cycle Time ... 81
4.18.3 Maximum Transmission Time ... 81
4.18.4 Maximum Transmission Time (Output as Input) ... 81
4.18.5 Filter Time ... 81
4.18.6 Cross Circuit Detection ... 81
4.18.7 Hardware Revision ... 82
4.18.8 Parameter Revision ... 82
4.18.9 Simplified function block symbols ... 83
4.18.10 Enable CRC writing ... 84
4.18.11 Enable Easy Instance setting for instances ... 84
4.18.12 Enable setting for download with additional information ... 84
4.18.13 Disable flashing for an LED (local modules) ... 84
4.19 Enable signal for safe outputs ... 85
4.20 Optional modules ... 87
4.21 Validating and generating download data ... 89
4.22 Calculating the monitoring settings ... 97
4.23 Resource consumption and response time ... 98
4.23.1 Resource consumption ... 99
4.23.2 Cycle time estimate for user program ... 100

4.23.3 Total response time 100
4.23.4 PDO frame length 100
4.24 Debugging 101
4.25 Forcing 102
4.26 Watch window 104
4.27 Viewing the online values of modules in the hardware tree 106
4.28 Reading and viewing Safety module log files 109
4.29 Printing 114
4.29.1 Settings when printing networks 116
4.29.2 Settings for printing the project tree 117
4.30 CRC comparison dialog 118
4.31 Global search 120

5. Constants 121

6. Version control and archiving 122
6.1 Versioning 122
6.2 Archiving 122
6.3 Saving as an XML file 122

7. Protection level settings for password mechanism 123

8. Using unsafe variables through the CAN bus 125
8.1 Prerequisite for use 125
8.2 Unsafe variable structure 125
8.3 Creating a project that uses unsafe variables through the CAN bus 126
8.4 Displaying the variables in online mode 128
8.5 Transferring status information to the Safety CPU through the CAN bus 129

9. Use of unsafe variables in stand-alone operation 130
9.1 SDO communication 130
9.2 SDO frame structure 131
9.3 Writing an SDO frame 131
9.4 Reading an SDO frame 132
9.5 Overview of SDO commands 132

9.6 Detailed description of SDO READ command 133
9.7 Detailed description of SDO WRITE command 133
9.8 Detailed description of SDO GET_STATE command 134
9.9 Detailed description of SDO GET_DIAG_INFO command 135
9.10 Detailed description of SDO GET_SAFENBR command 136
9.11 Detailed description of SDO GET_FIRMWARE_VER command 136
9.12 Detailed description of SDO GET_NOP command 137
9.13 Detailed description of SDO READ_VALUES command 138
9.14 Detailed description of SDO WRITE_VALUES command 138
9.15 Important Safety CPU virtual addresses 139
9.16 Configuration block types 139
9.17 Reading the configuration 140
9.18 Structure of the individual list entry block types 141
9.18.1 CFG_MODULE_V3 141
9.18.2 CFG_REV 144
9.19 Status of expansion module I/Os 144
9.20 Detected expansion modules 145
9.21 Fast non-safe input and output variables 145
9.22 Unsafe release signals for local outputs 145

10. Determining the sequence for the compilation of a safety CPU 146
10.1 Order of the networks 146
10.2 Sequence of instances within the networks 147
10.3 Showing the processing sequence 147

11. Online actions and download 149
11.1 Target/actual comparison of the hardware 151
11.2 Online actions 152
11.3 Download Wizard 154
11.3.1 Step 1: Compare the hardware tree and check the safety number of the module. 154
11.3.2 Step 2: Deleting the program on the Safety CPU, downloading a new project, uploading the code from the Safety CPU and comparing it with the project. 155
11.4 Passwords 157

11.4.1 Overview of actions allowed by each password level 158
11.5 Upload Log-Data 159
11.6 Get Module State 159

12. Export of download data for a safety CPU 160
12.1 Creating the binary file 160
12.1.1 Error during the check in the external tool: 161
12.1.2 Error during Reload and Compare: 162
12.2 Load binary file on a Safety CPU 163

13. Function blocks 164
13.1 Data Types 164
13.2 Special implementation details 165
13.3 Time monitoring for function blocks 166
13.4 Standard function blocks 167
13.4.1 General Information 167
13.4.2 Function blocks for type conversion 169
13.4.2.1 BOOL_TO_SAFEBOOL 169
13.4.2.2 DINT_TO_SAFEDINT 170
13.4.2.3 SAFEBOOL_TO_BOOL 170
13.4.2.4 SAFEDINT_TO_DINT 170
13.4.2.5 SAFEDINTERROR_TO_SAFEDINT 171
13.4.2.6 SERROR_TO_SFALSE 171
13.4.3 Standard numerical function blocks 172
13.4.3.1 ABS (Absolute value) 173
13.4.3.2 ABS_NS (absolute value, not safe) 173
13.4.3.3 ADD (Addition) 173
13.4.3.4 ADD_NS (Addition, not safe) 174
13.4.3.5 SUB (subtraction) 174
13.4.3.6 SUB_NS (subtraction, not safe) 175
13.4.3.7 MULDIV_NS (Multiplication and Division, not safe) 175
13.4.3.8 MULDIV (multiplication and division) 176
13.4.3.9 SF_SampleHold 176
13.4.3.10 SHL (Shift-Left) 177
13.4.3.11 SHL_NS (Shift-Left, not safe) 177

13.4.3.12 SHR (Shift-Right) ... 178
13.4.3.13 SHR_NS (Shift-Right, not safe) ... 178
13.4.3.14 AND_BW (AND, bitwise) ... 179
13.4.3.15 OR_BW (OR, bitwise) ... 179
13.4.3.16 XOR_BW (XOR, bitwise) ... 180
13.4.3.17 NOT_BW (NOT, bitwise) ... 180
13.4.3.18 Restart_Interlock ... 181
13.4.3.19 FW_Check ... 182
13.4.3.20 Path Speed ... 183
13.4.3.21 Speed ... 185
13.4.3.22 Direction ... 187
13.4.4 Comparative function blocks ... 189
13.4.4.1 GT (greater than) ... 189
13.4.4.2 GT_NS (greater than, not safe) ... 190
13.4.4.3 GE (greater than or equal to) ... 190
13.4.4.4 GE_NS (greater than or equal to, not safe) ... 190
13.4.4.5 LT (less than) ... 191
13.4.4.6 LT_NS (less than, not safe) ... 191
13.4.4.7 LE (less than or equal to) ... 191
13.4.4.8 LE_NS (less than or equal to, not safe) ... 192
13.4.4.9 EQ (equal) ... 192
13.4.4.10 EQ_NS (equal, not safe) ... 192
13.4.4.11 MIN (Minimum) ... 193
13.4.4.12 MIN_NS (Minimum, not safe) ... 193
13.4.4.13 MAX (Maximum) ... 194
13.4.4.14 MAX_NS (Maximum, not safe) ... 194
13.4.4.15 LIMIT (within limits) ... 195
13.4.4.16 LIMIT_NS (within limits, not safe) ... 195
13.4.4.17 SDINT_Compare ... 196
13.4.4.18 DINT_Compare ... 197
13.4.5 Logical function blocks ... 198
13.4.5.1 AND ... 198
13.4.5.2 AND_5I ... 199
13.4.5.3 AND_NS - AND NOT SAFE ... 199

13.4.5.4 NOT ... 200
13.4.5.5 OR ... 200
13.4.5.6 OR_5I ... 201
13.4.5.7 XOR ... 201
13.4.5.8 AND_XI ... 202
13.4.5.9 OR_XI ... 203
13.4.6 Timer function blocks ... 204
13.4.6.1 TOFF ... 204
13.4.6.2 TON ... 205
13.4.6.3 TP ... 205
13.4.7 Trigger function blocks ... 205
13.4.7.1 F_TRIG ... 206
13.4.7.2 R_TRIG ... 206
13.4.8 Counter function blocks ... 206
13.4.8.1 CTD - Down counter ... 207
13.4.8.2 CTU - Up counter ... 207
13.4.8.3 CTUD - Up/down counter ... 208
13.4.9 FlipFlop function blocks ... 208
13.4.9.1 RS_Flipflop ... 209
13.4.9.2 SR_Flipflop ... 209
13.5 Complex function blocks ... 210
13.5.1 Collision_Detection function ... 211
13.5.1.1 Functionalities ... 211
13.5.1.2 Workspace ... 212
13.5.1.3 Tool Center Point and enveloping sphere ... 212
13.5.1.4 Creating a complex working space ... 213
13.5.1.5 Interfaces ... 213
13.5.1.6 Example showing how to connect the function block ... 214
13.5.2 Safe_Area function ... 215
13.5.2.1 Functionalities ... 216
13.5.2.2 Workspace ... 216
13.5.2.3 Tool Center Point and enveloping sphere ... 217
13.5.2.4 Interfaces ... 217
13.5.2.5 Example showing how to connect the function block ... 218

13.5.3 SF_Antivalent function 219
13.5.3.1 Error detection 220
13.5.3.2 Error behavior 220
13.5.3.3 Function block-specific errors and status codes 220
13.5.3.4 Status Diagram SF_Antivalent 221
13.5.3.5 Time Diagram SF_Antivalent 222
13.5.4 SF_EDM function 223
13.5.4.1 Error detection 224
13.5.4.2 Error behavior 224
13.5.4.3 Characteristics 227
13.5.4.4 Status Diagram SF_EDM 228
13.5.4.5 Time Diagram SF_EDM 229
13.5.5 SF_EmergencyStop function 230
13.5.5.1 Error detection 231
13.5.5.2 Error behavior 232
13.5.5.3 Status Diagram SF_EmergencyStop 233
13.5.5.4 Time Diagram SF_EmergencyStop 234
13.5.6 SF_EnableSwitch function 235
13.5.6.1 Error detection 237
13.5.6.2 Error behavior 237
13.5.6.3 Status Diagram SF_EnableSwitch 239
13.5.6.4 Time Diagram SF_EnableSwitch 240
13.5.7 SF_Equivalent function 241
13.5.7.1 Error detection 242
13.5.7.2 Error behavior 242
13.5.7.3 Status Diagram SF_Equivalent 243
13.5.7.4 Time Diagram SF_Equivalent 244
13.5.8 SF_ESPE function 245
13.5.8.1 Error detection 246
13.5.8.2 Error behavior 247
13.5.8.3 Status Diagram SF_ESPE 248
13.5.8.4 Time Diagram SF_ESPE 249
13.5.9 SF_GuardLocking function 250
13.5.9.1 Operating sequence 251

13.5.9.2 Error detection ... 251
13.5.9.3 Error behavior ... 252
13.5.9.4 Status Diagram SF_GuardLocking ... 254
13.5.9.5 Time Diagram SF_GuardLocking ... 255
13.5.10 SF_GuardMonitoring function ... 256
13.5.10.1 Error detection ... 257
13.5.10.2 Error and reset behavior ... 257
13.5.10.3 Status Diagram SF_GuardMonitoring ... 259
13.5.10.4 Time Diagram SF_GuardMonitoring ... 260
13.5.11 SF_ModeSelector function ... 261
13.5.11.1 Error detection ... 263
13.5.11.2 Error behavior ... 263
13.5.11.3 Characteristics ... 264
13.5.11.4 Status Diagram SF_ModeSelector ... 265
13.5.11.5 Time Diagram SF_ModeSelector ... 265
13.5.12 SF_MutingPar function ... 267
13.5.12.1 Error detection ... 269
13.5.12.2 Error behavior ... 269
13.5.12.3 Status Diagram SF_MutingPar ... 274
13.5.12.4 Time Diagram SF_MutingPar ... 275
13.5.13 SF_MutingPar_2Sensor function ... 276
13.5.13.1 Error detection ... 278
13.5.13.2 Error behavior ... 278
13.5.13.3 Status Diagram SF_MutingPar_2Sensor ... 281
13.5.13.4 Time Diagram SF_MutingPar_2Sensor ... 281
13.5.14 SF_MutingSeq function ... 282
13.5.14.1 Error detection ... 284
13.5.14.2 Error behavior ... 284
13.5.14.3 Status Diagram SF_MutingSeq ... 287
13.5.14.4 Time Diagram SF_MutingSeq ... 288
13.5.15 SF_Optional_Pwd function ... 289
13.5.15.1 Error detection ... 290
13.5.15.2 Error behavior ... 290
13.5.15.3 SF_Optional_Pwd Status Diagram ... 292

13.5.15.4 Time Diagram SF_Optional_Pwd ... 293
13.5.16 SF_Optional_PwdII function ... 294
13.5.16.1 Error detection ... 295
13.5.16.2 Error behavior ... 295
13.5.16.3 Status Diagram SF_Optional_PwdII ... 298
13.5.16.4 Time Diagram SF_Optional_PwdII ... 298
13.5.16.5 Safety measures and instructions ... 298
13.5.17 SF_Optional_Switch function ... 300
13.5.17.1 Table of values ... 300
13.5.18 SF_Optional_Switch_SDint ... 301
13.5.19 SF_Optional_Switch_Dint ... 302
13.5.20 SF_OutControl function ... 303
13.5.20.1 General ... 304
13.5.20.2 Optional conditions for ProcessControl ... 304
13.5.20.3 Optional startup inhibits ... 304
13.5.20.4 Error detection ... 304
13.5.20.5 Error behavior ... 304
13.5.20.6 Status Diagram SF_OutControl ... 306
13.5.20.7 Time Diagram SF_OutControl ... 307
13.5.21 SF_SafetyRequest function ... 308
13.5.21.1 Error detection ... 309
13.5.21.2 Error behavior ... 309
13.5.21.3 Status Diagram SF_SafetyRequest ... 311
13.5.21.4 Time Diagram SF_SafetyRequest ... 312
13.5.22 SF_TestableSafetySensor function ... 313
13.5.22.1 Test mode ... 315
13.5.22.2 Optional Startup inhibits ... 315
13.5.22.3 Error detection ... 315
13.5.22.4 Error behavior ... 315
13.5.22.5 Status Diagram SF_TestableSafetySensor ... 320
13.5.22.6 Time Diagram SF_TestableSafetySensor ... 321
13.5.23 SF_TwoHandControlTypeII function ... 322
13.5.23.1 Error detection ... 322
13.5.23.2 Error behavior ... 322

13.5.23.3 Status Diagram SF_TwoHandControlTypeII ... 325
13.5.23.4 Time Diagram SF_TwoHandControlTypeII ... 326
13.5.24 SF_TwoHandControlTypeIII function ... 327
13.5.24.1 Error detection ... 327
13.5.24.2 Error behavior ... 327
13.5.24.3 Characteristics ... 329
13.5.24.4 Status Diagram SF_TwoHandControlTypeIII ... 330
13.5.24.5 Time Diagram SF_TwoHandControlTypeIII ... 331
13.5.25 SF_HGW_Login ... 332
13.5.25.1 Error detection ... 334
13.5.25.2 Error behavior ... 334
13.5.25.3 SF_HGW_Login Status Diagram ... 338
13.5.26 Complex numeric function blocks ... 338
13.5.26.1 SF_LimitComparator ... 338
13.5.26.2 SF_LimitComparator_Dint ... 342
13.5.26.3 SF_Numeric_Selector ... 343
13.5.26.4 SF_RampMonitor ... 347
13.5.26.5 SF_DirectionMonitor ... 351
13.5.26.6 SF_SkewMonitor ... 355
13.6 Function block macros ... 359
13.6.1 Description ... 359
13.6.2 Terms ... 359
13.6.3 Creating macros ... 359
13.6.3.1 Assembling a macro ... 360
13.6.3.2 Create a macro from existing elements ... 361
13.6.3.3 Inserting connections into the macro network ... 362
13.6.3.4 Creating embedded macros ... 364
13.6.3.5 Creating a copy of a macro ... 364
13.6.4 Macro libraries ... 365
13.6.4.1 Creating macro libraries ... 365
13.6.4.2 Saving macro libraries ... 366
13.6.4.3 Displaying macro libraries ... 367
13.6.4.4 Using macros from libraries ... 367
13.6.4.5 Local macro library ... 368

13.6.4.6 Locking macro libraries ... 369
13.6.4.7 Comparison of macros ... 371
13.6.5 Debugging function block macros ... 373

14. Safety modules ... 374
14.1 Description of XS-102-C00-000 Safety CPU module ... 374
14.2 Description of XS-322-4AI-I2 Safety Analog input module ... 375
14.3 Description of XS-322-10DI-PF Safety Digital input module ... 376
14.4 Description of XS-322-8DO-P20 Safety Digital output module ... 377
14.5 Description of XS-322-8DIO-PF20 Safety Digital input/output module ... 378
14.6 Description of XS-322-2DO-RNODC Safety DC Relay output module ... 379
14.7 Description of XS-322-2DO-RNOAC Safety Relay output module ... 380
14.8 Description of XS-322-2INC Safety Encoder module ... 381

15. Error codes ... 382

Appendix ... 440
A.1 Further usage information ... 441
A.2 Shortcuts - Keyboard shortcuts and special keys ... 442
A.2.1 Function keys and key combinations ... 442
A.2.1.1 Keyboard shortcuts in the FILE menu ... 442
A.2.1.2 Keyboard shortcuts in the EDIT menu ... 442
A.2.1.3 Function key in BUILD menu ... 442
A.2.1.4 Function keys and keyboard shortcut in DEBUG menu ... 442
Alphabetical index ... 443

0.1 About this documentation

This documentation contains all the information you will need in order to use the XN Safety modular safety PLC safely and correctly.

This User Help is an integral part of the XN Safety Designer programming software program.

The XN Safety Designer programming software program is an integrated part of the XN Safety modular safety PLC system. This software is used to program the corresponding Safety projects.

Make sure to always work with the latest documentation for the XN Safety modules; please refer to → section "Further usage information", page 441

Please send any comments, recommendations, or suggestions regarding this document to: DocumentationEGBonn@eaton.com

0.1.1 List of revisions

The following significant amendments have been introduced since previous issues:

Publication date	Page	Keyword	New	Modification	Deleted
01/2025	ff.	Issue for certification	✓		
06/2025	ff.	Product launch edition	✓		

0.1 About this documentation

0.1.2 Target group

This user help contains all the information you will need in order to use the SafetyDesigner_EN to configure the Safety modules.

This description is intended for electricians and electrical engineers, as well as for the people who will be in charge of performing the electrical installation and people who will be using the XN Safety Designer:

- Project planners, automation technicians, and automation engineers
- Installers, commissioning technicians
- Machine operators
- Maintenance personnel / testing technicians

This manual assumes that the reader has extensive knowledge of automation and the specifics behind it.

For additional help, information, and compatible accessories, please visit our website: Eaton.com.

If you have any questions, our Support Team will of course be happy to assist you. For our emergency phone number and office hours, please visit our website.

XN Safety module must be installed and connected exclusively by electricians and people who are familiar with electrical installation work.

CAUTION

Installation requires qualified electrician

Follow all safety instructions for the modular XN Safety safety PLC!
The section on safety instructions must be read and understood by everyone who will be working with the XN Safety CPU module before the actual work is performed Safety modules.

0.1.3 Legal disclaimer

All the information in this manual has been prepared to the best of our knowledge and in accordance with the state of the art. However, this does not exclude the possibility of there being errors or inaccuracies. We reserve the right to make technical changes that serve to improve the devices.
We assume no liability for the correctness and completeness of this information. These operating instructions are purely a product description. In particular, the information does not contain any assurance of specific properties within the meaning of warranty law.
The machine manufacturer is responsible for proper installation and product configuration. The machine operator is responsible for safe handling and proper operation.

Do not use the XN Safety before reading and understanding this manual.

It is assumed that the user of this manual is thoroughly familiar with the information found in the manuals for incorporating the safety controller into automation processes.

Hazards posed by the safety controller cannot be ruled out if the safety instructions are not observed – especially if the safety controller is installed and commissioned by inadequately qualified personnel or if it is used improperly. Eaton accepts no liability for damage resulting from failure to follow these instructions or the relevant regulations.

The following notes and rules of use apply to the use of sample programs and to the use of illustrations of program parts used in the documentation:

1. The program examples provided were created to the best of our knowledge and belief and in accordance with the current state-of-the-art. The program examples provided were created to the best of our knowledge and belief and in accordance with the current state-of-the-art. However, errors cannot be totally excluded, and the example programs do not cover all function blocks and applications that are available for the safety controller.
2. Electrical engineering skills and know-how are required in order to be able to program and commission the safety controller. An incorrectly wired or incorrectly configured safety controller will pose a property damage risk and an injury hazard when active components such as motors and pressure cylinders are being driven.
3. When using the provided sample programs and generating a program with XN Safety Designer, the user has the sole responsibility to observe the following:
   - All relevant rules and practices for preparing circuit diagrams for the safety controller as specified in the latest documents for these XN Safety.

- All occupational health and safety and accident prevention directives, standards, and regulations applicable to the commissioning, circuit diagram creation for, and use of the safety controller for your planned application, in particular those imposed by employers' liability insurance associations (Berufsgenossenschaften).
- Acknowledged rule of technology and state of science.
- All other general due diligence regarding the prevention of damages to life and physical condition of persons as well as material damage.

4. The manufacturer cannot accept any liability for any damages that are caused by customers not using the program examples provided in accordance with the conditions of use specified here under points 1 to 3.

The latest version of this manual can be found on our Download Center by using the following search term: MN050028.

Eaton.com/documentation

0.1.4 Short designations

The following general terms are used throughout this manual:

Short designation	Explanation
XN Safety	Entire series, used to refer to all the modules in the product family
Safety CPU, SafeCPU	XN Safety CPU module XS-102-C00-000
XN Safety module	Extension of XN Safety CPU module
XN Safety Designer	XN Safety Designer

You can find the exact designation for your Safety modules from the inscription on the modules.

0.1.5 Writing conventions

Tbl. 1: Format conventions used throughout this manual

Award	Description
`Monospaced` text	Used for displays, elements at the file level, source code command lines
Button	Used for the button labels on the device and in the software
Inscription	Used for the labels on the software user interface
Menu path\submenu\...\item	Used for paths to views and dialog boxes in the software
Menu/command	Used for commands found in the menu
<name>	Angle brackets are used to indicate variable values that you must replace with your own values

0.1.5.1 Warning labels

Risk of personal injury warning.

DANGER

"Danger" means that death or serious injury will occur if the specified measures are not taken.

- Follow all instructions in order to prevent death and serious injury.

WARNING

"Warning" means that death or serious injury may occur if the specified measures are not taken.

- Follow all instructions in order to prevent death and serious injury.

CAUTION

"Caution" means that moderate or minor injury may occur if the specified measures are not taken.

- Follow all instructions in order to prevent moderate to minor injury.

Property damage warning

ATTENTION

Warns about the possibility of material damage.

Prohibited use

Prohibited uses, actions, etc.

Prohibition signs interdict actions or
the use of certain objects

Requirements

Requirement

Mandatory signs require a certain behavior

Notes

Additional information, information worth knowing, useful additional information

Provides important information regarding the product, handling, use, or relevant parts of the documentation that require special attention.

▶ Indicates instructions to be followed

0.1.5.2 Additional information for use

Documents (such as manuals) are listed after the icon together with the corresponding name and Eaton number.

Publication title For identifying the Eaton publication code

Links to external Internet addresses. They will be shown after the icon.

Destination address without http(s)://www.

Links to external content are shown in blue.

1. Using the XN Safety Designer programming software

1.1 Definition: Project

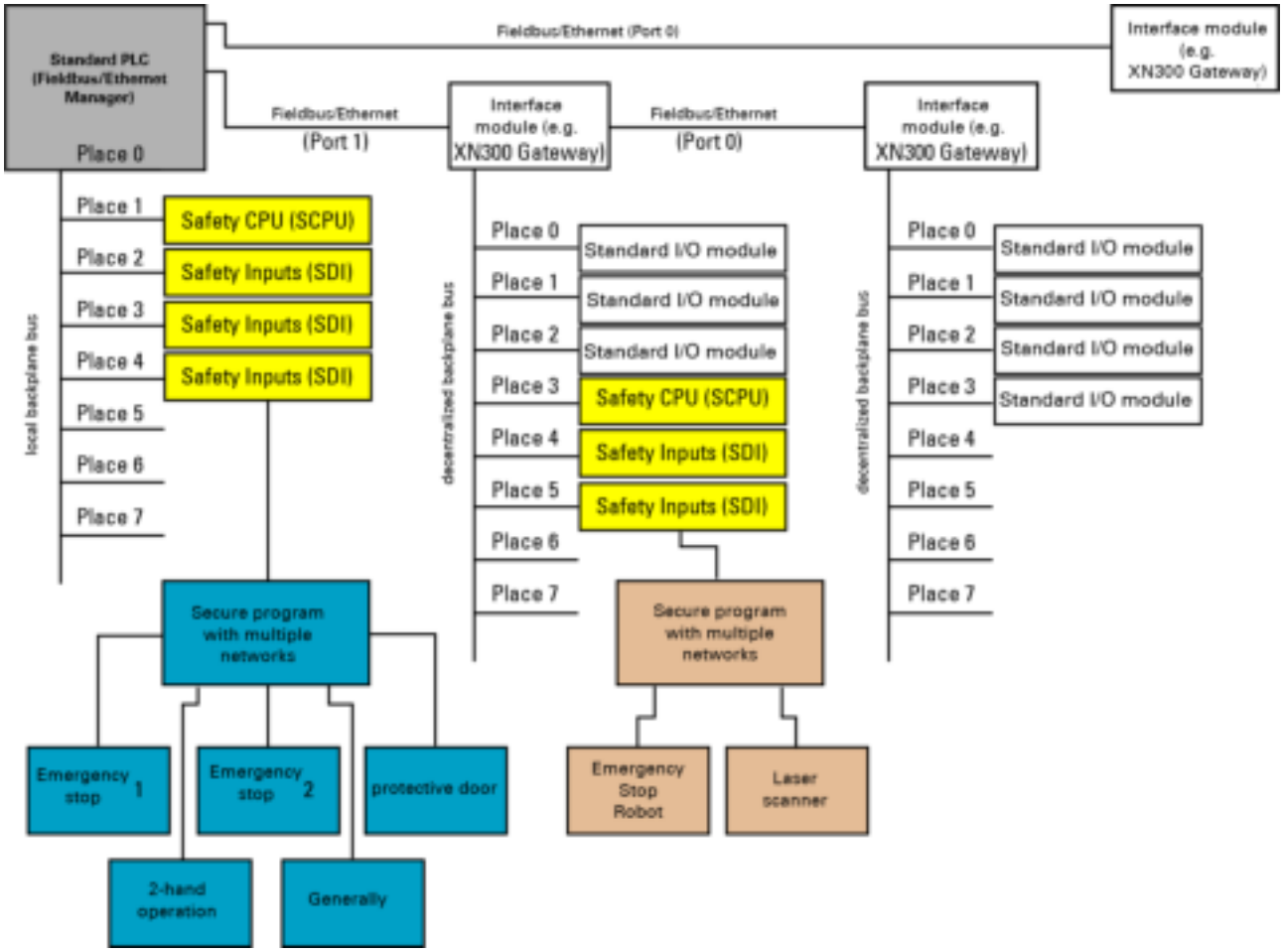
XN Safety Designer can be used to create and load projects. Within this context, a project contains the configuration data for the entire Safety system, as well as all Safety modules (Safety CPU and Safety I/O modules).
Please note that changes to a project can only be made offline, i.e., the moment there is an online connection to the Safety CPU (XS-102-C00-000), it will not be possible to make any changes to the project until the connection is terminated.

When using the XS-102-C00-000 Safety CPU in standalone mode, the Safety CPU will function as the root element.

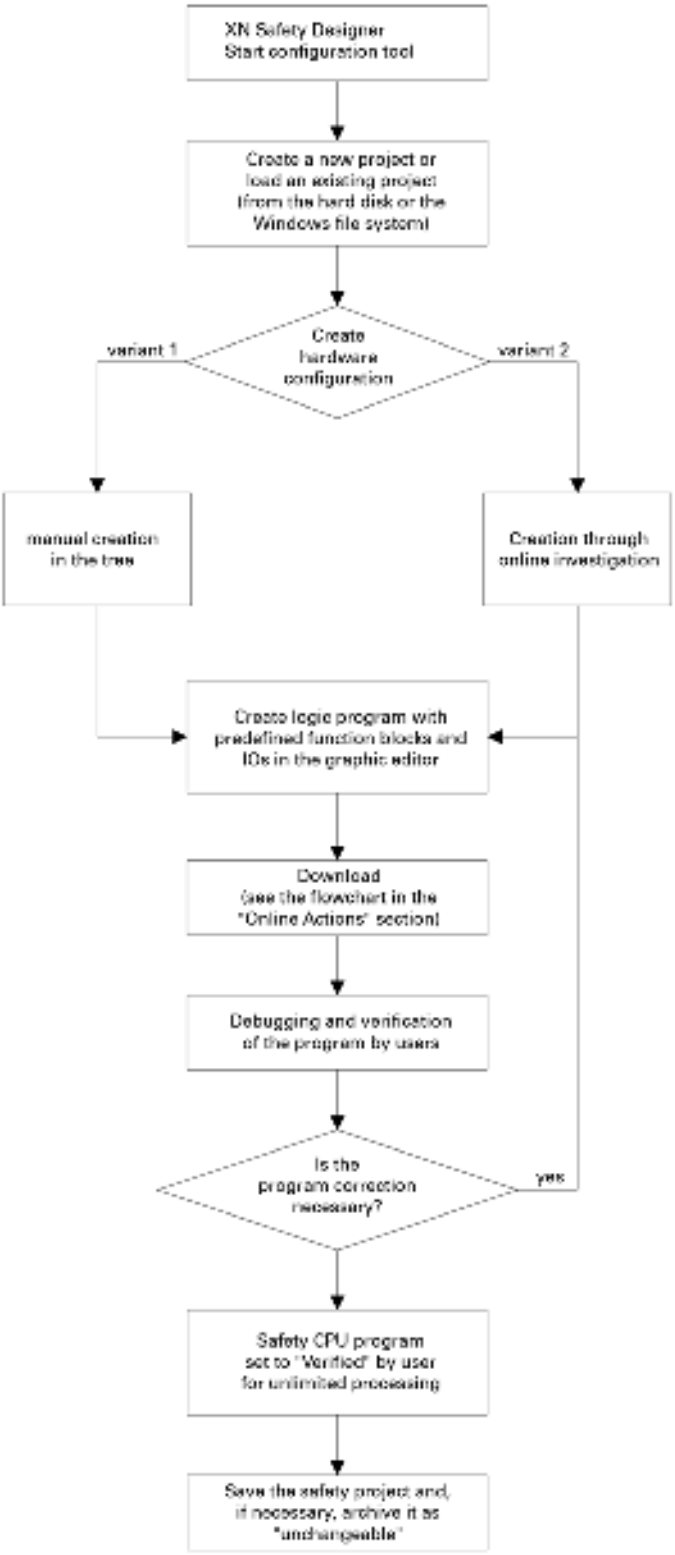
A program will be processed for each Safety CPU. This program, in turn, can be made up of multiple graphic networks (subprograms).

The XS-102-C00-000 Safety CPU can accommodate a maximum of 16 expansion modules (input cards, output cards, relays, etc.). The program can consist of multiple graphic networks (subprograms).

1.2 General diagram showing an example of a project, hardware topology, and program networks

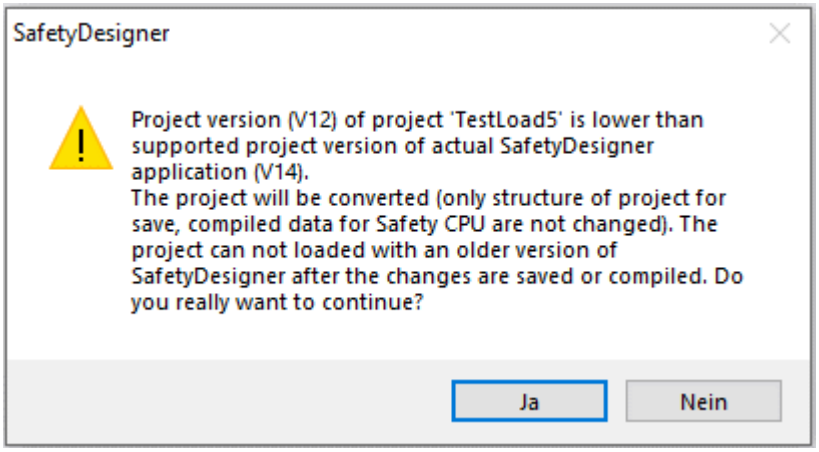


1.3 Flowchart for creating a project


XN Safety Designer
Start configuration tool
Create a new project or
load an existing project
(from the hard disk or the
Windows file system)
Create
hardware
configuration
variant 1
variant 2
manual creation
in the tree
Creation through
online investigation
Create logic program with
predefined function blocks and
IOs in the graphic editor
Download
(see the flowchart in the
"Online Actions" section)
Debugging and verification
of the program by users
Is the
program correction
necessary?
yes
Safety CPU program
set to "Verified" by user
for unlimited processing
Save the safety project and,
if necessary, archive it as
"unchangeable"

1.4 Loading projects with a newer version of XN Safety Designer

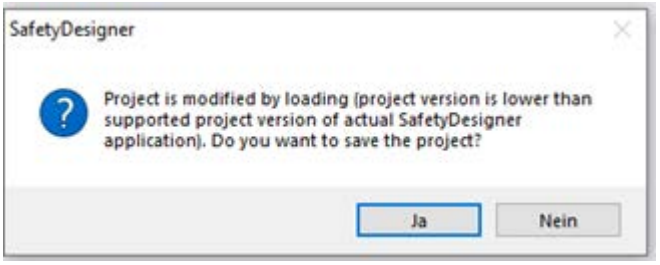
Projects created with the first version of XN Safety Designer can be loaded with a later version as well. When you load the project, you will be shown a prompt indicating that the XN Safety Designer version has changed and that once you save the project, you will no longer be able to load it in the old version.



You can choose whether to load the project in this prompt.

The differences between XN Safety Designer versions only have to do with the project configuration structure (there are additional configuration options in the Properties pane, for example). The actual compiled file, and the CRC for the compiled file on the Safety CPU, will not change.

If you do not make any changes to the project, you will be asked to confirm whether you want to save the project when trying to save or compile it.



Relationship between project version and version number

Project version	XN Safety Designer	Remark
V15	11.01.076	First build

2. Conditions and requirements for users

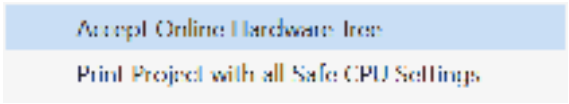
1. Without exception, XN Safety Designer must be used as intended. Among other things, this includes always checking all the parameters of the individual function blocks, Safety groups, constants, etc. to make sure they are correct.
2. It is also important to keep in mind that each project must be self-contained. This means that a Safety project must not, under any circumstance, attempt to access Safety modules from a different project – for example, a Safety CPU from project A should not try to access a Safety input module from project B. This is strictly prohibited!
3. Users must not, under any circumstance, update the firmware on XN Safety modules without authorization. This is strictly prohibited!
4. The safety system must be disconnected from XN Safety Designer during operation.
   Do not connect XN Safety Designer to the safety system except for configuration and diagnostic purposes.

3. Creating the hardware configuration

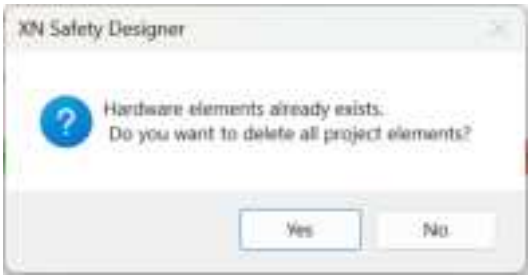
3.1 Determining the hardware configuration through an online connection

The hardware modules that are actually physically present can be identified by establishing an online connection to the PLC. The modules identified this way can be added to an empty Safety project with the context menu in the Hardware Tree pane.

To do this, right-click inside the Hardware Tree pane and click on the "Accept Online Hardware Tree" option.



If there is an existing Safety project (if there is already a hardware tree, for example), you will be shown a prompt letting you know that the existing project will be deleted. You can cancel this by clicking on "No".



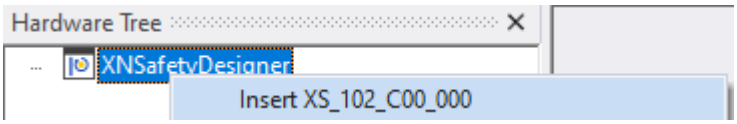
If you click on "Yes" instead, you will be able to open the various nodes in the hardware tree to see the modules that have been added. Please note that only the modules that are relevant to the Safety project will be added (Safety modules, interface modules).

Please note that you will need to modify the hardware tree as necessary before you can create an application.

The online connection can be established through the USB interface on the Safety CPU.

3.2 Manually editing an XS-102-C00-000 in standalone mode

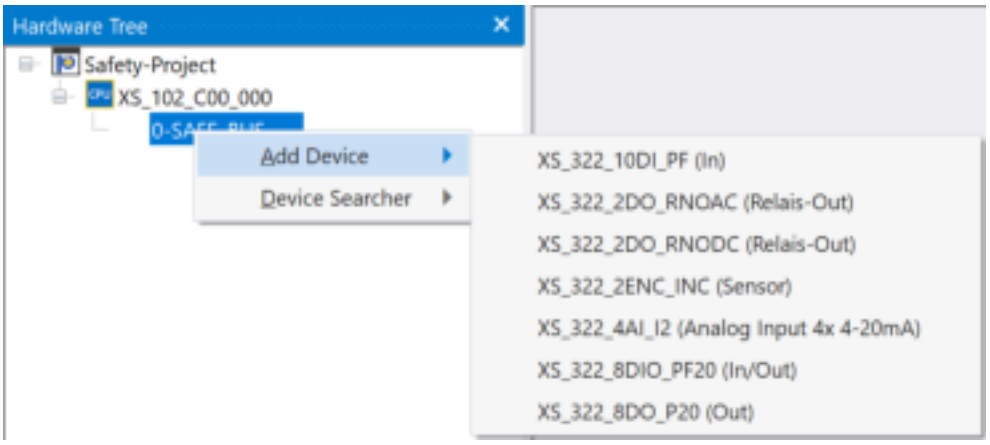
The program provides the option of configuring an XS-102-C00-000 in standalone mode. This method can be used to create a hardware configuration without an online connection to the XS-102-C00-000 module. To configure the corresponding tree, you will need to use the context menu in the Hardware Tree pane by right-clicking in it. In this case, the first step is to use the context menu for the root element in the Hardware Tree pane to add the XS-102-C00-000.



After you add the module, you will need to enter a name for the XS-102-C00-000 in the edit field or use the suggested name.

The next step is to expand the tree structure with an expansion slot (you can add up to 16 expansion modules). Please note that only one empty expansion slot will be shown at any one time at the end, and that no empty slots will be shown anymore once all 16 slots are used up.

To assign modules to the empty elements being shown, right-click on these elements to open the corresponding context menu. Only the modules that are allowed for the corresponding element will be shown.



You can use the steps above to put together your whole configuration with the modules you want. In addition, you can drag and drop the modules to the position you want within the hardware tree. Within this context, a cursor icon will be shown to

indicate whether the new position is allowed for the module you are dragging (dark rectangle if allowed; red cross if not allowed).

If you add the wrong module by accident, you can delete it with the "Delete Module" context menu option or by pressing the DEL key.

For the expansion modules, there is a maximum number of relay modules and sensors allowed. A maximum of eight XS_322_2DO_RNOAC, XS_322_2DO_RNODC and/or XS_322_2ENC_INC modules can be added, since each one of these modules takes up two slots.

After you add a module, you will need to enter a name for it in the edit field or use the suggested name.

In addition, you can use the "Replace Device" and "Replace Searcher" context menu options with Safety modules in order to change the respective module (from XS_322_10DI_PF to XS_322_8DO_P20, for instance). Please note that only modules allowed for replacement will be shown. In addition, existing elements in the networks that are affected by a replacement will be checked and deleted if necessary. If this happens, a prompt will be shown to let you know and ask you to explicitly confirm the replacement.

3.3 Easily finding out specific information for a Safety CPU

You can find out the names and version numbers of loaded projects by establishing an online connection to a Safety PLC without a project loaded in XN Safety Designer. A dialog box with the modules that are physically present will appear, and the name and version number of the loaded project for each Safety CPU will be shown between square brackets after each Safety CPU. The color of the corresponding text will indicate the Safety CPU's state:

- Green = Operation Mode
- Violet = Temporary Operation Mode
- Yellow = Service Mode
- Red = Error
- Gray = Other

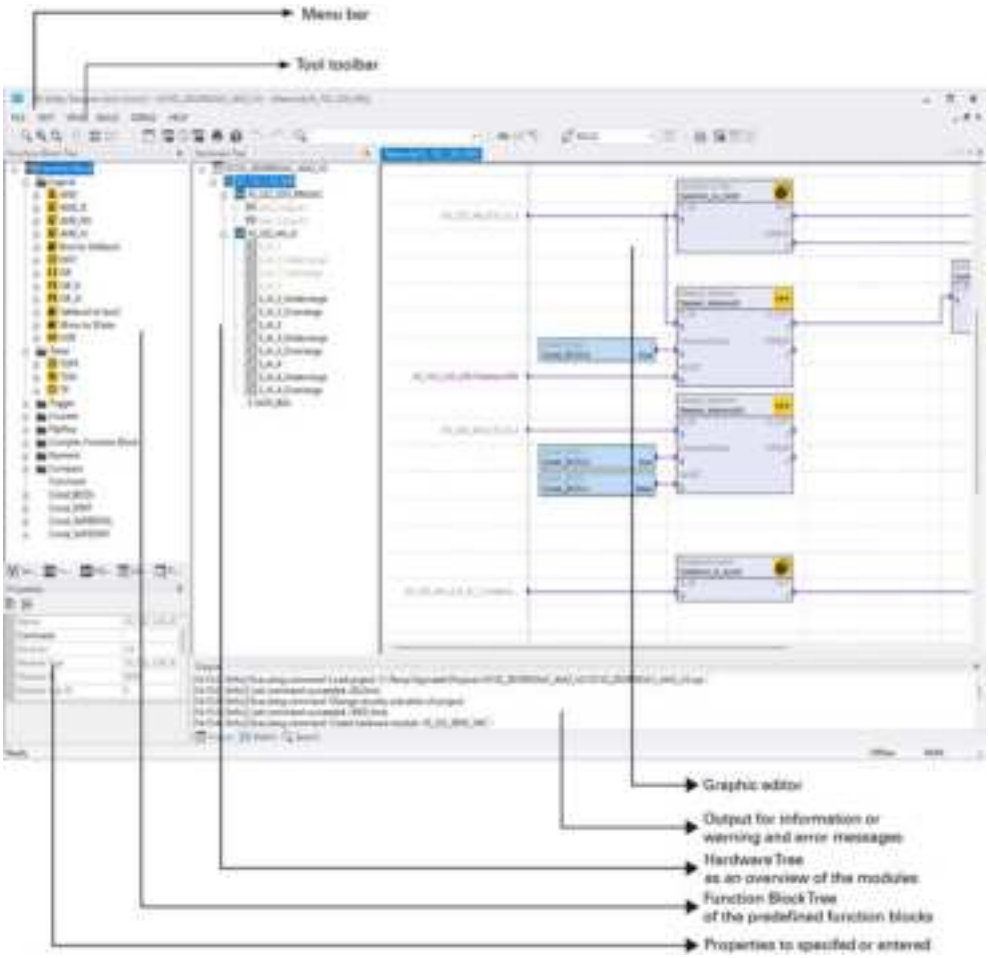
You can easily find out the firmware version and safety number of the Safety CPU. To do this, make sure the online hardware tree is open. Then move the cursor over the Safety CPU in the Online Hardware Tree pane and a tooltip showing the afore-mentioned information will appear.

4. Graphic editor

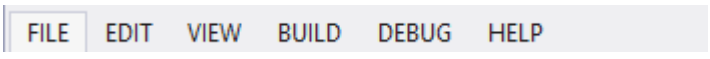
The basic configuration and use specifics are based on the usual user interface. The sequence code for the Safety CPU needs to be created graphically in the Safety CPU's network. Finally, the required parameters for the individual elements can be configured in the Properties pane.

Please note that you will only be able to make changes to the project if an online connection to the XS-102-C00-000 has not been established.

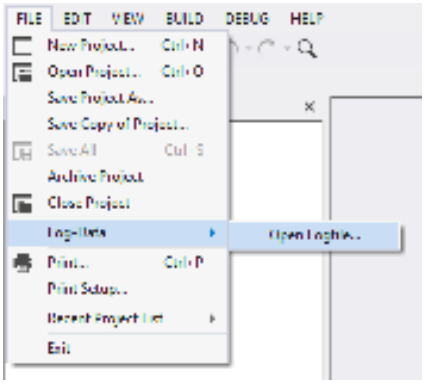
4.1 User Interface



4.2 Menu bar

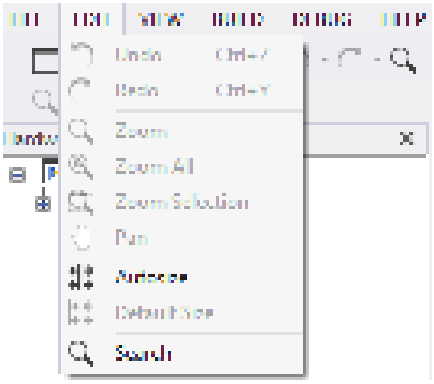


4.2.1 File



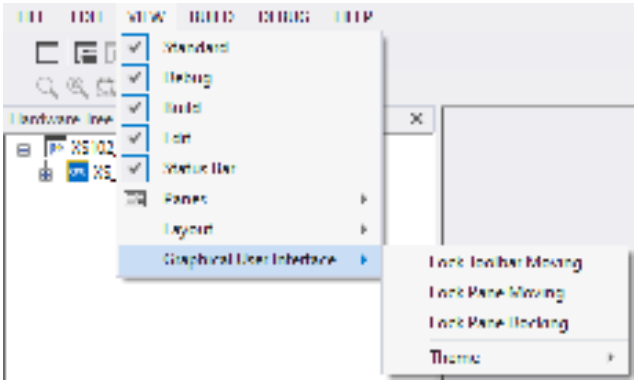
Overview		
	New Project	Opens the Create New Project dialog box used to create a new project.
	Open Project	Opens an existing project. The Open a Safety Project File dialog box can be used to select and open the project (*.ssp) you want.
	Save Project As	Used to save the currently open project with a different name and in a different path. The project will then be opened with the new name/path.
	Save Copy of Project	Used to save the currently open project with a different name and in a different path.
	Save All	Saves the project that is currently open.
	Archive Project	Archives the project that is currently open. Please note that the project will no longer be editable after it is archived!
	Close Project	Closes the currently open project.
	Log Data	Shows saved log files. The submenu will show the project's available log files.
	Print	Prints the entire project with all networks.
	Print Setup	Print menu settings
	Recent Project List	Shows a list of the most recently opened projects.
	Exit	Closes XN Safety Designer. If you have made any changes that have not yet been saved, a prompt for saving the project will appear before closing.

4.2.2 Edit



Overview		
	Undo	Undoes the last operation (the last action).
	Redo	Restores the last action that was undone.
	Zoom	Zoom tool for zooming into and out of objects in the network.
	Zoom All	Adjusts the zoom level for all the objects in the network in such a way that they are all visible.
	Zoom Selection	Used to adjust the zoom level for a specific area of the network.
	tbd	Manual tool for navigating through the network.
	Autosize	Automatically calculates the column width for a network.
	Default Size	Sets the network's column width to the default value.
	Search	Opens a dialog box for the global search function.

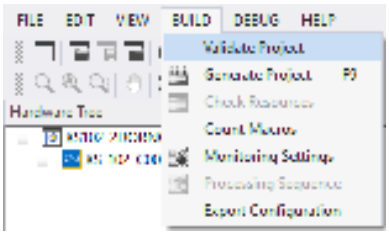
4.2.3 View



Overview		
	Normal Debug Build Edit Status Bar	Enable / disable the corresponding checkbox to show and hide the individual toolbars (Standard, Debug, Build, Edit, and Status Bar). A clearly laid out list and definition of the various toolbars can be found under → section "Toolbars", page 38.
	Panes	Enable / disable the corresponding checkbox to show or hide the following panes. → section "Tree views", page 40 goes over the individual panes in greater detail. Function Block Macro Tree Pane Library Tree Pane Variable Tree Pane Online Hardware Tree Pane Function Block Pane Hardware Tree Pane Project Tree Pane Properties Pane Watch Pane Output Pane
	Layout	Menu option for loading and saving the pane layout. Reset Pane Layout Export Pane Layout Import Pane Layout

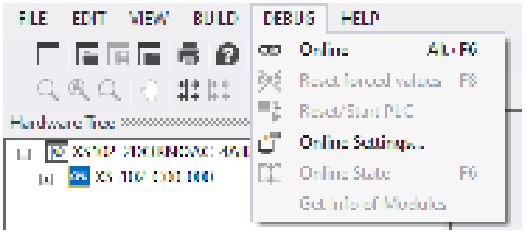
Overview	
Graphical User Interface	Enable / disable the corresponding checkbox in order to lock and unlock the following commands. Lock Toolbar Moving Lock Pane Moving Lock Pane Docking Theme – Visual user interface theme, set permanently to Eaton.

4.2.4 Build



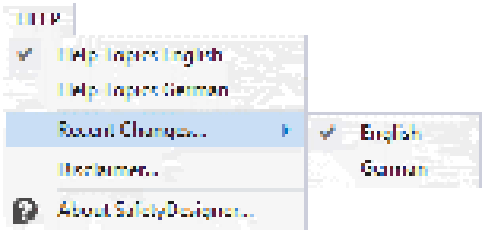
Overview		
	Validate Project	Checks (validates) the project that is currently open.
	Generate Project	Generates the download data for the project.
	Check Resources	Opens a dialog box with the expected memory use, cycle times, and response times.
	Count Macros	Determines all the use points of function blocks and macros and indicates the number of use points.
	Monitoring Settings	Opens a dialog box and calculates the expected cycle times and transmission times.
	Processing Sequence	Shows the processing sequence in networks; not available until the project has been compiled successfully.
	Export configuration	Exports a binary file with the download data.

4.2.5 Debug



Overview		
	Online	Establishes an online connection between the PC (project) and the PLC and generates the corresponding download data.
	Reset all forced values	Resets the forced values on the Safety CPU.
	Reset/Start PLC	Resets and starts the PLC before the safety application starts.
	Online Settings...	Opens the XN Online configurations dialog box.
	Online State	Opens and closes the online status indicator.

4.2.6 Help



Overview		
	Help Topics English	Opens the XN Safety Designer Help in English.
	Help Topics German	Opens the XN Safety Designer Help in German.
	Recent Changes...	Opens a Release Note file with the changes in this version in German or English.
	Disclaimer...	Legal disclaimer
	About XN Safety Designer...	Shows an information prompt for XN Safety Designer with the date and version.

4.3 Toolbars

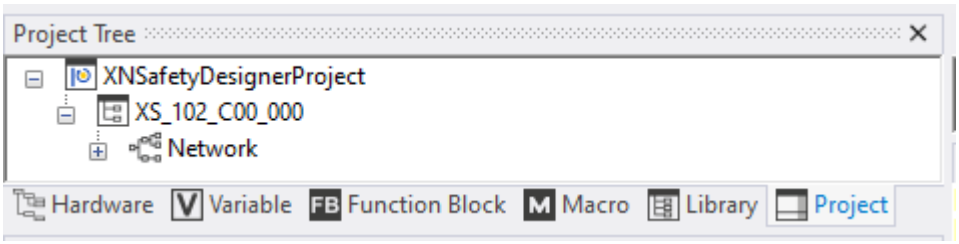
Overview			
	New Project	Creates a new project (opens the Create New Project dialog box).	Default toolbar
	Open Project	Opens an existing project. The Open a Safety Projectfile dialog box will appear, allowing you to select and open the project you want.	
	Save Project	Can be used to save the currently open project with a different name and in a different path. The project will then be opened with the new name/path.	
	Close Project	Closes the currently open project.	
	Print	Prints the entire project with all networks.	
	About XN Safety Designer	Shows the relevant software information with the corresponding date and version.	
	Undo	Undoes the last action.	
	Redo	Restores the last action that was undone with Undo.	
	Generate Project	Generates the download files for the project.	Build-Toolbar
	Monitored Settings	Opens a dialog box and calculates the expected cycle times and transmission times.	
	Check Resources	Opens a dialog box with the expected memory use, cycle times, and response times.	
	Online	Establishes an online connection between the PC (project) and the PLC and generates the corresponding download data.	Debug-Toolbar
	Reset forced values	Resets the forced values on the Safety CPU.	
	Reset/Start PLC	Resets and starts the PLC before the Safety application is started.	
	Online Settings...	Opens the XN Online configurations dialog box.	
	Current Online Configuration	Online connection to the Safety CPU.	
	Online State	Opens and closes the online status indicator.	
	Zoom	Zoom tool for zooming into and out of objects in the network.	Edit-Toolbar
	Zoom All	Adjusts the zoom level for all the objects in the network in such a way that they are all visible.	
	Zoom Selection	Used to adjust the zoom level for a specific area of the network.	
	tbd	Manual tool for navigating through the network.	
	Autosize	Automatically calculates the column width for a network.	
	Default Size	Sets the column width for the network to the default value.	

4.4 Tree views

The trees (tree structures) will be found on the left side of the editing tool by default and consist of six different panes:

- → section "Project Tree", page 41
- → section "Hardware Tree", page 43
- → section "Function Block Tree", page 46
- → section "Variable Tree", page 48
- → section "Library Tree", page 51
- → section "Function Block Macro Tree", page 52

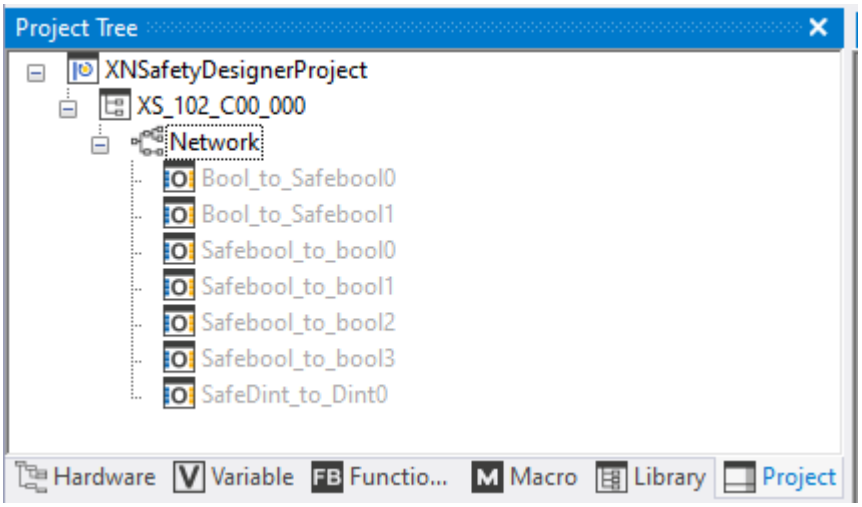
You can select each of these panes by clicking on the corresponding tab.



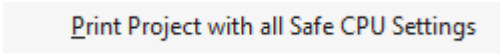
You can modify all trees with the exception of the function block tree. When you select the root element, the "Temporary Operation Time" property will be available in the Properties pane. This time (in minutes) specifies how long Temporary Operation Mode will run until the Safety CPU shuts down if the user does not complete the verification of the program with "Set Verified." The value range for this setting is five minutes to 480 minutes (eight hours) (→ section " Settings in the project", page 81).

4.4.1 Project Tree

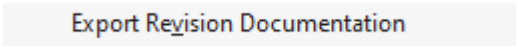
This tree structure shows the general project layout. The Safety CPU, the corresponding networks, and the placed function blocks will all be listed.



The root element will show the project name. You can use the context menu to print out a complete project with the corresponding Safety CPU and networks.



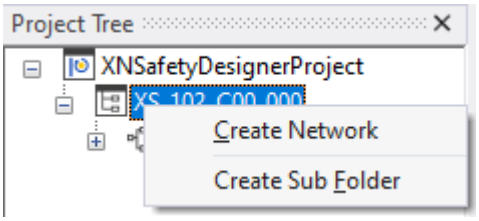
You can also use the context menu to export a revision history to an XML file.



The Safety CPU will be shown as a sub-element of the root element.

Please note that you will not be able to rename the Safety CPU here. You can only enter a new name for the element of the same name in the hardware tree. This name will then be automatically applied in the project tree.

You can use the Create Network option in the context menu for the Safety CPU to create a network.



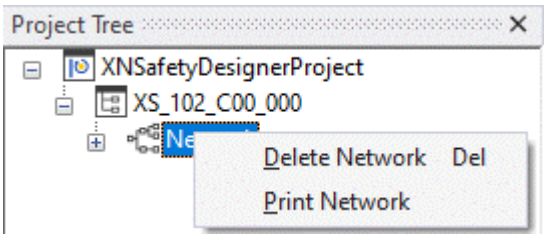
You can also add a network by pressing the [Insert] key.

This new network will be added as a sub-element of the Safety CPU. To open the network editor, simply double-click on the network name in the tree.

Placed function blocks will be shown as network sub-elements. These elements will be added automatically when you place them in the network editor.

You can change the name of the network in the tree, but each name must be unique within the Safety CPU.

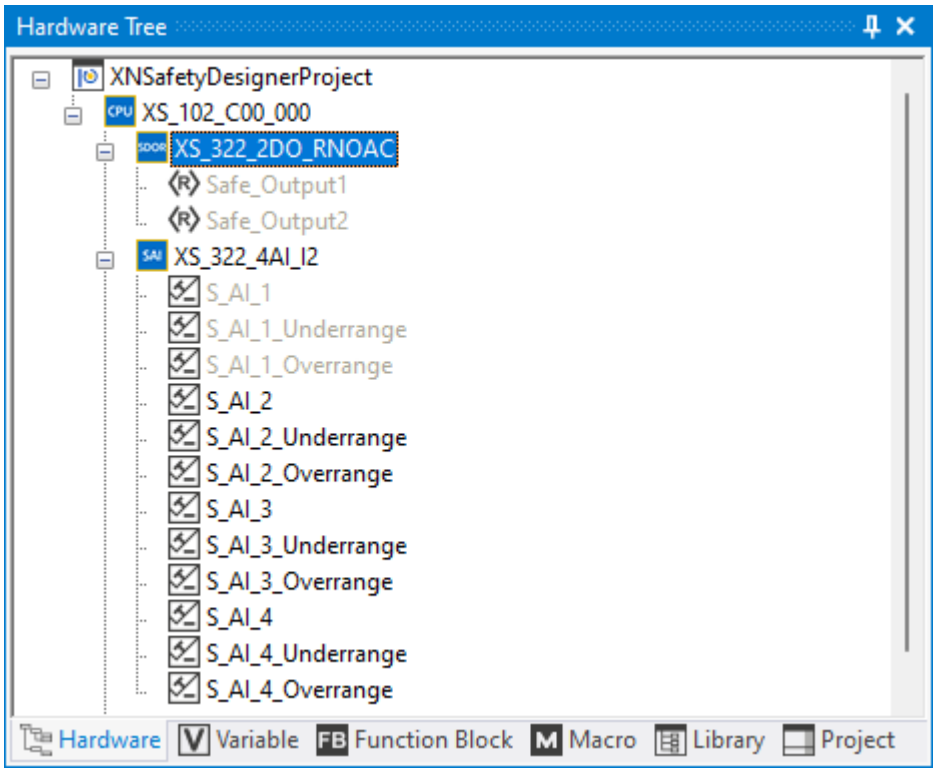
You can also delete or print out the network with the corresponding context menu options.



When you select an element (project, Safety CPU, network, function block instance), the corresponding properties will be shown in the Properties pane.

4.4.2 Hardware Tree

This tree structure will show the topology of the Safety modules together with the corresponding inputs and outputs. This topology can be identified with an upload from the PLC. If this is not possible, you can also create a topology manually by using the context menu → section "Creating the hardware configuration", page 28).



The various sub-elements will reflect the topology of the Safety modules. All modules will have their physical inputs and outputs listed as sub-elements.

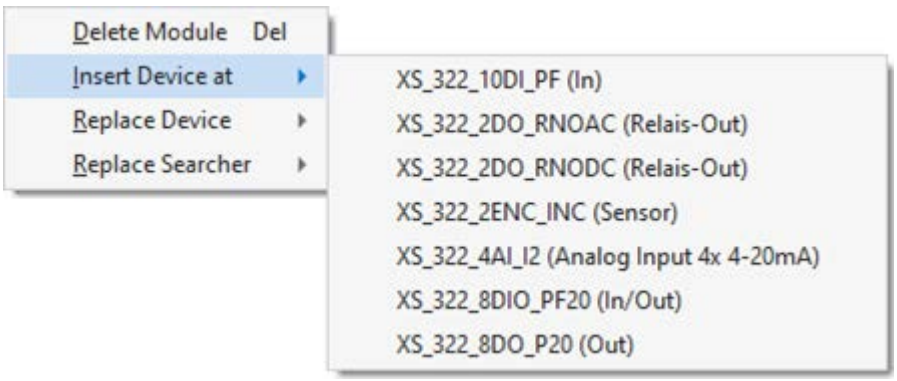
You can rename the various elements (modules and inputs/outputs) by pressing the F2 key or with a long click. Please note that the names you enter must be unique within the project.

Inputs and outputs being used will be indicated with a lighter color (light gray) in the tree.

You can move hardware modules within the tree by dragging and dropping them. The modules can only be moved within the area intended for them, and the new spot must not be occupied already.

You can transfer the properties of the inputs and outputs of a module and their points of use in networks to another module in the tree. To do this, you simply need to drag the input/output with the data to be transferred to the new input or output, as the case may be. Please note that inputs and outputs can only be moved separately.The cursor icon will let you know if moving is allowed (dark rectangle if allowed; red cross if not allowed). The properties (cross-circuit, for example) of the elements that are moved will be transferred to the new inputs and outputs. In addition, the names of the moved elements will be applied (exception: moving within the same module). The points of use of the moved inputs/outputs in the networks will be replaced.Existing connections in the networks will remain intact. AND operators in the input columns will only be relocated if both inputs / outputs are moved.

Moreover, you can replace safe hardware modules of a specific type with a different type. The context menu will show all the modules available for the replacement.

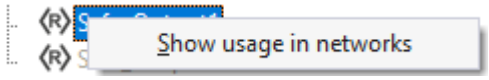


A prompt will appear asking you to confirm that you want to replace the module, since any inputs and outputs not found on the new module will be deleted.

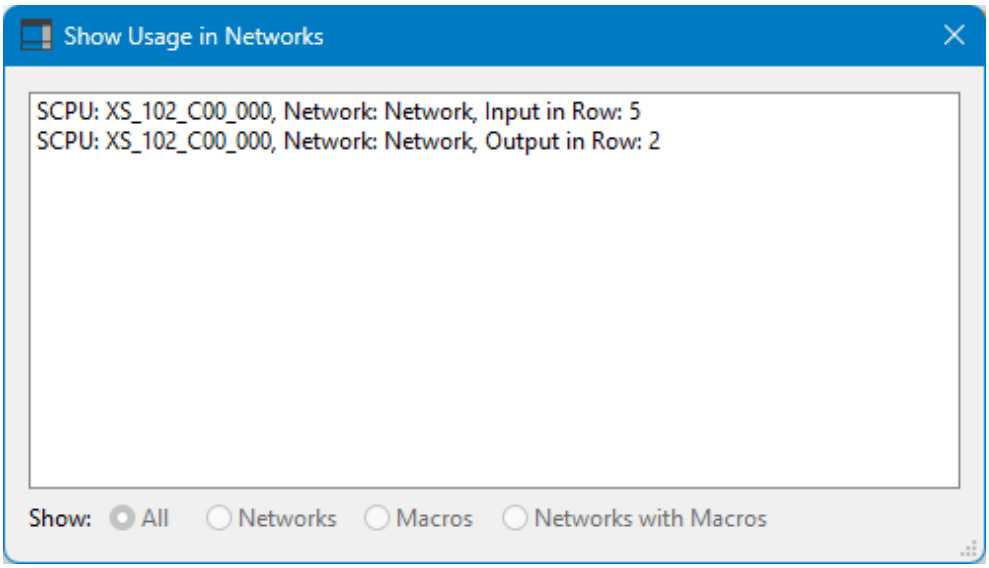
You can drag and drop the safety modules' inputs/outputs to the → section " Editor", page 58 in order to place them as necessary.

Selecting an element will show the corresponding properties in the Properties pane, where you can edit them if they are editable.

The Show usage in networks option in the context menu for inputs and outputs will take you the the points in the network where the corresponding element is placed.

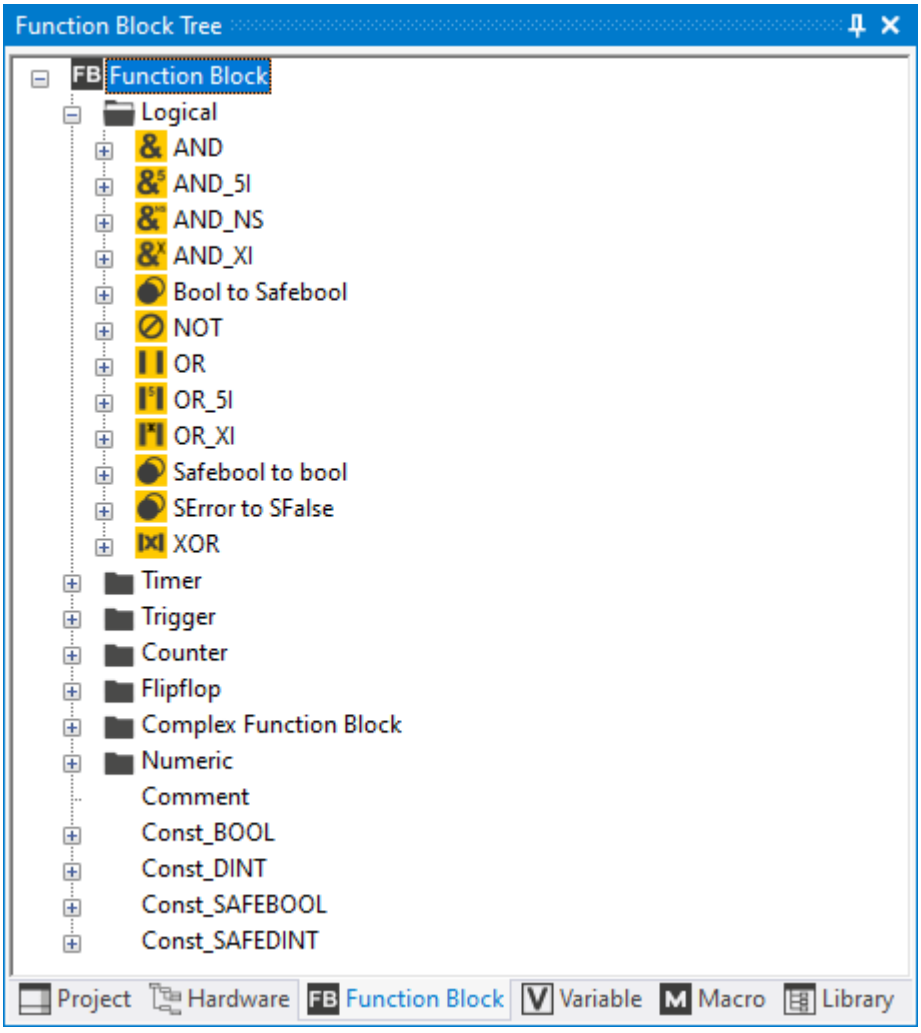


Please note that the context menu will only appear if the corresponding input or output is being used in a network.
If the channel is only being used once, the software will jump immediately to the corresponding point in the network.
If the channel is being used multiple times, the Show Usage in Networks dialog box will appear so that you can select the point of use you want by double-clicking on the corresponding line.

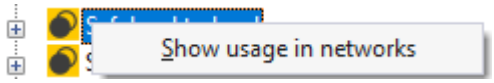


4.4.3 Function Block Tree

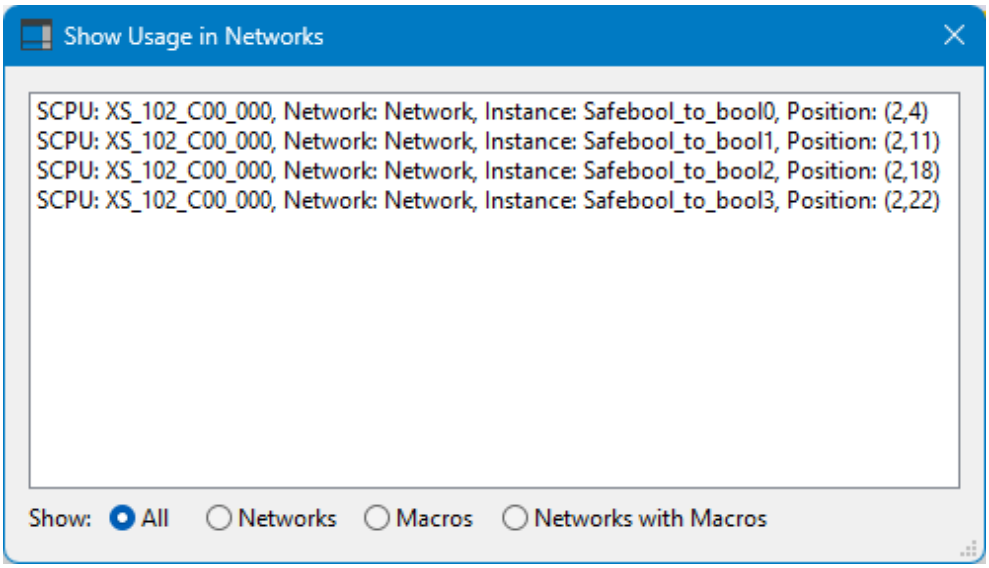
This tree structure contains all the predefined function blocks.
You can drag and drop the elements in this tree to place them as an instance in the network→ section " Editor", page 58.
As shown in the screenshot below, the various function blocks are organized into function-specific nodes (→ section "Function blocks", page 164).



You can use the context menu for the function blocks to jump to the point in the network where the corresponding function is being used.

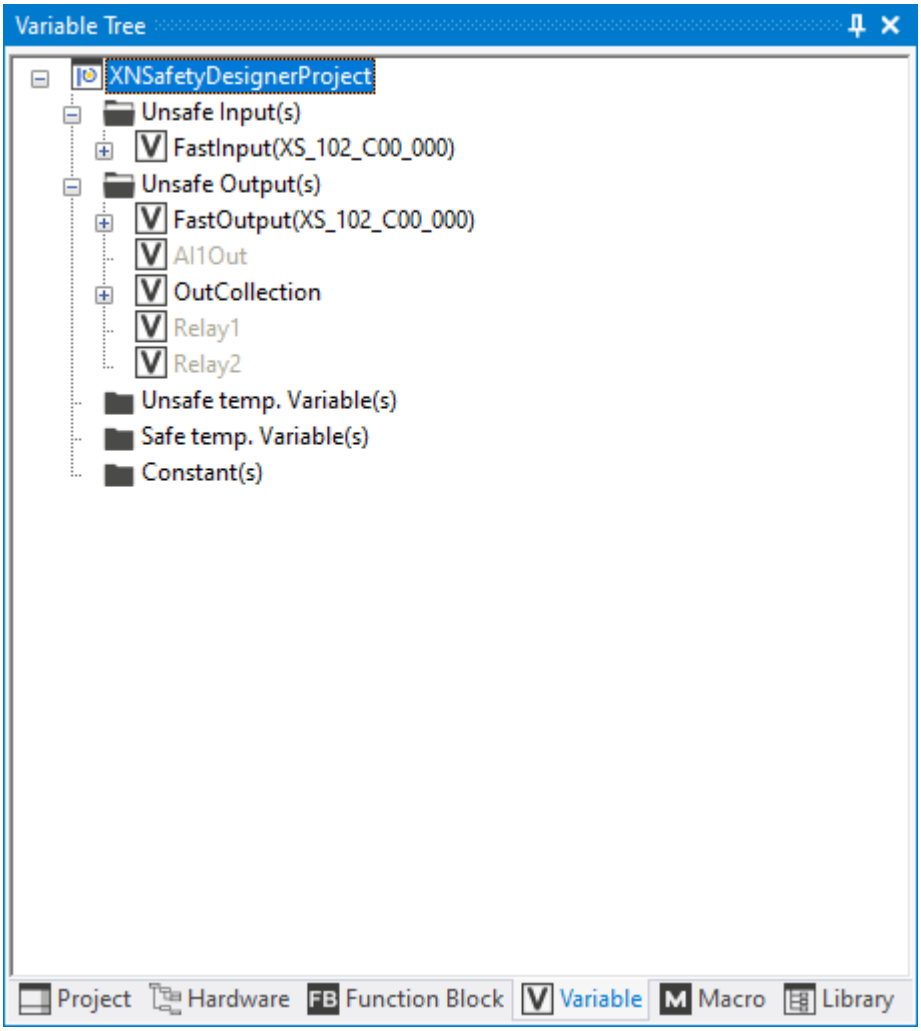


Please note that the context menu will only appear if the corresponding function block is being used.
If the function block is only being used once, the software will jump immediately to the corresponding point in the network.
If the function block is being used multiple times, the Show Usage in Networks dialog box will appear so that you can select the point of use you want by double-clicking on the corresponding line.

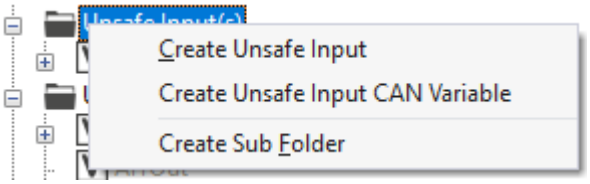


4.4.4 Variable Tree

You can use this tree structure to manage the various unsafe variables, temporary variables, and constants.



The first sub-elements in the structure will be the folders for unsafe inputs and outputs.
These variables are used to transfer data from and to the standard PLC. You can create these inputs/outputs with the context menu or the [Insert] key.

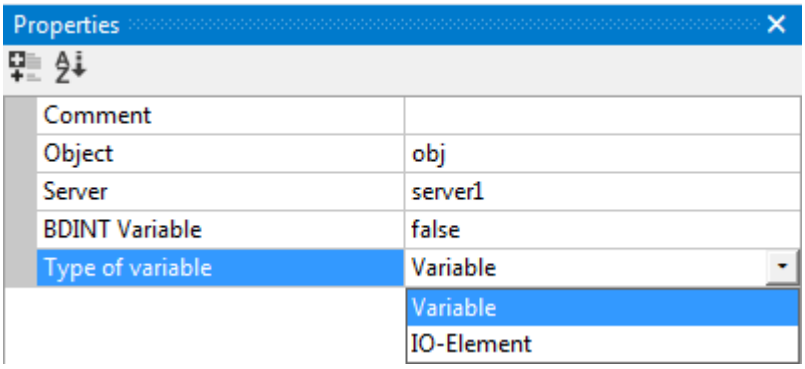


The name entered in the Variable Tree pane will be the symbolic name for the variable and is solely intended to make it easier to identify it in the Network, Tree, and Watch panes.

You can enter the symbolic name you want.
The software will always suggest an obj.server (Objekt.Server name) separated by ".". The first time it is entered, this name will be interpreted and, if possible, applied as an object and server name. If you do not include a "." in the name when creating the variable, you will need to enter the object and server names manually.

If you change the symbolic name for a variable, the object and server names will not change together with the symbolic name change. Please note that the object and server names will not be checked to see if they are valid.

An unsafe variable can support two different types of variables. One is a standard variable, which simply represents an object with the corresponding server in the unsafe application project. The other is an IO-Element variable, which represents an I/O element in the unsafe application project. You can select the variable type you want in the Properties pane for the corresponding variable.

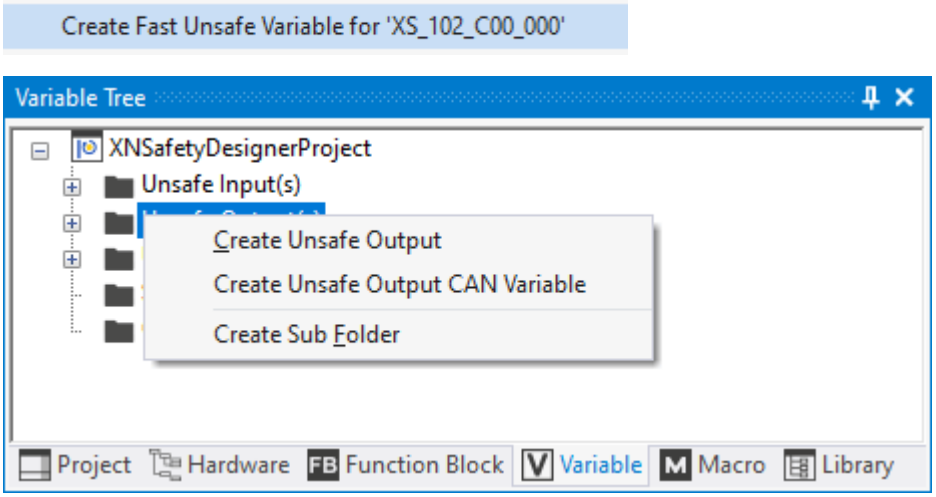


A specific data type is not specified here, and all servers will correspond to a four-byte value. Depending on the corresponding use in a specific function block, the value will be interpreted as a BOOL or DINT value. If it is not possible to create unsafe variables, a corresponding tooltip will be shown:

Tooltip	Description
Variables are not allowed in a stand alone Safety CPU	When a Safety CPU is run in standalone mode, unsafe variables are not supported.

4.4.4.1 Bit tags

You can use the Properties pane to convert the variables you create to a bit variable. When you use a bit variable, 32 individual bits will be created so that they can be placed in the network. Within this context, the bit numbers will be shown with an icon with the corresponding number. In addition, you can use the context menu for each Safety CPU to create a fast 32-bit input and output variable.



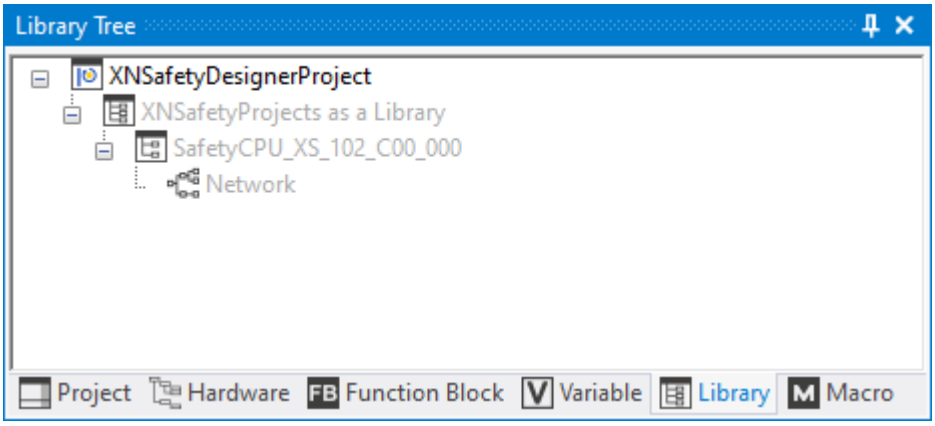
The next sub-elements will consist of the folders for temporary safe and unsafe variables. These variables are flags, meaning that they can be set in a specific section in the network and can be reused later on in the program. Within this context, the system will check whether the temporary variable is first set and then reused in the processing sequence.
You can create these variables with the context menu.
The corresponding names must be unique within the project.The elements can be used as inputs and outputs (configurable in the Properties pane).

Please note that temporary variables are only placeholders in the networks. When the application is created, the placeholders will be removed and a direct command for copying the values between two function blocks will be added. Temporary variables can only be used within a Safety CPU.

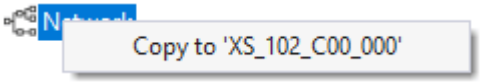
You can place the elements in the network → section " Editor", page 58 by dragging and dropping them.

4.4.5 Library Tree

You can use this tree structure to load other Safety Designer projects as a library. The individual Safety CPUs and their networks will be shown.



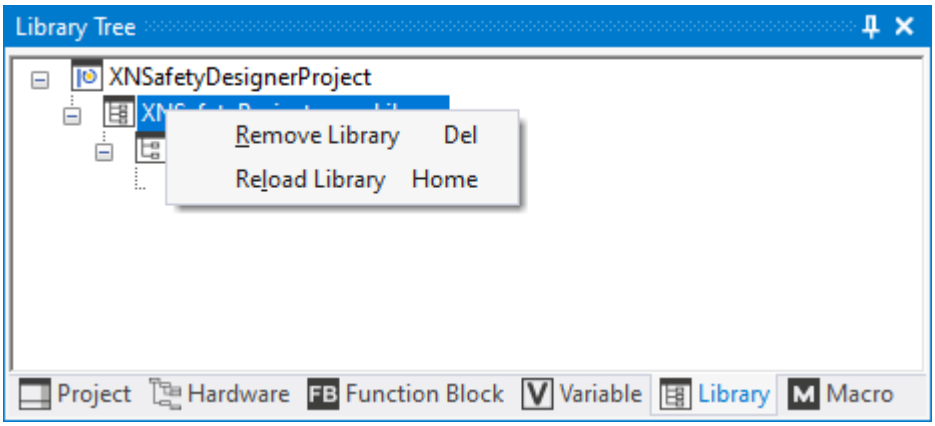
You can use the context menu to copy the networks directly to the Safety Designer project.



When a network is copied, the software will check whether all the corresponding data can be copied correctly. If any conflicts are detected, the corresponding elements in the network will be replaced with placeholders.

Double-clicking on a network name in the tree will open the network editor (in read-only mode). You can then copy elements from the network and paste them to a different network; please refer to → section " Copying, cutting, and pasting elements", page 74

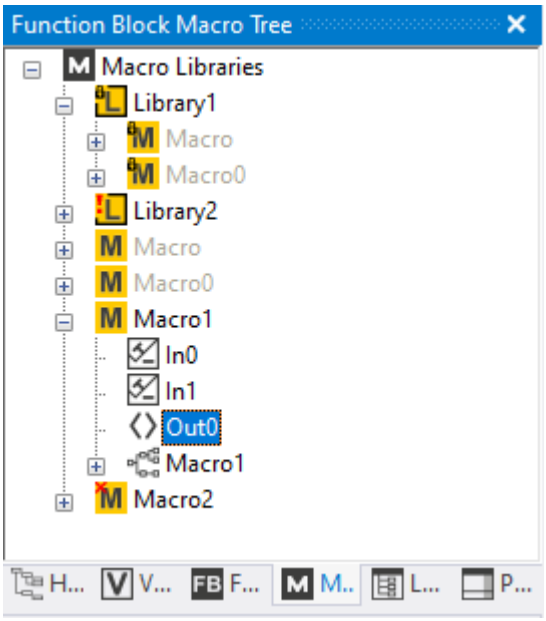
If you modify a project that is loaded in a different project as a library, you can use the Reload Library context menu option to load the modified library.



4.4.6 Function Block Macro Tree

You can use function blocks to create macros and group them together as libraries in this tree structures. Within this context, a macro is a combination of individual function blocks that can be used as an individual function block in networks. Macros can be managed in libraries that are independent of individual projects.

You can drag and drop the macros to place them as an instance in the network → section " Editor", page 58.



The tree will first list the libraries, which will be shown with an icon consisting of an "L" inside a box. This will be followed by the macros used in the current project, with these macros being symbolized with an "M" icon. If the macros are used in a network, the corresponding name in the tree will be shown in gray. In this case, just like with function blocks, you will be able to use the context menu to jump to the corresponding point in the network → section "Function Block Tree", page 46).

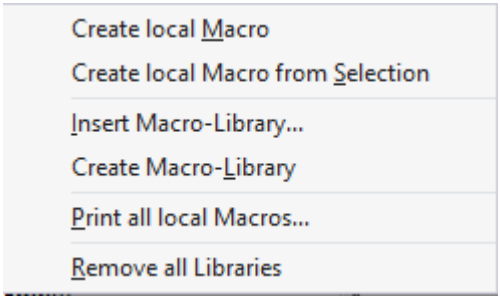
You can drag and drop macros to libraries in order to copy them. If an icon has a small padlock in its upper left corner, this means that the corresponding element is protected and can be used, but not modified.

If you expand a macro, the corresponding inputs and outputs will be shown the same way as for a function block. The macro network icon will then be shown below them. This macro network icon will have the name of the macro and can be opened by double-clicking on it. If the macro is being used, a folder called Instance(s) that lists all the points of use in the networks will appear just like for function blocks. You can either double-click on these entries or use the context menu to jump to the corresponding instances in the networks.

When the icon for a library has a red exclamation mark in its upper left corner, this means that the library has been modified, but has not yet been saved (again).

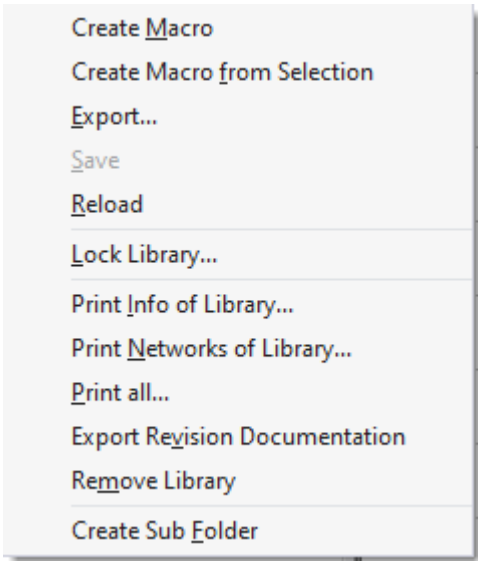
Meanwhile, when the icon for a macro has a red "X" in its upper left corner, this means that the macro is invalid, i.e., that it does not have at least one input and output and accordingly cannot be used.

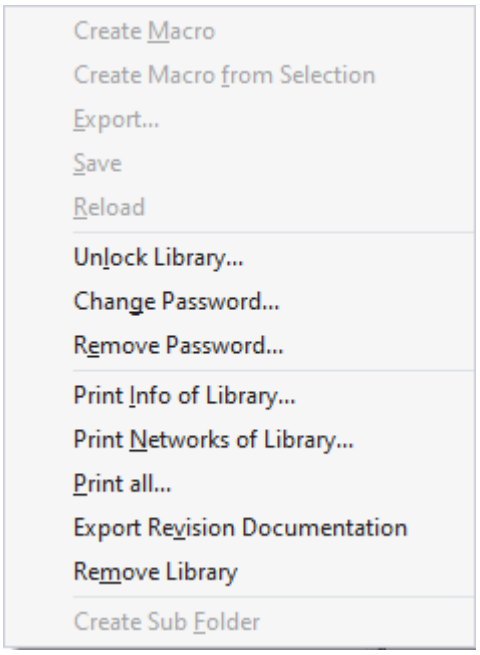
The context menu for the Macro Libraries root element features options for creating macros and libraries. You can load a library from a file with the InsertMacro-Library… option.



Print all local Macros will print out the networks for all the macros belonging to the project. Meanwhile, Remove all Libraries will remove all libraries from the view. In this particular case, a prompt will appear for each modified library to ask whether you want to save it.

The context menu for libraries not only offers options for creating macros, but also the option of saving the library in a new file with "Export..." Once you have exported a library, you will be able to save any changes made to it with "Save".

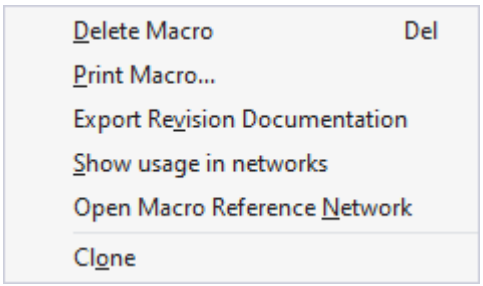




To protect a library against further changes with a password, click on the "Lock Library..." option. Once you do this, the context menu will show additional options – "Unlock Library..." can be used to temporarily unlock the library for modifications, for example. The library will then remain unlocked until you lock it again with "Relock Library" or reload it. There will also be the "Change Password" and "Remove Password" options for password management.

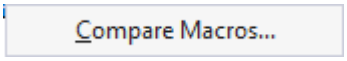
"Print Info of Library" will print the information, comments, version documentation, etc. for the library and all the macros it contains. Meanwhile, "Print Networks of Library" will print out the networks for all macros, while "Print all…" will print all the information with all macro networks. "Export Revision Documentation" will write the version documentation (if any) to an XML file. Finally, "Remove Library" will remove the corresponding library from the view.

The "Print Macro…" option in the macro context menu will print out the macro network, while "Export Revision Documentation" will write the version documentation to an XML file.

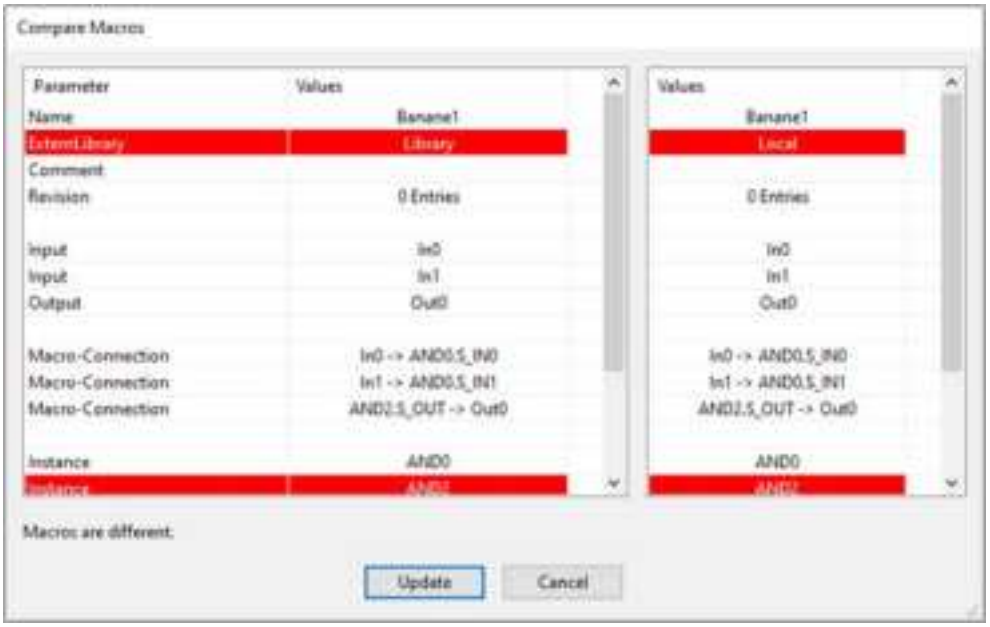


Clone will create a copy of the macro with another name in the project or in a library. Moreover, the macro context menu in libraries will have the Copy to local library option, which can be used to copy the macro to the project.

If you select a macro in the project and a macro with the same name in a library at the same time (by holding down the Ctrl key while selecting them), you will be able to open a special context menu.



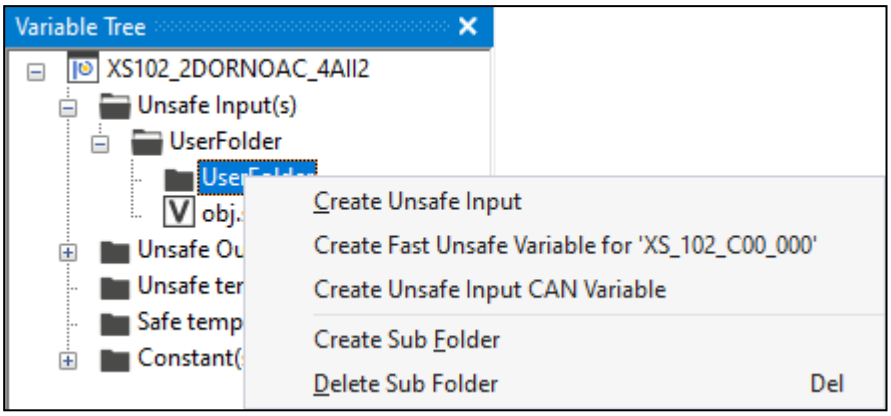
Selecting this option will open a dialog box that will indicate the differences between the two macros in rows highlighted in red.



If the macro is not being used in the project, you can click on Update to replace it with the one from the library.

4.4.7 Subfolders in the Tree view

Folders are used to organize variables, interfaces, function block macros, and networks. You can create subfolders with the Create Sub Folder option in the context menu, which will create them with the **<UserFolder>** or **<LibFolder>** default name as applicable. Subfolders can contain objects from the folder to which they belong ("Unsafe Input(s)," for example), as well as additional subfolders.



→ The subfolders of a folder must have different names.

→ When you delete a subfolder, all the elements in it, as well as all its subfolders with their own elements, will be removed from the project. Please note that if an element is being used, you will not be able to delete the corresponding subfolder.

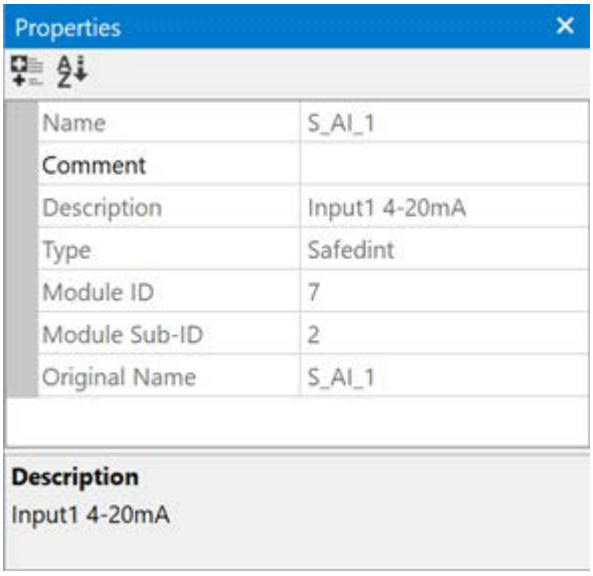
Subfolders will feature the context menu of the folder to which they belong – for example, a subfolder in the "Unsafe Input(s)" folder will have all the context menu options for unsafe inputs. You can move both subfolders and elements in a subfolder by dragging and dropping them.

→ If the target folder already contains a subfolder of the same name, you will not be allowed to move the original subfolder there.

→ Elements from a folder (unsafe inputs, for example) that are placed in different subfolders must still have a unique name.

4.5 Property Browser

The Properties pane will show the properties of the relevant element based on your selection in the Project Tree pane or the network editor. These elements will either be read-only or editable. The Properties pane is the central tool for setting parameters.



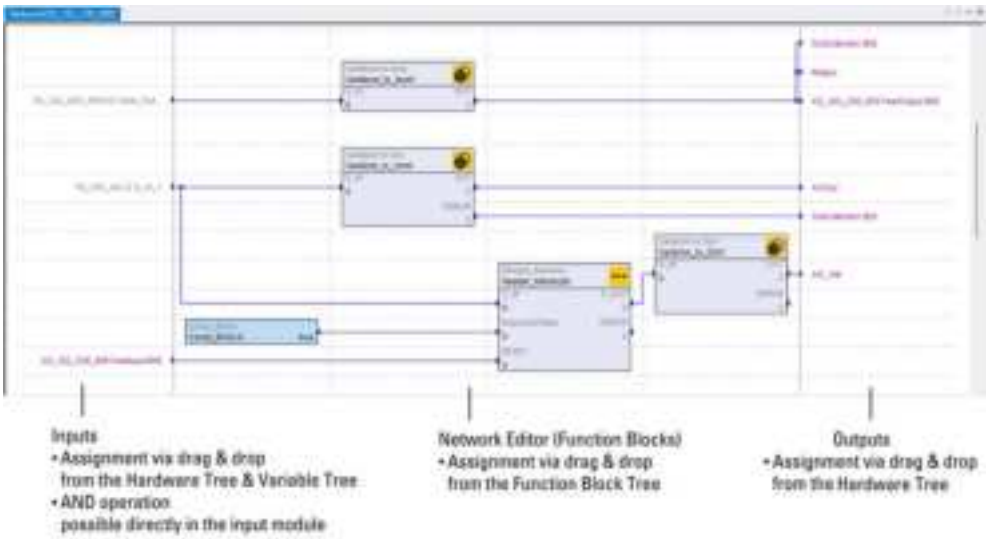
When in online mode, all settings will be grayed out (disabled).

4.6 Editor

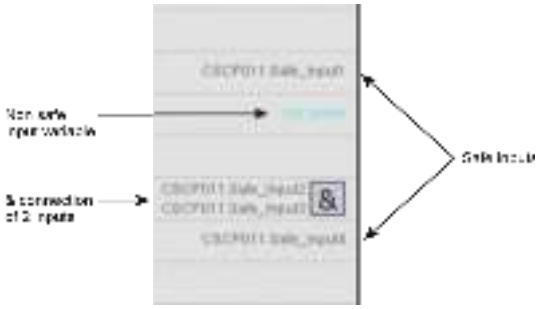
To create the sequence code for the Safety CPU, you will need to use the drag and drop function in the network editor. To do this, you will need to logically connect Safety module inputs and outputs with certified function blocks.

The network editor is subdivided into three areas:

- Column for inputs (left side of editor)
- Network editor (area between the input and output columns)
- Column for outputs (right side of editor)



4.7 Column for inputs



You can place inputs, outputs, unsafe variables, and bit variables in the individual rows.

The various elements can be added by dragging and dropping them from the Hardware Tree or Variable Tree pane. You can also connect two safe inputs with an AND operator in a row (in order to create a dual-channel emergency stop configuration, for example). To do this, you would need to add an input by dragging and dropping it and then drag a second input to the corresponding row. This second input can come either from the hardware tree or from the input column.This type of operator connection in the input column will be evaluated directly in the respective input module so that only the computed operator result will be transmitted to the Safety CPU. In other words, inputs connected with an AND operator must come from the same Safety module.

→ When using an AND operator in the input column, the two inputs will be connected to each other with the AND function block by default

→ If you need an operator with discrepancy time monitoring, do not use this AND operator. Instead, you will need to create your own operator with the Equivalent function block. More specifically, you will then be able to use the Properties pane to configure the AND operator for the input column with discrepancy time monitoring. Within this context, a distinction can be drawn between connections using an AND function block and connections using an Equivalent function block.

All inputs, outputs, and variables can be placed in the input column multiple times.

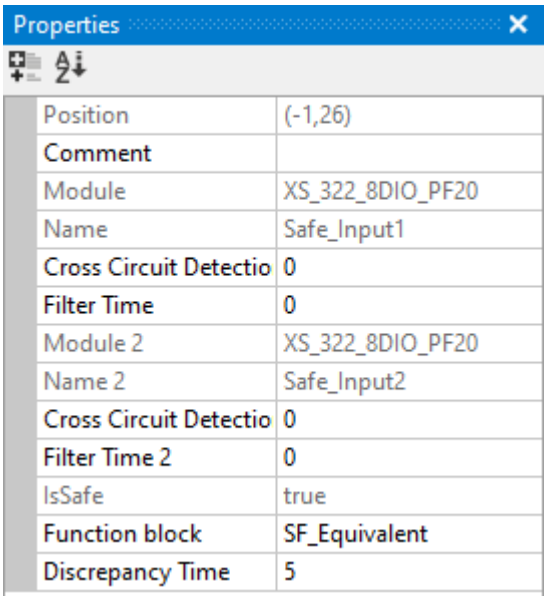
When you place an output in the input column, this means that the current output state will be used as an input value.

You can drag and drop existing elements from the Hardware Tree or Variable Tree pane while holding down the [SHIFT] key to replace existing inputs in the network. Please note that you will only be able to replace equivalent elements (unsafe variables with unsafe variables, for instance). The connections to the input will remain intact.

You can edit the properties of elements in Properties.

Equivalent operator in the input column

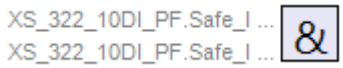
You can use an AND operator with an Equivalent function block (operator with discrepancy time monitoring). The type of AND operator can be configured in the Properties pane for the AND operator. The Function block property will contain a combo box that you can use to select between an AND function block and an Equivalent function block (default setting: AND). If you select the Equivalent block, you will additionally need to configure the discrepancy time.This time is configured with the Discrepancy Time property, which will only be visible for the Equivalent block. For this property, the default value will be 0, while the lower limit will be 0 ms and the upper limit will be 86400000 ms. This discrepancy time is the maximum delay that can occur when the two inputs are set; if it is exceeded, the Equivalent function block will enter its error state.



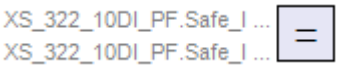
When you select the Equivalent function block, the SF_Equivalent function block will be used for processing. You can view the individual inputs and outputs of this function block in the Watch pane while in online mode. Error states will be signaled at the DiagCode output and can be read from the description for the function block. Moreover, errors can be reset by resetting the corresponding safe inputs.

Different symbols are used to represent these operators in the network:

AND:



Equivalence:

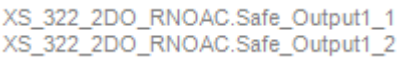


4.8 Column for outputs

The various elements can be added by dragging and dropping them from the Hardware Tree or Variable Tree pane.
A specific SafetyOutput can only be placed once in an output column in a Safety project. In addition, using unsafe variables in the output column is allowed, but they can only be placed once as well. This also applies to writing interfaces and bit variables, which can also be used but only placed once.

For safe outputs, you can set up an AND operator with an unsafe enable signal. Within this context, this "enable signal" refers to an unsafe variable that is not part of the Safety project.

Relay outputs will be shown in two rows, which is intended to illustrate the fact that the outputs are doubled in the actual hardware.

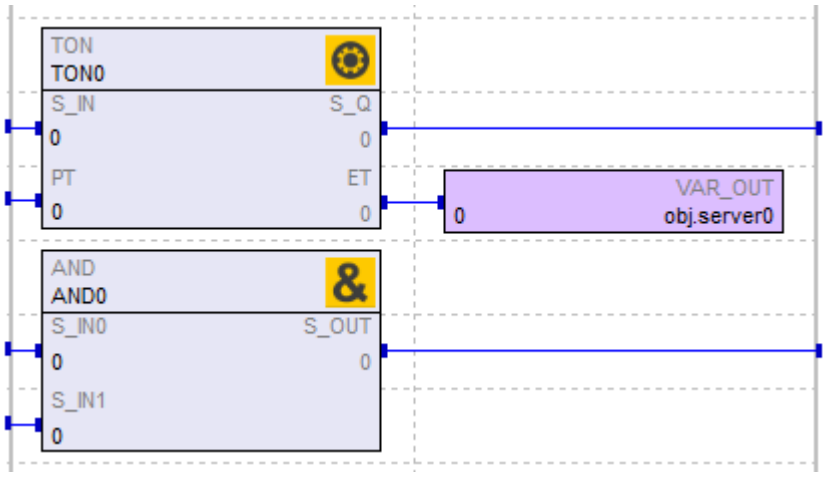


You can drag and drop existing elements from the Hardware Tree or Variable Tree pane while holding down the [SHIFT] key to replace existing outputs in the network. Please note that you will only be able to replace equivalent elements (unsafe variables with unsafe variables, for instance).
The connection to the output will remain intact.

The properties of the various elements can be modified in the Properties pane.

4.9 Network editor

The network editor shows the connected elements of a safe network.



The network editor features the following components:

- Background grid (for improved manageability), with each grid cell corresponding to one row and one column.
- Function block instances
- Constants
- Tags
- Temporary variables
- Bit tags
- Connections
- Comments

4.9.1 Network elements

In the network editor, the following element represents an instance of a function block. These elements can be added by dragging and dropping them from the Function Block Tree pane. Within this context, safe inputs and outputs can be easily identified in that they will always start with an S.

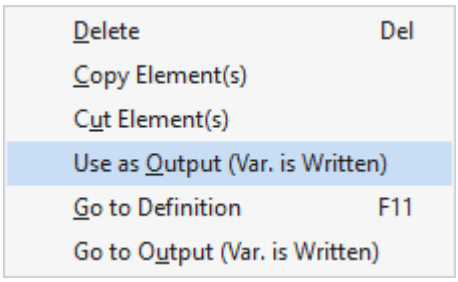
In addition, you can place blocks for constants, unsafe variables, and temporary variables. However, these blocks will not have a header section. Moreover, blocks for constants will only have one single output (and can be used as an input parameter for function blocks).

You can create blocks for unsafe variables either as input variables or output variables in the tree. However, safe variables cannot be created. The various elements can be added by dragging and dropping them from the Variable Tree pane. Unsafe variables will be taken from the standard PLC's unsafe application. In addition, you can set the values of unsafe input variables in the unsafe application. Unsafe input variables can only be connected to non-constant unsafe inputs on function blocks, with an example being the Activate input on complex function blocks. The system ensures that a safe output cannot be set exclusively by an unsafe variable. Finally, the unsafe variables used will be initialized with a value of 0 in the firmware.

4.9.1.1 Bit tags

You can create blocks for unsafe bit variables either as input variables or output variables in the tree. Meanwhile, these variables can be 32 fast bit variables in a Safety CPU or 32 bit variables. The various elements can be added by dragging and dropping them from the Variable Tree pane.

You can use blocks for temporary variables as an output or input. To define which one you want, you will need to use the context menu



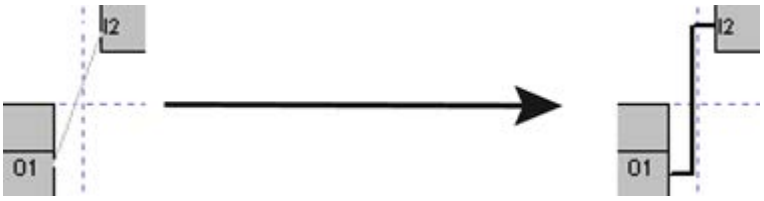
for the instance in the network or the Properties pane after selecting the instance. The various elements can be added by dragging and dropping them from the Variable Tree pane.

All inputs will be found on the left side of the instance, while all outputs will be found on the right side. The height of the header or of an input or output will be the same as that of one row in the editor. Finally, unsafe variables, constants, and temporary variables will be shown in a different color (constants = blue; unsafe variables = violet; temporary variables = green) so as to make it easier to distinguish between elements.

You can drag and drop existing elements from the Hardware Tree or Variable Tree pane while holding down the [SHIFT] key to replace existing function blocks for unsafe variables and temp. variables in the network. Please note that you will only be able to replace equivalent elements (unsafe variables with unsafe variables, for instance). The connection to the element will remain intact.

You can draw connections between the various network elements with the mouse.

All you have to do is set the start and end points for the connection and the actual line will be automatically generated by XN Safety Designer.



- Please note that you will only be able to draw connections if the data types (e.g., SAFEBOOK) of the end points match or are compatible.
- Each element in the output column can only be connected once.
- Inputs of instances can only be connected once.
- Intersections between connections are allowed. Branches will be indicated with a node.
- The cursor icon will let you know if you can draw a specific connection or not:

Valid connection:

Invalid connection:

You can add the Comment element by dragging and dropping it from the Function Block Tree pane. You can place the comment field anywhere you want as long as the corresponding position is not occupied already. The cursor icon will let you know if you can add the comment or not (dark rectangle if allowed; red cross if not allowed).

4.10 General instructions for using the editor

The following information explains how to use the editor.

When an attempted action is invalid, this will be indicated by a cursor icon, a color, or an error message in the Output pane.

When dragging and dropping and moving elements, an icon will be shown to indicate whether you can actually place the element where you want.

4.10.1 Filling the input column

You can drag Safety module inputs/outputs, unsafe variables, and temporary variables from the Hardware Tree / Variable Tree pane and drop them in individual cells. The cursor icon will let you know if you can place the element in the cell you want. The cell will assume a specific color based on the type of element placed (input/output, unsafe variable, temporary variable) (inputs/outputs = gray; unsafe variables = violet; temporary variables = green).

When you select a cell, the corresponding properties will be shown in the Properties pane. Elements are configured fully in this pane.

4.10.2 Filling the output column

You can drag Safety module outputs, unsafe variables, and temporary variables from the Hardware Tree / Variable Tree pane and drop them in individual cells. The cursor icon will let you know if you can place the element in the cell you want. The cell will assume a specific color based on the type of element placed (output, unsafe variable, temporary variable) (outputs = gray; unsafe variables = violet; temporary variables = green).

Please note that each physical output should only be placed once in a Safety project.

When you select a cell, the corresponding properties will be shown in the Properties pane. Elements are configured fully in this pane.

4.11 Working in the network editor

4.11.1 Adding network elements

You can drag network elements from the Function Block Tree / Variable Tree pane and drop them in the network editor. Function block instances are added from the Function Block Tree pane, while variables and temporary variables are added from the Variable Tree pane.

A dashed-border rectangle will show the size of the element you are trying to add, while the cursor icon will let you know if you can add the element where you want.

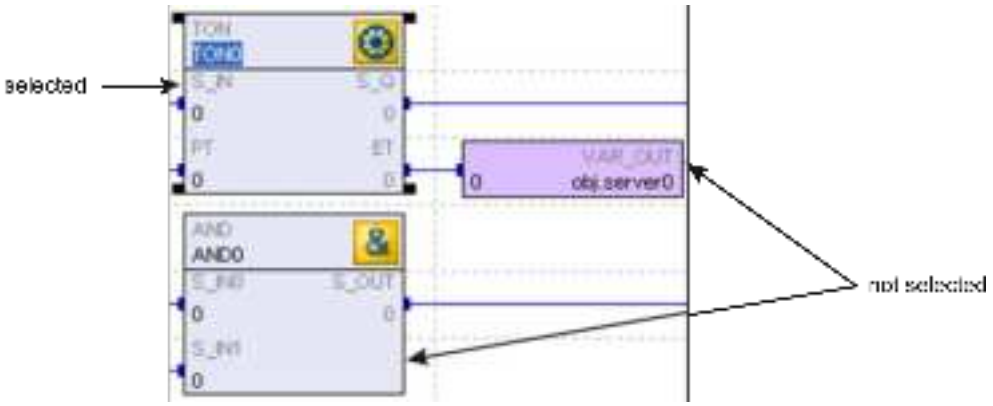
Representation

- Red cursor (cannot be added where you want)
- Gray rectangular cursor (can be added where you want)

You will only be able to place the element if there is enough space available in the location you want (the inputs and outputs of a function block will each take up their own row once the block is placed). If there is not enough space, you will need to manually move the objects you have already added previously.

4.11.2 Selecting network elements

To select a network element, click on it. When you select an element, solid rectangles will be shown at the corners in order to indicate that the element is selected.



You can select multiple elements at the same time. To do this, either use the [SHIFT] key or draw a rectangle around the elements you want by clicking and holding down the mouse button as you draw the rectangle. The Properties pane will show the selected element's properties. Please note, however, that if you select multiple elements, only the properties that they have in common will be shown.

4.11.3 Moving network elements

You can move selected network elements (instances, elements in the input and output columns) in the network editor. However, connections cannot be moved directly, and will only move if a corresponding element is moved. When you move an element, a dashed outline will be used to indicate the current position. Please note that when moving elements, you will have the same limitations outlined in "Adding network elements."

4.11.4 Deleting network elements

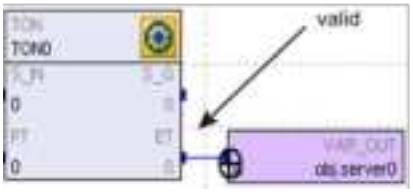
You can delete selected network elements with the context menu or the [DEL] key. When you do this, all the corresponding connections will be deleted as well.

4.11.5 Configuring network elements

You can modify the parameters for the individual selected network elements in the Properties pane. Parameters that cannot be edited will be shown in gray. For editable parameters, you will be able to configure the corresponding settings either by entering the value you want or by selecting an option from a list with predefined options.

4.11.6 Drawing connections

You can draw connections with the mouse. All it takes is two mouse clicks and releasing the mouse button between them. The cursor icon will let you know if you can draw the connection you want – in order to be able to draw a connection, the data types of both end points need to match (SAFEBOOL to SAFEBOOL, for example) or be compatible with each other (CONST to BOOL, for example). The actual connection line will be drawn automatically and cannot be manually changed.



Connections to the input column and output column can pass through several editor columns if the area in between is not blocked by a network element.

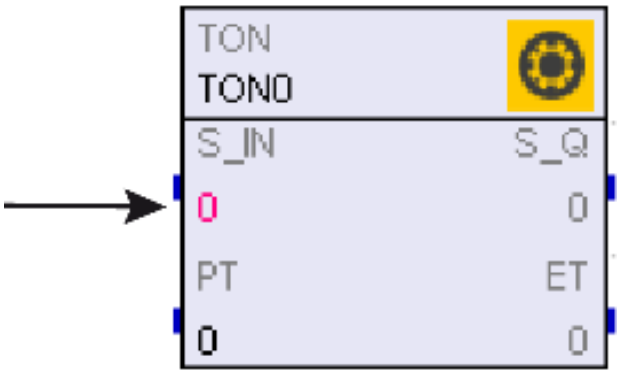
if you try to draw an invalid connection, a tooltip may be shown (at the pin that you are trying to use for the connection).

Following is a list of possible tooltips and what they mean:

Tooltip	Description
Connection is crossing another connection or an instance	The new connection intersects another connection or passes through an instance of a function block.
Connection with a dummy module	You are trying to generate a connection to a dummy element.
Input of instance is connected	The input of an instance of a function block is connected already.
Element in output column is connected	The element in the output column is connected already.
Connection between identical network elements (e.g. output column with output column)	You are trying to connect two identical elements (e.g., an element in the output column with an element in the output column).
Connection between different data types	You are trying to connect two different data types (e.g., bool with safebool)
Value of constant element is outside of allowed borders	The value of a constant falls outside of the allowed limits.
Sense of direction of connections must be left to right	The connection must always run from left to right.
Connection between two constant elements	You are trying to connect constants to constants.

The input parameters of a function block must always be connected
(this means that you will need to connect constants to any input parameters you do

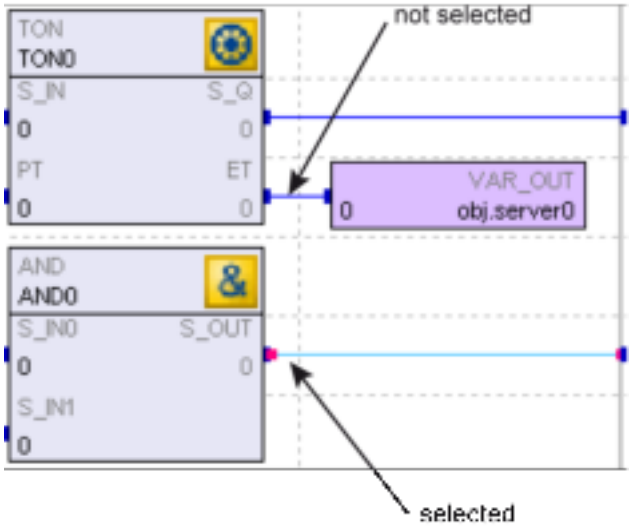
not need).
Meanwhile, connecting outputs is optional.
If a required parameter is not set, this will be indicated in color in the network element.



4.11.7 Selecting connections

To select a connection, click on it.

When you select a connection, a solid rectangle with a different color will be shown at each end.



4.11.8 Deleting connections

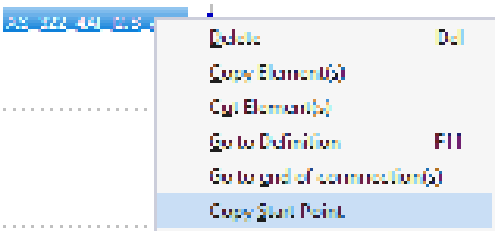
You can delete selected connections with the context menu or the [DEL] key.

4.11.9 Global connections

Global connections consist of a connection with a label name. Connecting lines are not drawn for these connections, and the connecting points are instead labeled with the label name. This provides greater clarity and manageability in complex networks.

Creating global connections

There are several ways to create and connect global connections.

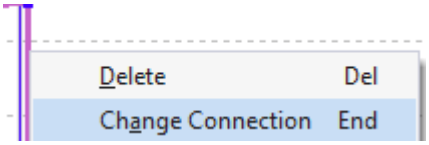


You can use the Copy Start Point and Paste Connection options to easily add connections to the connection points you want.

Create Connection will create a temporary global connection at the selected point.

All other connections extending from this point will be created as global connections. You can create the connections with the Copy and Paste options or by drawing a connection.

You can also change the connection type by clicking on the Change Connection option in the context menu for a connection.



When you delete a global connection, a prompt will appear asking whether you want to delete all connections with the same label name.

4.11.10 Adding comments

You can drag comments from the Function Block Tree pane and drop them in the network editor. A dashed-border rectangle will show the size of the comment you are trying to add, while the cursor icon will let you know if you can add the element where you want.

Representation

- Red cursor (cannot be added where you want)
- Gray rectangular cursor (can be added where you want)

4.11.11 Selecting comments

To select a comment, click on it. When you select a comment, solid rectangles will be shown at the corners in order to indicate that the comment is selected.

4.11.12 Moving comments

You can move selected comments in the network editor. When you move a comment, a dashed outline will be used to indicate the current position. Please note that when moving comments, you will have the same limitations outlined in → " Adding comments", page 73

4.11.13 Deleting comments

You can delete selected comments with the context menu or the [DEL] key.

4.11.14 Resizing comments

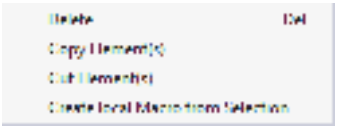
You can resize selected comments.
To do so, move the cursor over the solid rectangle on the right lower corner of the comment box – a double arrow for resizing the box will appear.
Now simply click and drag the mouse to resize the comment. A dashed-border rectangle will show the new size (please note that if you attempt to drag the mouse into an invalid area, the dashed-border rectangle will be limited to a valid size instead).
When you release the mouse button, the comment will be resized to the size of the dashed-border rectangle.

4.11.15 Entering comment texts

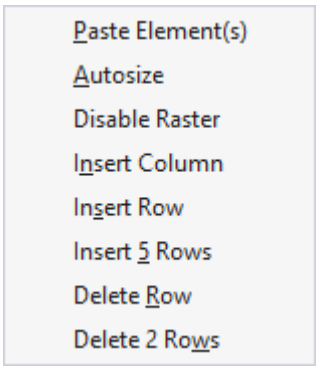
You can edit the text in the selected comment in the Properties pane or by pressing the [F2] key while the comment is selected in the network.

4.11.16 Copying, cutting, and pasting elements

You can copy selected elements from a network with the context menu or the [CTRL] + [C] keyboard shortcut. You can also cut selected elements from a network with the context menu or the [CTRL] + [X] shortcut. You can also use the copy command in library networks.



You can then paste the copied or cut elements with the context menu for the network or with the [CTRL] + [V] shortcut.



The elements being pasted will move together with the cursor and be shown with a dashed border. The cursor icon will let you know if you can paste the element where you want. Elements that are no longer unique after being pasted (for example, when a safe output from a Safety module is placed a second time) will be inserted as placeholders. These placeholders will be shown in orange.

4.11.17 Replacing placeholders

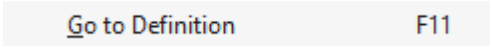
You can drag and drop elements from the Hardware Tree / Function Block Tree or Variable Tree pane to replace placeholders. The cursor icon will let you know whether the corresponding placeholder can be replaced.

4.11.18 Replacing existing elements

You can drag and drop elements from the Hardware Tree or Variable Tree pane while holding down the [SHIFT] key to replace existing elements in the network. The cursor icon will let you know whether the corresponding element can be replaced. Please note that function block instances (for the AND function block, for example) cannot be replaced.

4.12 Directly jumping to the definition of an element

You can use the context menu or press the [F11] key to jump to the definition of an element in the corresponding tree.



You can use this command with the elements in the input column, the elements in the output column, and any instances of function blocks.

For the elements in the input column / output column, either the channel of a hardware module or a variable / temp. variable will be selected in the corresponding tree.

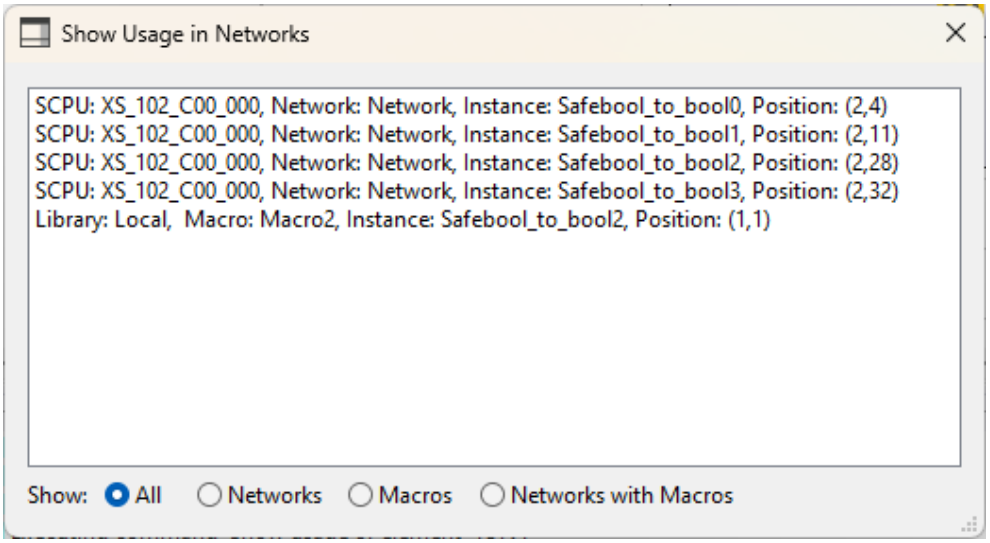
For the elements in the networks, either the function block or the variable / temp. variable will be selected in the corresponding tree based on the element type in question.

4.13 Directly jumping to the start points / end points of a connection

You can use the context menu to jump to the start point or end point of a connection.

You can jump to the start point of a connection from the output column or from the inputs of function blocks.

Meanwhile, you can jump to the end point of a connection from the input column or from the outputs of function blocks. If there is only one single connection, the system will immediately jump to the corresponding point in the network. If there are multiple connections, a dialog box where you can select the one you want will be shown (to select the connection, simply double-click on it).



In order to avoid having to constantly open the context menu, you can alternatively use the arrow keys to navigate through the various connections. The left and right arrow keys will take you to the start and end of a connection respectively, as well as from the output to the input side of an instance and vice versa. In addition, you can press the up and down arrow keys to navigate through the individual connections on the screen and through the network accordingly.If you get to the input or output column, the up and down arrow keys will take you to a different element.

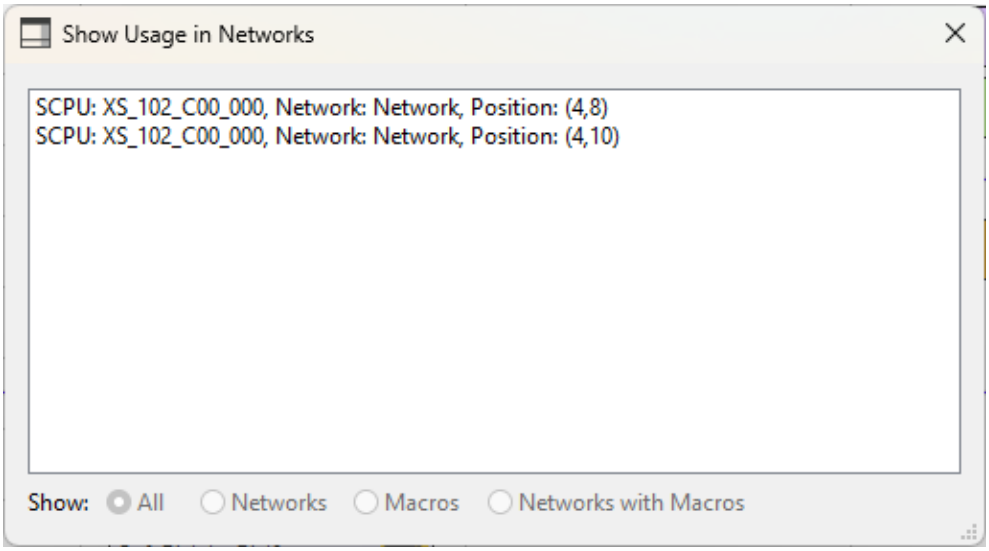
If you want to start from a connection of an instance, simply select it with the mouse. In the case of constants and other single-row elements, you will be able to navigate through the rest of the network right away; otherwise, the down arrow key will take you to the section with the connections, from where you can use the arrow keys to keep navigating accordingly.

4.14 Directly jumping to temporary variables

You can use the context menu to jump to the writing or reading side of temporary variables.

You can jump to the writing side with a reading temp. variable in the input column or in the network.

Meanwhile, you can jump to the reading side with a writing temp. variable in the output column or in the network. If there is only one single point of use, the system will immediately jump to the corresponding point in the network. If the variable is used multiple times, a dialog box where you can select the position you want will be shown (to select the position, simply double-click on it).



Just like with the connections in the network, you can use the arrow keys to move from the writing side to the reading side of a temp. variable and vice versa. Simply keep navigating on the side without a connection and the system will jump to the counterpart (this works both in the instance area and in the input and output columns). If there are multiple reading elements, the dialog box shown above will appear as well.

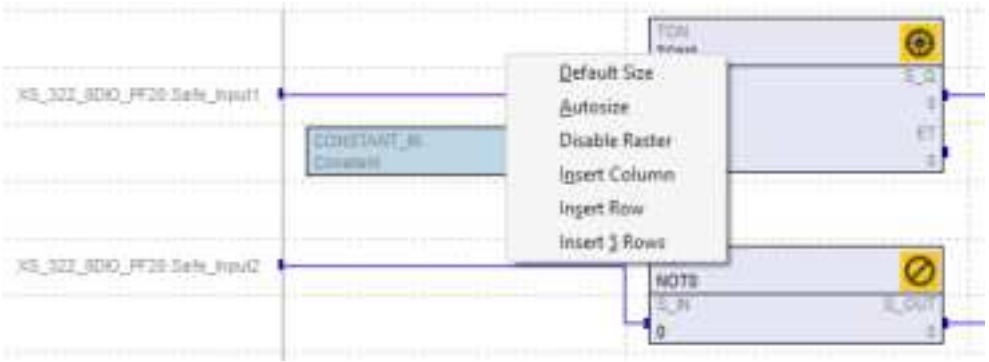
4.15 Pan and zoom

You can use the pan and zoom functions in any network. To select them, use the icons in the Edit toolbar, → section "Toolbars", page 38.

Zoom	When the zoom function is on, you can click the left mouse button or press the + key to zoom in or you can press the - key to zoom out.
Zoom All	When you use this function, the zoom level will be adjusted in such a way that the entire network is shown on a single page.
Zoom Selection	When you use this function, you can draw a rectangle to select the zoom area you want. Once you draw the rectangle, the program will zoom into the area you selected.
tbd	When you use this function, clicking and holding down the left mouse button while moving the mouse will move the area being shown around.

4.16 Changing the column width

To change the column width for a network, you can use the toolbar or the context menu in the network. In addition, different networks can have different column widths. Within this context, you can select between the Default Size and Autosize, options (the default size will be the minimum width for a column).When you select the Autosize option, all the texts in the network will be taken into account so that no texts are abbreviated in the default view (with zero zoom).



4.17 Error messages when trying to add an element to a network

When adding an element to a network, the cursor icon will let you know if you can add the element. If adding it is not allowed, a tooltip will appear to let you know why.

Following is a list of possible messages and what they mean:

Info text	Description
Element is only allowed in input column	The element can only be added to the input column.
SafeDint elements are not allowed in this version of Safety CPU	The element contains the SafeDint data type.
Position is not empty or not allowed	The location in the network is already occupied, or using it is not allowed.
Element is different to dummy element	The element does not match the dummy element being replaced (e.g., different data type).
Wrong second module for AND-Connection	If there is an AND operator in the input column, the hardware modules do not match (elements must be from the same module).
Element is only allowed in input or output column	The element can only be added in the input or output column.
Element is not allowed in input or output column	The element cannot be added in the input or output column.
Function blocks with SafeDint data are not allowed in this version of Safety CPU	The function block contains the SafeDint data type.
Replace element is different to placed element	The replacement element does not match the element being replaced (e.g., different data type).
Output variable can only used once	An output variable is set already and can only be set once.
Temp. variable can only used once as output	A temp. variable is set already and can only be set once.
Variables are not allowed in a stand alone Safety CPU	Variables cannot be used in a standalone Safety CPU.
Variable is not allowed in input column	The variable is not allowed in the input column.
Variable is not allowed in output column	The variable is not allowed in the output column.
Temp. Variables can only used in one Safety CPU	Temp. variables can only be used within a Safety CPU.
Write interface variable can only used once	A writing interface variable is set already and can only be set once.
Element is only allowed in output column	The element can only be added in the output column.
Output of a module can only used once as output	An output of a module is set already and can only be set once.
Outputs of a module can only used by 2 Safety CPUs	The outputs of a module are only allowed to be used by two different Safety CPUs.
Multi modules are not allowed in this version of Safety CPU	Multi-modules cannot be used with this Safety CPU version.
A Safety CPU can only used by one multi module	Only one multi-module can be used per Safety CPU.

Info text	Description
An unsafe variable can only used with object/IO-Element and server	An unsafe variable can only be used if the object name / I/O element name and the server name are set.
HGW Modules only allowed in own Safety CPU	HGW module elements can only be used in their own Safety CPU.
AND-Connections are not allowed for node channels	AND operators are not allowed for node modules.
Element is not allowed in macro network	The element cannot be added to macro networks.
Macro must have at least one input and one output	A macro must have at least one input and one output.
Constant variables are not allowed in this version of Safety CPU	Constants are not allowed in this Safety CPU version.
Module only allowed in one Safety CPU	Drive modules are allowed only in one Safety CPU.
Write enable of a section can only used once	The write enable flag for the parameter list can only be used once.
Only BDINT or CAN sub variables are allowed	Only three sub-variables of BDINT and CAN variables are allowed.
Parameters from parameter list are not allowed in this version of Safety CPU	Parameters from parameter lists are not allowed in this Safety CPU version
Function blocks with Array data not allowed in this version of Safety CPU	Function blocks with arrays are not allowed in this Safety CPU version.
Arrays not allowed in this version of Safety CPU	Arrays are not allowed in this Safety CPU version.

4.18 Settings in the project

4.18.1 Temporary Operation Time

Unit	Minutes (5 minutes to 8 hours; default: 1 hour)
Description	The time that the Safety CPU will run in Temporary Operation Mode. Once this time elapses, the Safety CPU will shut down if there has not been a verification.
Setting location	Root element of hardware tree, project tree, and variable tree

4.18.2 Max. Cycle Time

Unit	Milliseconds (1 ms to 50 ms, default value 3 ms)
Description	Monitored cycle time of a Safety CPU.
Setting location	Safety CPU module(s) in the hardware tree and project tree

4.18.3 Maximum Transmission Time

Unit	Milliseconds (1 ms to 100 ms, default value 10 ms)
Description	This value is used to check the maximum age of transmitted information. The value specifies the maximum bus transmission time – the transmitter and receiver cycle times are added to calculate the actual maximum age.
Setting location	Safety modules in the hardware editor or safe inputs and outputs in the graphical editor

4.18.4 Maximum Transmission Time (Output as Input)

Unit	Milliseconds (1 ms to 100 ms, default value 10 ms)
Description	This value is used to check the maximum age of readback outputs. The value specifies the maximum bus transmission time – the transmitter and receiver cycle times are added to calculate the actual maximum age.
Setting location	Safety modules in the hardware editor or safe readback outputs in the graphic editor

4.18.5 Filter Time

Unit	Milliseconds (0 ms to 100 ms, default value 0 ms)
Description	Time value for software filters for safe inputs.
Setting location	Safe inputs in the hardware tree and in the graphic editor

4.18.6 Cross Circuit Detection

Unit	Unitless; options: 0, A, or B (default: 0)
Description	Cross-circuit detection circuit for safe inputs (no cross-circuit detection, circuit A, or circuit B)
Setting location	Safe inputs in the hardware tree and in the graphic editor

4.18.7 Hardware Revision

Unit	Unitless; used to select hardware revisions • Less than HW: 4.0 means hardware before revision 4.0 • Greater than or equal to HW: 4.0 means hardware with revision 4.0 or higher
Description	Used to calculate the monitored cycle time of a Safety CPU as a function of the hardware revision that has been set. The formulas for the calculations are the same. Starting with revision 4.0, the calculated time will be increased by 15%.
Setting location	Safety CPU module(s) in the hardware tree and project tree. This selection is only available for Safety CPUs for which a different calculation is required.

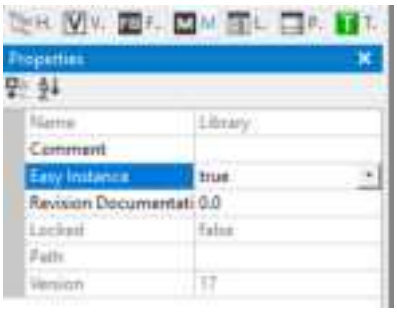
4.18.8 Parameter Revision

Unit	Unitless, used to select parameter revisions • Less than HW: 2.0 means parameters before hardware revision 2.0 • Greater than or equal to HW: 2.0 means parameters starting from hardware revision 2.0
Description	The set parameter revision specifies which parameters are available for a module. The parameters will be shown accordingly in the Properties pane.
Setting location	Safety module(s) in the hardware tree. This selection is only available for Safety modules for which different parameters can be set.

4.18.9 Simplified function block symbols

There are a number of function blocks for which you can select a simplified version for display. When you use this simplified display version, the block will be shown without a name.

You can make the corresponding change in the project's Properties pane. To do so, set the "Easy Instance" parameter to "true". You can also configure this setting in the macro library.



The following function blocks have simplified symbols available:

Function block	Representation
AND	&
OR	≥1
XOR	=1
NOT	⊘
Bool to Safebool	B2SB
Dint to SafeDint	D2SD
SError to SFalse	E2SBF
Safebool to bool	SB2B
SafeDint to Dint	SD2D
SafeDintError to SafeDint	SDE2SD

Please note that you will not be able to drag macros from an Easy Instance macro library to a normal project; you will first need to change the macro library or project.

If you want to reset the project or macro library to showing the standard version of the function blocks, first make sure that the networks have enough space available for the function block headers.

4.18.10 Enable CRC writing

Unit	Unitless; options: TRUE or FALSE (default: TRUE)
Description	This flag defines whether the CRCs for the application and hardware path should be written to the Safety CPU.
Setting location	Root element of hardware tree, project tree, and variable tree

4.18.11 Enable Easy Instance setting for instances

Unit	Unitless; options: TRUE or FALSE (default: FALSE)
Description	This flag defines whether select instances will be shown without a header.
Setting location	Root element of hardware tree, project tree, and variable tree

4.18.12 Enable setting for download with additional information

(Either as text or only as a CRC for the text). In large projects, this will make download times shorter and reduce the load on the Safety CPU flash memory.

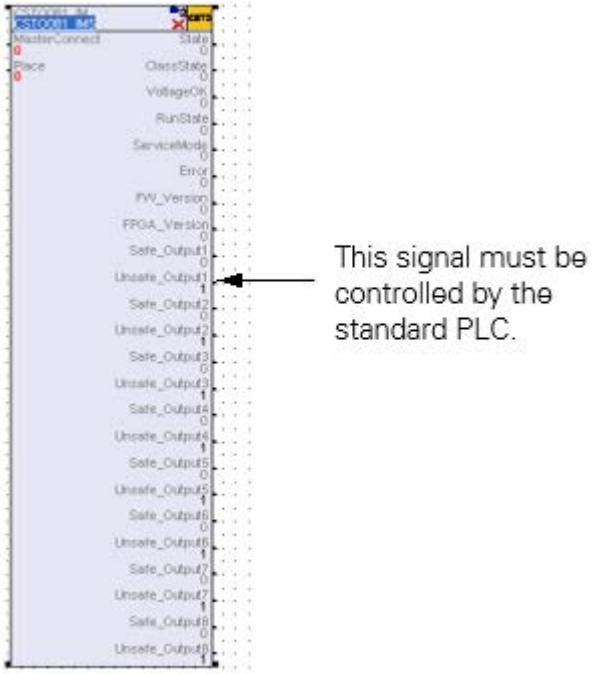
Unit	Unitless; options: TRUE or FALSE (default: FALSE)
Description	This flag defines whether additional information will be stored as the full text or only as a CRC.
Setting location	Root element of hardware tree, project tree, and variable tree

4.18.13 Disable flashing for an LED (local modules)

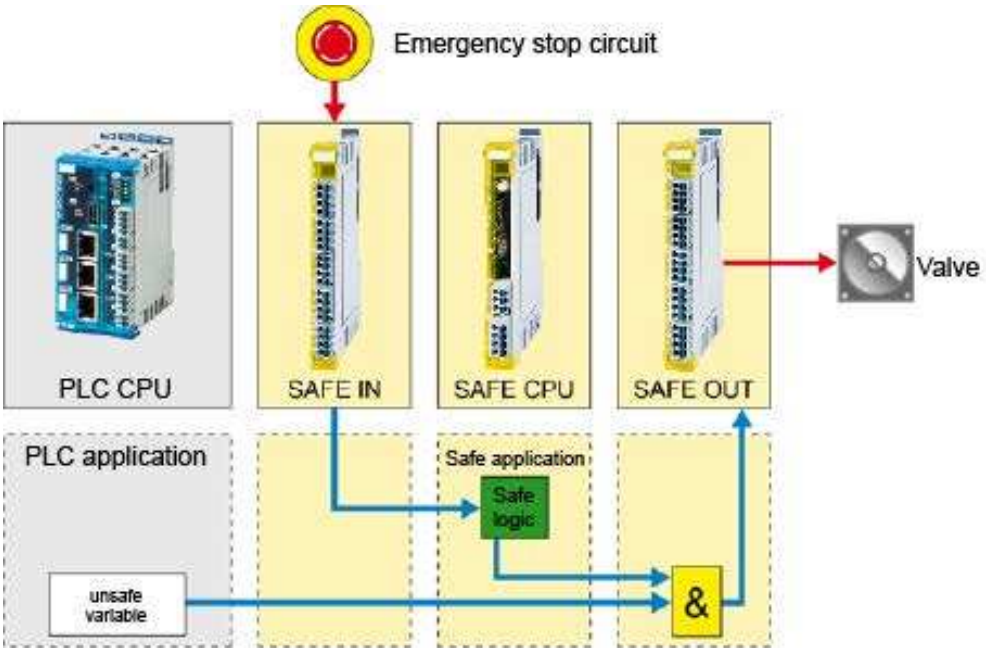
Unit	Unitless; options: TRUE or FALSE (default: FALSE)
Description	This flag defines whether the flashing of a red LED on the module will be disabled; allowed only if local optional modules are missing after verification.
Setting location	Safety module(s) in hardware tree. This selection is only available for Safety modules for which LED flashing can be disabled.

4.19 Enable signal for safe outputs

For safe outputs in Safety Designer, you can use an AND operator with an unsafe enable signal. Within this context, an enable signal is an unsafe variable that is not found in the Safety project, but that is instead set directly in the unsafe application on the corresponding hardware object.



This unsafe enable signal can only switch an output to its safe state (so that the output is turned off). The unsafe enable signal will be set with each bus cycle by the hardware classes in the Safety module and processed there in an interrupt. This makes it possible to quickly turn off a safe output with unsafe applications.



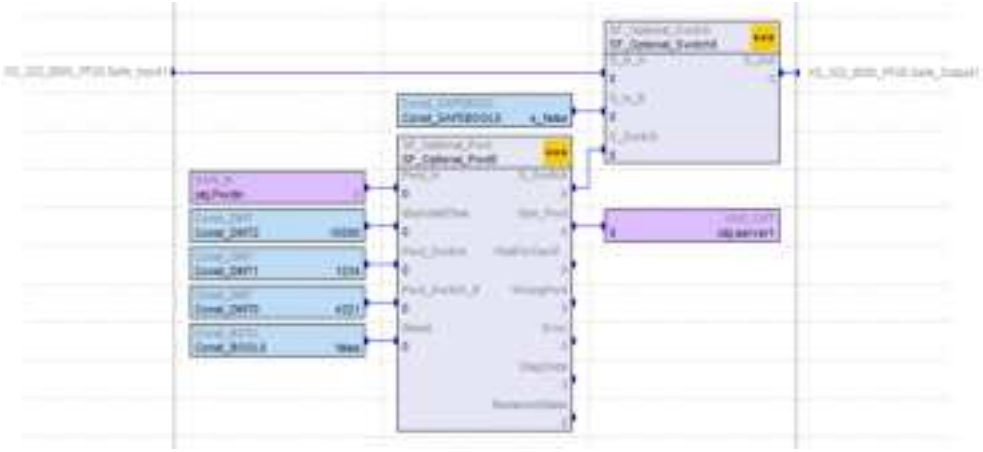
In order to be able to use an unsafe enable signal in XN Safety Designer, the "Enable signal used" flag must be set to "true" in the Properties pane. This value can be set either with the output in the hardware tree or in the network's output column. Please note that you will only be able to set the flag if the output is being used. The output will be labeled with an ampersand (&) icon in the network.

4.20 Optional modules

In XN Safety Designer, you can set Safety modules to "optional" in the Properties pane. When you do this, the corresponding individual modules will not need to be physically present in order for the Safety CPU to process the configuration. Instead, values preset by you will be used for these modules' data. Please note that using optional modules involves certain prerequisites.

To use this option, you will need to place special function blocks (SF_Optional_Pwd/SF_Optional_PwdII, and SF_Optional_Switch) after the inputs of the optional module and the remaining logic in the safety application. When a module is missing, these special function blocks will pass defined values (including values that do not correspond to a safe state) to the logic downstream of them.

→ Please note that a dangerous situation can arise if the state of these function blocks (in regard to the presence of an optional module) does not match the actual real-life situation.



Every used input of an optional module or a reading interface must be switched through an instance of the SF_Optional_Switch function block. The inputs used must be connected to the first input of the function block. The second input of the function block must be connected to a default value (such as a constant). This value will be used if the optional module is removed with a switch operation.The third input must be connected to the first output of the ST_Optional_Pwd/SF_Optional_PwdII function block and indicates whether the first or second input should be used. The SF_Optional_Pwd/SF_Optional_PwdII function block uses passwords to check that any switching between "optional" and "not optional" is intentional.The passwords required for switching are entered with an unsafe variable from the unsafe application, and the user must confirm them with an additional password (generated by the function block).

Every time the system switches between "optional" and "not optional," the user must check it to make sure everything is correct.

It must not be possible to copy the computed code (with "copy and paste," for example). Instead, it must be confirmed by being entered manually.

Safety measures and instructions

The use of the "SF_OPTIONAL_PWDII" function block affects all safety functions in general, which is why it is absolutely necessary to observe the following measures and hazard information. Failure to observe these warnings can result in serious injury or death!

DANGER

The machine's user must conduct a risk assessment regarding the potential hazards that can be posed by the use of the "SF_OPTIONAL_PWDII" function block while taking into account the standard-based requirements that apply to the specific application in question (e.g., hydraulic press brakes) and take all required risk mitigation measures.
The time during which the code calculated by the Safety CPU continues to be valid (login time) must always be as short as possible. In particular, it must always be ensured that this time does not cut across shifts.
It is absolutely necessary to ensure that the authorized operator that enters the password is exactly the same one that confirms the automatically calculated code.
The machine must be within sight of the machine's operator at all times while the automatically calculated code continues to be valid.
With the exception of the machine's operator, no one else should be within the machine's limits as defined in EN 12100. In particular, suitable measures must be used in order to ensure that the machine cannot be freely accessed.
Actuators without an emergency stop function must be installed in such a way that an emergency stop can be triggered within hand's reach from any position at all times.

4.21 Validating and generating download data

Safety Designer will generate the required download data based on the application. Before generating this data, Safety Designer will check the application you have created to make sure everything is OK. If any errors are detected in the application, error messages highlighted in red will be shown in the Output pane.
In most cases, double-clicking on the error message will select the corresponding point in the network.
In the case of errors for which a point in the network cannot be selected directly, you will need to determine what the problem is based exclusively on the error message.

Following is a list of possible error messages and what they mean:

Error message	Description
SafeCPU=<Name>, Network=<Name>, Object=<Name>, Input=<Name> is not connected.	The input of a function block instance in a Safety CPU network is not connected. Fix: Connect the instance's input.
SafeCPU=<Name>, Network=<Name>, Object= <Name>, Output=<Name> is not connected.	The output of a function block instance in a Safety CPU network is not connected. Fix: Connect the instance's output.
SafeCPU=<Name>, Network=<Name>, Object=<Name> is not connected.	A function block instance in a Safety CPU network is not connected. Fix: Connect the instance's inputs and outputs.
SafeCPU=<Name>, Network=<Name>, Row= <Line number>, Module1=<Name> is not equal Module2 =<Name>.	The AND operator in the input column has a problem; the inputs for the operator must come from a single module. Fix: Make sure the AND operator inputs come from one module only.
SafeCPU=<Name>, Network=<Name>, Row= <Line number>, Name1=<Name> is equal Name2 =<Name>.	The AND operator in the input column has a problem; the same input from a module has been used twice. Fix: Make sure that the AND operator uses two different inputs from the module.
SafeCPU=<Name>, Network=<Name>, Input <Name> in row <Line number> has different configurations to other input.	Elements in a Safety CPU network's input column that are used multiple times have different settings (e.g., cross-circuit, filter time). Fix: Make sure that the settings are configured the same way.
SafeCPU=<Name>, Network=<Name>, Input <Name> in row <Line number> is not connected.	An element in the input column is not connected. Fix: Connect the element in the input column.

Error message	Description
SafeCPU=<Name>, Network=<Name>, Output <Name> in row <Line number> is not connected.	An element in the output column is not connected. Fix: Connect the element in the output column.
SafeCPU=<Name>, Network=<Name>, Output =<Name.Servername> is used multiple times.	An output is used multiple times. Outputs can only be written to once. Fix: Only use the output once.
SafeCPU=<Name>, Network=<Name>, Connection <Startposition to Endposition> is invalid, start column is higher than end column.	A connection runs from right to left. The start point is to the right of the end point. Fix: Delete the network and create it again.
SafeCPU=<Name>, Network=<Name>, Position = (value) instance not found.	An instance could not be found at the specified position. Fix: Delete the network and create it again.
SafeCPU=<Name>, Network=<Name>, Connection <Startposition to Endposition>; Connection of constant block is invalid (delete connection and reconnect).	There is a connection between an editable constant and a constant input. Fix: Delete the connection and create it again.
SafeCPU=<Name>, Network=<Name>, Connection <Startposition to Endposition> has different data types (delete connection).	The data types of the connection's start and end points do not match. Fix: Delete the connection.
SafeCPU=<Name>, Network=<Name>, Instance= <Names>; init value is out of range.	The value set for a constant falls outside the valid range, Fix: Set a valid init value.
Temp. variable <Name> is never set.	A temp. variable is being used as an input without having been set first. Measure: Set the temp. variable as an output.
SafeCPU=<Name>, Network=<Name>, Element of temp. variable <Name> with invalid sequence. Sequence of networks in SafetyCPU <Name> cannot be dissolved. Temp. variables are used before they are set.	An unambiguous processing sequence for the networks with temp. variables cannot be created. All networks with temp. variables as an output in the download data must be processed first. If a temp. variable is needed as an input before the network with the temp. variables as an output has been processed, an error message will be generated. Accordingly, a network sequence is always determined when checking the temp. variables. This sequence must be unambiguous.The first temp. variable that can no longer be processed will be indicated as the "error variable." Crossovers when using temp. variables in the networks are not allowed. Fix: Check how all temp. variables are being used and set them differently as necessary.

Error message	Description
SafeCPU=<Name>, Network=<Name>; Using sequence of temp. variable <Name> is invalid. Temp variable is used before it is set.	A temp. variable is being used as an input before the variable has been set as an output. Fix: Set the output before the temp. variable is used as an input.
Temp variables <Name> can only be used within one SafeCPU. Variables used in Safety-CPU: <Name>	A temp. variable is being used in multiple Safety CPUs. Each temp. variable can only be used in one single Safety CPU. Fix: Delete the instances of the temp. variables from the other Safety CPUs.
SafeCPU=<Name>, Network=<Name>, Output=<Name> in Row= <Line number> has wrong domain number.	An unsafe variable has a non-existing domain number. Fix: Change the domain number for the unsafe variable.
SafeCPU=<Name>, Network=<Name>, Instance=<Name> has wrong domain number.	An unsafe variable has a non-existing domain number. Fix: Change the domain number for the unsafe variable.
PLC <Name> with domain number 0 has a invalid connection to module <Name>.	The PLC with domain number 0 is not the topmost standard PLC. Fix: Change the domain name or the hardware layout.
PLC <Name> has no connection to a other PLC.	The PLC has a domain number other than 0, but is not connected to any other standard PLCs. Fix: Establish a connection to another standard PLC.
Domain number <Number> is used in PLC <Name> and <Name>.	A domain number is being used in two standard PLCs. Fix: Change the domain number in one of the standard PLCs.
Domain number 0 must be used in project.	Domain number 0 is not found in the project. Fix: Set the domain number to 0 on the first standard PLC.
SafeCPU=<Name>, Network=<Name>, Input=<Name> in Row= <Line number> is a dummy element.	An element in the input column is a wildcard. Action: Replace wildcard.
SafeCPU=<Name>, Network=<Name>, Output=<Name> in Row= <Line number> is a dummy element.	An element of the output column is a wildcard. Action: Replace wildcard.
SafeCPU=<Name>, Network=<Name>, Instance=<Name> is a dummy element.	An instance in the network is a placeholder. Action: Replace wildcard.
SafeCPU=<Name>, Network=<Name>, Object=<Name>, Output=<Name> is remanent. Remanent outputs are not supported by module=<Name>.	An instance of a function block has a retentive server. The module does not support retentive servers. Fix: Delete the instance or replace the Safety CPU with a Safety CPU that supports retentive servers.

Error message	Description
SafeCPU=<Name>, Network=<Name>, Object=<Name> is a interface variable. Interface variables are not supported by module=<Name>.	An instance of a function block is an interface. The module does not support interface variables. Fix: Delete the instance or replace the Safety CPU with a Safety CPU that supports interface variables.
SafeCPU=<Name>, Network=<Name>, Object=<Name> is a bit variable. bit variables are not supported by module=<Name>.	An instance of a function block is a bit variable. The module does not support bit variables. Fix: Delete the instance or replace the Safety CPU with a Safety CPU that supports bit variables.
SafeCPU=<Name>, Network=<Name>, Input <Name> is a bit variable. Bit variables are not supported by module=<Name>.	An input element is a bit variable. The module does not support bit variables. Fix: Delete the input element or replace the Safety CPU with a Safety CPU that supports bit variables.
SafeCPU=<Name>, Network=<Name>, Output <Name> is a bit variable. Bit variables are not supported by module=<Name>.	An output element is a bit variable. The module does not support bit variables. Fix: Delete the output element or replace the Safety CPU with a Safety CPU that supports bit variables.
Can not find interface module of interface(read) <Name>	The configured interface module cannot be found in the project. Fix: Configure a different interface module.
SafeCPU=<Name>, Network=<Name>, Instance=<Name> is a interface variable. Interface variables are not supported by standalone PLC <Name>.	The instance is an interface variable. A standalone PLC does not support interface variables. Action: Delete instance.
SafeCPU=<Name>, Network=<Name>, Instance=<Name> is a variable. Variables are not supported by standalone PLC <Name>.	The instance is a variable. A standalone PLC does not support variables. Action: Delete instance.
SafeCPU=<Name>, Network=<Name>, Input=<Name> in row=<Line numbers> is a variable. Variables are not supported by standalone PLC <Name>.	The input element is a variable. A standalone PLC does not support variables. Action: Delete input element.
SafeCPU=<Name>, Network=<Name>, Output=<Name> in row=<Line numbers> is a variable. Variables are not supported by standalone PLC <Name>.	The output element is a variable. A standalone PLC does not support variables. Action: Delete output element.
Module <Name> is no child of module <Name>.	The sub-element is not allowed in this module. Action: Delete elements.

Error message	Description
Position of Module <Name> is wrong.	The position of the sub-element is wrong in this module. Action: Delete or move elements.
Module <Name> is wrong. It is not allowed to leave gaps between two modules of the same bus type.	It is not allowed to leave empty spots between two modules on the same bus system. Action: Delete or move elements.
Module <Name> is wrong. Order of child elements is not correct.	The sub-elements' sequence is wrong. Action: Delete or move elements.
Only one PLC as standalone module allowed, PLC <Name> is wrong.	Only one standalone PLC is allowed. Action: Delete PLC.
Module <Name> has no master module.	There is a module in slave mode with no module set to master mode. Action: Set module.
Master <Name> of module <Name> not found.	There is a module in slave mode and the module set to master mode for it is the wrong one. Action: Set another master module.
Module <Name> has different max. transmission times, only one time allowed.	A module has different. max. transmission times configured for its inputs/outputs. Action: Align times.
Module <Name> has different max. transmission times for output as input, only one time allowed.	A module has different. max. transmission times configured for its readback outputs. Action: Align times.
Invalid parameter <Name> of module <Name>. Value has to be within range <Value> - <Value>.	The value of a dynamic parameter falls outside the corresponding limits. Action: Specify value within the limits.
Invalid parameter <Name> of module <Name>. X-Value has to be within range <Value> - <Value>. Y-Value has to be within range <Value> - <Value>.	The values of a dynamic parameter fall outside the corresponding limits. Action: Specify values within the limits.
Invalid parameter <Name> of module <Name>. Unkown data type <Name>.	The data type of a dynamic parameter is unknown. Action: Replace parameter with another parameter.
Number of used enable signals of module <Name> is too high (max. number is 16).	The number of safe enable signals in the outputs is too high. Action: Use a maximum of 16 release signals.
Number of bit variables are not supported by module <Name>. Only 8 bits allowed.	The number of bit variables in a fast variable is too high. A max. of eight bits are allowed. Action: Delete variable and create again.
SafeCPU=<Name>, Network=<Name>, Interface=<Name> has a safe dint variable. Safe dint variables are not allowed.	The Safety CPU does not support SafeDint variables. Fix: Delete the interfaces in the network.

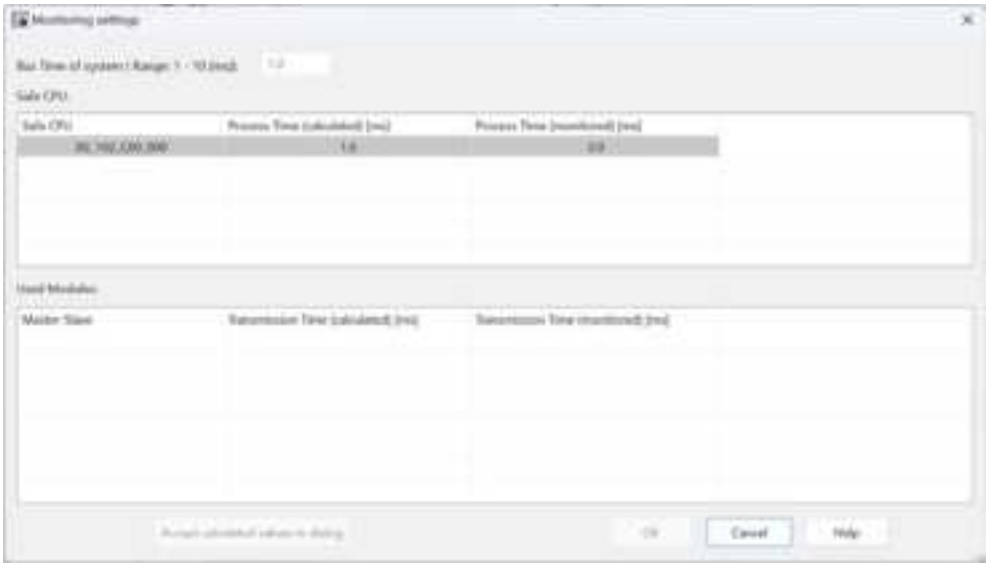
Error message	Description
SafeCPU=<Name>, Network=<Name>, Module=<Name> has an safe dint channel. Safe dint channels not supported by module <Name>.	The Safety CPU does not support SafeDint variables. Fix: Delete the element in the network.
SafeCPU=<Name>, Network=<Name> Object=<Name> has wrong revision of function block (must be lower than 2.0)	The Safety CPU does not support the function block version. Fix: Delete the instance in the network.
Module<Name>: Number of modules at bus <Name> is too high (max. number: <Quantity>. Module(s) <Names> needs more than one place at bus. Delete one or more modules.	Only a specific number of modules can be placed. There is at least one module that is taking up more than one spot on the bus (e.g., relay). Fix: Delete modules in the hardware tree.
Module <Name> is not allowed in standalone mode.	A module is not allowed in standalone mode. Fix: Delete the module.
Slave address <Number> of module <Name> is outside of allowed ranges.	The slave address falls outside of the permissible limits. Fix: Assign an address within the limits.
Connection ID <Number> of module <Name> is out-side of allowed ranges.	The connection ID falls outside of the permissible limits. Fix: Assign a connection ID within the limits.
Module <Name> has no FSoE master for slave address <Number>.	There is no FSoE master for a specified slave address. Fix: Delete the module.
Connection ID <Number> in module <Name> is used multiple times.	A connection ID is being used multiple times. Fix: Generate unique connection IDs.
Slave address <Number> in module <Name> is used multiple times.	A slave address is being used multiple times. Fix: Generate unique slave addresses.
Slave addresses must be unique in whole system! Ranges of slave addresses in project <Name> and project of interface <Name> are not unique. Please change your ranges of slave addresses (range of project:<Value>-<Value> range of interface:<Value>-<Value>) in one of the projects.	The limits for the slave addresses in the project and in the project of the writing interface overlap. Slave addresses must be globally unique in the whole system. Fix: Adjust the limits in one of the projects and adjust the slave addresses accordingly.

Error message	Description
Connection IDs must be unique in whole system! Ranges of Connection IDs in project <Name> and project of interface <Name> are not unique. Please change your ranges of Connection ID's (range of project:<Value>-<Value> range of interface:<Value>-<Value>) in one of the projects.	The limits for the connection IDs in the project and in the project of the writing interface overlap. Connection IDs must be globally unique in the whole system. Fix: Adjust the limits in one of the projects and adjust the connection IDs accordingly.
The module <Name> can be only used by a SafeCPU with master mode. Please change your configuration.	A module (e.g., XS-322-2INC) has been added downstream of a Safety CPU in slave mode. Fix: Switch the Safety CPU to master mode.
More than one function block <Name> has been found, only one function block is allowed.	A function block (e.g., Sf_HGW_Login) is being used multiple times. The function block is only allowed to be used once. Fix: Delete the extra function blocks.
The flag of safety interface communication of SafeCPU <Name> is wrong. Module <Name> is used in SafeCPU. Please change your configuration.	Safety interface communications have been disabled even though a module (HGB, SIB) requires them. Fix: Change the flag or delete the module.
Macro Network <Name>, Object <Name>, Input <Name> is not connected	The input of a function block instance in a macro network is not connected. Fix: Connect the instance's input.
Macro Network <Name>, Object <Name>, Out-put <Name> is not connected	The output of a function block instance in a macro network is not connected. Fix: Connect the instance's output.
Address of memory variable or constant <Name> cannot connected with <Name> in SafeCPU <Name>	The address of the memory variables or constants cannot be connected correctly in the Safety CPU.
Address of memory variable or constant <Name> cannot connected with element in output row SafeCPU <Name>	The address of the memory variables or constants cannot be connected correctly in the Safety CPU.
Adjustable CANY or DIM inputs must have same data type	Modifiable CANY variables and dimensions must have the same data types.
Adjustable inputs exeeds range	Modifiable inputs of a function block must have a limit.
Wrong number of inputs	The number of modifiable inputs is wrong.
Input <Name> in function block not found	The input cannot be found in the function block.
<Name> of <Name> must be connected with dint array	The element of a table must be connected to the array correctly.
Please check the length and dimensions of array input	The size or dimension of arrays is incorrect.
Data type is different from 'StartAddress' in <Name>	The data type is different from the data type of the start address.

Error message	Description
Memory variable <Name> must not be connected with address in SafeCPU <Name>	The flag variable cannot be connected to an address of the Safety CPU.
Parameter(s) of Section <Name> are not in correct order	The parameters within a sector are in the wrong order.
Type array <Name> of memory variable <Name> not found	The configured array data type of the memory variable cannot be found.
Type array <Name> of constan variable <Name> not found	The configured array data type of the constant cannot be found.

4.22 Calculating the monitoring settings

XN Safety Designer assists users with calculating the maximum required transmission times for the modules and determining the cycle time for the Safety CPUs. Clicking on the Monitoring Settings option will open the following dialog box with the automatically determined data that has been configured by the user.



XN Safety Designer can determine the cycle time for the Safety CPU based on this data and the built project. The project must be built for this purpose. Please note that the values in the dialog box can only be calculated correctly if the project can be built correctly. If the project cannot be built, a calculated cycle time will not be shown.

You can manually change the values for the Monitored data in the dialog box (selecting the cells and pressing F2 will open the editor).

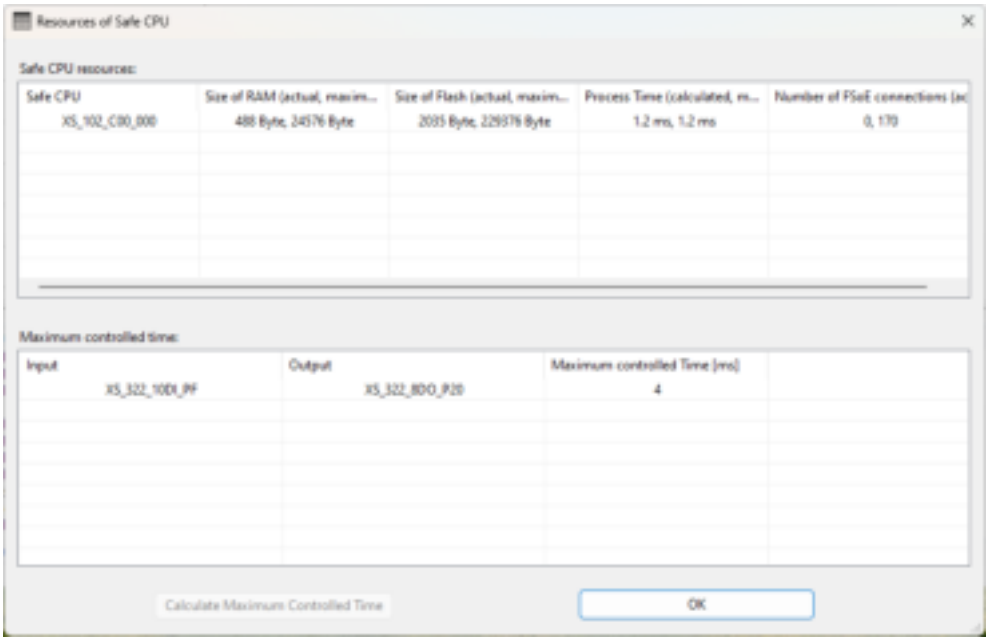
Clicking on the Accept calculated values in dialog button will copy the suggested data to the columns for the Monitored values.

Clicking on the OK button will apply the Monitored data in the project.

→ Make absolutely sure to check the values suggested by XN Safety Designer while the system is running to make sure they are correct!

4.23 Resource consumption and response time

XN Safety Designer supports users by calculating resource consumption, estimating the maximum cycle time, and calculating the maximum (monitored) total response time from a safe input to a safe output. This information will be available to you after generating a program (compiling operation), and you will need to confirm it.This dialog box will also appear when you go online, but the total response time will not be shown automatically in this case (you will need to click on the "Calculate Maximum Controlled Time" button).



Additional information regarding the compilation will be output.

This information is as follows:

- Number of COPY calls
- Number of CALL calls for the function blocks
- Size of interpreter code in bytes
- Size of function block code in bytes
- Size of used retentive variable space
- Size of additional information

→ The amount of memory used (flash and RAM) will depend on the aforementioned data among other things. Every function block, every unsafe variable, every I/O card, etc. will take up some memory.
Moreover, additional information such as the number of networks, the networks' names, the function blocks' names, the function blocks' code, etc. will also take up flash memory.

4.23.1 Resource consumption

The following resources will be taken into account in the corresponding calculations:

Flash memory consumption by program configuration

The flash memory required by the program will be compared to the memory available on the Safety CPU. If the program in XN Safety Designer exceeds the available size, an error message will be generated during compilation and a download will not be possible.

RAM consumption by program configuration

The RAM memory required by the program will be compared to the memory available on the Safety CPU. If the program in XN Safety Designer exceeds the available size, an error message will be generated during compilation, preventing a download.

A number of reference guidelines for a rough calculation of RAM use are listed below so that you can carry out a preliminary calculation to see how large the Safety CPU load can be or whether the selected Safety CPU will be adequate. In order to estimate the amount of memory used, you will need to know what your application will look like. In many applications, the function blocks make up the lion's share of RAM use.

Following is a list of the most important variables that take up RAM memory.

I/O modules

- 48 bytes are required per input
- 32 bytes are required per SAFEDINT input
- 16 bytes are required per output
- 24 bytes are required per SAFEDINT output
- …

Tags

- 4 bytes are required per unsafe variable
- …

Function blocks (please refer to I/O memory use table)

- 4 bytes are required per placed function block
- 4 bytes are required per input/output
- 8 bytes are required per SAFEDINT input/output
- …

Tbl. 2: I/O module memory use

Module	RAM [Byte]
XS_322_10DI_PF	480
XS_322_8DIO_PF20	320
XS_322_4AI_O2	512
XS_322_2ENC_INC	448
XS_322_2DO_RNOAC	32
XS_322_2DO_RNODC	32
XS_322_8DO_P20	128

4.23.2 Cycle time estimate for user program

Depending on the complexity and size of the user program, there can be long cycle times when processing the user program in the Safety CPU. After a compilation operation, you will be able to see the maximum calculated cycle time. Please note that the maximum processing time for each function block was determined when the function block was being developed, and it is these function block processing times that will be added up based on the user program.

4.23.3 Total response time

For transmitted input and output signals, you can configure a maximum transmission time (maximum age). In addition, you can configure the maximum program runtime (cycle time), which the firmware will monitor. These values can be used to calculate a maximum monitored total response time from a module (or input of a module) to an output and show the result.If an output is set by various modules, the times will be shown separately for each module.

Check the values suggested by XN Safety Designer in the system while it is running to make sure they are correct (e.g., by repeatedly measuring the response times in the running system)!

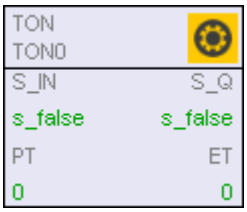
4.23.4 PDO frame length

XN Safety Designer will calculate the lengths of the PDO frames that are generated by Safety modules or needed by Safety modules. An error message will be generated if the lengths are too long.

4.24 Debugging

In order to be able to run debugging, you must be online with the PLC and the Safety project must match the program on the Safety modules. Values can be viewed without logging in. However, you will need to at least be logged in in debug mode if you want to force values. Please note that values can only be changed in Temporary Operation Mode.

During debugging, input/output states and variables will be shown. You will be able to change (force) the values of constants and individual elements in the input and output columns and transmit them to the Safety CPU.



The elements in the input and output columns, as well as the temp. variables and unsafe variables (BOOL only) will be highlighted in color. The connections extending from elements and from BOOL elements will be shown in color.

Color highlighting for input/output column; color-coded representation of connections:

- Safebool TRUE/TRUE …green
- Safebool FALSE/FALSE …blue
- Safebool Error … red
- Bool TRUE …green
- Bool FALSE … blue

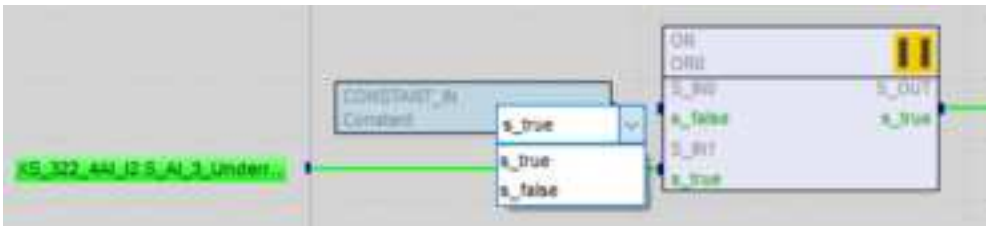
4.25 Forcing

You can change the values of constants, safe inputs (input column) and safe outputs (output column) and transmit them to the Safety CPU. In order to be able to change the values on the Safety CPU, you will need to be logged in with the Debug password level at least. In addition, the Safety CPU must be running in Temporary Operation Mode. In order to be able to force values for modules in slave mode, you must be logged in at least in the debug level on the module with master mode.

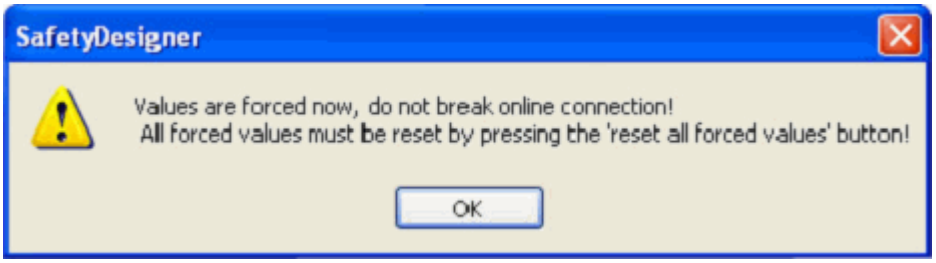
For the constants, you can press the F2 key or use a long click to open an editor for entering the value you want. Once you confirm the value, it will be transmitted to the Safety CPU and set there.

For safe inputs and outputs, you can change the corresponding values with the context menu.

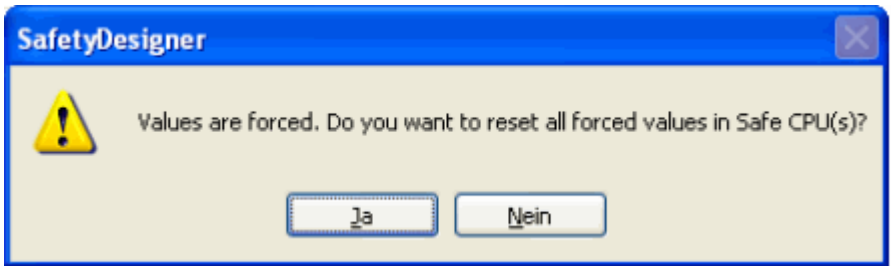
Forced values will be shown underlined.



You will get a warning indicating that the values were changed manually. This prompt (which also points out that the online connection must not be terminated during forcing) needs to be confirmed.



The Safety CPU will now run with forced values. The buttons for going online and closing the Online State View will be disabled. You can click on an empty area in the network editor or click on the Reset all forced values button at any time to reset all forced values. If you click on the Login or Service Mode button in the Online State View, you will need to confirm that you want to reset the values.



The commands for changing the login or setting Service Mode will only be run if you have explicitly confirmed that you want the values to be reset.

If the connection to XN Safety Designer is interrupted and the forced values have not been explicitly reset, the Safety CPU will switch to its error condition.

→ If the values are changed, the machine MUST be within sight of the technician changing the values!

→ Do not, under any circumstance, disconnect the online connection when the values are being changed!

4.26 Watch window

In online mode, you can use the Watch pane to view the values of elements in the network. Values can be viewed without logging in.

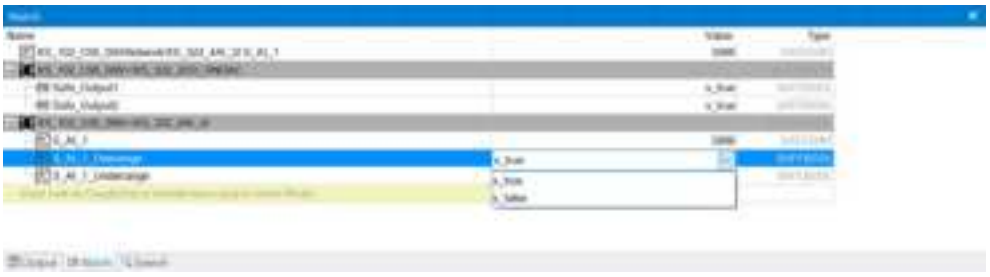
In online mode, you can drag and drop various elements from networks or trees to the Watch pane. These elements include elements in the input and output columns and instances of variables, constants, interfaces, and whole function blocks. If you drag elements from a tree to the Watch pane (an input card module, for example), an entry will be generated for the element with all the inputs used by it.

The elements shown in the Watch pane will be shown in the order in which they were added. You can sort the rows by clicking on the "Name" column header so that they will be sorted alphabetically by Safety CPU and networks. The data in the Watch pane will be saved when the project is closed and will be available again next time you load the project.

The element name will be shown in the "Name" column (syntax: name of Safety CPU, name of network, name of element). The unsafe variables will be shown with their symbolic name (name from the tree). Meanwhile, the online values will be shown in the "Value" column. Finally, the data type of the various elements will be shown in the "Type" column.

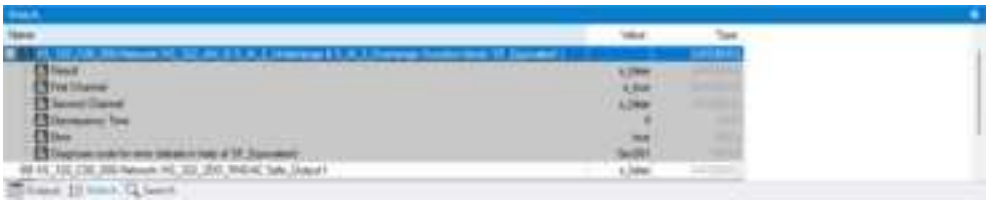
Elements can be forced in the Watch pane. All elements that cannot be forced will be highlighted in gray. Meanwhile, forced values will be shown with brown text. Please note that if you want to force values, you will have to be logged in in Debug Mode at least. Also note that values can only be changed in Temporary Operation Mode.

In order to be able to force values on modules in slave mode, you must be logged in at least in the debug level on the module in master mode. You can change BOOL and SAFEBOOL values with a combo box, while DINT and SAFEDINT values need to be entered with an editor.



The same instructions as in → section " Forcing", page 102 apply to the forced values.

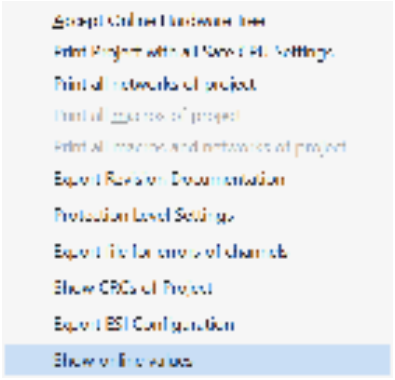
A tree will show the AND operator for the input column with an Equivalent function block. The individual information for the function block will be listed. Please note that the values cannot be forced.



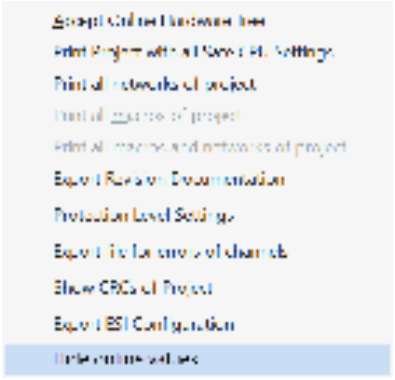
Result: Operator result
First Channel: Value of first safe input
Second Channel: Value of second safe input
Discrepancy Time: Set time difference
Error: Flag indicating whether the function block is in its error condition
Diagnose code for error: Diagnostic code for the error condition (you can read what the code means in the SF Equivalent block)

4.27 Viewing the online values of modules in the hardware tree

You will only be able to view online values in the hardware tree if the corresponding option has been enabled. To enable it, use the Show online values option in the project context menu.



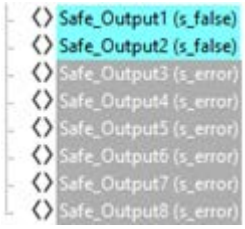
In order to avoid slowing down the online displays in the network and Watch pane unnecessarily, you can also disable the option with Hide online values. The last setting you select in this regard will be saved in the project.



For SafeBool inputs and outputs, the state of the input or output (s_true, s_false, or s_error) will be shown in parentheses after the name. In addition, the name will be highlighted in color: green for s_true, blue for s_false, and red for s_error.

For SafeDint inputs and outputs, the value will be shown in parentheses after the name and will only be highlighted in red in the event of an error.

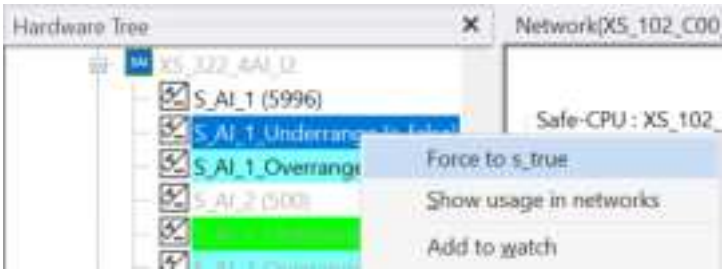
All outputs not used in networks will have an "s_error" state, but will not be highlighted in red and will be highlighted in gray instead.



Values will be shown for the Safety CPU to which they are connected on the PLC.

Forcing SafeBool values in the hardware tree

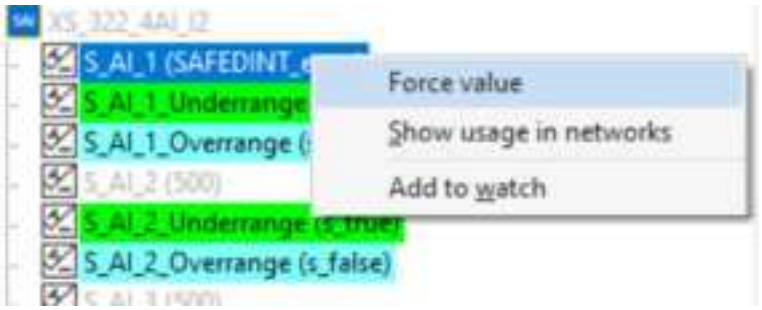
You can change SafeBool values in the hardware tree with the context menu the way you can in the network's input/output columns. In the event of an error (s_error), there can be a change to either s_true or s_false.



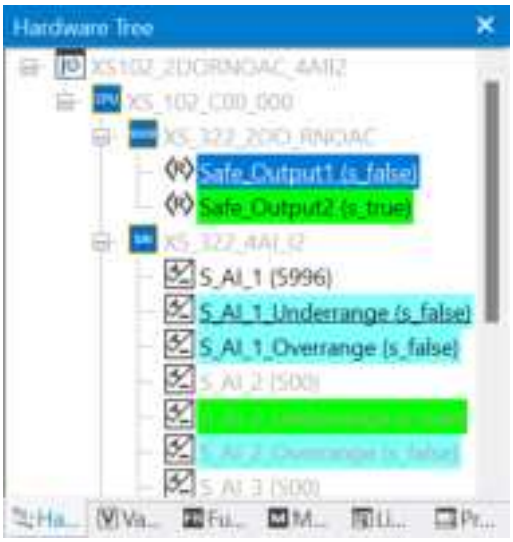
Forcing SafeDint values in the hardware tree

SafeDint values can also be changed with the context menu in the hardware tree.

When you do this, a small input window in which you can edit the value will appear.



Inputs and outputs with values that have been forced will be shown with underlined text just like in input and output columns.



Values can only be forced if the input/output is used in a network of the CPU to which the module is connected.

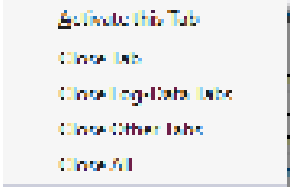
4.28 Reading and viewing Safety module log files

You can use the context menu in the Hardware Tree pane (for the module in question) to read the data from the log memory of a safety module (e.g., Safety CPU) and save it in two binary files (one file for controller 1 and another for controller 2). The information will be processed in XN Safety Designer, saved accordingly, and shown.If controller 1 and controller 2 contain different information, the corresponding entries will be shown in separate lines, one below the other, highlighted in red, and the second line will be preceded by a note saying "Controller 2." The first and last lines in the view will provide, highlighted in gray, information regarding the module name, the firmware version, and, if applicable, the project loaded in the Safety CPU.

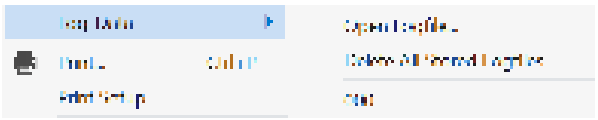
Log entries on different days will be indicated with a line highlighted in yellow, with the respective date shown between the entries.

Entries showing a system (re)start will be highlighted in violet.

To close the log data view(s), click on the Close Log-Data Tabs option in the context menu for the Register tab.

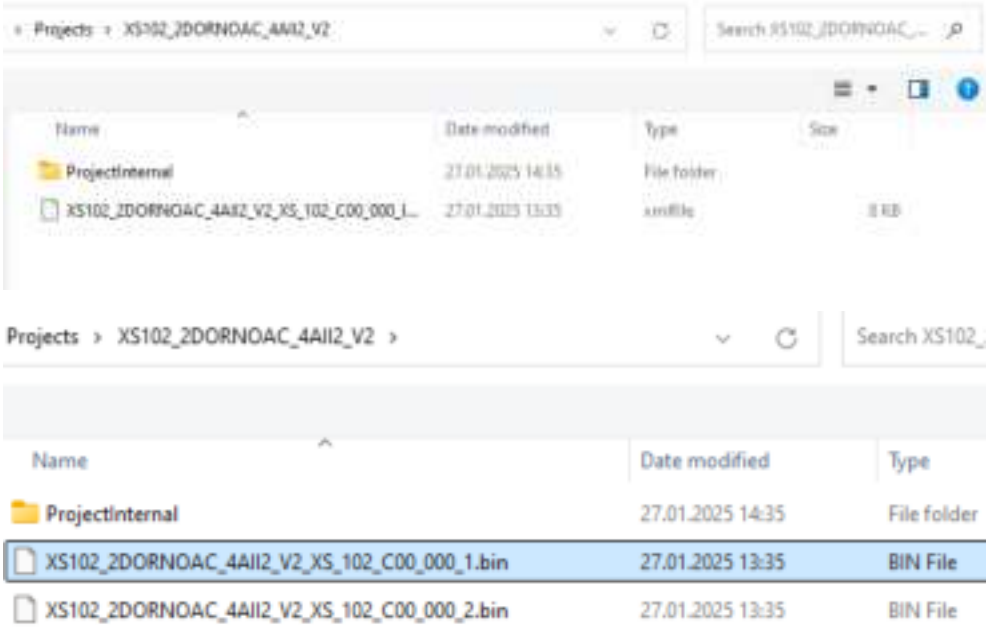


Once log data has been transferred, you can view it again at any time with the corresponding name by opening the File menu and clicking on the Log-Data option.



The Delete All Stored Logfiles menu option will delete all the log files corresponding to the project that is currently open.

You can also use the Open Logfile...option to view log data that belongs to other projects. When you select this option, the Open Log-Data File dialog box will appear.



The **<..._log.xml>** files will contain information that can be used to interpret the data. If these files are not available and only the raw data from memory is available (in the form of BIN files), select the "Log-Data file(*.bin)" filter instead of "Log-Data info(*_ log.xml)" in the combo box."..._1.bin" files will contain data for controller 1 and "..._ 2.bin" files will contain data for controller 2. If both files are available and you open the "..._1.bin" file, the other file will also be opened automatically and both files will be compared.

What the log data entries mean

„Change State of Login from 'old' to 'new'"

Login or logout for servicing and data transfer to a Safety CPU.

Possible connection types:

not logged in: No connection

debug: For troubleshooting

configure: For modifying the configuration and transferring projects

general: Firmware download (for authorized people only)

"Error Login from 'old' to 'new', returned message code: Errornumber"

Login was unsuccessful.

Possible causes: No authorization or typing mistake.

Possible connection types: Please refer to "Change state of Login …"

„Password changed for 'level'"

The password for the "level" connection type has been changed.

Possible connection types: Please refer to "Change state of Login …"

„Password change failed for 'level' returned message code: Errornumber"

A password change was rejected.

Possible causes: No authorization or typing mistake.

Possible connection types: Please refer to "Change state of Login …"

„Password loaded from SD-card"

The password was retrieved from a memory card.

„Set Service-Mode"

The Safety CPU was stopped in order to make changes, e.g., to load a different project or verify the project.

„Set temporary Service-Mode"

The Safety CPU was switched to a temporary Service Mode.

„Start Application"

The project in the Safety CPU was started.

„Change state of Configuration from 'old' to 'new"

The Safety CPU changed the operating mode from "old" to "new".

Possible operating modes:

unknown: Defaultwert

invalid: The loaded project was discarded and will not be processed anymore

not configured: There is no loaded project

configured but not deployed and not verified: A project is loaded, but it has not been transferred to the slave modules yet and has not been verified yet

configured and deployed but not verified: The project is loaded and has been deployed, but has not been verified yet

configured and verified: The project is loaded and verified

„Configuration loaded from SD-card"

The configuration and project data were loaded from a memory card.

„Got safety number from remote module 'XYZ': 0x000000yy"

„Got safety number from remote module with index 123: 0x000000yy"

Safety number was retrieved from module "XYZ" or module with index 123. Result: 0x000000yy.

„Module 'XYZ' changed"

„Module with index 123 changed"

A module change in "XYZ" or index 123 was detected.

„Deployed configuration to module ‚XYZ' with safety number 0x000000yy"

„Deployed configuration to module with index 123 with safety number 0x000000yy"

The remote module with name "XYZ" or index 123 and safety number 0x000000yy was configured.

„Error 123, …."

General error message with an explanatory text.

„Quit Error 123"

The error with number 123 was reset.

„Too many messages, no more messages added"

At least one log entry could not be added.

„Date and Time: 12.01.2013 12:13:51 overrun of internal timestamp: 2" (yellow background)

Date and time. The information regarding the internal time overrun counter is only of interest to technicians.

„Date: 12.01.2013" (yellow background)

„New Date: 12.01.2013" (yellow background)

Indicates the date of the following entries. In this case, the time information for the messages entered on the same day will be found on the left side. This message is added with every date change and after every system restart.

„Time passed: 13 days, 2 hours, 32 minutes, 12 seconds" (yellow background)

If the time cannot be resolved and the seconds counter has started again from zero on the left of the pane, this message will be used to indicate how much time has passed between the upper and lower messages in days, hours, minutes, and seconds.

„Startup of system"

„Startup of system at: 12.01.2013 12:13:51"

The Safety CPU was (re)started. The date and time will be specified if they can be determined.

„Shut down of system"

The Safety CPU has been switched off or power was interrupted.

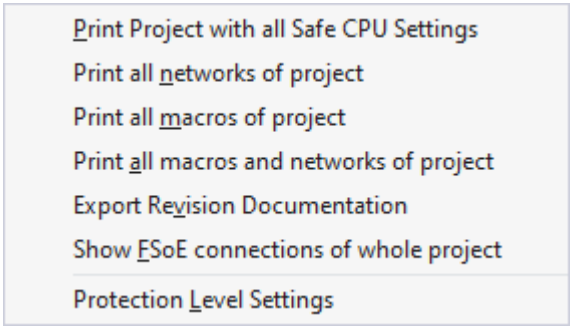
„Missing optional module(s): Safety-CPU1, Safety-CPU2, #15"

„Missing optional module(s) with indexes: 1, 5, 17”

The optional modules with names Safety-CPU1, etc. or index 15, etc. were not found.

4.29 Printing

When printing, you can print out the project or the individual networks. When the networks are printed out, the printout will correspond to the graphic display in the editor. For the project, meanwhile, you can print out the hardware tree, the variables, the project settings, the Safety CPU settings, and the Safety module settings. You can access these print options with the context menu in the Project Tree or Hardware Tree pane. You can also print the network currently being shown directly from the File menu.There is also the option of printing out all networks with a command call. To print all the networks, you will need to use the corresponding context menu option in the Project Tree or Hardware Tree pane.



You can print the entire project with all its networks with the Print option in the File menu, in which case the project settings will be printed first, followed by the networks.

If you use the option for printing all networks, all the networks will be shown simultaneously in the preview dialog box. The corresponding zoom level will be calculated based on the biggest network.

A header and footer with the project settings are inserted on all printed pages:

Header contains:

- Project names
- Company of the project
- Author of the project
- Safety system with version of the XN Safety Designer
- Revision of the project

Footer contains:

- Date and original time of printing
- Page with total number of pages

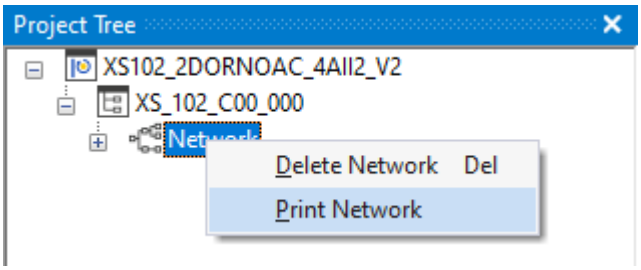
With the Safety CPUs, the CRCs of the respective configuration are displayed when the project is printed. The CRCs can only be displayed if the project has been built.

The CRCs are displayed in hexadecimal format; if there is no CRC, the text "no CRC" is displayed.

In online mode, additional module data is printed when the project is printed.

Online data included:

- Module names
- Security number of the module
- Firmware version of the module
- Configuring state
- Project name and revision of the project
- For optional modules, whether the module is present



There are two tabs in the print preview dialog.
In the first tab, the printer can be selected and general print settings can be set.
The second tab depends on the element to be printed.

4.29.1 Settings when printing networks

Network" tab:

Zoom factor: Zoom in or zoom out factor can be set as a percentage.

Landscape = Landscape format

Autosize of columns = Automatically adjust column width for printing

Navigation through the individual pages

A selection can be made via the context menu:

- Print this Page
- Select this Page
- Select All
- Deselect All

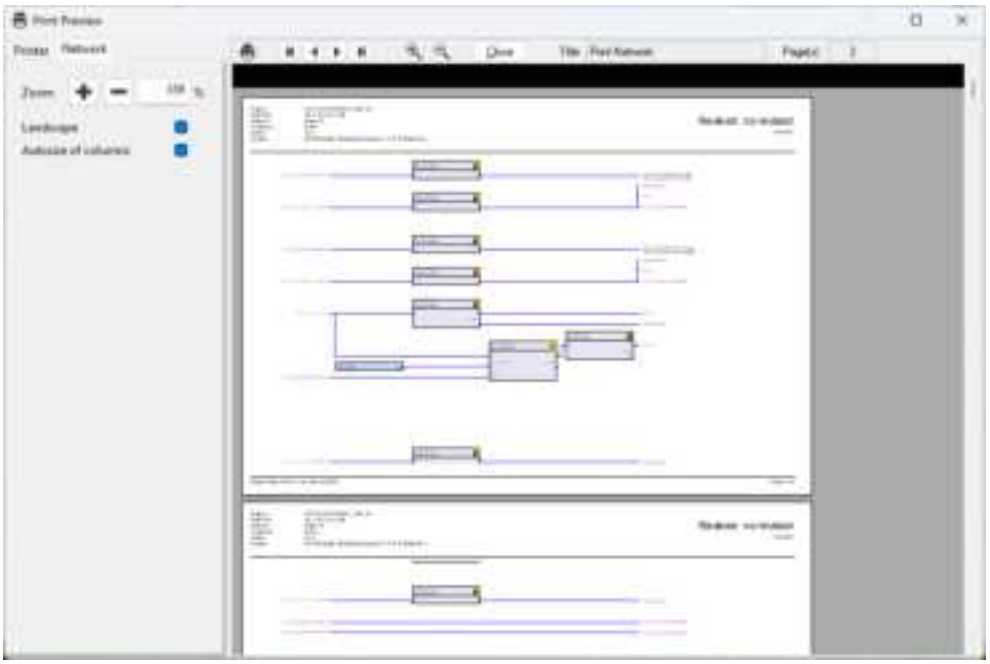


Abb. 1: Example showing the preview for a network printout

4.29.2 Settings for printing the project tree

Project" tab:

You can select the individual modules, together with the hardware tree, that should be printed.

Selectable are:

- Safety CPU
- Safe Modules
- Hardware Tree
- Unsafe
- Temp. Variables
- Revision documentation
- Memory variables
- Constants
- Macros
- Parameter Lists

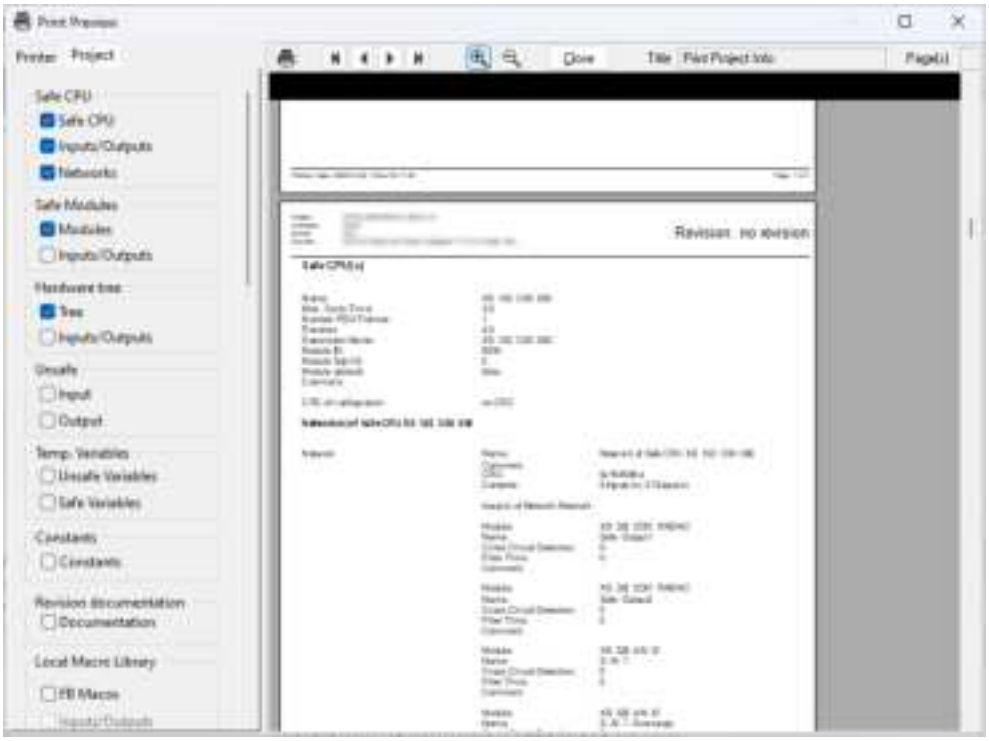


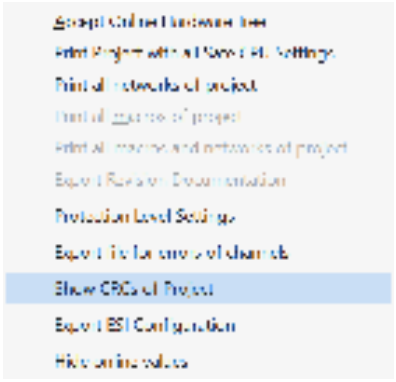
Abb. 2: Print example for Project Tree

4.30 CRC comparison dialog

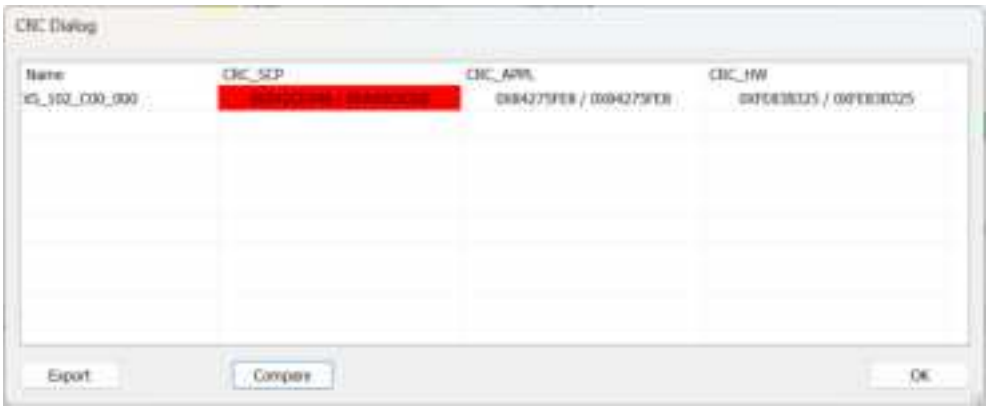
Three CRCs are calculated for a Safety CPU. One CRC represents the hardware layout, one represents the application, and the last one represents the whole compilation. These CRCs are not calculated until after a project build. After this build, the project will be saved again if the values of the CRCs have changed. These CRCs will be shown in the Properties pane for the Safety CPU, as will the online values.

Please note that moving modules can result in a change to the transmission time for the hardware tree (e.g., moving downstream of a CIV 512). This can change the response time of individual outputs. The application's CRC will not change, but the hardware tree CRC will. In this case, you will need to analyze the machine times again.

You can export and compare the CRCs. If the project has been compiled successfully, you will be able to access the CRC dialog box with the context menu in the Hardware Tree or Project Tree pane.



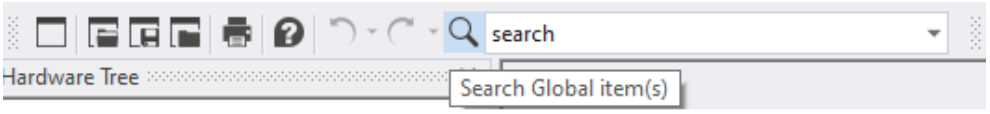
The dialog box will show the computed CRCs for each Safety CPU. Clicking on the Export button will export an XML file containing the information. The three CRCs for each Safety CPU and the date of creation in the file will be shown here. In addition, the file will be protected with its own CRC. Finally, clicking on the Compare button will allow you to import an exported XML file and compare it automatically to the current CRCs.



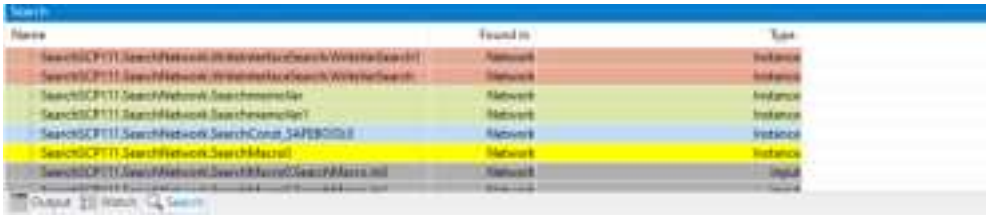
Red cells will be used to indicate a difference between the respective CRCs. Yellow cells mean that the data is only found in the imported file. White cells with a single CRC mean that the data is only found in the current configuration. Finally, white cells with two CRCs have identical CRCs in the configuration and imported file.

4.31 Global search

XN Safety Designer features a global search function.



The edit box shown above is a quick search that will use the last search setting that was used. You can confirm your selection with Enter, and the results will be shown in the Search pane.



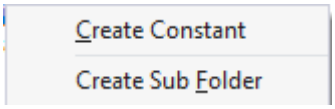
Pressing the Enter key or double-clicking will make the software jump to the corresponding elements.

You can set up a more detailed search by clicking on the button in the toolbar. The dialog box that appears will let you specify which areas you want to search.

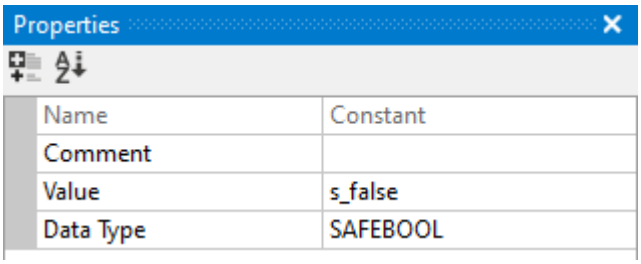
5. Constants

Constants are elements with a set value (you can set this value). You can place constants multiple times in a network.

To create a constant, use the context menu for the Constant(s) folder in the Variable Tree pane.



Each constant has a data type and a value. You can enter the corresponding data in the Properties pane for the constant.



You can drag and drop constants into a network.

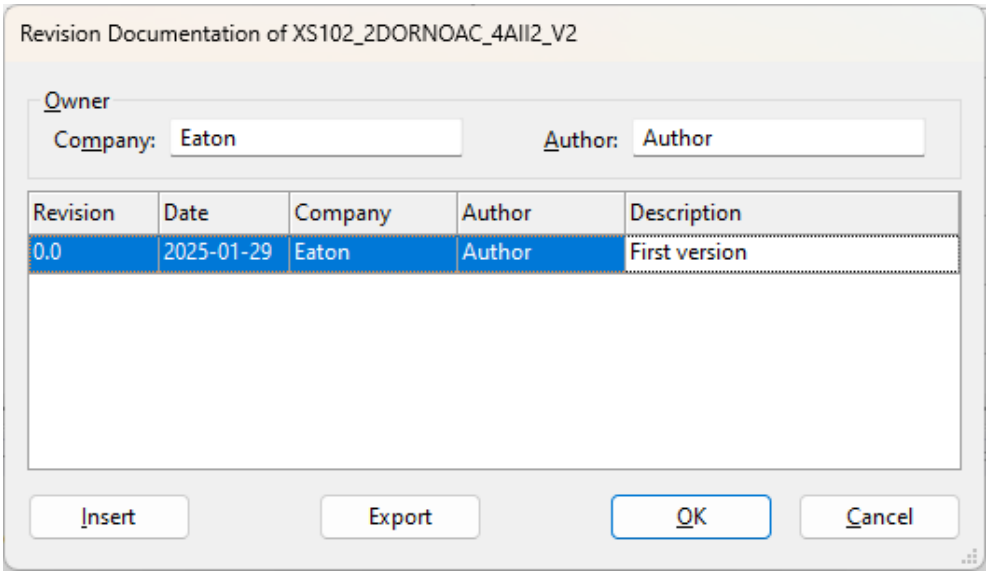
The differences from function block constants are as follows:

1. Constants can be placed multiple times.
2. All instances of a constant have the same value.
3. Constants cannot be used in function block macros.
4. Constants can be moved in parameter lists and vice versa.

6. Version control and archiving

6.1 Versioning

Every time you save a project, you will be asked to make a new entry in the revision documentation, that is, to enter a version number and a description of the version. Entering this information is not mandatory and will be up to you. If you want to export your entries to an XML file, simply click on the Export button.



6.2 Archiving

To save a project in such a way that it can no longer be modified, select the Archive Project option in the File menu. Once you do this, it will only be possible to load the project in read-only mode. Please note that if you need to make a change to an archived project, you will need to save it under a different name and work on the new copy instead. In addition, there is a save option that you can use so that it will not be possible to save copies of an archived project either.

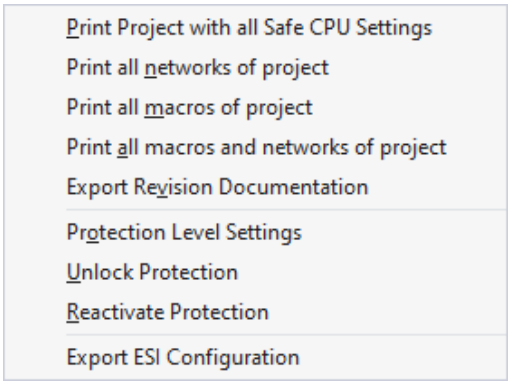
6.3 Saving as an XML file

You can use the "Save in XML-File" setting to save the project as an unencrypted XML file. The XML file will contain the CRC corresponding to the project, ensuring that the file is protected from tampering. You will also be able to use this XML file to open the project.

7. Protection level settings for password mechanism

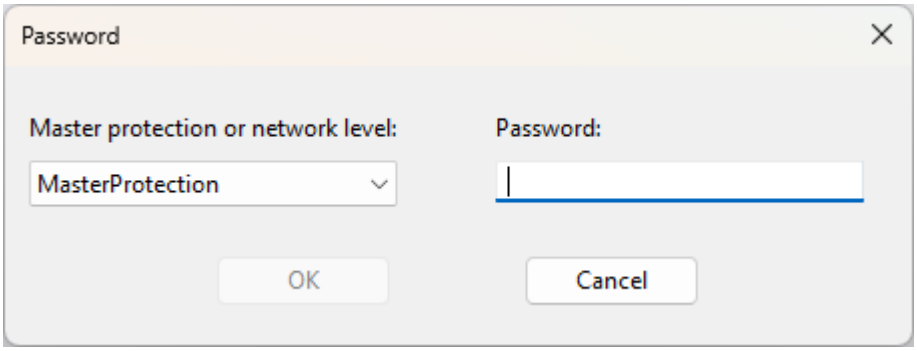
You can use a password to protect either the whole project or individual networks from changes.

You can access the Network Level dialog box used to create and change protection levels and passwords with the context menu in the Hardware Tree, Project Tree, and Variable Tree panes. You can also use the context menu to unlock and reactivate protection settings.

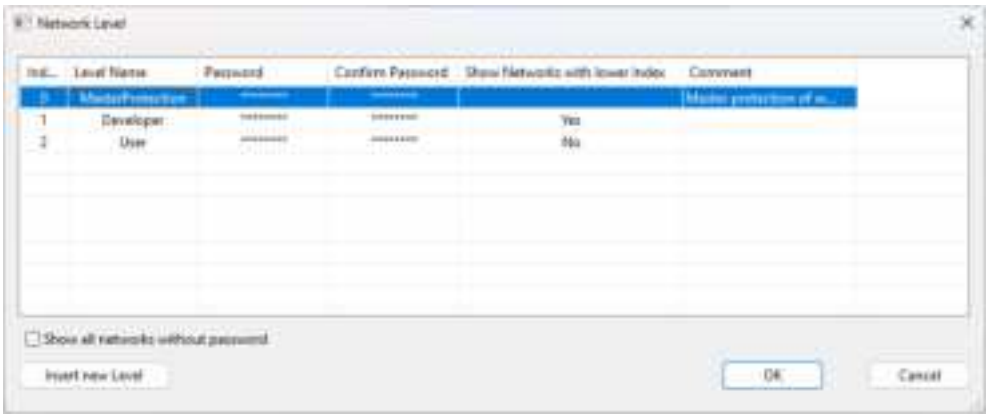


You will only be able to access the dialog box for the passwords and protection levels if the project is not write-protected or, in the event that it is write-protected, if you first log in with the MasterProtection password.

To log in, you will need to select the protection level and enter the password into the corresponding dialog box.



You can edit the Master Protection settings and the settings for individual networks in the Network Level dialog box.



Master Protection settings will already be pre-configured. All you will need to do is enter the password you want and confirm it (the password must be between three and eight characters long). The Master Protection level cannot be deleted. Please note that if you remove the password for Master Protection, all password protection for the project will be removed.

Once the Master Protection level has been configured, you can use the "Insert new Level" button to create individual levels for the various networks. When you click on the button, a level with the default index number and a default name will be created. The index number is unique and can be any number between 1 and 100 (and is editable). Levels will be sorted based on this number. You can also edit the level name, which must also be unique. Finally, you will need to enter a password for each level,and you can also enter a comment for each level.

If you enable the "Show all networks without password" checkbox, it will be possible to open all networks without a password. If the checkbox is disabled, you will be able to decide, for the individual levels, whether networks with a lower index can be opened in read-only mode.

The settings you configure in this dialog box will not be applied to the project until you click on OK. In turn, the OK button will not be enabled until after you have entered all required data (e.g., all network levels must have a password).

You can set the level you want for the individual networks in the Properties pane. The network will only be editable if the user has logged in to the project with the corresponding level. Logging in with a higher level (index number lower than the specified level) will also make it possible to edit the network. If you do not define a level for a specific network, it will only be possible to edit that network with the Master Protection password.

The tree for the networks will show networks that can only be opened with a specific protection password in blue.

8. Using unsafe variables through the CAN bus

8.1 Prerequisite for use

Using unsafe variables through the CAN bus is only possible in the Safety CPU's standalone mode.

Moreover, the Safety CPU must be connected to an appropriate counterpart through the CAN interface. If it is, the Safety CPU will be able to transmit and receive the unsafe variables through this interface.

8.2 Unsafe variable structure

The unsafe variables being sent through the CAN bus will have a size of eight bytes. These eight bytes can be subdivided in various different ways in Safety Designer:

- Two 4-byte values
- A 4 byte value and 32 bit values
- 64 bit values

The specific structure and size of the corresponding transmission data will depend on the use of these sub-variables. CAN data is always structured in such a way that the DINT values come first, followed by the BIT values. The size of the data will depend on which DINT values and BIT values are used. For example, if you use a variable with 64 BIT values, the bits used in the network will determine the size of the CAN packet.When bits 1 to 8 are placed, a CAN length of one byte will be used up, but if you use bits 57 to 64, the data size will be the full eight bytes. Meanwhile, if you use a four-byte data type (DINT) and bits 1 to 8, the packet will have a length of 5 bytes.

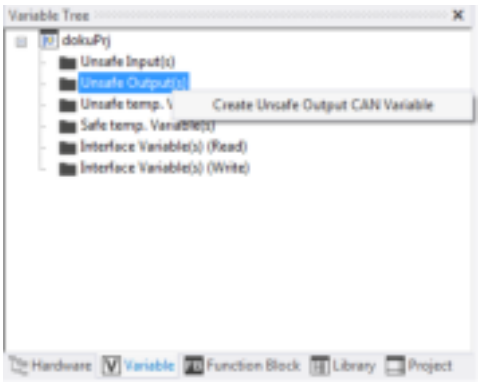
Make sure to use the CAN sub-variables in continuous order from bit 1 to bit 64 and to use DINT 1 before DINT 2. This will help you minimize the size of the CAN packets used.
For example, if you instead skip right to bits 57 to 64 and use only those eight bits, the CAN data length will still be eight bytes!

The CAN packets have a baud rate of 500,000.

8.3 Creating a project that uses unsafe variables through the CAN bus

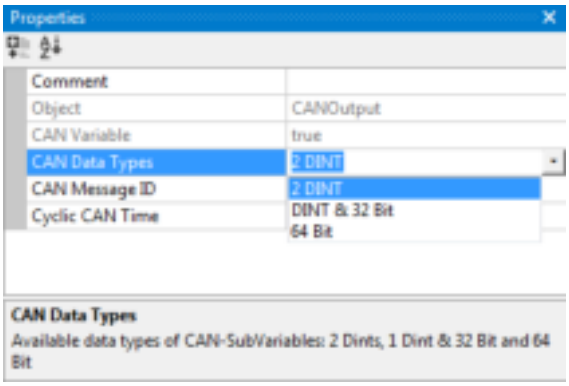
You must have created a standalone CPU in the hardware tree. When you create this Safety CPU, the option of creating unsafe variables that will be sent through the CAN bus is enabled.

You can use the context menu to create the unsafe input and output variables in the Variable Tree pane. Please note that the context menu will only be enabled up to the maximum number of variables (with the max. number being five input variables and five output variables).



The created variables will be subdivided into the corresponding sub-variables with DINT values and/or BIT values. This subdivision will be shown in the Variable Tree pane and in the Properties pane. Please note that when you create a variable, 2 DINT sub-variables will be created by default.

You can change how a variable is subdivided in the Properties pane. Please note, however, that you will only be able to change the subdivision if none of the sub-variables have been placed in a network yet.



Options:

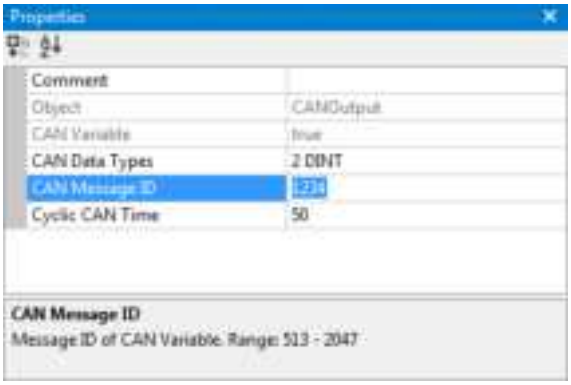
- 2 DINT
- 1 DINT and 32 BIT
- 64 BIT

Your selection will be shown automatically in the Variable Tree pane.

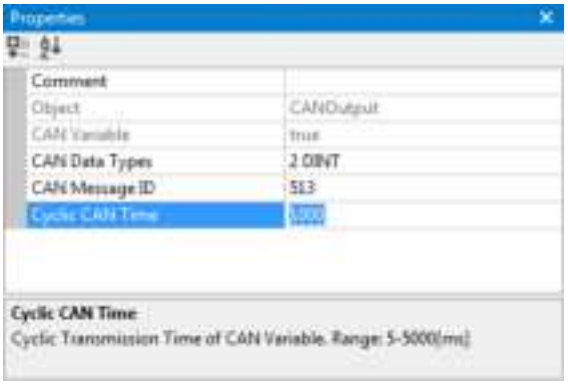
You can change the name of these variables and sub-variables in the Variable Tree pane. However, you will not be able to change their order.

A Message ID needs to be assigned to each variable. This Message ID is a unique ID for CAN communications. XN Safety Designer will assign Message IDs automatically starting from 513, but you can also set the Message ID you want in the Properties pane. Please make sure that there are no duplicate Message IDs and note that the ID range is between 513 and 2047.

→ The Message IDs must be globally unique in the entire system. Within this context, it is important to note that XN Safety Designer can only check uniqueness within a project. If multiple Safety projects are being used in a single system, you will need to check the Message IDs to make sure they are globally unique.



You can additionally set a cycle time for output variables in the Properties pane. The Safety CPU will transmit the corresponding unsafe variable at the set interval. Please note that the set cycle time can increase a bit, since data can only be written at the end of a Safety CPU cycle. In other words, the actual transmission time will consist of the cycle time plus the configured transfer rate. As a recommended value, you can set a value between 5 and 5000 [ms]. The default value is 50 [ms].



You can drag the sub-variables and drop them into the network to place them. While the outputs can only be used once, inputs can be placed multiple times. The cursor icon will let you know whether you can place a sub-variable.

The screenshot above shows CAN variables being used in a network.

It is important to keep in mind that the variables being sent through the CAN bus contain unsafe data.

Using unsafe variables affects the Safety CPU's maximum cycle time. The reason for this is that the CAN packets need to be put together and transmitted / received. XN Safety Designer takes this into account when calculating the Safety cycle time. Please note that the CAN data packets will be transmitted at the end of a CPU cycle.

8.4 Displaying the variables in online mode

In online mode, the values of variables will be shown in the network and in the Watch pane. To show these values in the Watch pane, simply drag them from the network and drop them in the Watch pane.

Bit variables can assume a value of true or false. DINT variables have a numeric value. Values cannot be forced.

8.5 Transferring status information to the Safety CPU through the CAN bus

CAN Message ID 2047 is used by default to read status information for a standalone Safety CPU through the CAN bus.
Meanwhile, CAN Message ID 511 is used to send a response.
To change these values, you can enter the IDs you want from the range of CAN variables (513–2046) in the Property pane for the CPU. The corresponding parameters are CAN Request ID and CAN Reply ID. These numbers will then be blocked so that they cannot be used in variables.

9. Use of unsafe variables in stand-alone operation

In standalone mode, the user must take care of the process of writing unsafe variables to a Safety CPU and reading unsafe variables. The corresponding communications consist of SDO communications and must be structured accordingly in order to be able to exchange data with the Safety CPU.

9.1 SDO communication

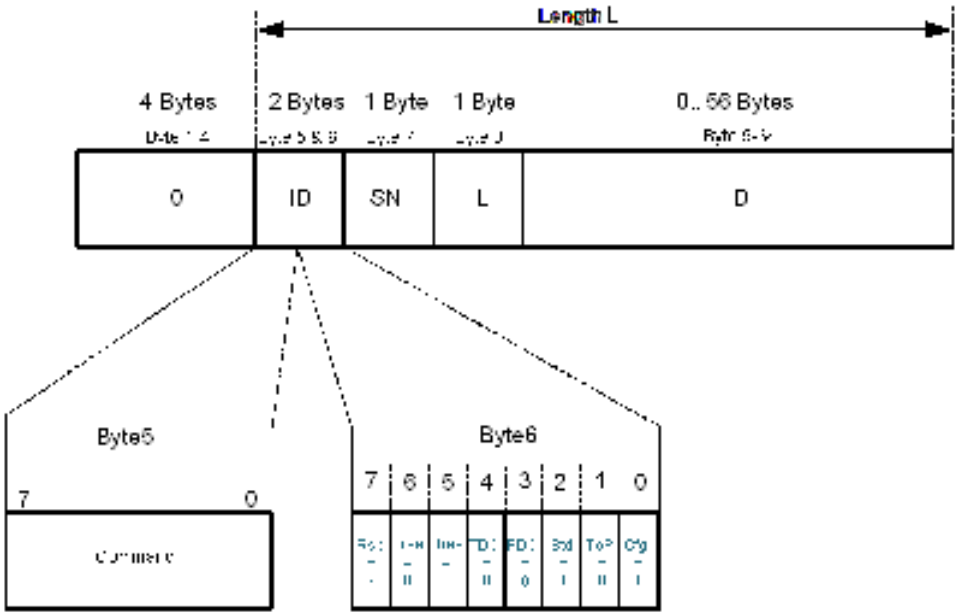
SDO communications can be used to send various commands to the Safety CPU. These commands will be processed by the Safety CPU, after which the corresponding response will be sent back.

SDO command examples:

- Reading unsafe variables
- Writing unsafe variables

For writing and reading SDO frames, the Safety CPU features an SDO write buffer with an SDO write status register and an SDO read buffer with an SDO read status register.

9.2 SDO frame structure



The SDO command being used is entered in the Command field (refer to the overview).

The frame type is specified in byte 6, and is always 00000101_2 ($=5_{10}$) for the request and 10000101_2 ($=133_{10}$) for the response when it comes to unsafe SDO requests.

This is followed by a sequence number in the SN field that is incremented with every new request.

Meanwhile, the L field contains the length of the entire frame without counting the CRC field that is always filled with zeroes (the first four bytes).

The length field is followed by the data, which varies depending on the specific command in question.

The Safety CPU's firmware will mirror this frame structure for the response (same command, same sequence number, and response bit set in the type field), add the command-specific response data to the data field, and then set the length accordingly.

9.3 Writing an SDO frame

The SDO write status register (offset: 0x01D1 for XS-102-C00-000) will be read. If bit 7 in this register has a value of 9, then the SDO write buffer (offset: 0x00C0 for XS-102-C00-000) can be written to.

9.4 Reading an SDO frame

The SDO read status register (offset: 0x01D0 for XS-102-C00-000) will be read. If bit 7 in this register has a value of 1, then the SDO read buffer (offset: 0x0188 for XS-102-C00-000) contains data. The number of bytes in the buffer will be found in buts 0 to 6 in the SDO read status.

9.5 Overview of SDO commands

Cmd	Designation	Description
0	READ	Used to read a memory area (reading the configuration, cyclical check of config CRC) → "Detailed description of SDO READ command", page 133
1	WRITE	Used to write to a memory area → "Detailed description of SDO WRITE command", page 133
4	GET_STATE	Returns the current runtime state, config state, login level, and saved CRCs from the configuration → "Detailed description of SDO GET_STATE command", page 134
6	GET_DIAG_INFO	Returns diagnostic information (error code if there is an error present) → "Detailed description of SDO GET_DIAG_INFO command", page 135
15	GET_SAFENBR	Returns the safety number of a module → "Detailed description of SDO GET_SAFENBR command", page 136
17	GET_FIRMWARE_VER	Returns the firmware version → "Detailed description of SDO GET_FIRMWARE_VER command", page 136
19	NOP	Dummy command for clearing the Out buffer in a Safety module → "Detailed description of SDO GET_NOP command", page 137
28	READ_VALUES	General read command for reading one or more memory areas in a virtual memory area → "Detailed description of SDO READ_VALUES command", page 138
30	WRITE_VALUES	General write command for writing to one or more memory areas in a virtual memory area → "Detailed description of SDO WRITE_VALUES command", page 138

9.6 Detailed description of SDO READ command

General read command for reading a memory area. You can use this command to read the configuration and unsafe variables. The address is used to specify which data should be read. To read an unsafe variable, you will need the offset for that variable in the application memory. You will then need to add the offset to the application memory address.

The variable offset is determined by parsing the configuration data.

Command: 0

Request data	
4 byte	Virtual address of memory area
1 byte	Number of bytes to be read

Response data	
1 byte	Return code
1 byte	Number of read bytes. A value of 0 indicates that there is no additional data.
n Byte	Read data

9.7 Detailed description of SDO WRITE command

You can use this command to write to unsafe variables. The address is used to specify which data should be written to. To write to an unsafe variable, you will need the offset for that variable in the application memory. You will then need to add the offset to the application memory address.

The variable offset is determined by parsing the configuration data.

Command: 1

Request data	
4 byte	Virtual address of memory area
1 byte	Number of bytes to be written
n Byte	Data that will be written

Response data	
1 byte	Return code

9.8 Detailed description of SDO GET_STATE command

Returns the current runtime state, config state, login level, and saved CRCs from the configuration. The CRCs from the configuration will only be valid if the config state has a value > NOT_CONFIGURED.

Description of config states:

Designation	Value	Description
UNKNOWN	0	Unknown (e.g., while the firmware is being downloaded)
INVALID	1	Invalid (if, for example, an error has been detected in the configuration)
NOT_CONFIGURED	2	Not configured, i.e., the configuration has been explicitly deleted.
CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED	4	Configured, but not deployed and not verified.
CONFIGURED_AND_VERIFIED	8	Configured, deployed, and verified
CONFIGURED_DEPLOYED_NOT_VERIFIED	16	Configured; deployed but not yet verified
CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED_DEV	36	Configured, but not deployed or verified + Developer Mode
CONFIGURED_DEPLOYED_NOT_VERIFIED_DEV	48	Configured; deployed but not verified yet + Developer Mode

Command: 4

Request data	
1 byte	optional: Flags

If bit 0 in set in Flags, then the counter for the change to the device addresses will be sent. Otherwise, it will not be sent.

Response data	
1 byte	Return code
1 byte	Runtime state 1 POST 2 SERVICE 4 ERROR 8 IDLE 16 CHK_CFG 32 OP_TEMP 64 OP

Response data	
1 byte	Config state 1 INVALID 2 NOT_CONFIGURED 4 CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED 8 CONFIGURED_AND_VERIFIED 16 CONFIGURED_DEPLOYED_NOT_VERIFIED 36 CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED_DEV 48 CONFIGURED_DEPLOYED_NOT_VERIFIED_DEV
1 byte	Login-Level
4 byte	Configuration container CRC. When a Safety CPU is involved, the configuration container will contain its own configuration and the configurations that must be transmitted to the remote modules.
4 byte	CRC of the configuration's list header.
2 byte	I/O status error counter. Is incremented every time the error state of a local input or output changes.
optional	
4 byte	CRC of the interface frame created by the module. A value of 0 means that an interface frame will not be created.
	Counter for the change to the device addresses in the address assignment service (can be retrieved at address V_ADDR_FSOE_DEVADDR). This value will only be sent if bit 0 is set in the Flags element in the request.

9.9 Detailed description of SDO GET_DIAG_INFO command

Returns diagnostic information. The diagnostic information returned by the function can be interpreted with the help of the error code list.

Command: 6

Request data	
1 byte	Used to select µC1 or µC2 0 µC1 ≠0 µC2

Response data	
1 byte	Return code
1 byte	Used to select µC1 or µC2 0 µC1 ≠0 µC2
1 byte	Current error code
1 byte	The first error that triggered the error condition
4 byte	Reason code 0 corresponding to the error code (first error) (What the reason code means will depend on the specific error code)
4 byte	Reason code 1 corresponding to the error code (first error) (What the reason code means will depend on the specific error code)

9.10 Detailed description of SDO GET_SAFENBR command

Returns the safety number for a module.

This command has a unique characteristic in that a separate SSDO type (SSDO-Cfg or SSDO-Mod Frame) is used in which the specified target address is not the module safety number, but rather the topology path or a compressed version of the topology path.

A Safety module is unable to interpret what the topology path means and how it is structured. If a C_GET_SAFENBR request with a topology path as the target address is received by a Safety module, the topology path will not be interpreted, and a response with the module's own safety number as a sender address will be returned.

Command: 15

Request data	
	None

Response data	
1 byte	Return code
4 byte	Module safety number

9.11 Detailed description of SDO GET_FIRMWARE_VER command

Returns the firmware version

Command: 17

Request data	
	None

Response data			
1 byte	Return code		
8 byte	Firmware Version 4 byte Minor version If the module is in the "Bootloader" runtime state, then the minor version will contain the bootloader version.	2 byte Major version	2 byte Module type
4 byte	Boot loader version		

9.12 Detailed description of SDO GET_NOP command

Dummy command for clearing the Out buffer in a Safety module.

This command is required by the standard PLC in order to clear the Out buffer in a Safety module after a timeout. After the response to this command is received, the PLC can assume that there are no responses to an earlier command stored in the Safety module's Out buffer.

Required login level: 0

Command: 19

Request data	
	None

Response data	
1 byte	Return code

9.13 Detailed description of SDO READ_VALUES command

General read command for reading one or more memory areas in a virtual memory area.

In contrast to C_READ, this command makes it possible to specify multiple addresses at the same time. The size of the memory area specified with an address will be implicitly defined by the type of virtual memory. For the application memory, this size is four bytes (corresponding to the size of a cell in application memory).

Command: 28

Request data	
1 byte	Number of following address(es)
n * 4 byte	Virtual addresses of memory area

Response data	
1 byte	Return code
1 byte	Number of values read
n * x Byte	Read memory areas, with x bytes each. For the application memory, x has a value of 4.

9.14 Detailed description of SDO WRITE_VALUES command

General write command for writing to one or more memory areas in a virtual memory area.

The address of the memory area in this command is not a physical address, but a virtual address instead. The values of the virtual addresses can be found in the description of the service commands.

In contrast to C_WRITE, this command makes it possible to specify multiple addresses at the same time. The size of the memory area specified with an address will be implicitly defined by the type of virtual memory. For the application memory, this size is four bytes (corresponding to the size of a cell in application memory).

Command: 30

Request data	
1 byte	Number of following address(es)
n * (4+4) Byte	Virtual addresses of the memory areas and their values

Response data	
1 byte	Return code

9.15 Important Safety CPU virtual addresses

Adress offset	Designation	Description
0x2000 0000	CONFIG_DATA	Base address for configuration data
0x2010 0000	SER_NR_DATA	Base address for serial and safety numbers
0x2070 0000	OPTIONAL_CFG	Optional flags and the IDs of the expected modules can be read here. They are structured as follows: 2 byte Optional flags (each bit indicates whether the module at the corresponding slot downstream of the Safety CPU has been configured as optional) 1 byte HBG optional flag (bit 0 indicates whether the hand-held terminal has been configured as optional) 16 byte Array with 16 single bytes for the IDs of the modules expected at the respective slot 1 byte Expected ID for the hand-held terminal
0x3000 0000	APP_DATA	Application memory The application memory is the RAM that is required when processing the safety application. You can determine the values of variables by reading from this area and force values by writing to this area.
0x3400 0000	UNSAFE_BDINT	Address range for unsafe BDINT variables (32 bits, where each individual bit can be separately connected in Safety Designer).

9.16 Configuration block types

Number	Designation	Description
19	CFG_CONTAINER_ V3	Block type of entire configuration (includes module configurations)
21	CFG_REV	Safety Designer project name and revision number string
40	CFG_MODULE _V3	Block type of a module configuration (incl. SafeDINT).

9.17 Reading the configuration

The configuration consists of four-byte values. However, strings are an exception and the specified length must be used there.

At the beginning, the first 12 bytes in the configuration are read from the CONFIG_DATA address: CRC, length of entire configuration, and block type of configuration (CFG_CONTAINER).

If the block type of the configuration is CFG_CONTAINER_V3, then there will be a dynamic header right after the data read so far (the length of the data will be specified in the header's first four bytes = configuration offset: 0x0C).

The length of the dynamic header will be added to its address in order to get to the first list entry.

If the block type of the configuration is not CFG_CONTAINER_V3, then the first list entry will start four bytes after the data read so far (configuration offset: 0x10).

Based on the length of the entire configuration as an upper limit for the offset, a loop will then start in order to go through the individual lists.

General structure of a list entry (independently of block type)

4 byte	Length of module path, followed by the path string with the corresponding length.
4 byte	Length of compressed module path, followed by the path string with the corresponding length.

After this, the memory contains the CRC, total length, block type, and ModuleID of the current list. To get to the next list, add the total length value to the total length offset.

9.18 Structure of the individual list entry block types

9.18.1 CFG_MODULE_V3

This block type describes a module configuration, including safe and unsafe inputs and outputs.

How to calculate the addresses in the application memory for later access to the unsafe variables (the memory structure results from the configuration memory parsed here):

The address starts with 4 (0–3 used for time).

Each SafeBool takes up four bytes, and each SafeDINT takes up eight bytes.

In the application memory, the bit variables (unsafe BDINT) take up four bytes per bit used.

The addresses must be added up for the individual elements in line with their usage. For later access to the unsafe variables, you will need to add, to the respective offset, the base address (APP_DATA or UNSAFE_BDINT for unsafe BDINT variables) for access to the application memory.

Block structure:

	Interface frame area (there are no frames in standalone mode).
	Structure: Four-byte CRC
4 byte	Path length
	Path
	4 Byte length comp. Pfad
	Comp. path

FSoE connection area (there are no connections in standalone mode); only the length needs to be read here. The length takes up four bytes.

Area with the COPY connections and the CALL calls for the function blocks. This area is ended with the END command.
The end command is represented with the number 2.
16 bytes are required per call.

Read four bytes and check the value. If the value is 2, you have reached the end of the area. Make sure to always increase the address by 16 bytes.

Local input area (SafeBool).
The number of inputs will be at the beginning (four-byte value).
16 bytes are required for each quantity.

Output as input area (SafeBool).
The number of outputs will be at the beginning (four-byte value).
Four bytes are required for each quantity.

Safe output area (SafeBool).
The number of outputs will be at the beginning (four-byte value).
Four bytes are required for each quantity.

Local input area (SafeDINT).
The number of inputs will be at the beginning (four-byte value).
16 bytes are required for each quantity.

Output as input area (SafeDINT).
The number of outputs will be at the beginning (four-byte value).
Four bytes are required for each quantity.

Safe output area (SafeDINT).
The number of outputs will be at the beginning (four-byte value).
Four bytes are required for each quantity.

Fast unsafe output area.
The number of outputs will be at the beginning (four-byte value).

Unsafe output area.
The number of outputs will be at the beginning (four-byte value).

Per output:	4 byte length of the name
	Name of the variable

Unsafe output area (BDINT).
The number of outputs will be at the beginning (four-byte value).

A four-byte bit mask per output.
The bit masks for all outputs will be before the actual outputs (for example, if there are three unsafe outputs, the 3 x 4 bytes for the bit masks will be first, followed by the name length + names).

Per output:	4 byte length of the name
	Name of the variable

Safe input area (SafeBool).
The number of inputs will be at the beginning (four-byte value).

Per input	4 byte path length
	Path
	4 Byte length comp. Pfad
	Comp. path
	4 Byte Index

Safe input area (SafeDINT).
The number of inputs will be at the beginning (four-byte value).
Per input 4 byte path length
Path
4 Byte length comp. Pfad
Comp. path
4 Byte Index

Safe interface input area (SafeBool).
The number of inputs will be at the beginning (four-byte value).
Per input 4 byte path length
Path
4 Byte length comp. Pfad
Comp. path
4 Byte Index

Safe interface inputs (SafeDINT).
The number of inputs will be at the beginning (four-byte value).
Per input 4 byte path length
Path
4 Byte length comp. Pfad
Comp. path
4 Byte Index

Fast unsafe input area.
The number of inputs will be at the beginning (four-byte value).

Unsafe input area.
The number of inputs will be at the beginning (four-byte value).
Per input: 4 byte length of the name
Name of the variable

Unsafe input area (BDINT).
The number of inputs will be at the beginning (four-byte value).

A four-byte bit mask per input.
The bit masks for all inputs will be before the actual inputs.
Per input: 4 byte length of the name
Name of the variable

...

9.18.2 CFG_REV

This block type contains a string (block length) containing the project name and the project revision (the two values will be separated by a tab character).

The length of the string will be the total length of the list minus 12 bytes (= header without CRC, since the CRC is not included in the length).

9.19 Status of expansion module I/Os

The status of the inputs/outputs can be read directly from the Safety CPU address range (0x008A) (i.e., not with SDO commands).

The configuration of the I/O status register is not fixed, but instead depends on how the modules are plugged in. First come the I/Os of the CAN HBT module, then the I/Os of the connected local inputs, and then the I/Os of the connected local outputs:

Byte 0:

Bit 1…0 Validation Button State
Bit 7…2 free

Byte 1:

Bit 15…8 Status LEDs (Can, Run, Service, Error)

Bits LED

0.1 Can
2.3 Run
4.5 Service
6.7 Error

Byte 2-3:

Bit 17 … 16 I/O Status SafetyCan (HBG) Input 1
...
Bit 31 … 30 I/O Status SafetyCan (HBG) Input 8

Byte 4..67

Bit 32 Status of connected I/Os
…543

1. local inputs
2. Local outputs (digital outputs and relay outputs may be intermixed)

There will be two bits with the following meaning for each input, output, and status:

Input, Output:

b00	not available
b01	Fault State
b10	Low
b11	High

Button-State, Status LED:

b00	Slow flashing
b01	Fast flashing
b10	AUS
b11	ON

9.20 Detected expansion modules

You can use the I/O modules register (0x01D4) to find out which expansion modules are plugged in.

9.21 Fast non-safe input and output variables

For the XS-102 Safety CPU, you can read and write to up to 32 unsafe bit input and output variables directly (without SDO commands) (0x0080 of read and write areas respectively).

9.22 Unsafe release signals for local outputs

You can connect local safe outputs to an unsafe enable signal with an AND operator. The enable signals are handled with offset 0x0088 for the write area on the module.

10. Determining the sequence for the compilation of a safety CPU

→ "Order of the networks", page 146
→ "Sequence of instances within the networks", page 147
→ "Showing the processing sequence", page 147

10.1 Order of the networks

A distinction must be made here between several cases

Case 1: There are no temporary variables being used in the networks

The order will be determined based on the network names (in ascending alphabetical order).

Case 2: Temporary write variables are being used

The order will be determined based on the names of the temporary variables (in ascending alphabetical order). In other words, the network that will be processed first will be the one that has the temporary variable with the first letter (in ascending alphabetical order).

Case 3: Mix of networks with temporary write variables and without temporary variables

The networks with temporary write variables will be processed first (in the temporary variables' ascending alphabetical order) and the other networks will be processed afterwards (in the networks' ascending alphabetical order).

Case 4: Mix of networks with temporary variables (read and write) and without temporary variables

The networks with temporary write variables will be processed first. The points of use of the temporary variables will be checked against each other, meaning that the order no longer has to be sorted alphabetically. Please note that you must always make sure that the network with the temporary write variables comes before the networks with the temporary read variables.After these networks come the networks without temporary variables or with temporary read variables only (in the networks' ascending alphabetical order).

10.2 Sequence of instances within the networks

Within the networks, the system generally works from the top left to the bottom right.

Starting from the topmost function block on the left (first row, then column), the system will look to the right until it finds an end (e.g., output, temporary write variable). From there, it will then look back for all required function blocks to the left. This means that the processing sequence will be inverted. The list is then simply inverted, resulting in the order in which the function blocks must be processed.

All the function blocks that have already been processed will be flagged, and the process will then start again with the next function block on the top left that has not yet been processed.

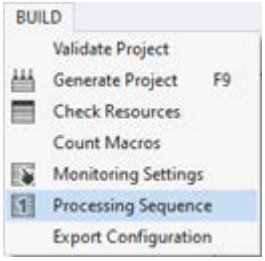
This process will be repeated until all function blocks have been processed.

10.3 Showing the processing sequence

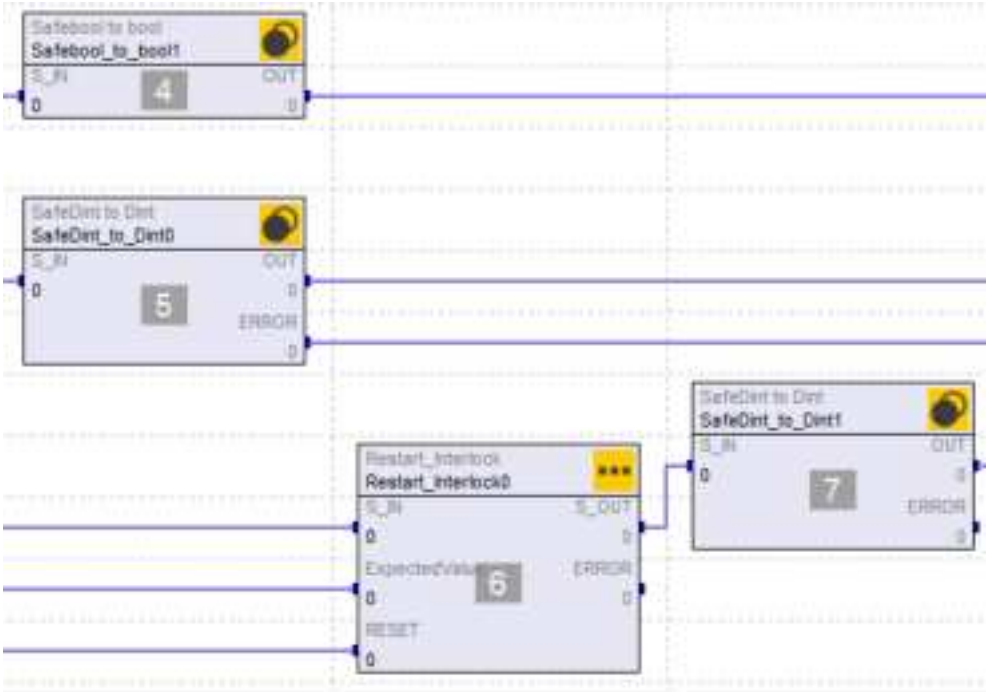
You will not be able to view the processing sequence until after successfully compiling the project. If you make any changes after that, you will need to compile the project again.

To show the sequence, click on the button in the Build toolbar.

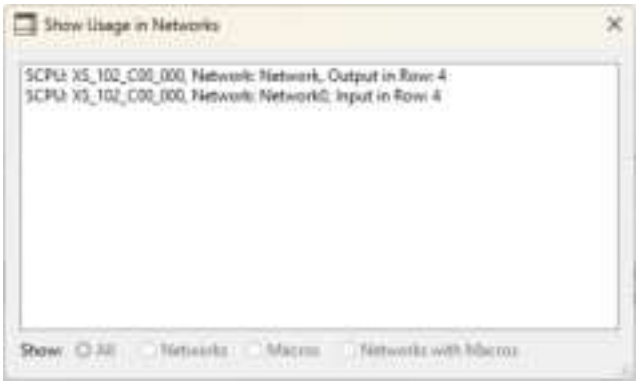
You can also show it by selecting the Processing Sequence option in the Build menu.



The processing sequence will be shown in the network with numbers highlighted in gray.



The Go to Previous Output(s) context menu option will take you to the point in another network where the values were set. If there are square brackets, that can mean that there are multiple networks involved. In that case, a dialog box with all points in question will appear.



Double-clicking on the entry you want will take you to the corresponding point.

11. Online actions and download

You can open the "Online State" dialog box after establishing an online connection.

This dialog box will show all Safety CPUs with their status. This status will be continuously refreshed.

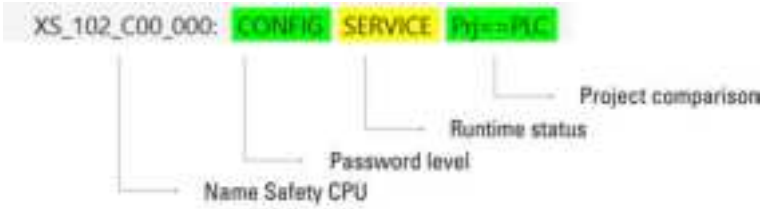
This applies to the following information:

- Password level
- Service Mode Status
- Configuration status
- General information (e.g., firmware version, Safety CPU safety number)

The following information will be shown:

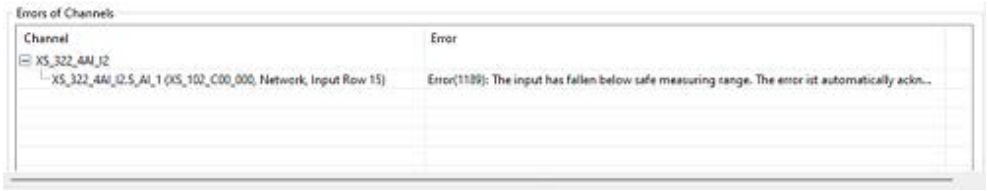
- Safety CPUs without modules in slave mode
- All Safety CPUs
- Safety CPU name
- Safety CPU status
- Download, project status
- Cycle time
- Project comparison
- The LEDs will reflect the status. These LEDs can also be seen on the corresponding hardware.
- Password level
- Firmware version of Safety CPU
- Safety number of Safety CPU (check to make sure that the number is correct)
- Project name and project version

After an online connection is established, the status bar at the bottom of the dialog box will show the state of the Safety CPUs. This status bar will show the name of the Safety CPU, the password level, the runtime state, and the project comparison for the Safety CPU.



The online editor will show a pane containing the errors in affected module channels. You can double-click on a line to select the corresponding element in the network. This error pane is needed, since a module will not go into its general error condition if individual channels are experiencing an error. Each channel and its errors will be listed in the pane, and you can reset these errors with the Quit Error command, after which the Safety module will attempt to resume processing.

In addition, the pane will show the slave modules' errors / states. This information is necessary in order to be able to see slave module errors and states faster. Slave modules with an error will be shown in red. Moreover, the IDLE, Check configuration, and Service runtime states will be shown. Finally, you can double-click to select the corresponding element in the hardware tree in order to get more detailed information.



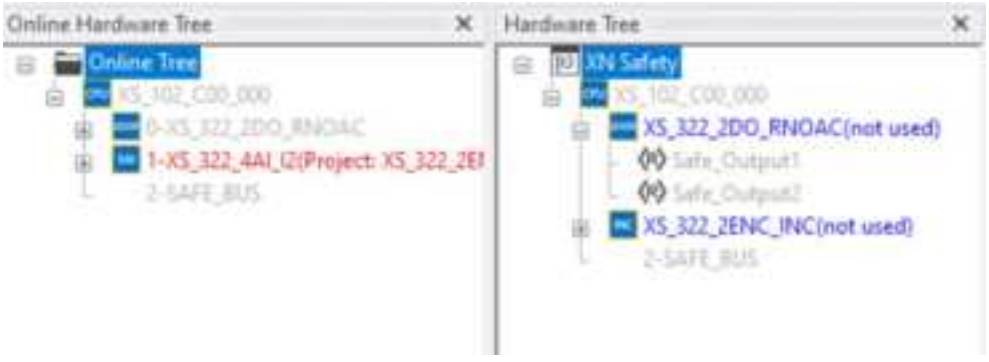
11.1 Target/actual comparison of the hardware

XN Safety Designer features a function that you can use to compare the hardware path (topology path) of the Safety modules in the Safety project with the Safety modules that are actually physically present with the PLC. You will only be able to configure the Safety CPU if the comparison result is positive. In other words, all the Safety CPUs configured in the project must be physically present at the corresponding spot in order for you to be able to configure the Safety CPU.

This comparison will be carried out automatically after establishing an online connection. If the comparison fails, a hardware tree showing the modules that are physically present will be opened. You can compare this tree to your configuration in the Safety project in order to correct any differences (please note that the error can be due to an incorrectly connected hardware module on the machine or to an erroneous configuration in XN Safety Designer).

The differences will be indicated with color coding in the hardware tree (red for different modules at the same spot, brownish for modules that are only found on the PLC, and blue for modules that are only found in the project). In addition, texts will be appended in order to make the color coding clearer. Finally, the position (slot number) will be shown at the beginning of all modules in order to make comparisons easier for expansion modules, for example.

You can use the context menu to copy individual modules from the tree to the project. Please note, however, that the context menu will only appear if copying the corresponding module to the project is possible to begin with. You will be given the option of inserting the module or, if the modules are different, of replacing the module in the project. Copying to the project is done in online mode, but the online dialog box will be closed after a copy operation, since this operation will modify the project.

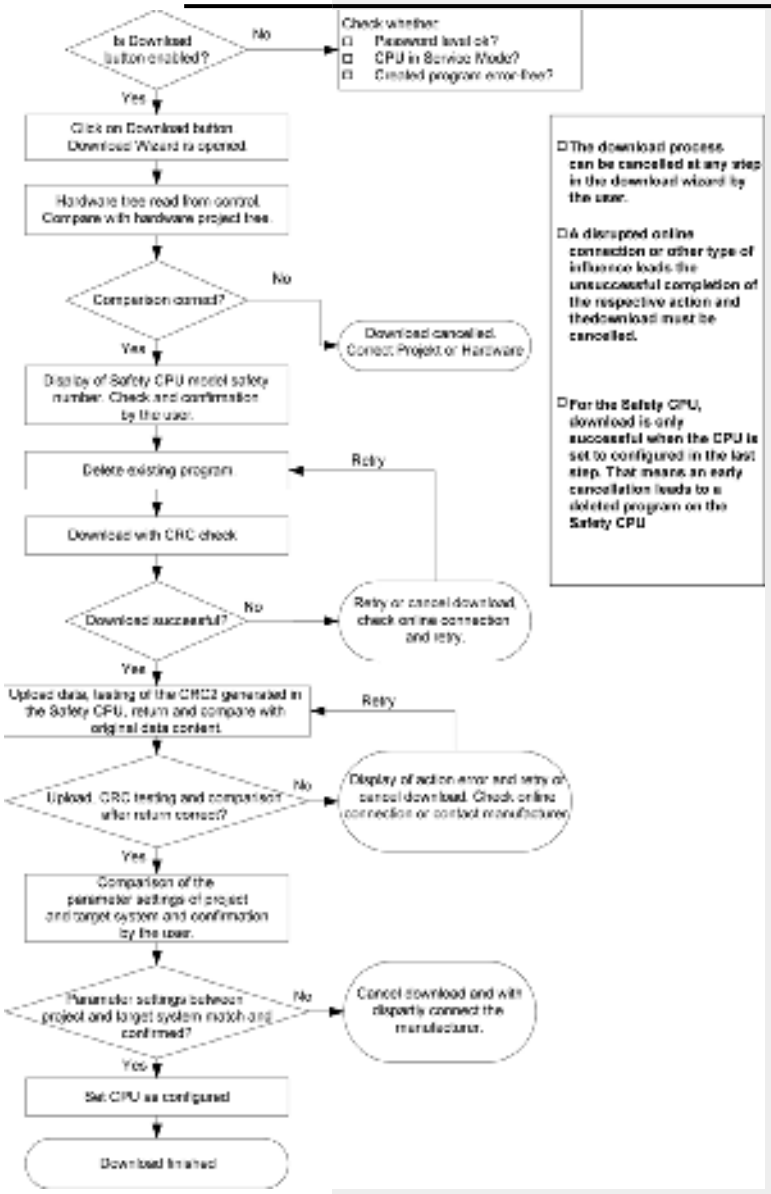


In addition, the XN Safety Designer hardware tree will show the unique module safety number for a specific Safety module in the Properties pane. You can also check it with the physically present hardware.

11.2 Online actions

With the exception of "Quit Error," "Service Mode," and "Start," the online actions described below all require the "Configuration" password level. "Quit Error" is available without a password level, while "Service Mode" and "Start" are available starting from the "Debug" level. For an overview of the actions that can be used with a password level, please refer to → " Passwords", page 157.

Button	Action
Reset/Start PLC	You can use a button in the debug toolbar to select whether the PLC should be reset / started. If you click on the button, the PLC will be reset before a download. It will also be reset and restarted before the safety application starts. The button status will be stored in the registry.
Quit Error	If there is an error present on the Safety CPU or on modules that the Safety CPU uses, this action will confirm the error. Please note that errors in slave Safety CPUs need to be confirmed with a Quit Error on the master Safety CPU. After you run a Quit Error command, you will be automatically logged back in to the Safety CPU to which you were logged in before the command.
Service Mode	Switches the Safety CPU to Service Mode. This mode is required in order to run a Download, Clear, Set Verified, Firmware Update, etc. command.
Start	Exits Service Mode on the Safety CPU. Depending on the specific system state, the Safety CPU will switch to Running Mode, Temporary Running Mode, Error Mode, or Idle.
Clear Program	Deletes the program on the Safety CPU. Deleting the configuration in slave modules with XN Safety Designer is not possible. In order to delete the configuration, the module must either be set to master mode or run as a standalone CPU.
Set Verified	This will complete the program check in the Safety CPU with a positive result. Please note that you will need to run this confirmation twice. With this action, the Safety CPU will switch from temporary processing to indefinite processing. When running this command on Safety CPUs, this process will also check which optional modules are connected, and if any optional modules are missing, a warning will be shown. The prompt will let you know that connecting optional modules during operation after the fact can trigger a fault.
Download	The following conditions must be met in order for the Download button to be enabled: ■ An error-free code must have been generated. ■ The password level must be valid. ■ The Safety CPU in question must be in Service Mode.

Button	Action
	After the upload, you will need to confirm all the program parameters (filter time, etc.) that are not immediately apparent in the network view. A separate dialog box will show the parameters. The uploaded values will be compared to the values in the project, and you will need to enable a checkbox to confirm that the values are correct.It will not be possible to set the Safety CPU to the configured state unless you first confirm that the values are correct. This confirmation may be omitted if you are logged in in "Developer Mode" (please refer to the "Login" button section). In "Developer Mode," downloads are carried out without the wizard, meaning that individual confirmations from the user are not required. In addition, when Developer Mode is active, the PLC will only be able to run in Temporary Operation Mode. To switch to the standard (indefinite) Operation Mode with the "Set Verified" button instead, a download with a confirmation of the individual parameters is required.

Button	Action
Login	See also chapter.→ " Passwords", page 157 Used to log in with a password to a Safety CPU. If a password has not been assigned yet, you will first need to assign one (with Change Password). If a corresponding password is found on the Safety CPU, the system will check to see if the login credentials are correct. When you log in in the "Configuration" level, you can set the "Developer Mode" option in order to make it possible to download data faster during development. If Developer Mode is enabled, you will not need to confirm the download steps individually, but the Safety CPU operation mode will be limited to Temporary Operation Mode.In other words, the "Set Verified" button will not be available if you have carried out a download with Developer Mode enabled and switching to standard (indefinite) Operation Mode will not be allowed. When logging in in the "Configuration" level in order to make program changes, you will only be able to log in to one single Safety CPU. In other words, XN Safety Designer will prevent attempts to simultaneously log in to multiple Safety CPUs in the "Configuration" level. The login state in a Safety CPU will be temporarily stored for the duration of an online session; in other words, you will only need to log in once, after which XN Safety Designer will automatically log you in (if you use the general functions).
Change Password	When changing the password, the old password must be entered. Each new password must be confirmed by entering it twice. When you first use the system, a password will not have been assigned to the Safety CPU yet (the Safety CPU comes without a password when supplied). You will accordingly have to set a password on the Safety CPU. In this case, the input field for the old password will be empty and you will simply need to fill out the input field for the new password and the field used to confirm the new password.

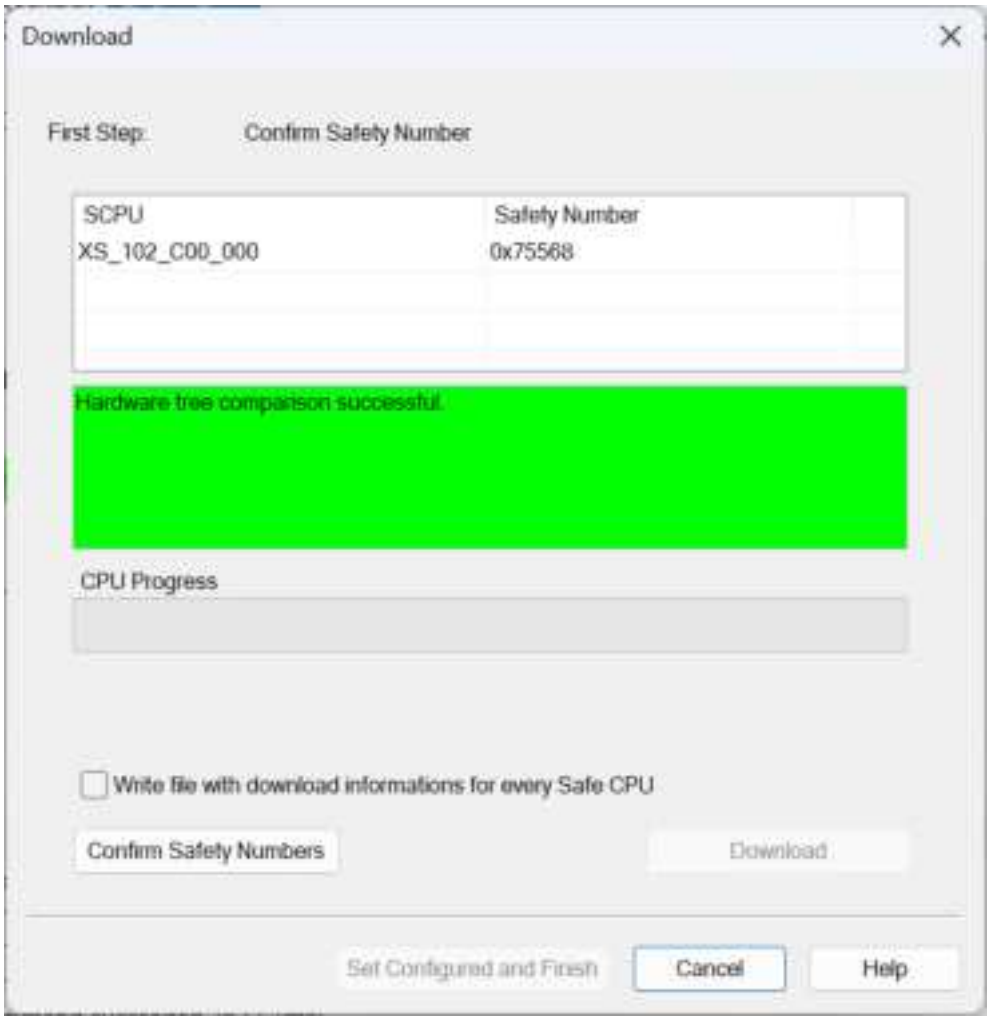
11.3 Download Wizard

The download wizard appears as soon as a project is to be downloaded to the CPU.

The wizard is subdivided into two steps.

11.3.1 Step 1: Compare the hardware tree and check the safety number of the module.

In this step, XN Safety Designer will compare the specified modules in your project with the modules that are actually physically present on the PLC. If a module needed by the Safety CPU is not present, a warning will be generated. However, you will still be able to carry out a download to the Safety CPU, since the download only requires the Safety CPU itself. After checking that the safety numbers being shown are correct, click on the Confirm Safety Numbers button.More specifically, the safety numbers for the Safety CPUs will be shown in a list, and need to be compared to the safety numbers on the modules.



11.3.2 Step 2: Deleting the program on the Safety CPU, downloading a new project, uploading the code from the Safety CPU and comparing it with the project.

As soon as you have confirmed the safety numbers, the Download All button will be enabled.
Clicking on the Download All button will delete the old project and load the new project onto the Safety CPU.

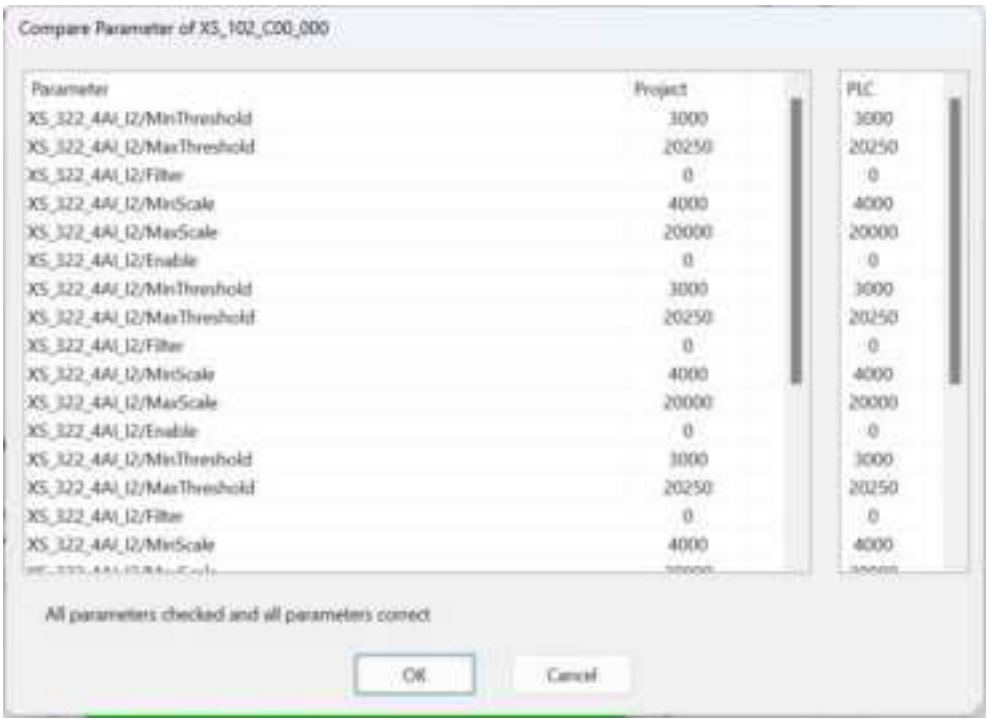
After a successful download, the code will be automatically uploaded and compared with the project.

After a successful upload, the Check Parameter dialog box will appear. Check all the parameters and confirm that they are correct.

This work sequence is automatically repeated for all listed Safety CPUs.

The progress bars will show the progress for the CPU currently in question and for the process overall.

Clicking on the Set Configured and Finish button will set the Safety CPU to "configured" and finish the download operation.



11.4 Passwords

- **Level 0 (No password)**
  Only read-only access is possible without a password.
- **Level 1:**
  PW1 can be used to work in debug mode (force mode).
- **Level 2:**
  The configuration of the safety modules can be changed with PW2.
- **Level 3 (manufacturer password):**
  A firmware update can be carried out with PW3.

The passwords are compared, evaluated, and stored by the Safety CPU. This ensures that it will not be possible to make a change to every machine with a single password.

You will only be able to change the password after first entering the old one. New passwords need to be entered twice (in order to avoid typing mistakes) and must be between six and eight characters long. Within this context, the level in which you are logged in does not matter, since the ability to change the password is protected by the old password.

After one hour without any actions, XN Safety Designer will automatically switch back to level 0.

11.4.1 Overview of actions allowed by each password level

green..... Action possible

red..... Action not possible

gray..... Combination makes no sense

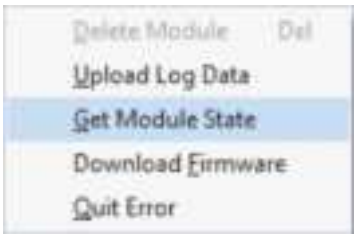
Action	Offline	Online				
	Offline	No PW level 0	Level 1 Debugging	Level 2 Configuration	Level 2 Configuration (Dev.-Mode)	Level 3 Manufacturer password
Program change						
Debugging						
Debugging (Display only)						
Debuggen (Force-Mode)			only in temp. OpMode	only in temp. OpMode	only in temp. OpMode	only in temp. OpMode
Actions in online dialog						
Login						
Change Pass-word						
Quit Error						
Service Mode			when in run mode	when in run mode	when in run mode	when in run mode
Start			when in Service Mode	when in Service Mode	when in Service Mode	when in Service Mode
Clear Program				when in Service Mode	when in Service Mode	when in Service Mode
Download				when in Service Mode	when in Service Mode	when in Service Mode
Write SD Card				when in Service Mode		when in Service Mode
Set Verified				when in Service Mode		when in Service Mode

11.5 Upload Log-Data

You can use the context menu in the Hardware Tree pane (for the corresponding module) to read a module's log memory, save it in a binary file, and show it in a tab. This data can be used for servicing purposes. Please refer to the "Show log files" section as well.

11.6 Get Module State

You can use the context menu in the Hardware Tree pane (for the corresponding module) to read a module's status data. This data will be shown in the Output pane and can be saved as well.



The following will be output:

- Runtime status (e.g., Operation Mode)
- Errors (if any)
- Configuration status (e.g., verified)
- Security number of the module
- Firmware version
- Cycle time, max. cycle time, and configured max. cycle time in [ms]
- Used flash memory in [bytes]
- Total flash memory in [bytes]
- Used RAM memory in [bytes]
- Total RAM memory in [bytes]

The following data will be shown separately for the two controllers:

- Application processing time in [ms]
- Max. application processing time in [ms]
- Time that the tests need in [ms]
- Max. time that the tests need in [ms]
- Used RAM memory in [bytes]
- Size of operating system images in [bytes]

12. Export of download data for a safety CPU

The download data of a Safety CPU can be exported to a binary file.

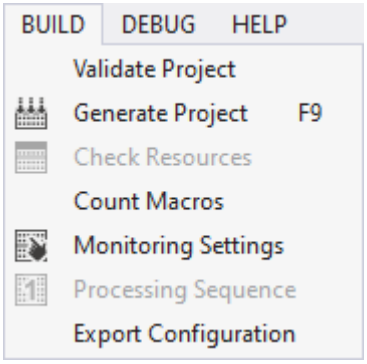
Structure of the binary file:

- Exportheader
- Configuration (identical to the configuration when downloading via the XN Safety Designer)
- an additional CRC at the end

This binary file can then be imported into a safety CPU without the XN Safety Designer.

12.1 Creating the binary file

The export can be called up via the Build menu with Export Configuration.



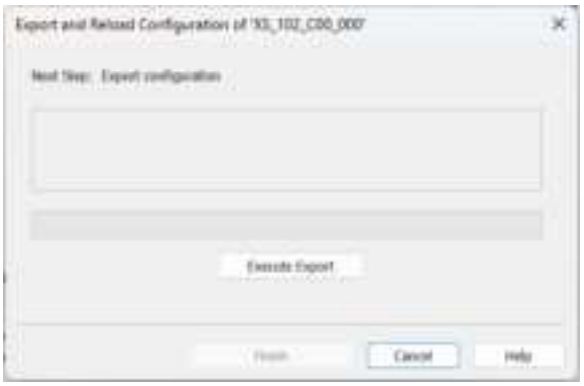
The project must be compilable in order to perform an export. The export runs in several steps via a wizard.

The Execute Export button first exports a binary file (the name of the binary file is made up of the project name and the safety CPU name,
e.g. Project-<Name>-Safety-CPU-<Name>.bin).

The export is stored in the project directory. After writing the file, an external tool is automatically started (CheckConfig.exe), which loads the binary file and checks the CRC of the configuration for validity. The external tool replaces the checked CRC with a second CRC and saves the binary file under a different name. If the check fails, an error message is displayed and the binary file is not saved again.

12. Export of download data for a safety CPU

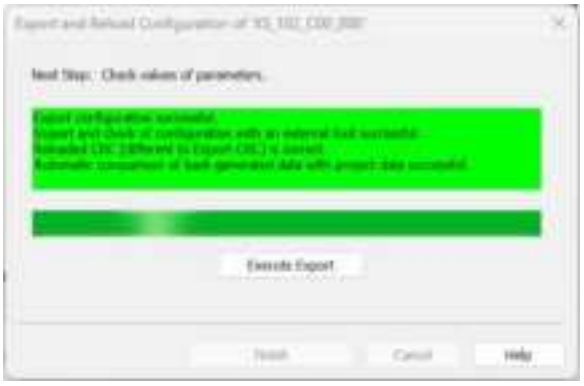
12.1 Creating the binary file



12.1.1 Error during the check in the external tool:

Error	Description
External tool 'CheckConfig.exe' not found	The external tool could not be started. The tool must be in the installation directory.
Import file <Name> failed	The binary file could not be loaded.
CRC is incorrect	The CRC check has failed.
Export file <Name> failed	The new binary file could not be written

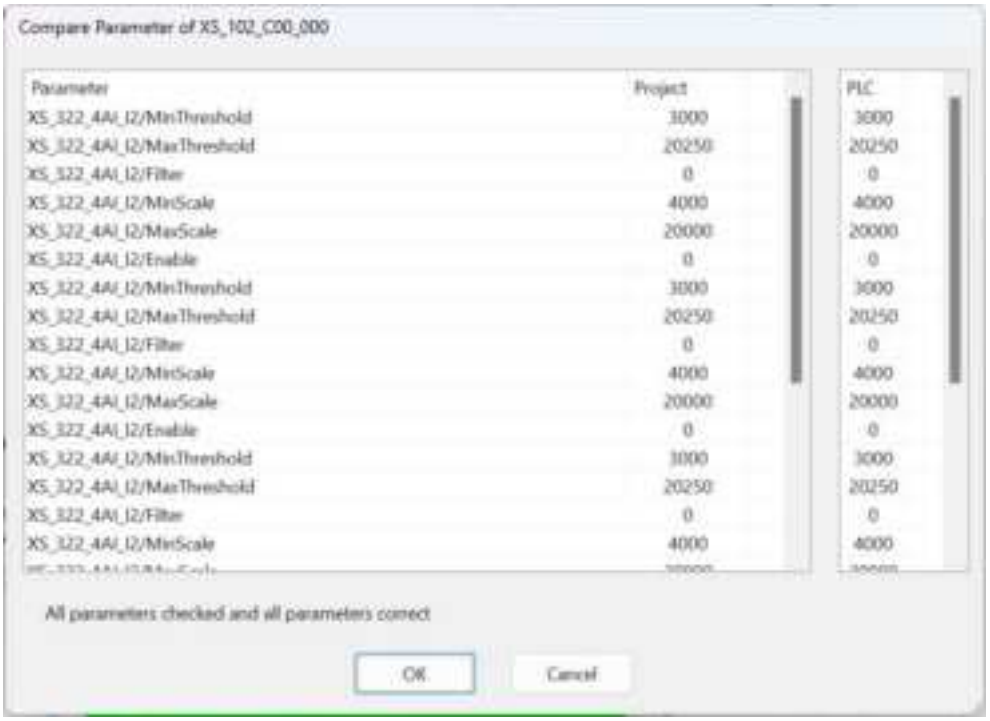
This is followed by a "Reload and Compare". The binary file created by the external tool is read in and the new CRC is checked for validity; the configuration read back is also compared with the project.



12.1.2 Error during Reload and Compare:

Error	Description
Reloaded CRC (different to Export-CRC) is incorrect	The CRC generated by the external tool is not correct.
Automatic comparision of back-generated data with project data failed	The reconverted data is different to the project.

After the successful process, the check parameter dialog opens. The user must check the parameters and confirm that they are correct.



The binary file is saved again with the "Finish" button (name is identical to the first name). The CRC generated by the external tool is replaced by the original CRC. The CRC generated by the external tool is written to the end of the binary file. The binary file is located in the project directory.

12.2 Load binary file on a Safety CPU

The user must perform two steps to download to a Safety CPU.

Step 1: Evaluating and checking the export header of the binary file

Structure of the export header:

- CRC of the header (4 Byte)
- Length of the header (4 Byte)
- Type of header (4 Byte), the type is assigned 0
- Length of the hardware path (4 Byte)
- Hardware path (path is aligned to 4 bytes, length of the path without 4-byte alignment is at the beginning of the path)
- Length of the project name (4 Byte)
- Name of the current project.
- Length of the revision number of the project (4 Byte)
- Revision number of the project
- Length of the name of the Safety-CPU (4 Byte)
- Safety-CPU Name

The information in the export header (project name, revision, ...) must be checked for correctness by the user.

Step 2: Download the binary file

The user must log in to the Safety CPU and download the configuration (binary file without the export header). After the successful download, the user must confirm the download and start and verify the Safety CPU.

13. Function blocks

13.1 Data Types

Generally, a distinction is drawn between safe and unsafe data types. In order to keep the corresponding complexity low, only the following data types are available:

1. SAFEBOOL (4 byte value)
2. SAFEDINT (8 byte value)
3. DINT (4 byte value)

As can be seen in the list of data types, Boolean values are also transmitted as four bytes. In this case, a value of 0 will be interpreted as `False`, while all values other than 0 will be interpreted as `True`.

.

13.2 Special implementation details

- BOOL (not SAFE_BOOL) values will only be read for FALSE and not FALSE.
- If a specific time is monitored by a function block, this time can be adjusted dynamically, and there is no defined error state (complex function blocks), the setting must not exceed a 31-bit (i.e., 0x7FFFFFFF) value. If the set time is higher, an error (ERR_BAD_INPUT_VALUE) will be returned to the safe module's firmware.
- On INIT, the function block will be initialized. The SAFEBOOL outputs will be set to SAFEBOOL_FALSE. The BOOL outputs will be set to FALSE. The edge flags will be initialized with the value of the corresponding input. Accordingly, no edges will be detected in the first pass. It is also possible for an invalid value to be written to the edge flags, i.e., the SAFEBOOL inputs will not be checked. In complex function blocks, the DiagCode will be set to 0x0000.The function block will be exited after initialization.
- At the end of the function block, the input value will be written to the edge flag.
- If a SAFEBOOL input is undefined (not True, not False, and not Error), all SAFEBOOL outputs will be set to SAFEBOOL_ERROR and an error code will be returned to the safe module's firmware. This also applies in the event that the function block has internal memory for edge detection. In the case of complex function blocks, the "READY", "ERROR", and "DiagCode" outputs will keep the value that has already been set.
- If there is a SAFEBOOL_ERROR on a SAFEBOOL input, all SAFEBOOL outputs will be set to SAFE-BOOL_ERROR. In addition, for complex function blocks, the DiagCode output will be set to a state of 0000. This ensures that the function block will have to restart if the SAFE-BOOL_ERROR switches back to SAFEBOOL_TRUE or SAFEBOOL_FALSE and the relevant function block conditions must be met again in order to reach the state in which the SAFEBOOL_ERROR occurred.
- For SAFEDINT inputs, there can be a value or a SAFEDINT_ERROR. If there is a SAFE-DINT_ERROR, a SAFEDINT_ERROR will be signaled at the output.
- If a SAFEDINT is invalid, the outputs will be set to SAFEDINT_ERROR and an error code will be sent to the Safety module's firmware. The Safety CPU can signal an error condition this way. The error must be reset.
- The error values in online mode are signaled with an s_error in the function block instances.
- If the input of a function block is constant (Constant Value setting in the Properties pane), you will not be able to connect an unsafe variable. Unsafe variables can be connected indirectly through function block outputs, in which case you must verify that the interconnection is correct.

13.3 Time monitoring for function blocks

The times that can be configured for the function blocks are constant and can only be configured with Safety Designer. These values cannot be changed during operation.

The allowed times are listed in the descriptions for the individual function blocks. If the maximum valid range can be set for function blocks, the times will not be indicated explicitly.

The maximum time monitoring value range, which is 0 to `0x7FFFFFFF` for a four-byte value, applies to the following function blocks:

- TON
- TOFF
- TP
- SF_Equivalent function
- SF_Antivalent function
- SF_ModeSelector function
- SF_GuardMonitoring function
- SF_SafetyRequest function
- SF_EDM function
- Path Speed
- Speed
- Direction

13.4 Standard function blocks

13.4.1 General Information

The function blocks take up the RAM listed in the RAM memory column. Some function blocks require SRAM memory instead, since values need to be stored. These values will then be available the next time the Safety CPU starts. Memory is also taken up by retentive memory variables.

The SRAM memory is

- XS-102-C00-000: 120 Byte (30 values 4 Byte each)

The memory allocation of the function blocks is listed in the SRAM memory column.

Function block	RAM memory (byte)	SRAM memory (byte)
AND	16	
AND_5I	28	
AND_NS	16	
AND_XI	110	
ABS	20	
ABS_NS	12	
ADD	32	
ADD_NS	24	
AND_BW	28	
Bool_to_Safebool	12	
Collision_Detection	60	
CTD	28	
CTU	28	
CTUD	44	
Dint_to_SafeDint	16	
DINT_Compare	36	
EQ	24	
EQ_NS	16	
F_TRIG	16	
FW_Check	20	
GT	24	
GT_NS	16	
GE	24	
GE_NS	16	
LT	24	
LT_NS	16	
LE	24	
LE_NS	16	

Function block	RAM memory (byte)	SRAM memory (byte)
LIMIT	36	
LIMIT_NS	20	
MULDIV	52	
MULDIV_NS	32	
MIN	28	
MIN_NS	16	
MAX	28	
MAX_NS	16	
NOT	12	
NOT_BW	20	
OR_BW	28	
OR	16	
OR_NS	16	
OR_5I	28	
OR_XI	110	
Restart_Interlock	36	
R_TRIG	20	
SUB	32	
SUB_NS	20	
SHL	24	
SHL_NS	16	
SHR	24	
SHR_NS	16	
Safebool_to_Bool	12	
SafeDInt_to_Dint	16	
SError_to_SFalse	12	
SDINT_Compare	40	
Safe_Area	60	
SampleHold	24	
SF_Antivalent	44	
SF_EDM	72	
SF_EmergencyStop	44	
SF_EnableSwitch	48	
SF_Equivalent	44	
SF_Espe	44	
SF_GuardLocking	64	
SF_ModeSelector	128	
SF_MutingPar	108	
SF_MutingPar_2Sensor	80	
SF_MutingSeq	108	
SF_Optional_PWD	64	4
SF_Optional_PWDII	72	4
SF_Optional_Switch	20	
SF_Optional_Switch_SDINT	32	
SF_Optional_Switch_DINT	20	
SF_OutControl	56	

Function block	RAM memory (byte)	SRAM memory (byte)
SF_SafetyRequest	56	
SF_TestableSafetySensor	84	
SF_TwoHandControllTypeII	32	
SF_TwoHandControllTypeIII	40	
SF_LimitComparator	68	
SF_LimitComparator_Dint	20	
SF_NumericSelector	132	
SF_RampMonitor	80	
SF_DirectionMonitor	68	
SF_SkewMonitor	92	8
SF_HGW_Login	104	4
TOFF	32	
TON	32	
TP	32	
XOR_BW	28	
XOR	16	

13.4.2 Function blocks for type conversion

Unless otherwise specified, each function block is available as a SAFE variant. Pure UNSAFE function blocks are not provided.

13.4.2.1 BOOL_TO_SAFEBOOL

Conversion of a BOOL value into a SAFEBOOL value.

- When converting, the "In" input is set to the value of the SAFEBOOL value.
- If there is no valid value at the input, SAFEBOOL_ERROR will be output.

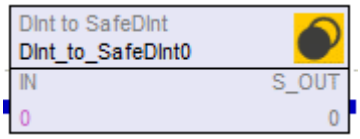
Input variables	
IN	Input (BOOL)

Output variables	
S_OUT	Output (SAFEBOOL)

13.4.2.2 DINT_TO_SAFEDINT

Conversion of a DINT value into a SAFEDINT value. A SAFEDINT value consists of a DINT value and a value that indicates whether the value is valid.

- When converting, the "In" input is set to the value of the SAFEDINT value; the additional value is set to the valid status.

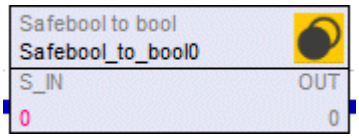


Input variables	
IN	Input (DINT)

Output variables	
S_OUT	Output (SAFEDINT)

13.4.2.3 SAFEBOOL_TO_BOOL

Converts a SAFEBOOL value to a BOOL value

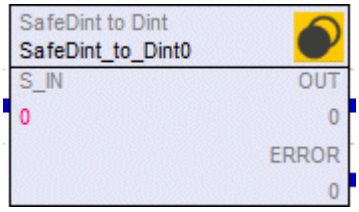


Input variables	
S_IN	Eingang (SAFEBOOL)

Output variables	
OUT	Output (BOOL)

13.4.2.4 SAFEDINT_TO_DINT

Converts a SAFEDINT value to a DINT value

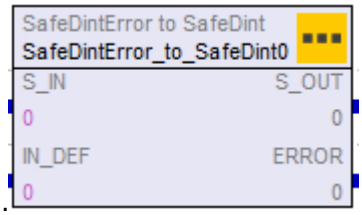


Input variables	S_IN	Input (SAFEDINT)
Output variables	OUT	Output (DINT)
	ERROR	TRUE if the input is invalid (BOOL)

13.4.2.5 SAFEDINTERROR_TO_SAFEDINT

Converts SAFEDINT_ERROR states to a default value.

- If there is a SAFEDINT_ERROR value at the "S_IN" input, the value at the "IN_DEF" input (DINT) will be output as a SAFE-DINT value at the "S_OUT" output; otherwise, the value at the "S_IN" input will be connected through.

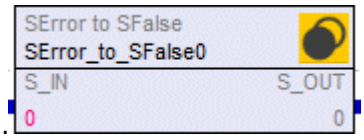


Input variables	
S_IN	Input (SAFEDINT)
IN_DEF	Output value for when S_IN (DINT) has SAFEDINT_ERROR

Output variables	
S:OUT	Output (SAFEDINT)
ERROR	Flag for SAFEDINT_ERROR at S_IN (BOOL)

13.4.2.6 SERROR_TO_SFALSE

Converts SAFEBOOL_ERROR states to SAFEBOOL_FALSE. Output S_OUT will only become SAFE-BOOL_TRUE if input S_IN also has a value of SAFEBOOL_TRUE; otherwise, S_OUT will always be SAFE-BOOL_FALSE.



Input variables	
S_IN	Eingang (SAFEBOOL)

Output variables	
S:OUT	Output (SAFEBOOL)

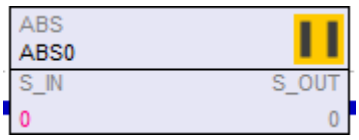
13.4.3 Standard numerical function blocks

Tbl. 3: Overview

→ " ABS (Absolute value)", page 173 → " ABS_NS (absolute value, not safe)", page 173		Absolute amount
→ " ADD (Addition)", page 173 → " ADD_NS (Addition, not safe)", page 174		Addition
→ " SUB (subtraction)", page 174 → " SUB_NS (subtraction, not safe)", page 175		Subtraction
→ " MULDIV (multiplication and division)", page 176 → " MULDIV_NS (Multiplication and Division, not safe)", page 175		Multiplication and division
→ " SF_SampleHold", page 176		Switching with hold function
→ " SHL (Shift-Left)", page 177 → " SHL_NS (Shift-Left, not safe)", page 177		Shift-Left of a SAFEDINT or DINT input
→ " SHR (Shift-Right)", page 178 → " SHR_NS (Shift-Right, not safe)", page 178		Shift-Right of a SAFEDINT or DINT input
→ " AND_BW (AND, bitwise)", page 179		Computes the bitwise AND of SAFEDINT inputs
→ " OR_BW (OR, bitwise)", page 179		Computes the bitwise OR of SAFEDINT inputs
→ " XOR_BW (XOR, bitwise)", page 180		Computes the bitwise XOR of SAFEDINT inputs
→ " NOT_BW (NOT, bitwise)", page 180		Bit-by-bit reversal of a SAFEDINT input
→ " Restart_Interlock", page 181		The Restart_Interlock FUB checks the correctness of an input from analog input cards
→ "FW_Check", page 182		The FUB checks the online FW version with that of the inbox
→ "Path Speed", page 183		Calculation of velocity in a 3-dimensional space
→ "Speed", page 185		Calculating the speed from a change in position
→ "Direction", page 187		Calculating the direction of movement from the change in position

13.4.3.1 ABS (Absolute value)

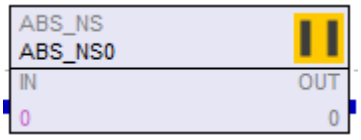
Returns an unsigned value.



Input variables	
S_IN	Input (SAFEDINT)
Output variables	
S_OUT	Output (SAFEDINT)

13.4.3.2 ABS_NS (absolute value, not safe)

Returns a value with a plus sign

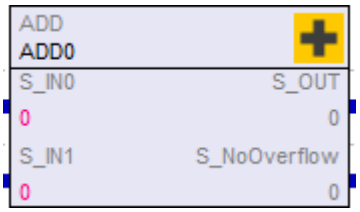


Input variables	
IN	Input (DINT)
Output variables	
OUT	Output (DINT)

13.4.3.3 ADD (Addition)

Adds S_IN0 and S_IN1.

Does not handle carries; if there is an overflow, the number will start again from 1.

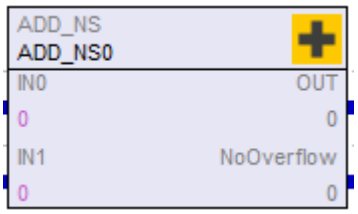


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Output (SAFEDINT)
S_NoOverflow	Overflow has not taken place (SAFEBOOL)

13.4.3.4 ADD_NS (Addition, not safe)

Adds IN0 and IN1.

Does not handle carries; if there is an overflow, the number will start again from 1.

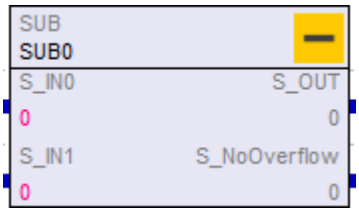


Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Output (DINT)
NoOverflow	Overflow has not taken place (BOOL)

13.4.3.5 SUB (subtraction)

Subtracts S_IN1 from S_IN0.

Does not handle carries; if there is an overflow, the number will start again from 1.

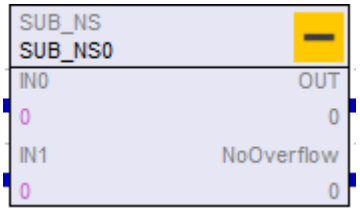


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Output S_IN0 - S_IN1 (SAFEDINT)
S_NoOverflow	Overflow has not taken place (SAFEBOOL)

13.4.3.6 SUB_NS (subtraction, not safe)

Subtracts IN1 from IN0.

Does not handle carries; if there is an overflow, the number will start again from 2_{31}-1.

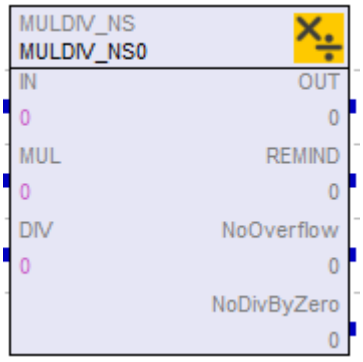


Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Output S_IN0 - S_IN1 (DINT)
NoOverflow	Overflow has not taken place (BOOL)

13.4.3.7 MULDIV_NS (Multiplication and Division, not safe)

Multiplies IN by MUL and divides the result by DIV.

The sign of REMIND depends on the multiplication result. The multiplication's interim result has double the data length (64 bits).

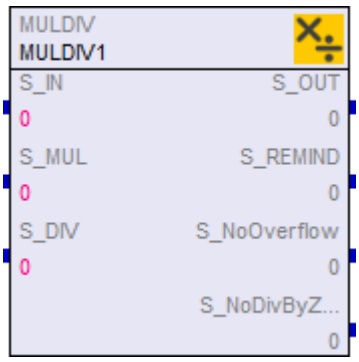


Input variables	
IN	Input value (DINT)
MUL	Factor (DINT)
DIV	Divisor (DINT)
Output variables	
OUT	Output (IN * MUL) / DIV (DINT)
REMIND	Remainder of the division (DINT)
NoOverflow	Overflow has not taken place (BOOL)
NoDivByZero	No division by 0 (BOOL)

13.4.3.8 MULDIV (multiplication and division)

Multiplies S_IN by S_MUL and divides the result by S_DIV.

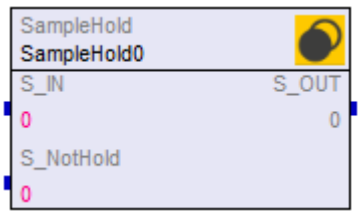
The sign of S_REMIND depends on the multiplication result. The multiplication's interim result has double the data length (64 bits).



Input variables	
S_IN	Input value (SAFEDINT)
S_MUL	Factor (SAFEDINT)
D_DIV	Divisor (SAFEDINT)
Output variables	
S_OUT	Output (S_IN * S_MUL) / S_DIV (SAFEDINT)
S_REMIND	Remainder of the division (SAFEDINT)
S_NoOverflow	Overflow has not taken place (SAFEBOOL)
S_NoDivByZero	No division by 0 (SAFEBOOL)

13.4.3.9 SF_SampleHold

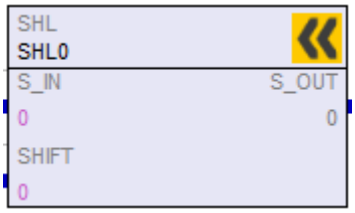
Switches S_IN to S_OUT if S_NotHold is set. If S_NotHold is not set, S_OUT will maintain its old value.



Input variables	
S_IN	Input (SAFEDINT)
S_NotHold	Used to activate the hold function. FALSE: The value at S_OUT will be maintained. TRUE: The value S_IN is switched through to S_OUT. (SAFEBOOL).
Output variables	
S_OUT	Output (SAFEDINT)

13.4.3.10 SHL (Shift-Left)

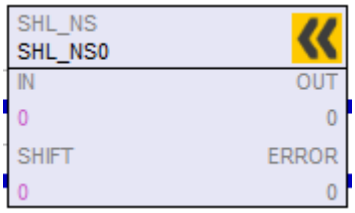
Shift-left operation for a SAFEDINT value. The input will be shifted left by the specified number of bits in a bitwise manner.



Input variables	
S_IN	Input (SAFEDINT)
SHIFT	The number of bits by which the value will be shifted (DINT)
Output variables	
S_OUT	Result (SAFEDINT)

13.4.3.11 SHL_NS (Shift-Left, not safe)

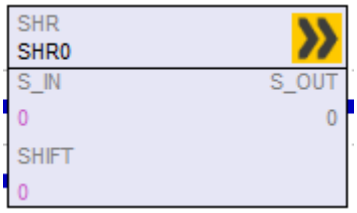
Shift-left operation for a DINT value. The input will be shifted left by the specified number of bits in a bitwise manner.



Input variables	
IN	Input (DINT)
SHIFT	The number of bits by which the value will be shifted (DINT)
Output variables	
OUT	Result (DINT)
ERROR	TRUE if errors occur

13.4.3.12 SHR (Shift-Right)

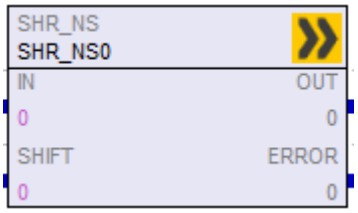
Shift-right operation for a SAFEDINT value. The input will be shifted right by the specified number of bits in a bitwise manner.



Input variables	
S_IN	Input (SAFEDINT)
SHIFT	The number of bits by which the value will be shifted (DINT)
Output variables	
S_OUT	Result (SAFEDINT)

13.4.3.13 SHR_NS (Shift-Right, not safe)

Shift-right operation for a DINT value. The input will be shifted right by the specified number of bits in a bitwise manner.

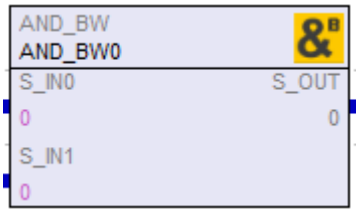


Input variables	
IN	Input (DINT)
SHIFT	The number of bits by which the value will be shifted (DINT)
Output variables	
OUT	Result (DINT)
ERROR	TRUE if errors occur

13.4.3.14 AND_BW (AND, bitwise)

Bitwise AND operator with two SAFEDINT inputs,

Computes the logical bitwise AND of S_IN0 and S_IN1 and outputs the resulting bit pattern on S_OUT.

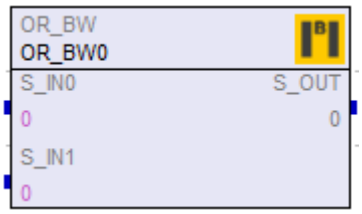


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Bitwise AND output (SAFEDINT)

13.4.3.15 OR_BW (OR, bitwise)

Bitwise OR result of two SAFEDINT inputs.

The bitwise OR of S_IN0 and S_IN1 will be computed and the resulting bit pattern will be output at S_OUT.

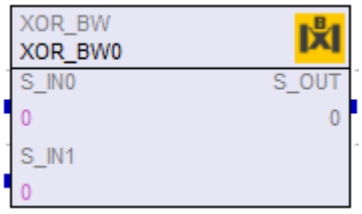


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Bitwise AND output (SAFEDINT)

13.4.3.16 XOR_BW (XOR, bitwise)

Bitwise XOR CONTINUE

This function block will compute the bitwise XOR of S_IN0 and S_IN1 and output the resulting bit pattern at S_OUT.

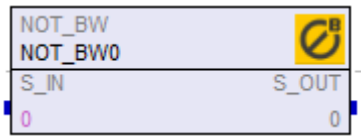


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Bitwise AND output (SAFEDINT)

13.4.3.17 NOT_BW (NOT, bitwise)

Bitwise inversion of a SAFEDINT input.

S_IN will be inverted in a bitwise manner and the resulting bit pattern will be output at S_OUT.

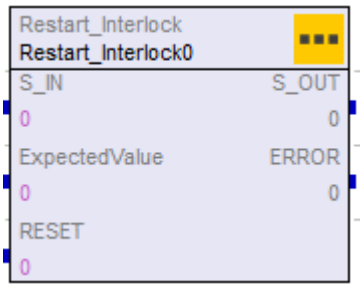


Input variables	
S_IN	Input (SAFEDINT)
Output variables	
S_OUT	Bitwise AND output (SAFEDINT)

13.4.3.18 Restart_Interlock

The Restart_Interlock function block checks the correctness of an input on analog input cards if the ExpectedValue input has a value of TRUE. The value of the S_IN input will be connected through to output S_OUT. If there is an incorrect value at the S_IN input, it will be continuously output at the S_OUT output. Moreover, the ERROR output will have a value of TRUE. The RESET input will then need to be used to trigger an error reset (transition from TRUE to FALSE).

If the ExpectedValue input has a value of FALSE, the input will be connected through to the output directly without a check.



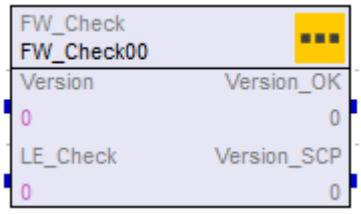
Input variables	
S_IN	Input (SAFEDINT)
ExpectedValue	Flag, whether check active (BOOL)
RESET	Reset Flag (BOOL)
Output variables	
S_OUT	Result (SAFEDINT)
ERROR	Fault scenario (BOOL)

13.4.3.19 FW_Check

This function block checks the online FW version with the version at the input. The Safety CPU FW is copied to a non-visible function block input.

The function block can only be placed once per Safety CPU.

Comparison: The version at the input is less than or equal to the version on the Safety CPU.



Input variables	
Version	Input, version to be compared (DINT)
LE_Check	Flag for less-than-or-equal comparison (BOOL) FALSE = Equal comparison TRUE = Less-than-or-equal comparison
Output variables	
Version_OK	Outcome, result of the comparison (BOOL)
Version_SCP	Output indicating which FW version is running on the PLC (DINT)

13.4.3.20 Path Speed

You can use the PATH_SPEED function block to compute the derivative of the position (X, Y, Z) in question in order to get the corresponding speed. In other words, this function block is used to calculate speed in a three-dimensional space. You can use the function block's input variables to configure its behavior.

Input variables S_xPosition, S_yPosition, S_zPosition are unitless. This means that you are responsible for how you interpret the position value. Meanwhile, you can select the time unit for the S_Speed output variable with the SampleUnit input variable. The S_Speed output variable specifies the speed along the trajectory.

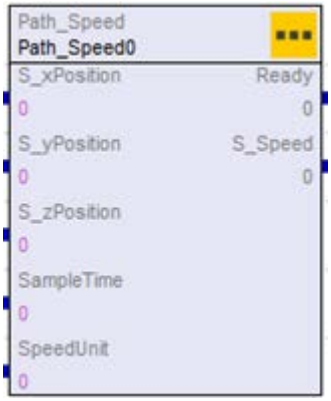
When it comes to very slow speeds relative to the Safety CPU's cycle time, the system will be unable to meaningfully compute the aforementioned speed. As a workaround, you can adjust the "sampling time" for the object with the SampleTime input variable. If you set the SampleTime input variable to 0, the object will be run in every cycle. If you set the SampleTime to a value of 100 (ms) instead, the object will not be run again until after 100 ms elapse.

When setting the SampleTime, make sure to take the position sampling rate into account. This time will be 1 ms for the inputs on the XS-322-2INC module, for example. If the position is transmitted through the bus, the transmission time must also be taken into account in order to get a correct calculation. The set SampleTime must always be longer than the time between position changes.

This function block assumes that there will not be an overflow in input variables S_xPosition, S_yPosition, and S_zPosition, since an overflow would not make sense for a spatially limited axis.

The function block's name, PATH_SPEED, was specifically chosen in order to make the function clearer for users. However, the function block can be generally used to compute the derivative of three variables over time by calculating the Euclidean norm of the individual terms. This means that the function block can also be used to determine acceleration.

The speed can then be filtered with a PT1 block.



Input variables	
S_xPosition	Position of the X-axis [unitless] (SAFEDINT)
S_yPosition	Position of the Y-axis [unitless] (SAFEDINT)
S_zPosition	Position of the Z-axis [unitless] (SAFEDINT)
SampleTime	This input variable is used to define the time interval for running the object. If a value of 0 is specified, the object will be run in every cycle. If a value greater than 0 is specified, the corresponding time will have to elapse since the last pass in order for the speed to be computed at the output. This value is specified in ms, and only positive values are allowed. (DINT)
SpeedUnit	Output unit (0 = ms, 1 = s, 2 = min) (DINT)
Output variables	
Ready	Signals whether the function block is active (BOOL)
S_Speed	Calculated speed [position unit per SpeedUnit] (SAFEDINT)

The function block features a start inhibit that needs to be run through before the function block will work correctly. After initialization or after an error is reset automatically, the function block will need, if applicable, to go through multiple cycles before it outputs a correct result and Ready is switched to TRUE. This inhibit function will depend on the set SampleTime. The safety CPU must run through at least 3 cycles before a correct calculation can be performed.

Calculation formula:

$$v = \sqrt{\left(\frac{x(t+h)-x(t)}{h}\right)^2 + \left(\frac{y(t+h)-y(t)}{h}\right)^2 + \left(\frac{z(t+h)-z(t)}{h}\right)^2}$$

v Calculated speed

x x-Position

y y-Position

z z-Position

t Timestamp of last calculation in [ms]

h Time interval for the calculation (SampleTime)

13.4.3.21 Speed

You can use the SPEED function block to compute the derivative of the position in question in order to get the corresponding speed. In other words, this function block is used to calculate the speed along a single axis. You can use the input variables to configure the function block's behavior.

Input variable S_Position is unitless. This means that you are responsible for how you interpret the position value. Meanwhile, you can select the time unit for the S_Speed output variable with the SpeedUnit input variable.

When it comes to very slow speeds relative to the Safety CPU's cycle time, the system will be unable to meaningfully compute the aforementioned speed. As a workaround, you can adjust the "sampling time" for the object with the SampleTime input variable. If you set the SampleTime input variable to 0, the object will be run in every cycle. If you set the SampleTime to a value of 100 (ms) instead, the object will not be run again until after 100 ms elapse.

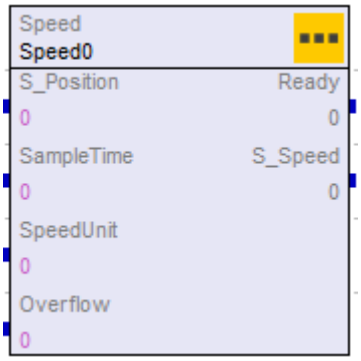
When setting the SampleTime, the position sampling rate must be taken into account. For XS-322-2INC module inputs, for example, this time will be 1 ms. If the position is transmitted through the bus, the bus transmission time must also be taken into account in order to ensure that the calculation is correct. Please note that the set SampleTime must always be longer than the time between position changes.

Since the position value at the S_Position input can have an overflow, this must be taken into account in the speed computation. Accordingly, you will need to know the value at which the position experiences an overflow. This value is specified with the Overflow input variable.

Please make sure that for very fast speeds, the distance covered in the time set in SampleTime is not greater than half of the distance specified in Overflow.

The function block's name, SPEED, was specifically chosen in order to make the function clearer for users. However, the function block itself can be generally used to compute the derivative of a variable over time. This means that the function block can also be used to determine acceleration (with an overflow of 0).

The speed can then be filtered with a PT1 block.



Input variables	
S_Position	Axis position. This value is unitless for the function block. (SAFEDINT)
SampleTime	This input variable is used to define the time interval for running the object. If a value of 0 is specified, the object will be run in every cycle. If a value greater than 0 is specified, the corresponding time will have to elapse since the last pass in order for the speed to be computed at the output. This value is specified in ms, and only positive values are allowed. (DINT)
SpeedUnit	Unit of output for S_Speed (0 = ms, 1 = s, 2 = min) (DINT)
Overflow	The Overflow input variable must be used to enter the value at which the position (S_Position) has an overflow. When you set a value here, the position will be checked from 0 to the set value. If you set a value of 0, the minimum value and maximum value of DINT will be used as the overflow position. (DINT)
Output variables	
Ready	Signals whether the function block is active (BOOL)
S_Speed	Calculated speed [position unit per SpeedUnit] (SAFEDINT)

The function block features a start inhibit that needs to be run through before the function block will work correctly. After initialization or after an error is reset automatically, the function block will need, if applicable, to go through multiple cycles before it outputs a correct result and Ready is switched to TRUE. This inhibit function will depend on the set SampleTime. The safety CPU must run through at least 3 cycles before a correct calculation can be performed. A value of 0 will be output at the S_Speed output as long as the Ready output has a value of false.

Calculation formula:

$$v = \frac{x(t+h) - x(t)}{h}$$

v Calculated speed (S_Speed)

x(t+h) Position value (s_Pos) in current call

x(t) Position value (s_Pos) in last call

t Timestamp of last calculation in [ms]

h Time interval for the calculation (SampleTime)

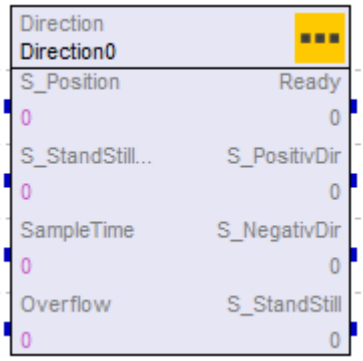
13.4.3.22 Direction

The function block calculates the direction of movement from the change in position.

You can set a tolerance (S_StandStillWindow) that must be exceeded in order for the function block to determine that the calculation is being output correctly. If this tolerance is not exceeded, the function block will output the direction of movement. The S_PositiveDir output will switch to SAFEBOOL _TRUE if the direction is positive. The S_NegativeDir output will switch to SAFEBOOL _TRUE if the direction is negative. The _Standstill output changes to SAFEBOOL _TRUE if no direction is selected.

When setting the SampleTime, the position sampling rate must be taken into account. For XS-322-2INC module inputs, for example, this time will be 1 ms. If the position is transmitted through the bus, the bus transmission time must also be taken into account in order to ensure that the calculation is correct. Please note that the set SampleTime must always be longer than the time between position changes.

Please make sure that for very fast speeds, the distance covered in the time set in SampleTime is not greater than half of the distance specified in Overflow.



Input variables	
S_Position	Axis position. This value is unitless for the function block. (SAFEDINT)
S_StandStillWindow	Position change tolerance. This tolerance is the maximum allowed change in position, within the configured time (SampleTime), in which a standstill will still be detected. The value set here will work in both the positive and negative directions. (SAFEDINT)
SampleTime	Calculation interval in ms. If a value of 0 is set, the calculation will always be carried out in a Safety CPU cycle. (DINT)
Overflow	The value at which the position (S_Position) has an overflow must be entered into the Overflow input variable. When a value is set, the position will be checked from 0 to the set value. If a value of 0 is set instead, the minimum value and maximum value of DINT will be used as the overflow points. (DINT)
Output variables	
Ready	Signals whether the function block is active (BOOL)
S_PositiveDir	Calculated direction of movement (SAFEDINT) SAFEBOOL_TRUE = positive direction of movement

	SAFEBOOL_FALSE = negative direction of movement and standstill
S_NegativeDir	Calculated direction of movement (SAFEDINT) SAFEBOOL_TRUE = negative direction of movement SAFEBOOL_FALSE = positive direction of movement and standstill
S_StandStill	Calculated direction of movement (SAFEDINT) SAFEBOOL_TRUE = standstill SAFEBOOL_FALSE = positive or negative direction of movement

The function block features a start inhibit that needs to be run through before the function block will work correctly. After initialization or after an error is reset automatically, the function block will need, if applicable, to go through multiple cycles before it outputs a correct result and Ready is switched to TRUE. This inhibit function will depend on the set SampleTime. The safety CPU must run through at least 3 cycles before a correct calculation can be performed.

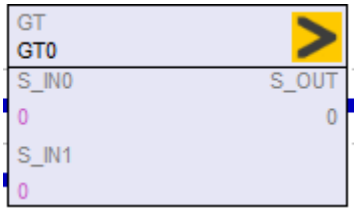
13.4.4 Comparative function blocks

Tbl. 4: Overview

→ " GT (greater than)", page 189 → " GT_NS (greater than, not safe)", page 190		Greater than
→ " GE (greater than or equal to)", page 190 → " GE_NS (greater than or equal to, not safe)", page 190		Greater than/equal to
→ " LT (less than)", page 191 → " LT_NS (less than, not safe)", page 191		Less than
→ " LE (less than or equal to)", page 191 → " LE_NS (less than or equal to, not safe)", page 192		Less than/equal to
→ " EQ (equal)", page 192 → " EQ_NS (equal, not safe)", page 192		Equal
→ " MIN (Minimum)", page 193 → " MIN_NS (Minimum, not safe)", page 193		Minumum
→ " MAX (Maximum)", page 194 → " MAX_NS (Maximum, not safe)", page 194		Maximum
→ " LIMIT (within limits)", page 195 → " LIMIT_NS (within limits, not safe)", page 195		Within limits
→ " SDINT_Compare", page 196		Comparison of two SAFEDINT inputs over a period of time
→ " DINT_Compare", page 197		Comparison of two DINT inputs over a period of time

13.4.4.1 GT (greater than)

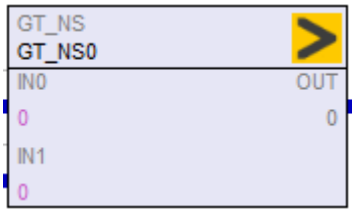
Indicates whether S_IN0 is greater than S_IN1.



Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Result of the comparison (SAFEBOOL)

13.4.4.2 GT_NS (greater than, not safe)

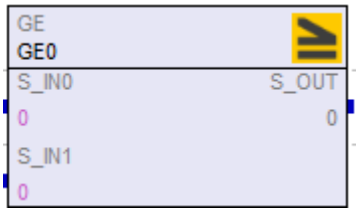
Indicates whether IN0 is greater than IN1.



Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Result of the comparison (BOOL)

13.4.4.3 GE (greater than or equal to)

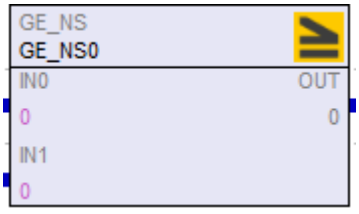
Indicates whether S_IN0 is greater than or equal to S_IN1.



Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Result of the comparison (SAFEBOOL)

13.4.4.4 GE_NS (greater than or equal to, not safe)

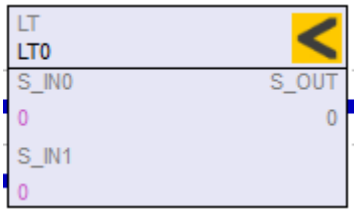
Indicates whether IN0 is greater than or equal to IN1.



Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Result of the comparison (BOOL)

13.4.4.5 LT (less than)

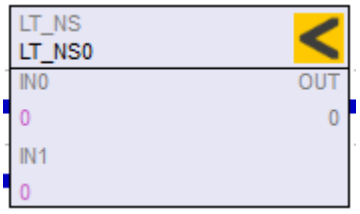
Indicates whether S_IN0 is smaller than S_IN1.



Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Result of the comparison (SAFEBOOL)

13.4.4.6 LT_NS (less than, not safe)

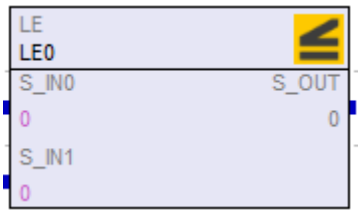
Indicates whether IN0 is less than IN1.



Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Result of the comparison (BOOL)

13.4.4.7 LE (less than or equal to)

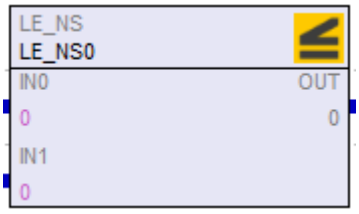
Indicates whether S_IN0 is less than or equal to S_IN1.



Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Result of the comparison (SAFEBOOL)

13.4.4.8 LE_NS (less than or equal to, not safe)

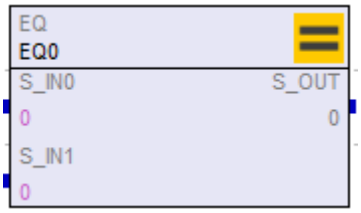
Indicates whether IN0 is less than or equal to IN1.



Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Result of the comparison (BOOL)

13.4.4.9 EQ (equal)

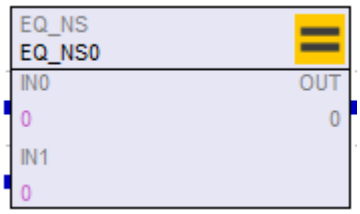
Indicates whether S_IN0 is equal to S_IN1.



Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Result of the comparison (SAFEBOOL)

13.4.4.10 EQ_NS (equal, not safe)

Indicates whether IN0 is equal to IN1.

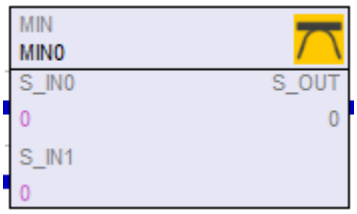


Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Result of the comparison (BOOL)

13.4.4.11 MIN (Minimum)

Outputs the lower value of S_IN0 and S_IN1.

If a constant is connected to an input, it will be the upper limit for the other value.

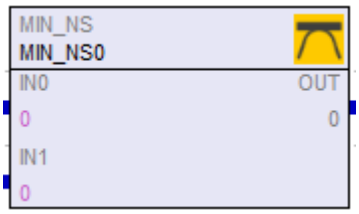


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Minimum of inputs (SAFEDINT)

13.4.4.12 MIN_NS (Minimum, not safe)

Outputs the lower value out of IN0 and IN1.

If a constant is connected to an input, it will be the upper limit for the other value.

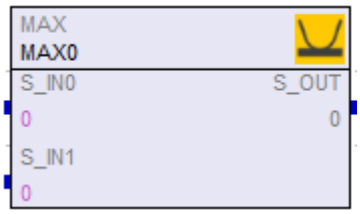


Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Minimum of inputs (DINT)

13.4.4.13 MAX (Maximum)

Outputs the larger value of S_IN0 and S_IN1.

If a constant is connected to an input, it will be the lower limit for the other value.

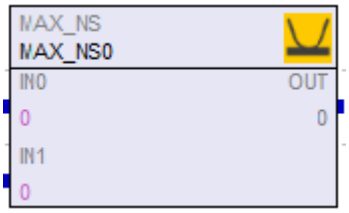


Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Output variables	
S_OUT	Maximum of inputs (SAFEDINT)

13.4.4.14 MAX_NS (Maximum, not safe)

Outputs the larger value of IN0 and IN1.

If a constant is connected to an input, it will be the lower limit for the other value.

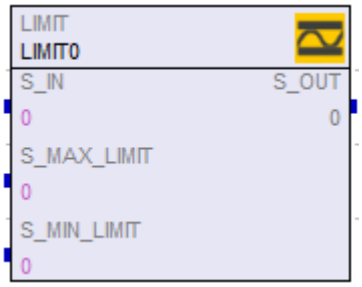


Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Output variables	
OUT	Maximum of inputs (DINT)

13.4.4.15 LIMIT (within limits)

Delivers an input value within the upper and lower limits. If S_IN is greater than S_MAX_LIMIT, then S_MAX_LIMIT will be output; if S_IN is less than S_MIN_LIMIT, then S_MIN_LIMIT will be output; otherwise, S_IN will be output.

If S_MAX_LIMT is less than S_MIN_LIMT, SAFEDINT_ERROR will be output.

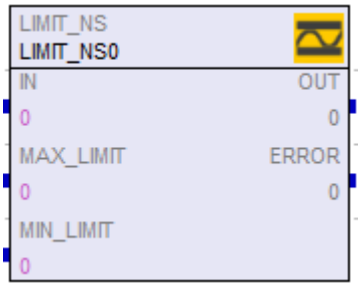


Input variables	
S_IN	Input (SAFEDINT)
S_MAX_LIMIT	Upper limit (SAFEDINT)
S_MIN_LIMIT	Lower limit (SAFEDINT)
Output variables	
S_OUT	Result (SAFEDINT), SAFEDINT_ERROR in the event of an error

13.4.4.16 LIMIT_NS (within limits, not safe)

Delivers an input value within the upper and lower limits. If IN is greater than MAX_LIMIT, then MAX_LIMIT will be output; if IN is less than MIN_LIMIT, then MIN_LIMIT will be output; otherwise, IN will be output.

If MAX_LIMIT < MIN_LIMIT, ERROR will be set to TRUE and OUT to IN.

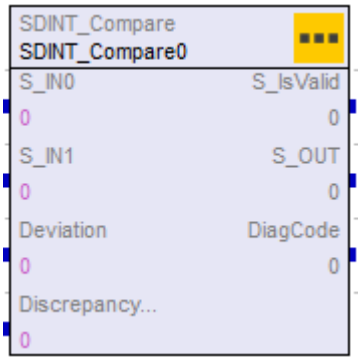


Input variables	
IN	Input (DINT)
MAX_LIMIT	Upper limit (DINT)
MIN_LIMIT	Lower limit (DINT)
Output variables	
OUT	Result (DINT)
ERROR	TRUE if MAX_LIMIT < MIN_LIMIT

13.4.4.17 SDINT_Compare

Compares two SAFEDINT inputs S_IN0 and S_IN1 with each other and outputs the average value.

Deviation indicates how large the difference is allowed to be. Meanwhile, DiscrepancyTime is the time period during which the difference is allowed to be larger. If the absolute value of the difference between S_IN0 and S_IN1 is greater than "Deviation" longer than "DiscrepancyTime", S_IsValid will be set to SAFEBOOL_FALSE and S_OUT will be set to 0 until the difference is less than or equal to "Deviation" again.



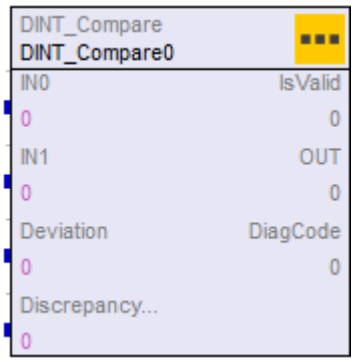
Input variables	
S_IN0	Input 1 (SAFEDINT)
S_IN1	Input 2 (SAFEDINT)
Deviation	Allowed difference (DINT)
DiscrepancyTime	Time during which the difference is allowed to be greater (DINT)
Output variables	
S_IsValid	SAFEBOOL_TRUE if S_OUT is valid (SAFEBOOL)
S_OUT	Average value of S_IN0 and S_IN1 or 0 in the event of an error (SAFEDINT)
DiagCode	Status display (HDINT)

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C000	SafeDintError	SAFEDINT_ERROR at S_IN0 or S_IN1
C001	DeviationError	Value for deviation is invalid (< 0)
C002	DiscrepancyError	Value for DiscrepancyTime is invalid (< 0)
C003	DiffError	The difference between S_IN0 and S_IN1 has been greater than Deviation longer than DiscrepancyTime
FB-specific status codes (no error)		
0000	Idle/Init	The function block is not active (initial state) or Activate = FALSE Ready = FALSE S_Y_Value = 0 Error = FALSE Error = FALSE
8000	NoError	S_OUT is a valid value

13.4.4.18 DINT_Compare

Compares two DINT inputs "IN0" and "IN1" with each other and outputs the average value.

Deviation indicates how large the difference is allowed to be. Meanwhile, DiscrepancyTime is the time period during which the difference is allowed to be larger. If the absolute value of the difference between IN0 and IN1 is greater than "Deviation" longer than "DiscrepancyTime", IsValid will be set to FALSE and OUT will be set to 0 until the difference is less than or equal to "Deviation" again.



Input variables	
IN0	Input 1 (DINT)
IN1	Input 2 (DINT)
Deviation	Allowed difference (DINT)
DiscrepancyTime	Time during which the difference is allowed to be greater (DINT)
Output variables	
IsValid	SAFEBOOL_TRUE if S_OUT is valid (BOOL)
OUT	Average value of IN0 and IN1 or 0 in the event of an error (DINT)
DiagCode	Status display (HDINT)

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	DeviationError	Value for deviation is invalid (< 0)
C002	DiscrepancyError	Value for DiscrepancyTime is invalid (< 0)
C003	DiffError	The difference between S_IN0 and S_IN1 has been greater than Deviation longer than DiscrepancyTime
FB-specific status codes (no error)		
0000	Idle/Init	The function block is not active (initial state) or Activate = FALSE Ready = FALSE S_Y_Value = 0 Error = FALSE
8000	NoError	OUT is a valid value

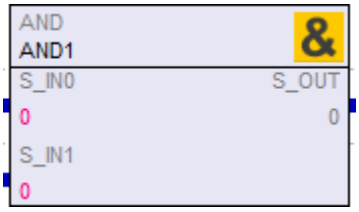
13.4.5 Logical function blocks

Tbl. 5: Overview

→ " AND", page 198		Logical AND operator
→ " AND_5I", page 199		Computes the logical AND of five inputs
→ " AND_NS - AND NOT SAFE", page 199		Computes the logical AND of a BOOL input and a SAFEBOOL input
→ " NOT", page 200		Inverts the input
→ " OR", page 200		Logical OR operator
→ " OR_5I", page 201		Computes the logical OR of five inputs
→ " XOR", page 201		Logical XOR operator
→ "AND_XI", page 202		Computes the logical AND of a variable number of inputs
→ "OR_XI", page 203		Computes the logical OR of a variable number of inputs
→ "SAFEBOOL_TO_BOOL", page 170		Converting a SAFE input into a UNSAFE output
→ "SAFEDINT_TO_DINT", page 170		Conversion of a SAFEDINT input into an UNSAFE output
→ "SERROR_TO_SFALSE", page 171		Conversion of SAFEBOOL_ERROR states to SAFEBOOL_FALSE.

13.4.5.1 AND

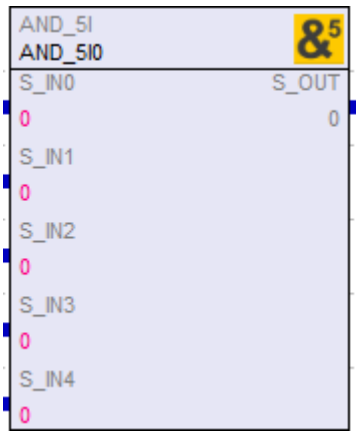
Computes the logical AND of two SAFEBOOL inputs.



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.2 AND_5I

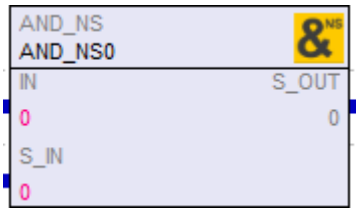
Computes the logical AND of five SAFEBOOL inputs



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
S_IN2	Input 3 (SAFEBOOL)
S_IN3	Input 4 (SAFEBOOL)
S_IN4	Input 5 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.3 AND_NS - AND NOT SAFE

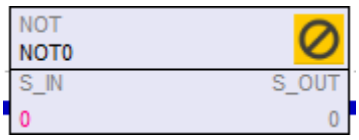
Computes the logical AND of a BOOL input and a SAFEBOOL input.



Input variables	
IN	Input 1 (BOOL)
S_IN	Input 2 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.4 NOT

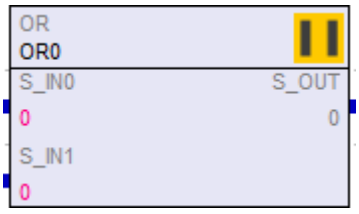
Reversing of the input.



Input variables	
S_IN	Eingang (SAFEBOOL)
Output variables	
S_OUT	Output (SAFEBOOL)

13.4.5.5 OR

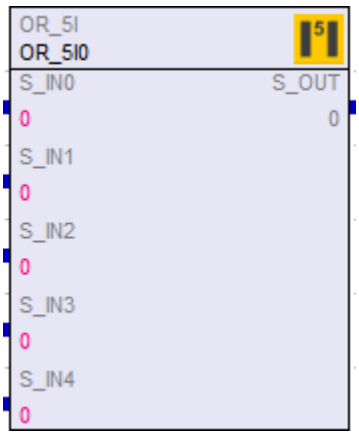
Computes the logical OR of two SAFEBOOL inputs.



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.6 OR_5I

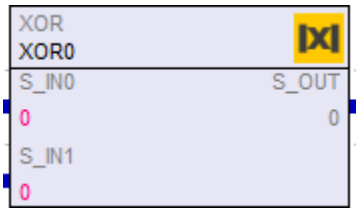
Computes the logical OR of five SAFEBOOL inputs.



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
S_IN2	Input 3 (SAFEBOOL)
S_IN3	Input 4 (SAFEBOOL)
S_IN4	Input 5 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.7 XOR

Computes the logical XOR of two SAFEBOOL inputs.

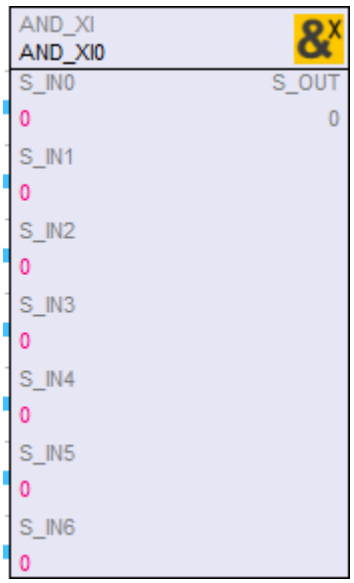


Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

13.4.5.8 AND_XI

Computes the logical AND of a variable number of SAFEBOOL inputs (maximum of 20).

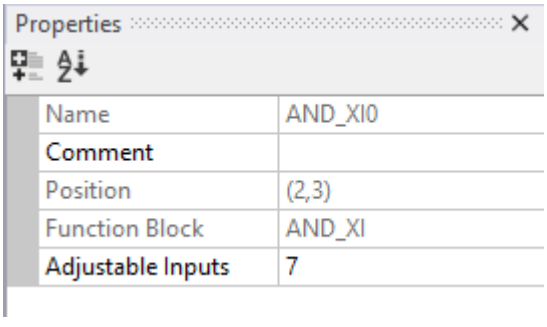
These variable inputs can be identified based on the corresponding blue symbols.



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
...	...
S_IN18	Input 19 (SAFEBOOL)
S_IN19	Input 20 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

Configuring the inputs

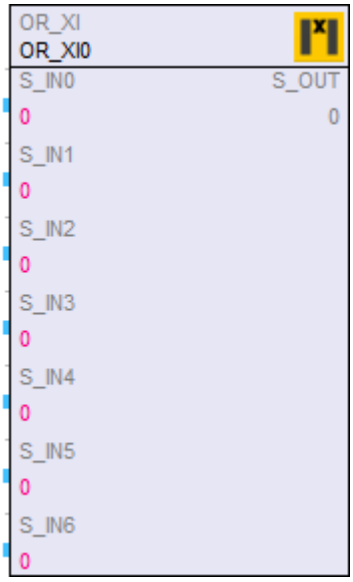
When you click on the function block, the information in the Properties pane will include the Adjustable Inputs entry. You can use this entry to specify how many inputs should be used.



13.4.5.9 OR_XI

Computes the logical OR of a variable number of SAFEBOOL inputs (maximum 20).

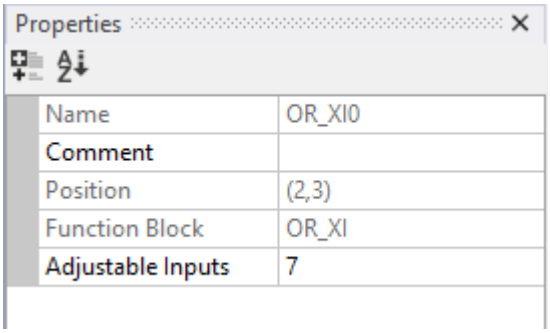
These variable inputs can be identified based on the corresponding blue symbols.



Input variables	
S_IN0	Input 1 (SAFEBOOL)
S_IN1	Input 2 (SAFEBOOL)
...	...
S_IN18	Input 19 (SAFEBOOL)
S_IN19	Input 20 (SAFEBOOL)
Output variables	
S_OUT	Output of logic link (SAFEBOOL)

Configuring the inputs

When you click on the function block, the information in the Properties pane will include the Adjustable Inputs entry. You can use this entry to specify how many inputs should be used.



13.4.6 Timer function blocks

Tbl. 6: Overview

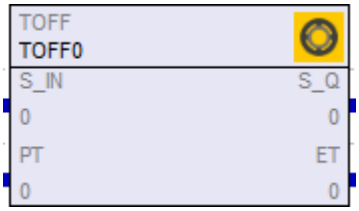
→ " TOFF", page 204		Switch-off delay
→ " TON", page 205		Switching-on delay
→ " TP", page 205		Pulse duration

Special features of the timer FBs Ton, Toff, TP, TP1:

If an edge flag (auxiliary variable used to detect a rising or falling edge) has a value of SAFEBOOL_ERROR, the output will be set to SAFEBOOL_FALSE and the timer will be stopped. The edge flag will continue to have a value of SAFEBOOL_ERROR until the input returns to a value of SAFEBOOL_FALSE.

13.4.6.1 TOFF

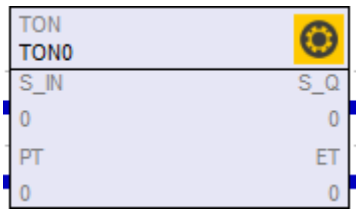
Off-delay for a SAFEBOOL signal.



Input variables	
S_IN	A falling edge will start the timer (SAFEBOOL)
PT	Delay time in ms (DINT)
Output variables	
S_Q	FALSE if the delay time has elapsed (SAFEBOOL)
ET	Amount of delay time that has already elapsed in ms (DINT)

13.4.6.2 TON

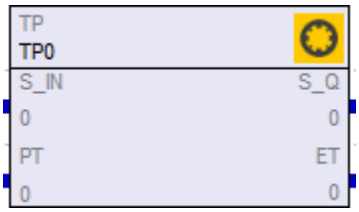
On-delay for a SAFEBOOL signal.



Input variables	
S_IN	A rising edge will start the timer (SAFEBOOL)
PT	Delay time in ms (DINT)
Output variables	
S_Q	TRUE if the delay time has elapsed (SAFEBOOL)
ET	Amount of delay time that has already elapsed in ms (DINT)

13.4.6.3 TP

Pulse time for a SAFEBOOL signal.



Input variables	
S_IN	A rising edge will start the timer (SAFEBOOL)
PT	Duration of pulse in ms (DINT)
Output variables	
S_Q	TRUE during the specified time (SAFEBOOL)
ET	Amount of delay time that has already elapsed in ms (DINT)

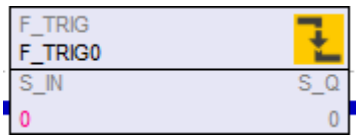
13.4.7 Trigger function blocks

Tbl. 7: Overview

→ " F_TRIG", page 206		Trigger detection of a falling edge
→ " R_TRIG", page 206		Trigger detection of a rising edge

13.4.7.1 F_TRIG

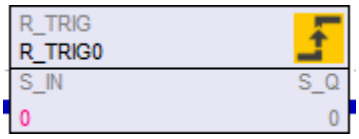
Used to detect a falling edge at its input.



Input variables	
S_IN	If there is a falling edge, Q will be set to TRUE for one cycle (SABEOOL)
Output variables	
S_Q	(SAFEBOOL)

13.4.7.2 R_TRIG

Used to detect a rising edge at the input.



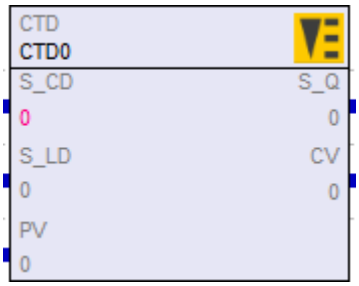
Input variables	
S_IN	When there is a rising edge, Q will be set to TRUE for one cycle (SAFEBOOL)
Output variables	
S_Q	(SAFEBOOL)

13.4.8 Counter function blocks

Tbl. 8: Overview

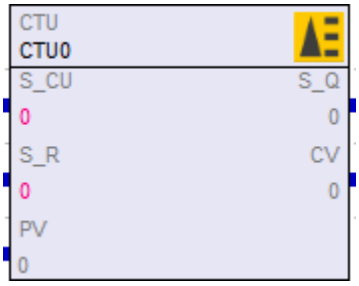
→ " CTD - Down counter", page 207		Down counter
→ " CTU - Up counter", page 207		Up counter
→ " CTUD - Up/down counter", page 208		Down and up counter

13.4.8.1 CTD - Down counter



Input variables	
S_CD	If there is a rising edge, the system will count down by 1 (SAFEBOOL)
S_LD	Initializes CV to PV (SAFEBOOL)
PV	Initial value from where counting starts (DINT)
Output variables	
S_Q	TRUE if CV reaches the minimum of 0 (SAFEBOOL)
CV	Current value of the counter (DINT)

13.4.8.2 CTU - Up counter

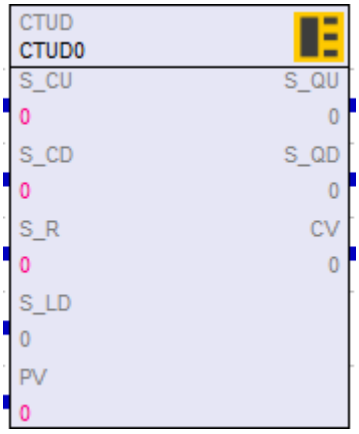


Input variables	
S_CU	On the rising edge, the system counts up by 1 (SAFEBOOL)
S_R	Initializes CV to 0 (SAFEBOOL)
PV	Maximum value (DINT)
Output variables	
S_Q	TRUE if CV reaches the maximum (PV) (SAFEBOOL)
CV	Current value of the counter (DINT)

13.4.8.3 CTUD - Up/down counter

CTUD will first check the Reset input, which takes precedence over the LD input. If the Reset input or LD is set, the counting direction inputs will not be checked further. Otherwise, the positive counting direction will be processed first and the negative counting direction afterwards. In other words, if the count value (CV) is 0 or < PV, CV will not change, since CV will first be incremented and then decremented.If CV = PV, CV will be decreased, since CV can no longer be incremented and CV will be decremented after the failed increment attempt is made.

If PV = 0, both outputs (QU and QD) are set to SAFEBOOL_TRUE.



Input variables	
S_CU	On the rising edge, the system counts up by 1 (SAFEBOOL)
S_CD	If there is a rising edge, the system will count down by 1 (SAFEBOOL)
S_R	Initializes CV to 0 (SAFEBOOL)
S_LD	Initializes CV to PV (SAFEBOOL)
PV	Initial value from where counting starts (DINT)
Output variables	
S_QU	TRUE if CV reaches the maximum (PV) (SAFEBOOL)
S_QD	TRUE if CV reaches the minimum of 0 (SAFEBOOL)
CV	Current value of the counter (DINT)

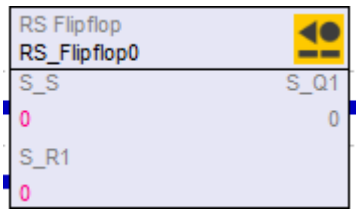
13.4.9 FlipFlop function blocks

Tbl. 9: Overview

→ " RS_Flipflop", page 209		Override reset
→ " SR_Flipflop", page 209		Override set

13.4.9.1 RS_Flipflop

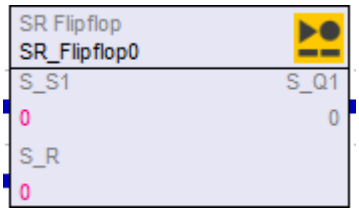
Flip-flop with preferential reset.



Input variables	
S_S	If TRUE, Q1 will be set to TRUE (SAFEBOOL)
S_R1	If TRUE, Q1 will be set to FALSE even if S_S has a value of TRUE (SAFEBOOL)
Output variables	
S_Q1	Output of logic link (SAFEBOOL)

13.4.9.2 SR_Flipflop

Flip-flop with preferential set.



Input variables	
S1	If TRUE, then Q1 will be constantly set to TRUE (SAFEBOOL)
R	If TRUE, then Q1 will be set to FALSE if S1 has a value of false (SAFEBOOL)
Output variables	
Q1	Output of logic link (SAFEBOOL)

13.5 Complex function blocks

Drive-specific function blocks are not listed.

Tbl. 10: Overview

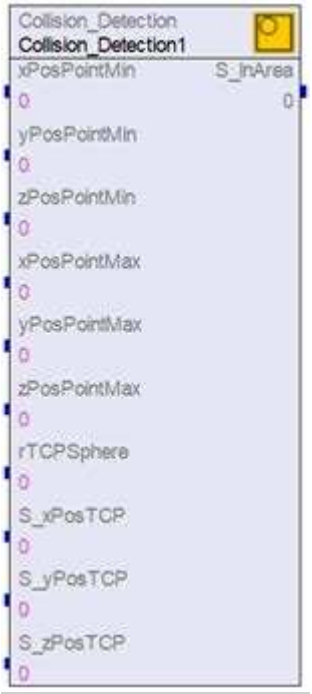
→ " SF_Optional_PwdII function", page 294	Used for password requests and password checks for optional modules. This function block should only be used together with the SF_Optional_Switch function block.
→ "SF_Optional_Switch function", page 300	This function block is used with optional modules to switch off inputs or to switch to another data source (e.g. constant). This function block is only to be used together with the SF_Optional_Pwd/SF_Optional_PwdII function block.
→ "SF_Optional_Switch_ SDint", page 301	This function block is used with optional modules to switch off inputs or to switch to another data source (e.g. constant).
→ "SF_Optional_Switch_ Dint", page 302	This function block is used with optional modules to switch off inputs or to switch to another data source (e.g. constant).
→ " SF_OutControl function", page 303	You can use this function block to control a Safety output with a signal from the functional application and a Safety signal with an optional start inhibit.
→ " SF_SafetyRequest function", page 308	This function block serves as an interface to a generic actuator (a safety drive or a safety valve, for example) and can be used to bring this actuator to a safe state.
→ " SF_ TestableSafetySensor function", page 313	This function block detects, for example, the loss of sensor detection, longer response times than specified, and static ON signals in single-channel sensor systems. It can be used for externally testable safety sensors (ESPE).
→ " SF_TwoHandControlTypeII function", page 322	This function block offers two-hand control functionality (see EN 574, section 4 type II).
→ " SF_TwoHandControlTypeIII function", page 327	This function block offers two-hand control functionality (see EN 574, section 4 type III). A fixed time difference of 500 ms is specified.
→ "SF_HGW_Login", page 332	This function block is used to connect an HGW module. It features two separate login mechanisms. The first login mechanism can be used to set a machine number for the system, while the second login mechanism is used to securely establish a connection between the HGW and Safety CPU.
Numerical function blocks	
→ " SF_LimitComparator", page 338	Checks the input signal for upper and lower limit.
→ "SF_LimitComparator_ Dint", page 342	Checks the input signal for upper and lower limit.
→ " SF_Numeric_ Selector", page 343	Like SF_ModeSelector, but with SafeDint values. You can use the S_ModeX SafeBool inputs to select a SafeDint input that will be connected through to the S_Out output.
→ " SF_RampMonitor", page 347	RampMonitor monitors the speed reduction in the event of an emergency stop. A rising edge at S_MonitoringOn will cause the current speed to be applied as the initial speed for the monitoring function.
→ " SF_DirectionMonitor", page 351	The S_Direction input determines the direction in which the input value should be checked (positive travel: S_Direction = FALSE; negative travel: S_Direction = TRUE).
→ " SF_SkewMonitor", page 355	A rising edge at the S_Calibrate input will start the calibration process and set S_CalibrationOk to FALSE

13.5.1 Collision_Detection function

The Collision_Detection function block was developed in order to detect collisions between a robot's tool center point (TCP) and other objects in its work environment. This function block accepts two points that span a working space when considered together. This working space, in turn, is the area within which the TCP is allowed to move. You can specify an enveloping sphere around the TCP.This enveloping sphere is a mathematical representation that represents the outer boundaries of the TCP on the robot in its current configuration.

The function block's output will signal whether the TCP coordinates with the enveloping sphere are located within the working space. If they are, a value of s_true will be output. If the TCP exits the working space, a value of s_false will be output instead.

The information provided by the output can be used in order to take actions to stop the robot or prevent a collision.



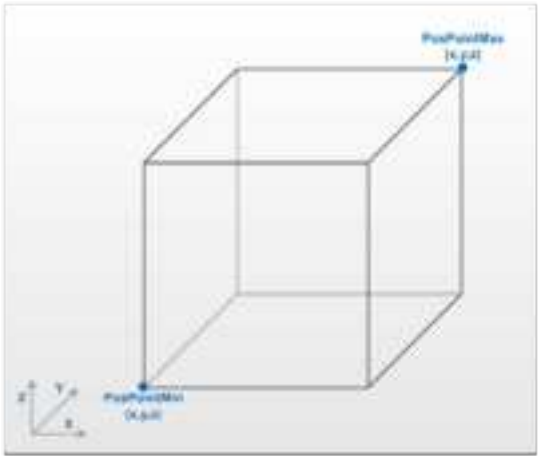
13.5.1.1 Functionalities

The following functions are included in the function block:

- The function block monitors the position of a robot (TCP) in the working space, places an enveloping sphere over its current position, and detects collisions by comparing the enveloping sphere with the defined working space boundaries.
- The output indicates, with a Boolean value, whether the working space boundaries have been crossed.

13.5.1.2 Workspace

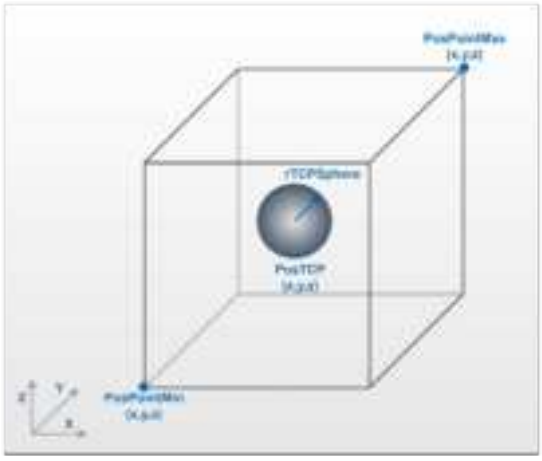
The working space will be defined based on the XYZ coordinates of the PosPointMin and PosPointMax points. These two points will span a cuboid. In order for the function block to work properly, all three PosPointMax coordinates (x, y, z,) must be greater than those of PosPointMin. Within this context, the coordinates will be relative to the kinematic structure's base coordinate system. You can use DINT constants to define the working space with the PosPointMin and PosPointMax points.



The figure above shows the base coordinate system and the working space spanned by the PosPointMin and PosPointMax points.

13.5.1.3 Tool Center Point and enveloping sphere

Depending on the robot's kinematic structure, the coordinates for the Tool Center Point (TCP) can be taken directly from a safe sensor system or calculated by the DH-Transformation function block. An enveloping sphere will be placed around the TCP, with the radius of this sphere being defined by the value of rTCPSphere. You can create this value with a DINT constant.

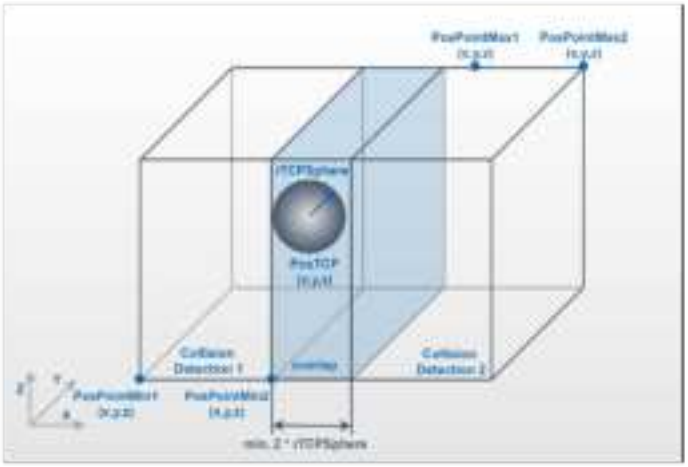


The figure above shows an enveloping sphere in the working space. This enveloping sphere is defined by PosTCP and rTCPSphere.

13.5.1.4 Creating a complex working space

You can create complex working spaces by connecting individual working spaces with a logical operator. To do this, you will need to connect the outputs of multiple Collision_Detection function blocks with an OR. The OR function block's output will then compute whether the TCP is located within the resulting complex working space.

If you are using a complex working space, the individual spatial areas of the working spaces must overlap in such a way that the enveloping sphere will fit the overlapping space along all axes. In other words, this overlap must be greater than two times rTCPSphere.



The figure above shows two overlapping working spaces. The overlap is greater than two times rTCPSphere, meaning that the TCP can move from working space 1 into working space 2.

13.5.1.5 Interfaces

Input variables	Description
xPosPointMin	The minimum X position of the workspace is defined via this input. (DINT)
yPosPointMin	The minimum Y position of the workspace is defined via this input. (DINT)
zPosPointMin	The minimum Z position of the workspace is defined via this input. (DINT)
zPosPointMin	The maximum X position of the workspace is defined via this input. (DINT)
yPosPointMax	The maximum Y position of the workspace is defined via this input. (DINT)
zPosPointMax	The maximum Z position of the workspace is defined via this input. (DINT)
rTCPSphere	The sphere radius of the enveloping sphere is defined via this input. (DINT)
S_xPosTCP	The X position of the TCP to be monitored is defined via this input. (SAFEDINT)
S_yPosTCP	The Y position of the TCP to be monitored is defined via this input. (SAFEDINT)
S_zPosTCP	The Z position of the TCP to be monitored is defined via this input. (SAFEDINT)
Output vari-	Description

ables	
S_InArea	The output variable will have a value of s_true if the sphere is completely confined within the working space that is spanned. The value will instead be s_false if the sphere intersects or leaves the area boundaries. (SAFEBOOL)

13.5.1.6 Example showing how to connect the function block

The following figure shows how the working space can be defined with DINT constants. The position can for example be calculated by the DH-Transformation function block.

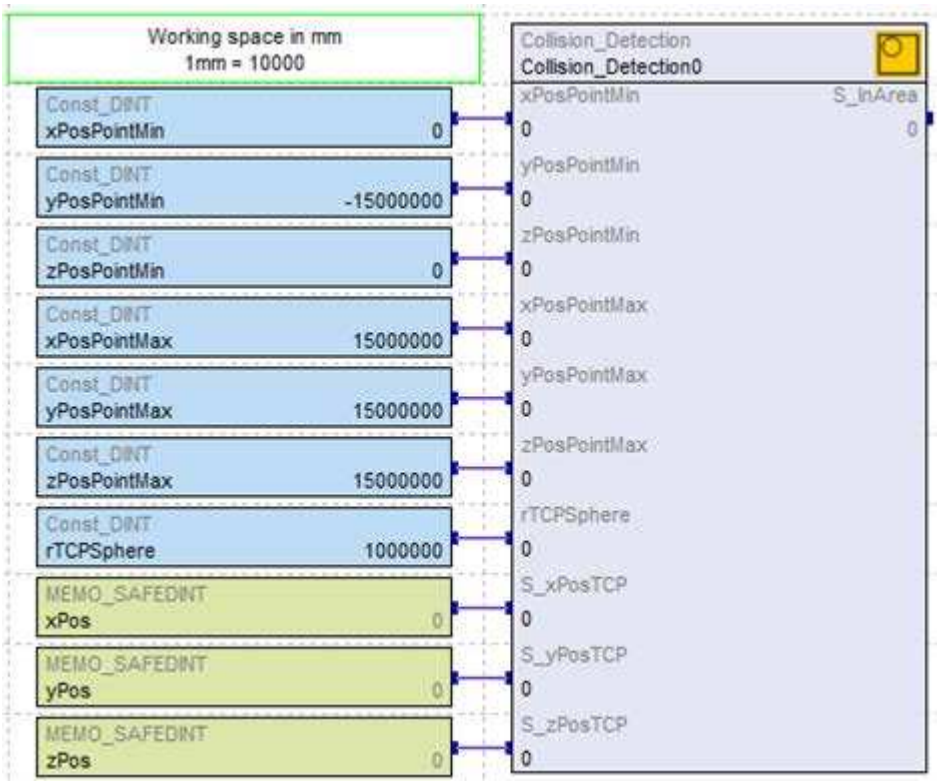


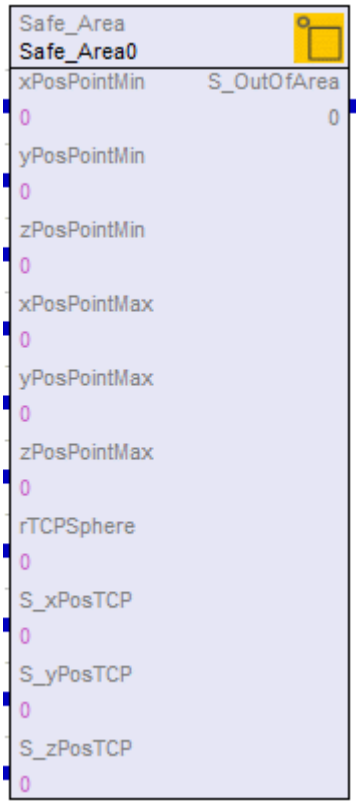
Abb. 3: Example Six-axis robot

13.5.2 Safe_Area function

The Safe_Area function block was developed in order to detect the entry of a robot's tool center point (TCP) into a safe work space. This function block accepts two points that span a working space when considered together. This working space is the area that the TCP is not allowed to touch or intersect. You can specify an enveloping sphere around the TCP.This enveloping sphere is a mathematical representation that represents the outer boundaries of the TCP on the robot in its current configuration.

The function block's output will signal whether the TCP coordinates with the enveloping sphere are located outside the safe working space. If they are, a value of s_true will be output. If the safety area is touched or entered, a value of s_false will be output instead.

The information provided by the output can be used in order to take actions to stop the robot or prevent a collision.



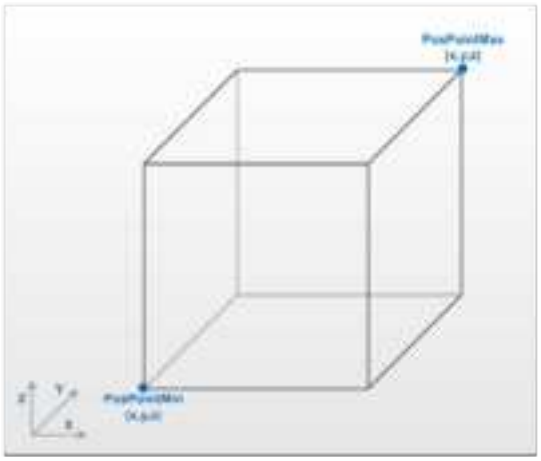
13.5.2.1 Functionalities

The following functions are included in the function block:

- The function block monitors the position of a robot (TCP) outside of the safe working space, places an enveloping sphere over its current position, and detects collisions by comparing the enveloping sphere with the defined safe working space boundaries.
- The output indicates, with a Boolean value, whether the safe working space boundaries have been crossed.

13.5.2.2 Workspace

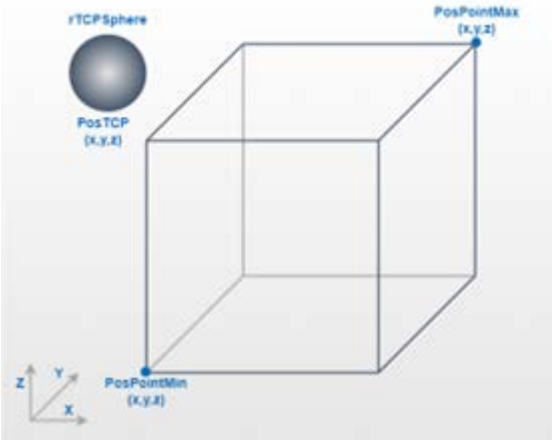
The safety area will be defined based on the XYZ coordinates of the PosPointMin and PosPointMax points. These two points will span a cuboid. In order for the function block to work properly, all three PosPointMax coordinates (x, y, z,) must be greater than those of PosPointMin. Within this context, the coordinates will be relative to the kinematic structure's base coordinate system. You can use DINT constants to define the safe working space with the PosPointMin and PosPointMax points.



The figure above shows the base coordinate system and the safe working space spanned by the PosPointMin and PosPointMax points.

13.5.2.3 Tool Center Point and enveloping sphere

Depending on the robot's kinematic structure, the coordinates for the Tool Center Point (TCP) can be taken directly from a safe sensor system or calculated by the DH-Transformation function block. An enveloping sphere will be placed around the TCP, with the radius of this sphere being defined by the value of rTCPSphere. You can create this value with a DINT constant.



The figure above shows an enveloping sphere outside of the safe working space. This enveloping sphere is defined by PosTCP and rTCPSphere.

13.5.2.4 Interfaces

Input variables	Description
xPosPointMin	This input is used to define the minimum X position of the safe area. (DINT)
yPosPointMin	This input is used to define the minimum Y position of the safe area. (DINT)
zPosPointMin	This input is used to define the minimum Z position of the safe area. (DINT)
xPosPointMax	This input is used to define the maximum X position of the safe area. (DINT)
yPosPointMax	This input is used to define the maximum Y position of the safe area. (DINT)
zPosPointMax	This input is used to define the maximum Z position of the safe area. (DINT)
rTCPSphere	The sphere radius of the enveloping sphere is defined via this input. (DINT)
S_xPosTCP	The X position of the TCP to be monitored is defined via this input. (SAFEDINT)
S_yPosTCP	The Y position of the TCP to be monitored is defined via this input. (SAFEDINT)
S_zPosTCP	The Z position of the TCP to be monitored is defined via this input. (SAFEDINT)
Output variables	Description
S_InArea	The output variable will have a value of s_true if the sphere is completely outside of the safety area that is spanned. The value will instead be s_false if the sphere touches or enters the area boundaries. (SAFEBOOL)

13.5.2.5 Example showing how to connect the function block

The following figure shows how the safety area can be defined with DINT constants. The position can for example be calculated by the DH-Transformation function block.

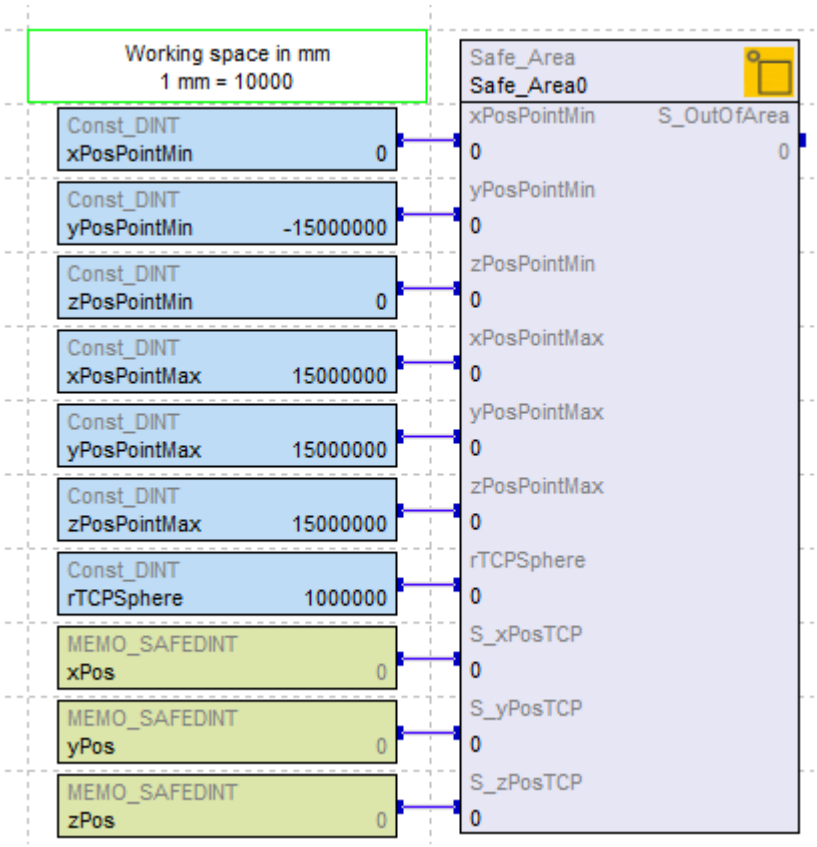
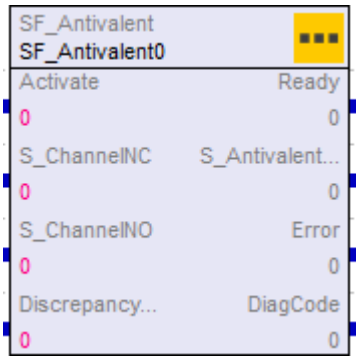


Abb. 4: Example: Six-axis robot

13.5.3 SF_Antivalent function

This function converts two antivalent SAFEBOOL inputs (NO/NC pair) to a SAFEBOOL output with discrepancy time monitoring. Please note that this function block should not be used as a standalone block, since it does not have a restart inhibit function. You will need to connect the output to other safety-relevant functions instead.



Input variables	
Activate	Activates the function block (BOOL).
S_ChannelNC	NC stands for "normally closed." Input for the NC connection. (SAFEBOOL) False: NC open TRUE: NC closed
S_ChannelNO	NO stands for "normally open." Input for the NO connection. (SAFEBOL) FALSE: NO open TRUE: NO closed
DiscrepancyTime	Maximum monitoring time for the discrepancy status of both inputs in ms. (DINT)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL).
S_AntivalentOut	Output of logic link (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

Both inputs are independent of each other. The function block's output will signal the evaluation result for both channels.

If S_AntivalentOut = TRUE and one of the safety-relevant inputs changes, then the output will immediately switch to FALSE.

Discrepancy time monitoring: The discrepancy time is the maximum time that both inputs can have the same state (for example, both inputs being either TRUE or FALSE) without the function block detecting an error. Discrepancy time monitoring starts when there is a state change at one of the inputs. The function block will detect an error if the two inputs do not have antivalent values when the discrepancy time elapses. The inputs must be switched symmetrically.In other words, monitoring will be carried out both for the switching-on and switching-off processes.

13.5.3.1 Error detection

The function block will monitor the discrepancy time between the NO and NC channels.

13.5.3.2 Error behavior

The S_AntivalentOut output will be set to FALSE. Error will be set to TRUE. DiagCode will indicate the error conditions.

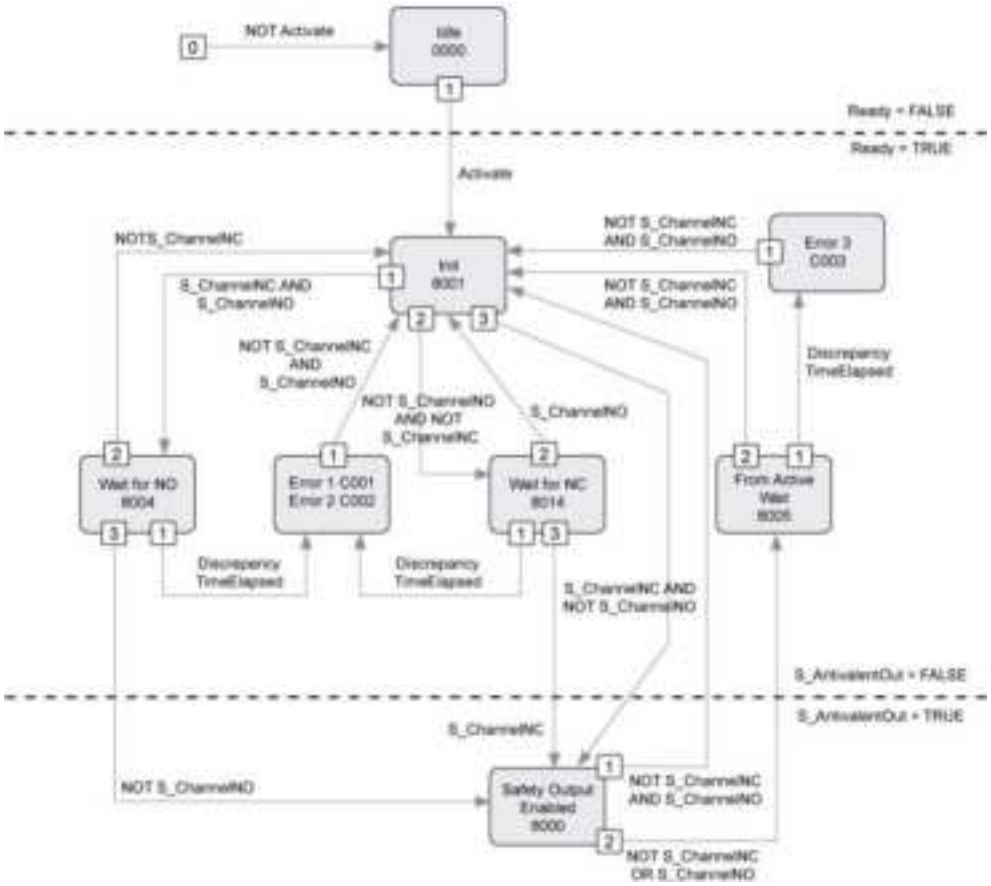
There is no reset input that will reset an error. If an error occurs at the inputs, the new input values with correct values must be able to reset the error flag. (Example: If a faulty switch has been replaced, reusing the switch will result in the correct output.)

13.5.3.3 Function block-specific errors and status codes

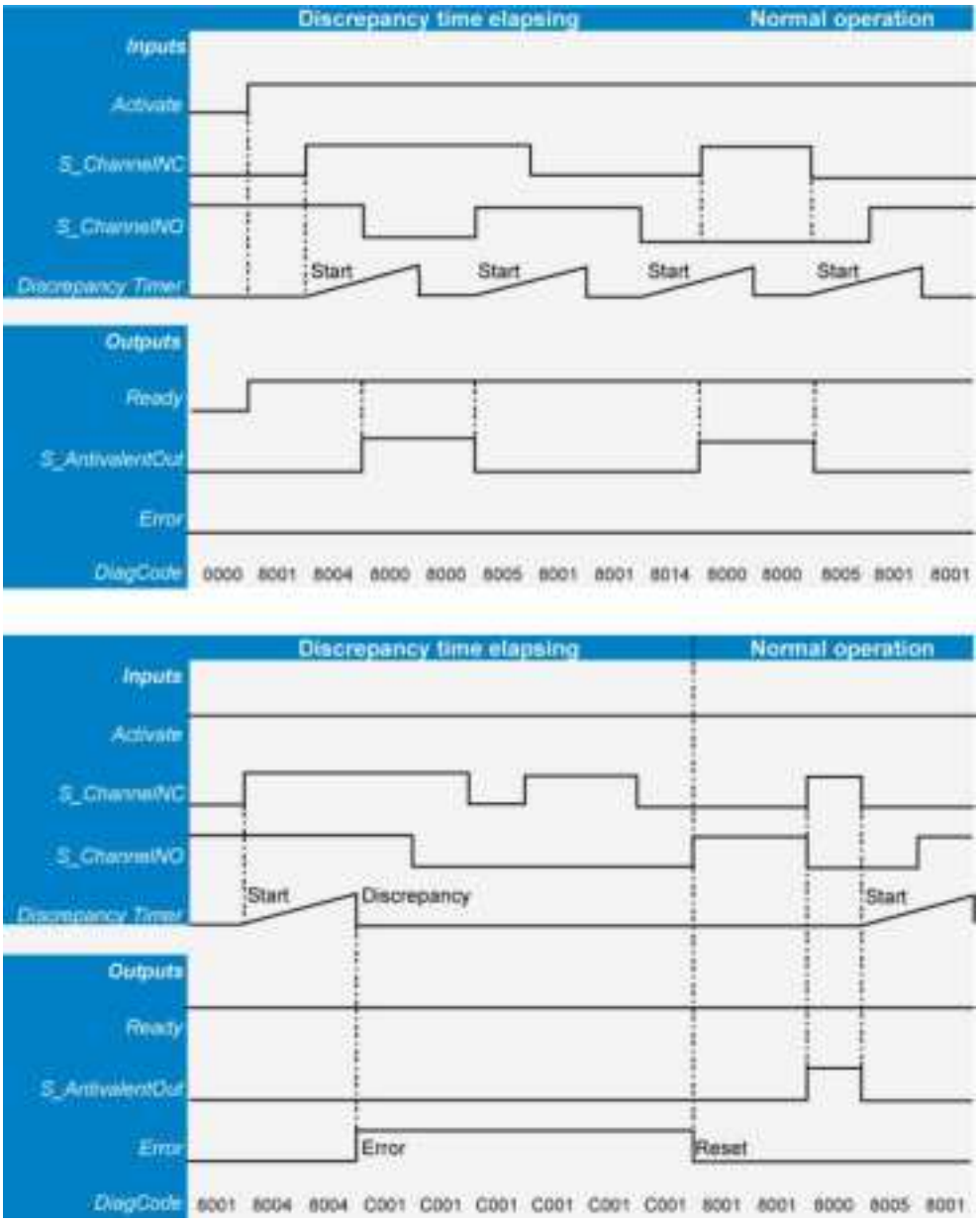
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error 1	Discrepancy time elapsed in state 8004 Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014 Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005 Ready = TRUE S_AntivalentOut = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AntivalentOut = FALSE Error = FALSE
8001	Init	The function block has detected an activation and the status is now activated. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs are in their active state and in antivalent mode. Ready = TRUE S_AntivalentOut = TRUE Error = FALSE

DiagCode	Status name	Status description and output setting
8004	Wait for NO	ChannelNC has been switched to TRUE and is waiting for ChannelNO to be switched to FALSE. Discrepancy timer starts. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8014	Wait for NC	ChannelNO has been switched to TRUE and is waiting for ChannelNC to be switched to FALSE. Discrepancy timer starts. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE
8005	From Active Wait	A channel was switched to inactive and is waiting for the other channel to be switched to inactive as well. Ready = TRUE S_AntivalentOut = FALSE Error = FALSE

13.5.3.4 Status Diagram SF_Antivalent

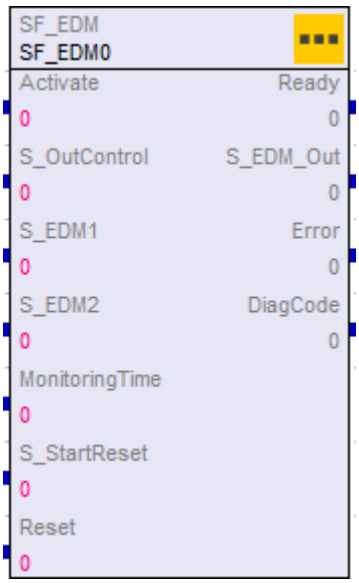


13.5.3.5 Time Diagram SF_Antivalent



13.5.4 SF_EDM function

External device monitoring. This function block controls a Safety output and monitors actuators.



Input variables	
S_OutControl	Control signal from the previous safety function blocks. Typical signals from the following FBs (e.g. SF_OutControl, SF_TwoHandControlTypeII, etc.). (SAFEBOOL) FALSE: Deactivates the safety output TRUE: Activates the safety output
S_EDM1	Feedback signal of the first connected actuator. (SAFEBOOL) FALSE: Switching state of first connected actuator TRUE: Output state of first connected actuator
S_EDM2	Feedback signal of second connected actuator. If only one signal is used in the application, the signal must be connected to S_EDM1 and S_EDM2. S_EDM1 and S_EDM2 will then be controlled with the same signal. (SAFEBOOL) FALSE: Switching state of second connected actuator TRUE: Output state of second connected actuator
MonitoringTime	Max. response time of connected and monitored actuators (DINT)
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU has been started (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_EDM_Out	Controls the actuator. The result is monitored through feedback signal S_EDMx. (SAFEBOOL) FALSE: Deactivates the connected actuator TRUE: Activates the connected actuator

The SF_EDM function block controls a Safety output and monitors controlled actuators.

This function block monitors the original state of the actuators with the feedback signals (S_EDM1 and S_EDM2) before the actuators are activated by the function block.

The function block monitors the switching state of the actuators (MonitoringTime) after the actuators have been activated by the function block.

When possible, you should use two individual feedback signals in order to accurately diagnose the connected actuators. While you can instead use a shared feedback signal from the two connected actuators to implement a simple actuator diagnostic function, this function will be limited. If you use a shared signal, you will need to connect it both to the S_EDM1 and S_EDM2 inputs. S_EDM1 and S_EDM2 will then be controlled by the same signal.

The selected switching devices used in the safety function must belong to the category specified in the risk assessment (EN ISO 13849).

Optional start inhibit:

- Start inhibit in the event of block activation.

The S_StartReset input should only be activated if it has been ensured that no potential hazardous situations can occur as a result of starting.

13.5.4.1 Error detection

The following conditions will force a switch to the error state:

- Invalid static reset signal in the process.
- Invalid EDM signal in the process.
- S_OutControl and Reset are incorrectly connected as the result of a programming mistake.

13.5.4.2 Error behavior

In error conditions, the output will be as follows:

- In the event of an error, the S_EDM_Out output will be set to FALSE and remain in this safe state.
- An EDM error message must always be reset with a rising edge at the Reset input.
- A reset error message (reset signal already set in the event of an error, despite there not being a rising edge) can be reset by setting the Reset input to FALSE.

After block activation, you can reset the optional start inhibit with a rising edge at the Reset input.

DiagCode	Status name	Status description and output setting
FB-specific error codes		

DiagCode	Status name	Status description and output setting
C001	Reset Error 1	Static reset signal in state 8001. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C011	Reset Error 21	Static reset signal or the same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C010. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C021	Reset Error 22	Static reset signal or the same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C020. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C031	Reset Error 23	Static reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C030. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C041	Reset Error 31	Static reset signal or the same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C040. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C051	Reset Error 32	Static reset signal or the same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C050. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C061	Reset Error 33	Static reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C060. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C071	Reset Error 41	Static reset signal in state C070. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C081	Reset Error 42	Static reset signal in state C080. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C091	Reset Error 43	Static reset signal in state C090. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

DiagCode	Status name	Status description and output setting
C010	EDM Error 11	The signal at EDM1 is invalid in the initial actuator state. In state 8010, the EDM1 signal has a value of FALSE when S_OutControl is enabled. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C020	EDM Error 12	The signal at EDM2 is invalid in the initial actuator state. In state 8010, the EDM2 signal has a value of FALSE when S_OutControl is enabled. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C030	EDM Error 13	The signals at EDM1 and EDM2 are invalid in the initial drive state. In state 8010, the EDM1 and EDM2 signals have a value of FALSE on activation. O_OutControl. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C040	EDM Error 21	The signal at EDM1 is invalid in the initial actuator state. In state 8010, the EDM1 signal has a value of FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C050	EDM Error 22	The signal at EDM2 is invalid in the initial actuator state. In state 8010, the EDM2 signal has a value of FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C060	EDM Error 23	The signals at EDM1 and EDM2 are invalid in the initial drive state. In state 8010, the EDM1 and EDM2 signals have a value of FALSE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C070	EDM Error 31	The signal at EDM1 is invalid in the actuator's switching state. In state 8000, the EDM1 signal has a value of TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C080	EDM Error 32	The signal at EDM2 is invalid in the actuator's switching state. In state 8000, the EDM2 signal has a value of TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE

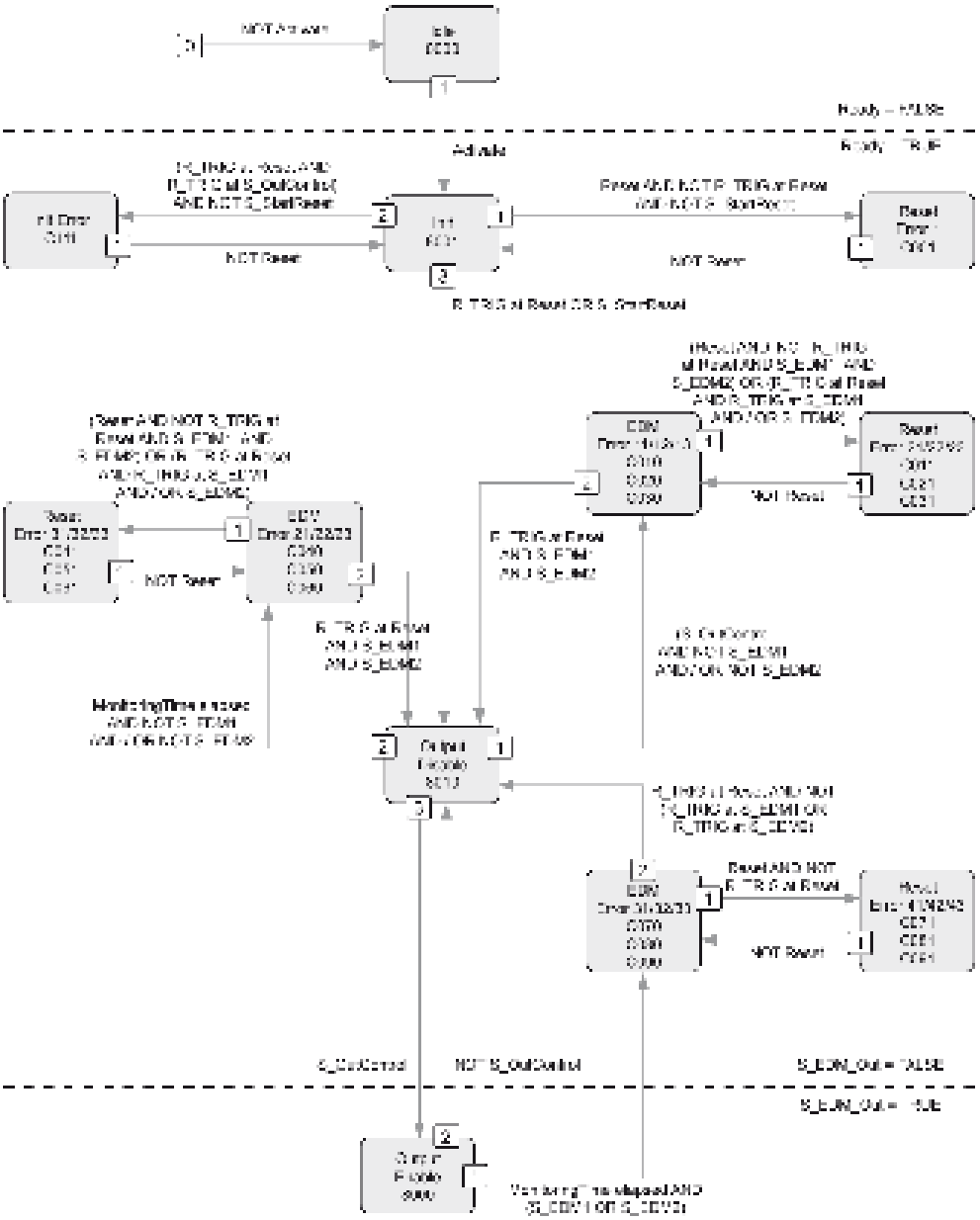
DiagCode	Status name	Status description and output setting
C090	EDM Error 33	Signals EDM1 and EDM2 are invalid in the initial actuator state. In state 8000, the EDM1 and EDM2 signals have a value of TRUE and the monitoring time has elapsed. Ready = TRUE S_EDM_Out = FALSE Error = TRUE
C111	Init Error	Similar signals were detected at S_OutControl and Reset (R_TRIG in the same cycle) (possible programming mistake) Ready = TRUE S_EDM_Out = FALSE Error = TRUE
FB-specific status codes (no error):		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EDM_Out = FALSE Error = FALSE
8001	Init	Block activation start inhibit active. Reset required. Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8010	Output	EDM control deactivation not active. Timer starts if the state is started. Ready = TRUE S_EDM_Out = FALSE Error = FALSE
8000	Output	EDM control activation active. Timer starts if the state is started. Ready = TRUE S_EDM_Out = TRUE Error = FALSE

13.5.4.3 Characteristics

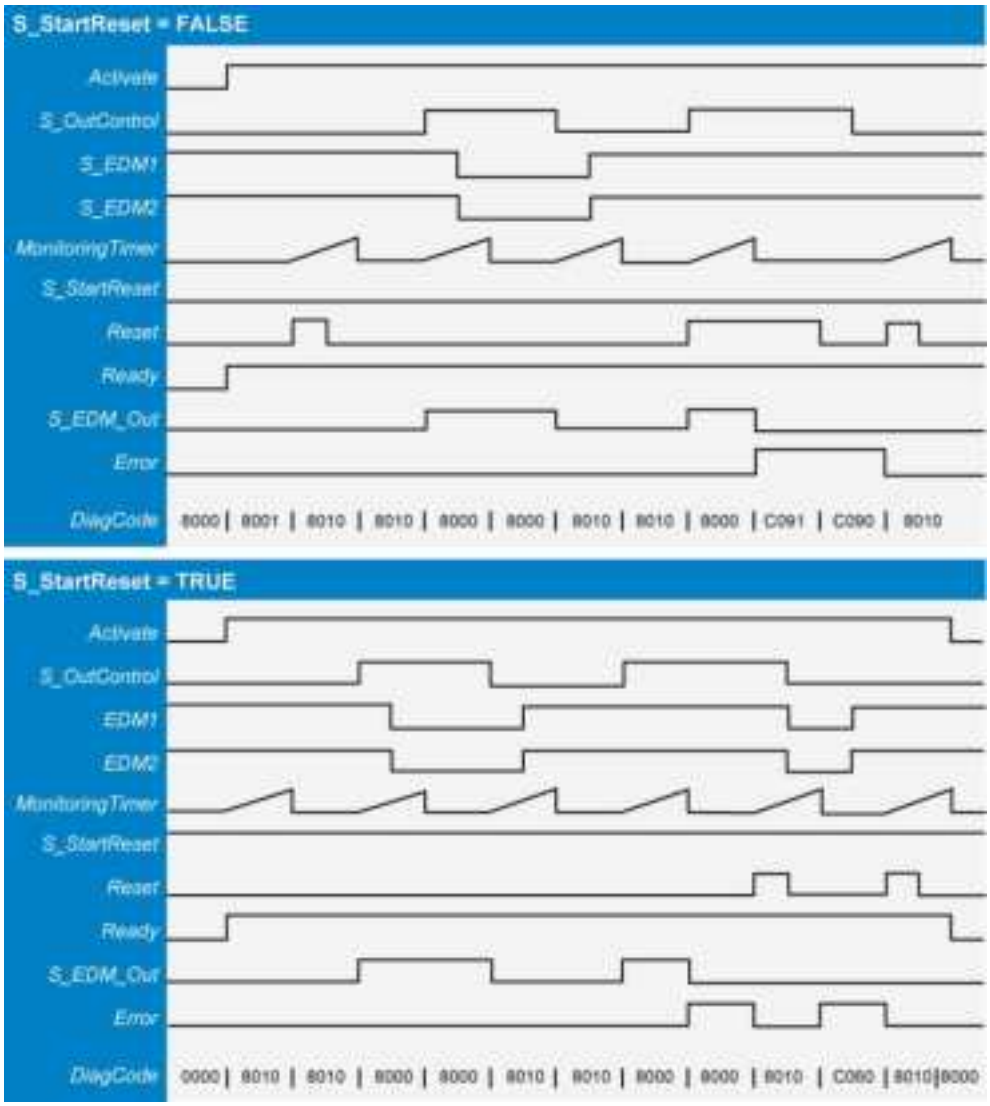
From state 8010 to states C010, C020, and C030, the conditions in the state diagram do not match the explanations for the individual states.

The condition(s) has/have been defined as follows (..... AND NOT S_EDM1 AND / OR NOT S_ED M2), just like described in the state diagram for states C040, C050, and C060.

13.5.4.4 Status Diagram SF_EDM

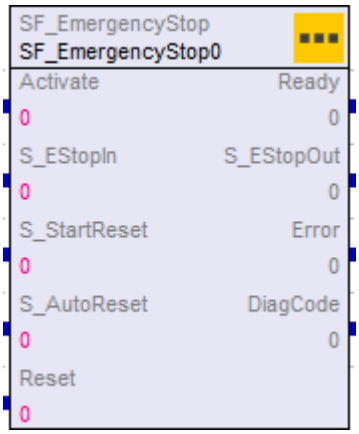


13.5.4.5 Time Diagram SF_EDM



13.5.5 SF_EmergencyStop function

This function block monitors an emergency stop button. You can use it for the uncontrolled emergency stop functionality (stop category 0) or, with additional peripheral support, for the controlled emergency stop functionality (stop category 1 or 2).



Input variables	
Activate	Activates the function block (BOOL)
S_EStopIn	Safety request input. (SAFEBOOL) FALSE: Request for a Safety response (e.g., emergency stop button is on) TRUE: No request for a Safety response (e.g., emergency stop button not on)
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL).
S_EStopOut	Safety response output. (SAFEBOOL) FALSE: Safety output deactivated. Request for a Safety response (e.g., emergency stop button pressed, reset required, or internal fault active) TRUE: Safety output activated. No request for a Safety response (e.g., emergency stop button not pressed, no internal fault active).
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

The S_EStopOut enable signal will be set back to FALSE as soon as the S_EstopIn input is set to FALSE. The S_EStopOut enable signal will be set back to TRUE if the S_EStopIn input is set to TRUE and a reset is carried out. The enable reset will depend on the defined S_StartReset, S_AutoReset, and the reset inputs.

If S_AutoReset = TRUE, resetting will be automatic.

If S_AutoReset = FALSE, there must be a rising edge at the Reset input in order to reset the activation.

If S_StartReset = TRUE, resetting will be automatic if the PLC was started for the first time.

If S_StartReset = FALSE, there must be a rising edge at the Reset input in order to reset the activation.

The S_StartReset and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PLC is started.

SF_EmergencyStop can be used to monitor both single-channel and dual-channel emergency stop buttons. For example, in the case of dual-channel applications, you can use the additional SF_Equivalent function block to detect whether contact synchronization remains in place. The corresponding category in conformity with EN ISO 13849 will depend on the final elements used.

SF_EmergencyStop will automatically detect a static TRUE signal at the Reset input. Additional error detection, such as for open wires and short-circuits, will depend on the hardware being used.

13.5.5.1 Error detection

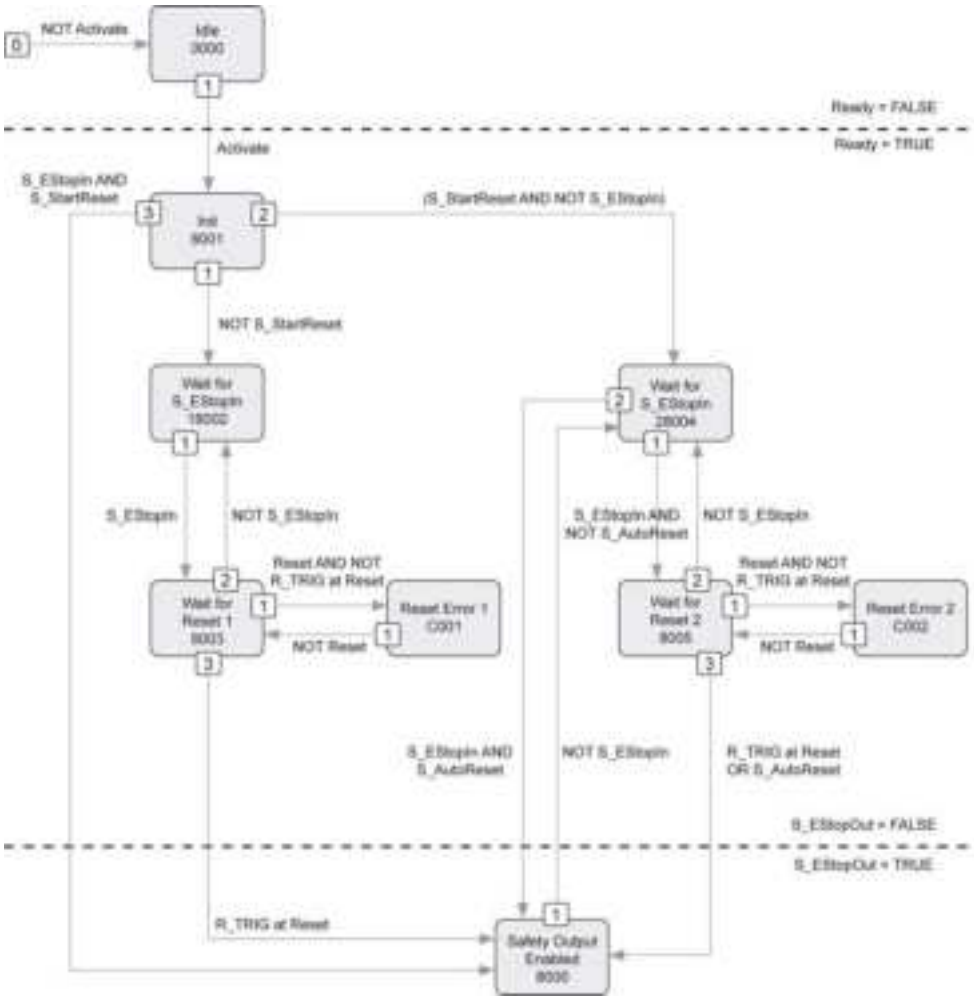
The function block will detect a static TRUE signal at the Reset input.

13.5.5.2 Error behavior

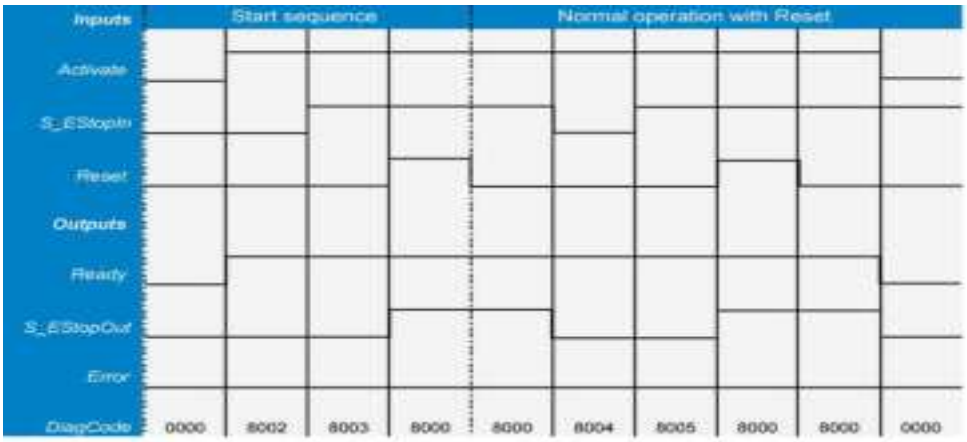
S_EStopOut will be set to FALSE if there is a static TRUE signal at the Reset input. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Reset has a value of TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE
C002	Reset Error 2	Reset has a value of TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = TRUE
FB-specific status codes (no error):		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EStopOut = FALSE Error = FALSE
8001	Init	The activation is TRUE. The function block has been activated. Check whether S_StartReset is required. S_StartReset is required. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8002	Wait for S_ EstopIn 1	The activation is TRUE. Check whether Reset is FALSE and wait until S_ EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8003	Wait for Reset 1	The activation is TRUE. S_EStopIn = TRUE. Wait for a rising edge of Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8004	Wait for S_ EstopIn 2	The activation is TRUE. Safety request detected. Check whether Reset has a value of FALSE and wait until S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8005	Wait for Reset 2	The activation is TRUE. S_EStopIn = TRUE. Check whether S_AutoReset has a value of TRUE or wait for a rising edge on Reset. Ready = TRUE S_EStopOut = FALSE Error = FALSE
8000	Safety Out- put Enabled	The activation is TRUE. S_EStopIn = TRUE. Functional mode with: S_EStopOut = TRUE. Ready = TRUE S_EStopOut = TRUE Error = FALSE

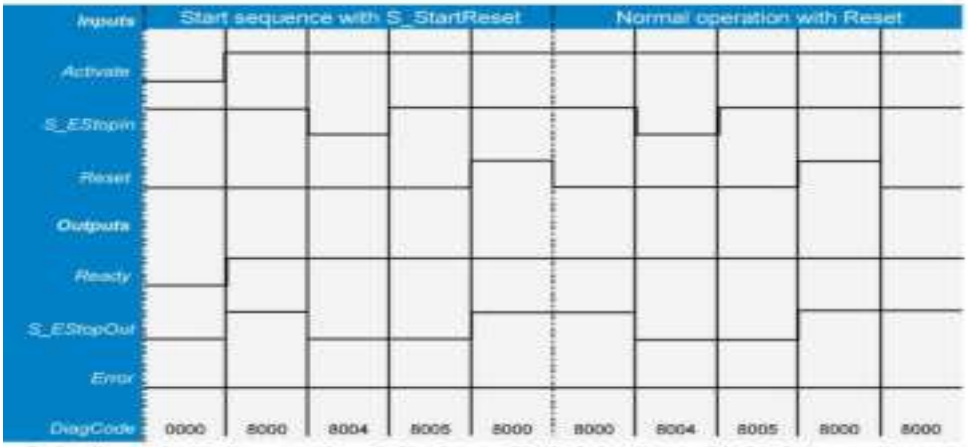
13.5.5.3 Status Diagram SF_EmergencyStop



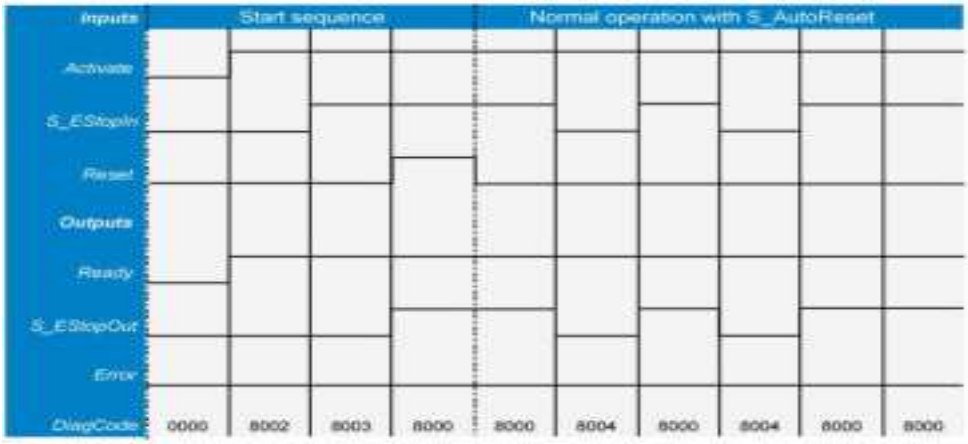
13.5.5.4 Time Diagram SF_EmergencyStop



Timing diagram for SF_EmergencyStop: S_StartReset = FALSE; S_AutoReset = FALSE; Start, reset, normal operation, safety demand, restart



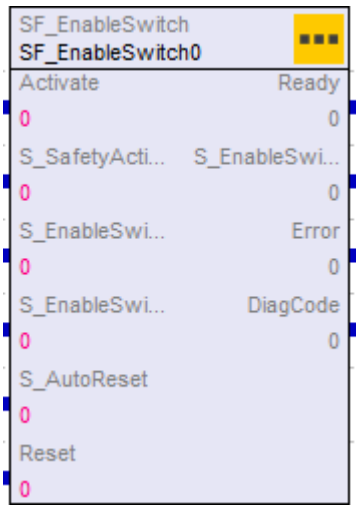
Timing diagram for SF_EmergencyStop: S_StartReset = TRUE, S_AutoReset = FALSE; Start, normal operation, safety demand, restart



Timing diagram for SF_EmergencyStop: S_StartReset = FALSE, S_AutoReset = TRUE; Start, normal operation, safety demand, restart

13.5.6 SF_EnableSwitch function

This function block evaluates the signals of an operating mode switch with three positions (e.g., OFF / Automatic / Manual).



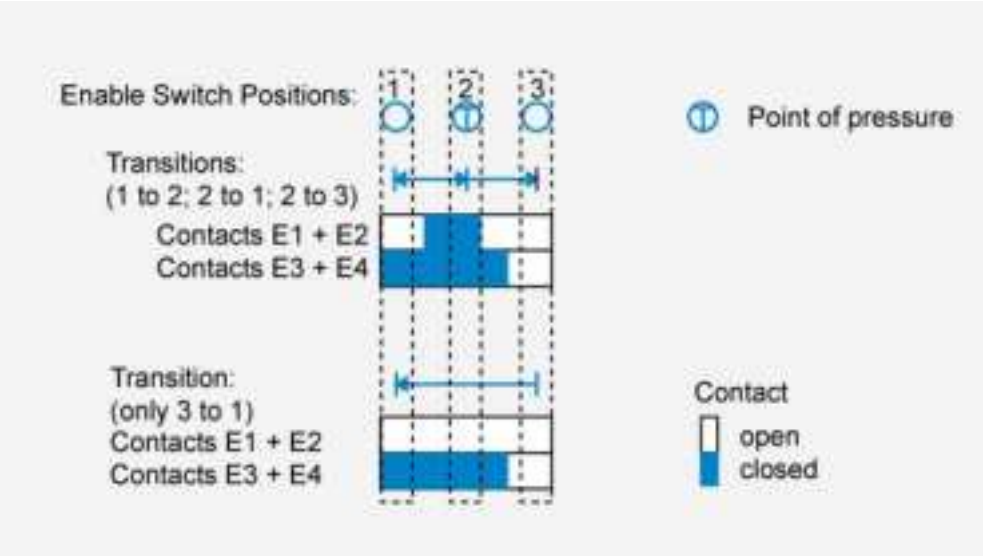
Input variables	
Activate	Activates the function block (BOOL)
S_SafetyActive	Used to confirm safe mode (limits the speed or the speed force, limits the movement range). (SAFEBOOL) FALSE: Safe mode is not enabled TRUE: Safe mode is enabled
S_ EnableSwitchCh1	Signals from contacts E1 and E2 on the connected operating mode switch (SAFEBOOL) FALSE: Connected switches are open TRUE: Connected switches are closed
S_ EnableSwitchCh2	Signals from contacts E3 and E4 on the connected operating mode switch (SAFEBOOL) FALSE: Connected switches are open TRUE: Connected switches are closed
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_ EnableSwitchOut	Safety-relevant output: Signals when monitoring is disabled. (SAFEBOOL) FALSE: Deactivates monitoring disabling TRUE: Activates monitoring disabling
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

You can use the SF_EnableSwitch function block to disable safeguarding (DIN EN 60204 section 9.2.4) implemented with an enabling switch (DIN EN 60204 section 9.2.5.8) if the corresponding operating mode is selected and active. The cor-

responding operating mode (speed or motion force limiting, freedom-of-movement limiting) must be selected outside of the SF_EnableSwitch function block.

The SF_EnableSwitch function block evaluates the signals of an enabled switch with three positions (DIN EN 60204 section 9.2.5.8).

The S_EnableSwitchCh1 and S_EnableSwitchCh2 input parameters process the contact states of E1 to E4:



The E1 and E2 signals must be connected to the S_EnableSwitchCh1 input. Meanwhile, the E3 and E4 signals must be connected to the S_EnableSwitchCh2 input. The position of the enable switch will be detected by the function block that uses this signal sequence. The transition from position 2 to 3 can be different from the one shown here.

The switching direction (position 1 => position 2/position 3 => position 2) can be detected in the function block with the specified signal sequence for the enable switching contacts. Safeguard disabling can only be activated by the function block after a movement from position 1 to position 2. Other switching directions or positions should not be used to activate safeguard disabling. This measure corresponds to the requirements in EN 60204 section 9.2.5.8.

To meet the requirements in DIN EN 60204 section 9.2.4, you will need to use an appropriate switching device. In addition, you will need to make sure that the respective operating mode (DIN EN 60204 section 9.2.3) is selected in the application (automatic operation must be disabled with suitable measures in this operating mode).

The operating mode will usually be indicated by an operating mode switch in conjunction with the SF_ModeSelector function block and the SF_SafeRequest function block or the SF_SafelyLimitedSpeed function block.

The SF_EnableSwitch function block processes the confirmation of the "safe mode" state with the "S_SafetyActive" input. When implemented in a safeguarded mode

application without a confirmation, a static TRUE signal will be connected to the "S_ SafetyActive" input.

The S_AutoReset input should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.6.1 Error detection

The following conditions will force a switch to the error state:

- Invalid static reset signal in the process.
- Invalid switch positions.

13.5.6.2 Error behavior

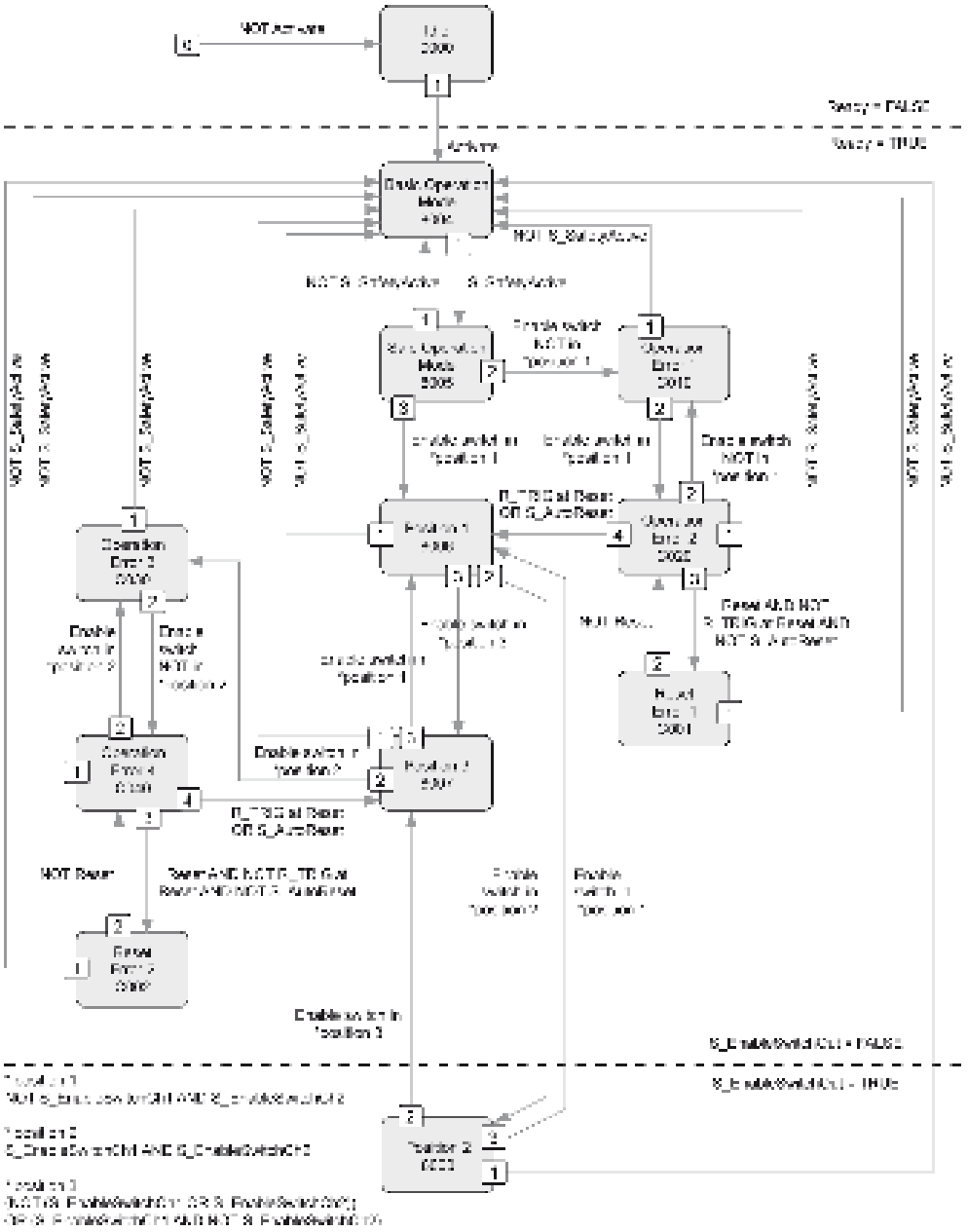
In the event of an error, the S_EnableSwitchOut safe output will be set to FALSE and remain in this safe state. Unlike with other function blocks, a reset error state resulting from the Reset = FALSE condition can be left alone, as can a reset error state if the S_SafetyActive signal has a value of FALSE.

Once the error has been fixed successfully, the switch must be switched to the starting position that was defined in the process before the enabling switch can be used to set the S_EnableSwitchOut output to TRUE. If S_AutoReset = FALSE, a rising edge will be required at the Reset input.

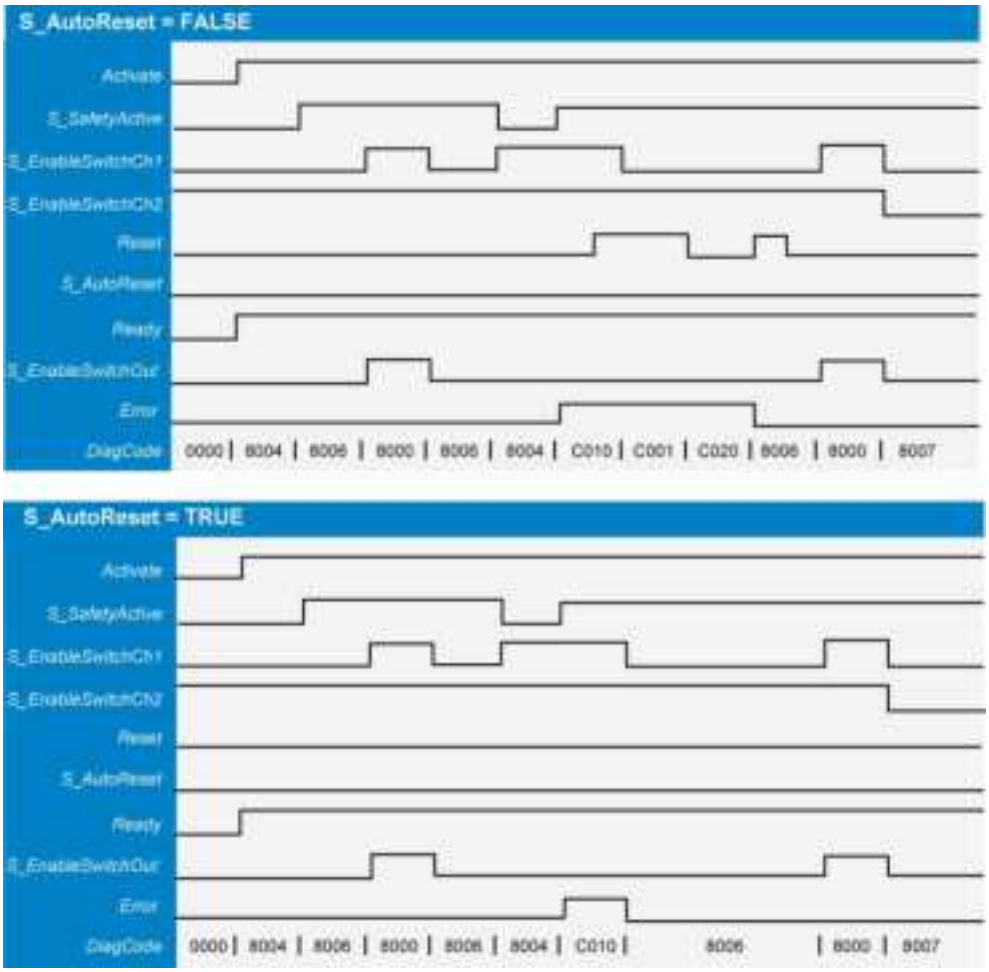
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal was detected in state C020. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state C040. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C010	Operation Error 1	Enabling switch not at position 1 when S_SafetyActive was enabled. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C020	Operation Error 2	Enabling switch at position 1 after C010. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
C030	Operation Error 3	Enabling switch at position 2 after position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE

DiagCode	Status name	Status description and output setting
C040	Operation Error 4	Enabling switch not at position 2 after C030. Ready = TRUE S_EnableSwitchOut = FALSE Error = TRUE
Function-block-specific error codes (no error):		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EnableSwitchOut = FALSE Error = FALSE
8004	Basic Operation Mode	Safe operating mode not active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8005	Safe Operation Mode	Safe operating mode active. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8006	Position 1	Safe operating mode active and enabling switch at position 1. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8007	Position 3	Safe operating mode active and enabling switch at position 3. Ready = TRUE S_EnableSwitchOut = FALSE Error = FALSE
8000	Position 2	Safe operating mode active and enabling switch at position 2. Ready = TRUE S_EnableSwitchOut = TRUE Error = FALSE

13.5.6.3 Status Diagram SF_EnableSwitch

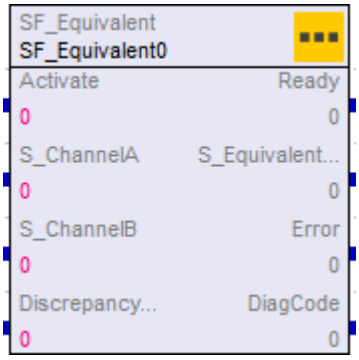


13.5.6.4 Time Diagram SF_EnableSwitch



13.5.7 SF_Equivalent function

This function block combines two equivalent SAFEBOOL inputs (both NO or NC) into a SAFEBOOL output with discrepancy time monitoring. Please note that this function block should not be used as a standalone block, since it does not have a restart inhibit function. You will need to connect the output to other safety-relevant functions instead.



Input variables	
Activate	Activates the function block (BOOL)
S_ChannelA	Input A for the logical connection. (SAFEBOOL) FALSE: Contact A open TRUE: Contact A closed
S_ChannelB	Input B for the logical connection. (SAFEBOOL) FALSE: Contact B open TRUE: Contact B closed
DiscrepancyTime	Maximum monitoring time for the discrepancy status of both inputs in ms. (DINT)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_EquivalentOut	Output of logic link (SAFEBOOL) FALSE: At least one input signal = "false" or a status change outside the monitored time. TRUE: Both input signals are "active" and a status change within the monitored time.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

If the value of a channel signal changes from TRUE to FALSE, the output will be switched off (FALSE) immediately due to safety reasons.

Discrepancy time monitoring: The discrepancy time is the maximum time for which both inputs can have different states without the function block detecting an error. Discrepancy time monitoring starts when there is a state change at one of the inputs. The function block will detect an error if the two inputs do not have the same state when the discrepancy time elapses.

The inputs must be switched symmetrically. In other words, monitoring will be carried out both for the switching-on and switching-off processes.

13.5.7.1 Error detection

On changes to TRUE and FALSE, the function block will monitor the discrepancy time between channel A and channel B.

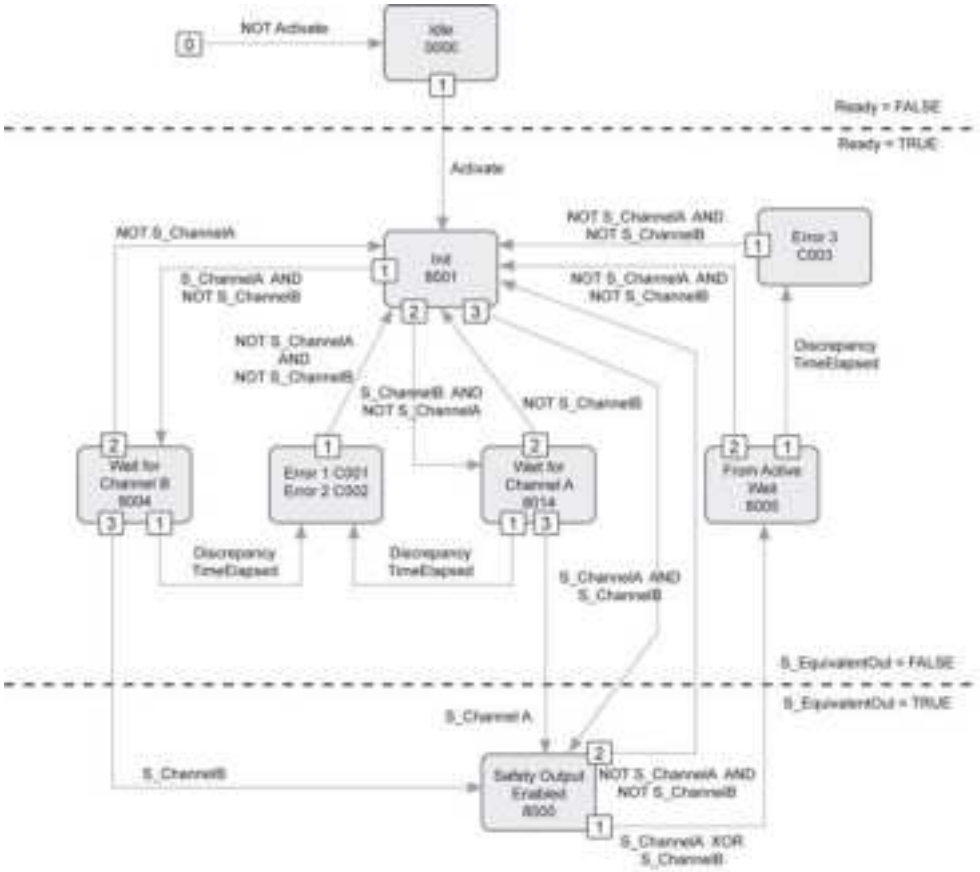
13.5.7.2 Error behavior

There is no defined Reset input that will reset an error. If an error occurs at the inputs, the new input values with correct values must be able to reset the error flag. (Example. If a faulty switch has been replaced, reusing the switch will result in correct output.)

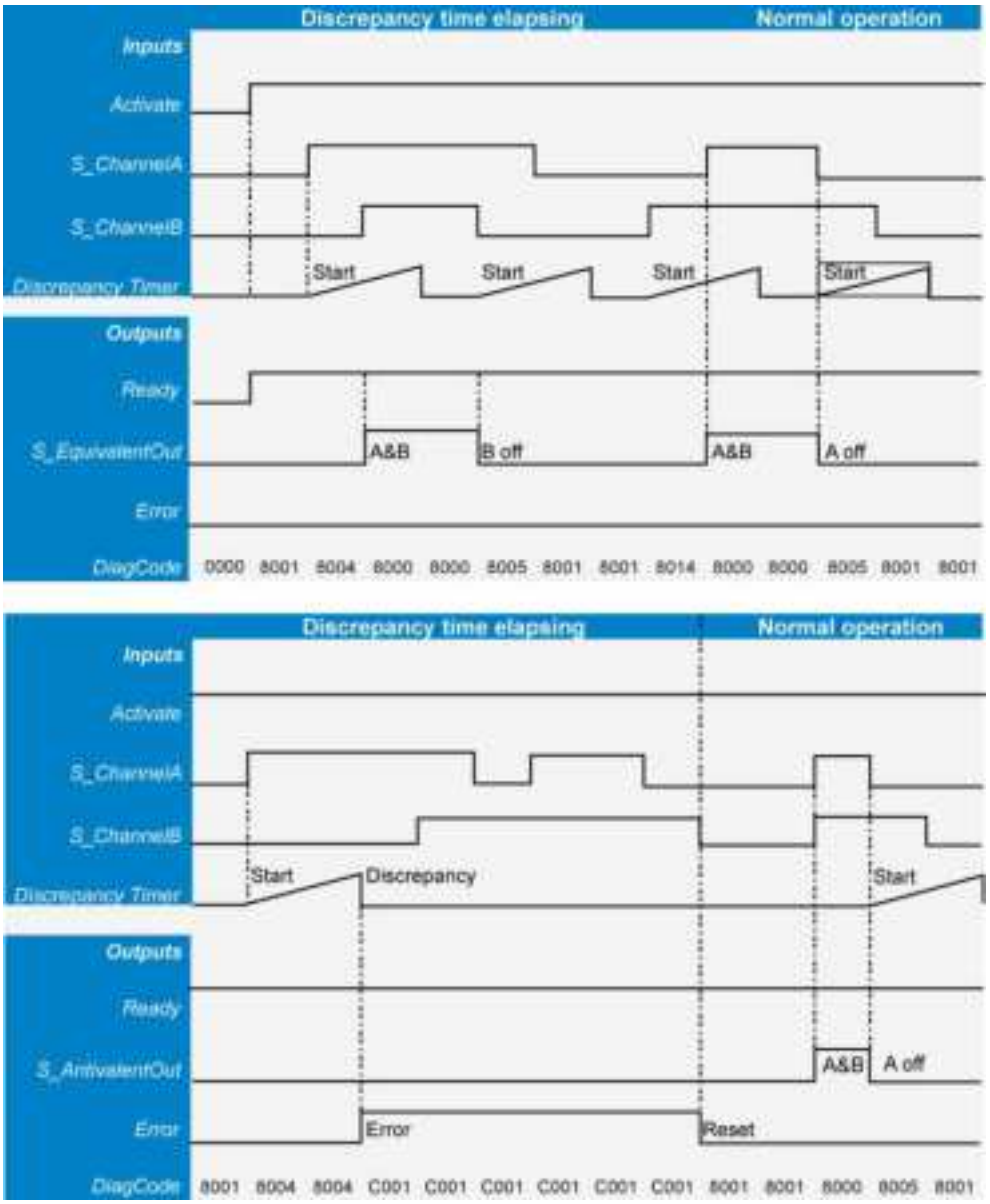
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error 1	Discrepancy time elapsed in state 8004 Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C002	Error 2	Discrepancy time elapsed in state 8014 Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
C003	Error 3	Discrepancy time elapsed in state 8005 Ready = TRUE S_EquivalentOut = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_EquivalentOut = FALSE Error = FALSE
8001	Init	An activation was detected by the function block and is now activated. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8000	Safety Output Enabled	The inputs have been set to TRUE in the corresponding mode. Ready = TRUE S_EquivalentOut = TRUE Error = FALSE
8004	Wait for Channel B	Channel A was set to TRUE and is waiting for channel B; the discrepancy timer was started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE
8014	Wait for Channel A	Channel B was set to TRUE and is waiting for channel A; the discrepancy timer was started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8005	From Active Wait	A channel was set to FALSE and is only waiting for the second channel to also be switched to FALSE; the discrepancy timer was started. Ready = TRUE S_EquivalentOut = FALSE Error = FALSE

13.5.7.3 Status Diagram SF_Equivalent

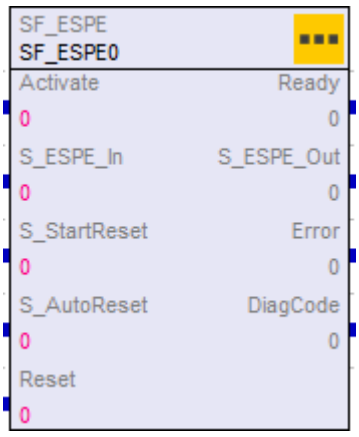


13.5.7.4 Time Diagram SF_Equivalent



13.5.8 SF_ESPE function

You can use this function block to monitor electro-sensitive protective equipment (ESPE).



Input variables	
Activate	Activates the function block (BOOL)
S_ESPE_In	Safety request input. (SAFEBOOL) FALSE: ESPE active; request for a Safety response. TRUE: ESPE not active; no request for a Safety response. The Safety system must be able to detect very brief sensor disruptions (at least 80 ms: Specified in standard 61496-1)
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released. (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_ESPE_Out	Safety response output. (SAFEBOOL) FALSE: Safety output deactivated. Request for a Safety response. TRUE: Safety output activated. No request for a Safety response.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

This function block is a safety-relevant function block for monitoring electro-sensitive protective equipment (ESPE). The function is identical to SF_EmergencyStop. The S_ESPE_Out output signal will be set to FALSE as soon as the S_ESPE_In input is set to FALSE. The S_ESPE_Out output signal will be set to TRUE if the S_ESPE_In input is set to TRUE and a reset is carried out.The enable reset will depend on the defined S_StartReset, S_AutoReset, and Reset inputs.

If S_AutoReset = TRUE, then resetting will be automatic.

If S_AutoReset = FALSE, then there must be a rising edge at the Reset input in order to reset the activation.

If S_StartReset = TRUE, resetting will be automatic if the PES is started for the first time.

If S_StartReset = FALSE, there must be a rising edge at the Reset input in order to reset the activation.

The S_StartReset and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

The SPE must be selected in relation to product standards EN IEC 61496-1, 61496-2, and 61496-3 and the required categories defined in EN ISO 13849.

13.5.8.1 Error detection

The function block will detect a static TRUE signal at the Reset input.

13.5.8.2 Error behavior

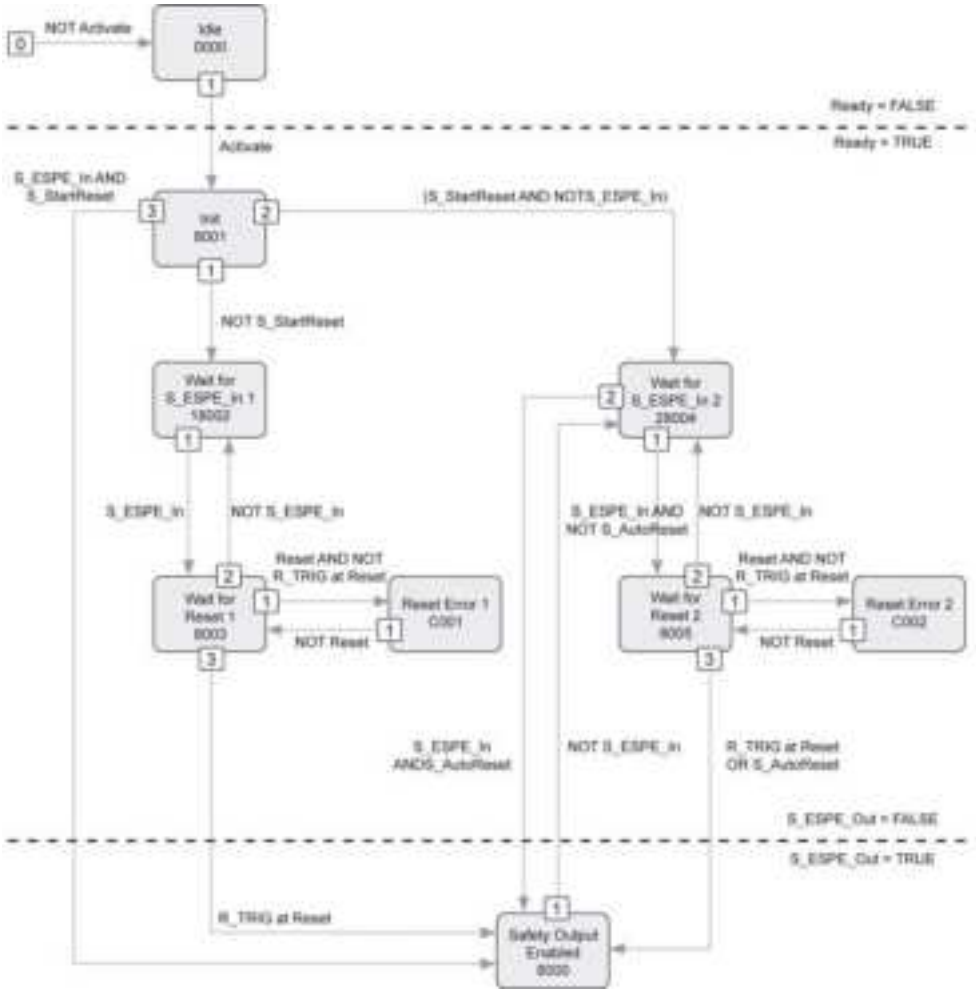
S_ESPE_Out will be set to FALSE if there is a static TRUE signal at the Reset input. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE.

In order to exit the error condition, Reset must be set to "FALSE".

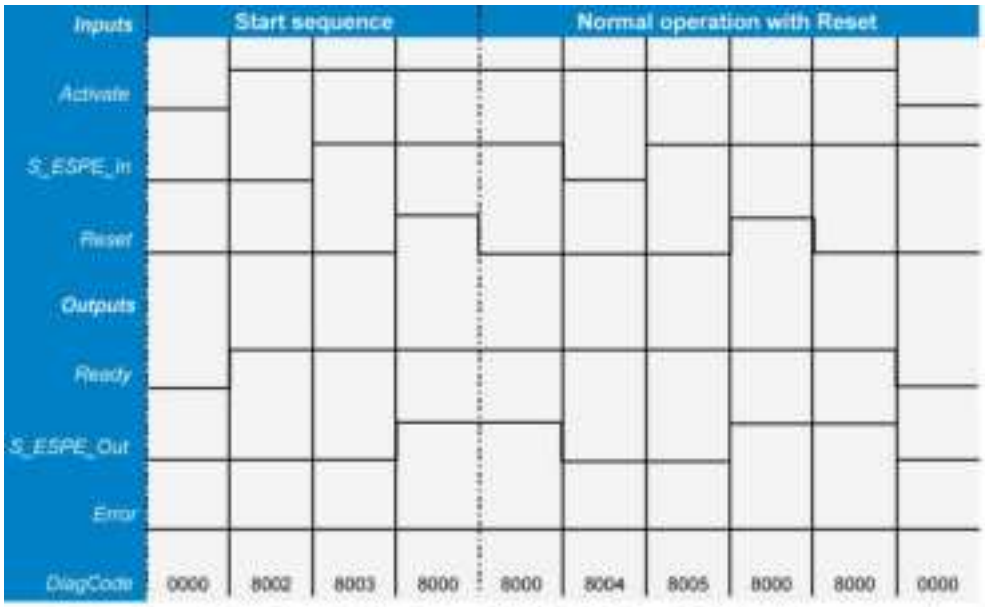
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Reset is TRUE when waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE
C002	Reset Error 2	Reset is TRUE when waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_ESPE_Out = FALSE Error = FALSE
8001	Init	The activation is TRUE. The function block has been activated. Check whether S_StartReset is required. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8002	Wait for S_ESPE_In 1	The activation is TRUE. Check whether Reset has a value of FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8003	Wait for Reset 1	The activation is TRUE. S_ESPE_In = TRUE. Wait for a rising Reset trigger. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8004	Wait for S_ESPE_In 2	The activation is TRUE. Safety request detected. Check whether Reset has a value of FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE
8005	Wait for Reset 2	The activation is TRUE. S_ESPE_In = TRUE. Check whether S_AutoReset is TRUE or wait for a rising edge at Reset. Ready = TRUE S_ESPE_Out = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8000	Safety Output Enabled	The activation is TRUE. S_ESPE_In = TRUE. Functional mode with S_ESPE_Out = TRUE Ready = TRUE S_ESPE_Out = TRUE Error = FALSE

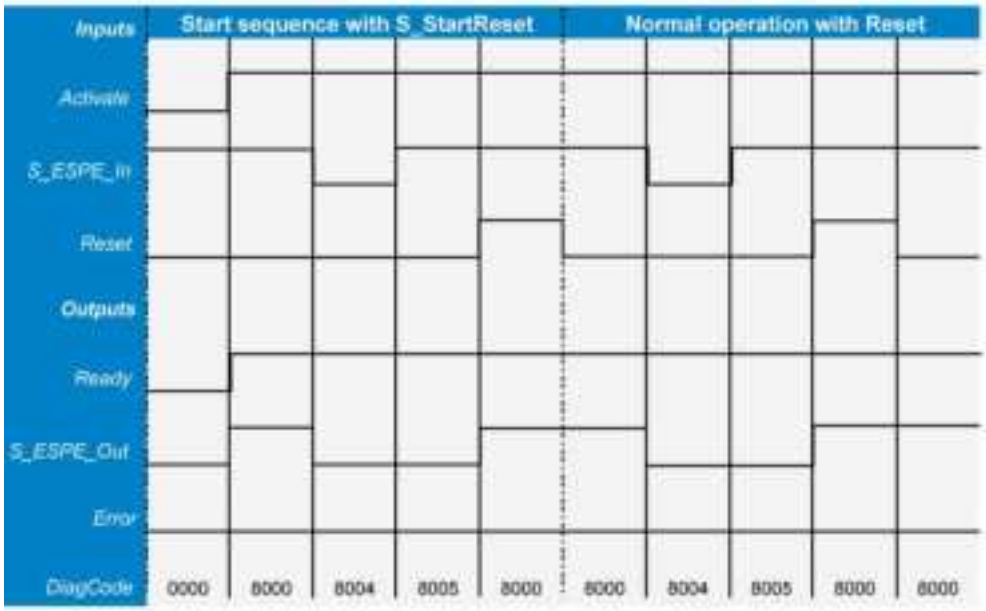
13.5.8.3 Status Diagram SF_ESPE



13.5.8.4 Time Diagram SF_ESPE



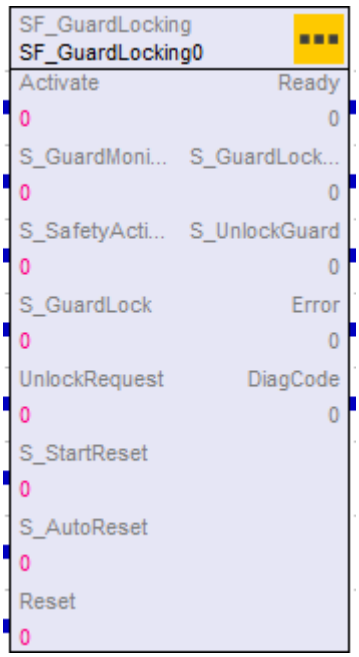
Timing diagram for SF_ESPE: S_StartReset = FALSE; S_AutoReset = FALSE; Start, reset, normal operation, safety demand, restart



Timing diagram for SF_ESPE: S_StartReset = TRUE; S_AutoReset = FALSE; Start, normal operation, safety demand, restart

13.5.9 SF_GuardLocking function

This function block controls access to a hazardous area with a protective guard that can be locked.



Input variables	
Activate	Activates the function block. (BOOL)
S_GuardMon-itoring	Monitors the protective guard. (SAFEBOOL) FALSE: Protective guard open. TRUE: Protective guard closed.
S_SafetyActive	Status of hazardous area, e.g., based on speed monitoring or safe delay monitoring. FALSE: Machine in "unsafe" state TRUE: Machine in safe state
S_GuardLock	Status of mechanical protective guard lock. FALSE: Protective guard is not locked. TRUE: Protective guard is locked.
UnlockRequest	Operator request for unlocking the guard. FALSE: No request TRUE: Request
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started. (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency button is released. (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_GuardLocked	Interface to hazardous area where people must be stopped. (SAFEBOOL)

	FALSE: No safe condition TRUE: Safe condition
S_UnlockGuard	Signal for unlocking the protective guard (SAFEBOOL) FALSE: Close protective guard TRUE: Unlock protective guard
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

The function controls the protective interlock and monitors the position of the guard and the interlock. This function block can be locked with a mechanical switch.

Users request access to the hazardous area. In this case, it will only be possible to unlock the protective device if the hazardous area is in a safe condition. The protective device can be locked if the protective door is closed.

The machine can be started if the protective door is closed and locked. If the door is open or unlocked, this will be detected in the case of a safety-critical situation.

The S_StartReset and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.9.1 Operating sequence

1. Request asking to bring a hazardous area to a safe condition – not part of this function block.
2. Feedback from the hazardous area indicating that it is in a safe condition (through S_SafetyActive).
3. Request from operator asking to unlock the protective door (through Unlock-Request).
4. Enable signal for guard that should be opened (through S_UnlockGuard).
5. Protective door unlocked (through S_GuardLock). The protective door can now be opened. (S_GuardLocked = FALSE).
6. The operator opens the protective door.
7. Protective door status monitoring through S_GuardMonitoring is used to signal when the protective door is closed again.
8. Feedback from operator for restarting the hazardous area (reset).
9. Lock Out Guard (S_UnlockGuard).
10. Check to see whether the protective door is locked (S_GuardLock).
11. The hazardous area is exited, the system can be started again (S_ GuardLocked = TRUE).
12. The system in the hazardous area is started up.

13.5.9.2 Error detection

Static signals will be detected in the event of a reset. Errors will be detected in the protective switch.

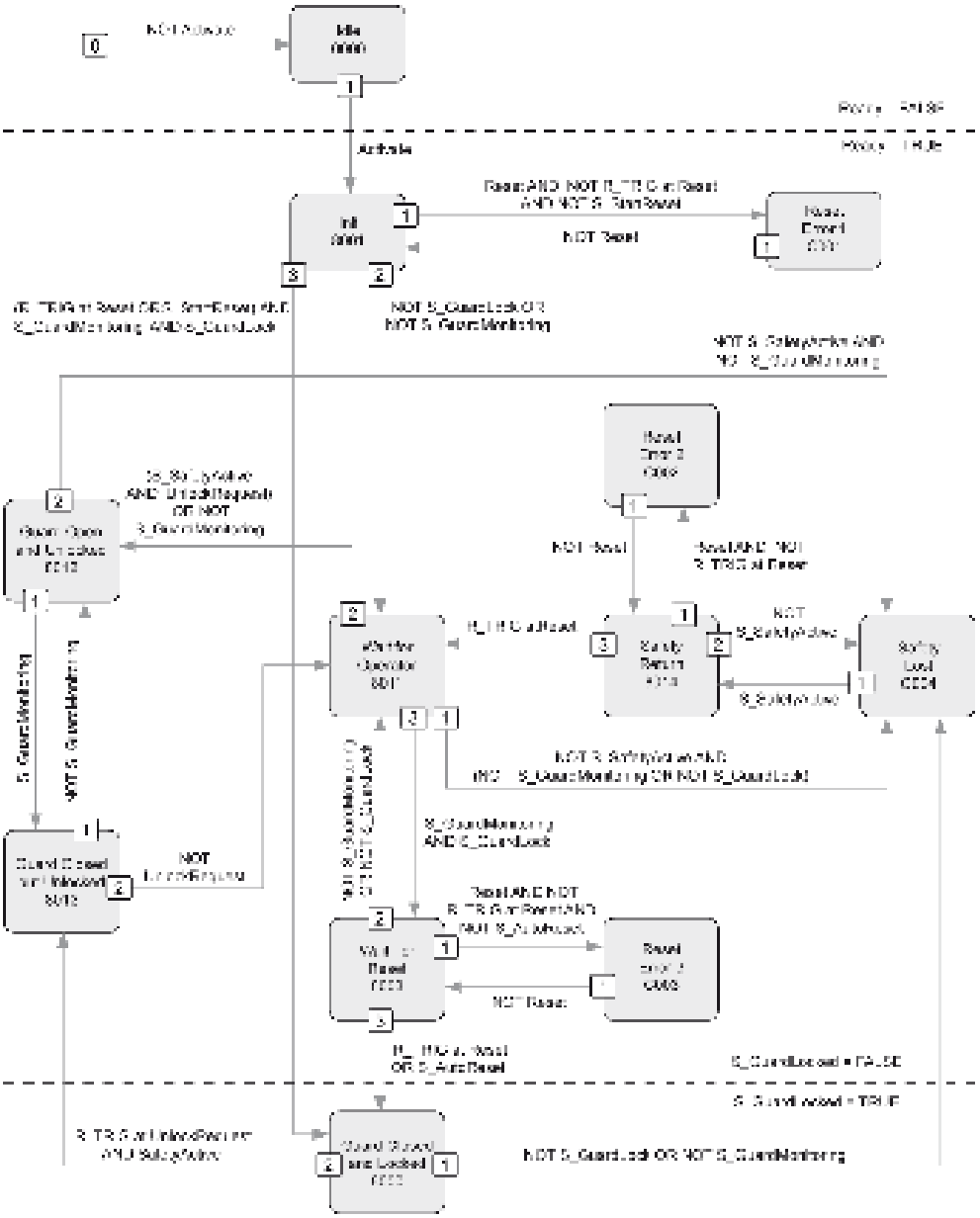
13.5.9.3 Error behavior

If there is an error, the S_GuardLocked and S_UnlockGuard-outputs will be set to FALSE. The DiagCode output will output the corresponding error code and Error will be set to TRUE. An error must be reset with a rising edge at the Reset input.

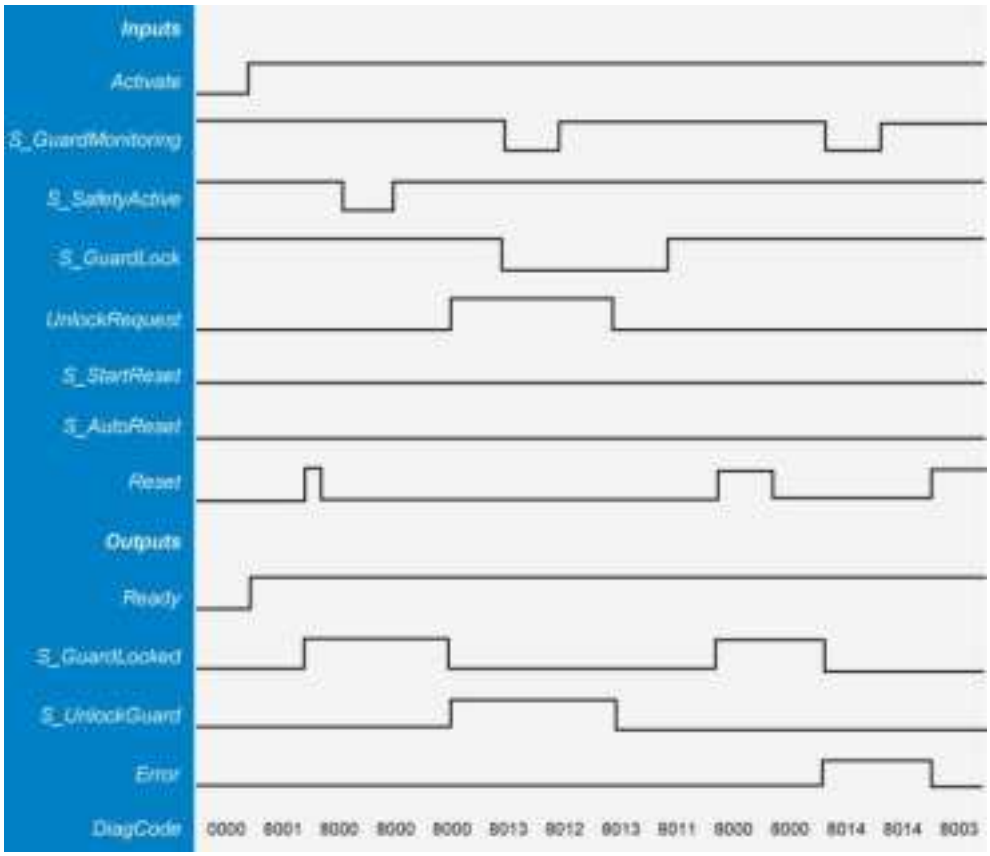
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error1	Static reset signal was detected in state 8001. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state C004. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C003	Reset Error 3	Static reset signal was detected in state 8011. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
C004	Safety Lost	Unsafe state, protective door open or unlocked. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8000	Guard Closed and Locked	Protective door is locked. Ready = TRUE S_GuardLocked = TRUE S_UnlockGuard = FALSE Error = FALSE
8001	Init	Function block has been activated and initialized. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8003	Wait for Reset	Door is closed and locked and is now waiting for an Operator Reset Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8011	Wait for Operator	Wait for an unlocking request or a reset from the operator. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE
8012	Guard Open and Unlocked	The lock is being unlocked and the protective door is open. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8013	Guard Closed but Unlocked	The lock is being unlocked, but the protective door is closed. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE Error = FALSE
8014	Safety Return	Return from S_SafetyActive signal, now waiting for actuation by operator. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE Error = FALSE

13.5.9.4 Status Diagram SF_GuardLocking

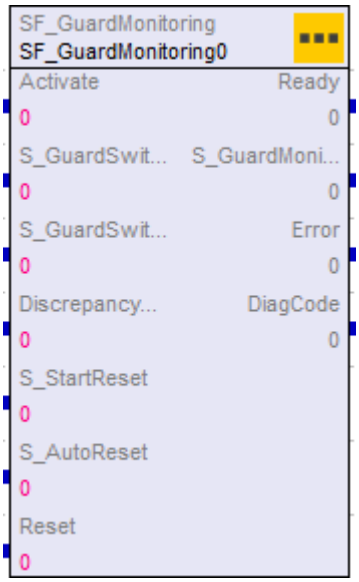


13.5.9.5 Time Diagram SF_GuardLocking



13.5.10 SF_GuardMonitoring function

This function block monitors a protective door. There are two independent inputs for the two switches at the protective door that work in conjunction with a time difference (MonitoringTime) in order to close the protective device.



Input variables	
Activate	Activates the function block (BOOL)
S_GuardSwitch1	Protective door switch input 1. (SAFEBOOL) FALSE: Protective door open. TRUE: Protective door closed.
S_GuardSwitch2	Protective door switch input 2. (SAFEBOOL) FALSE: Protective door open. TRUE: Protective door closed.
DiscrepancyTime	Configures the monitoring time between S_GuardSwitch1 and S_GuardSwitch2 (DINT)
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)
S_GuardMonitoring	This output signals the protective door status. (SAFEBOOL) FALSE: Protective door is not active. TRUE: Both S_GuardSwitches are TRUE; no error message. Protective door is active.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

The function block requires two inputs for the switches (in conformity with EN 1088) at the protective door, a discrepancy time, and a Reset input. If the protective door

only features one switch, the S_GuardSwitch1 and S_GuardSwitch2 inputs can be connected to each other. The monitoring time is the maximum time required in order to trigger both switches when the protective door is closed.The Reset, S_StartReset, and S_AutoReset inputs define how the function block will be reset after the protective door has been opened.

Both the S_GuardSwitch1 input and the S_GuardSwitch2 input should switch to FALSE when the protective door opens. The S_GuardMonitoring output will switch to FALSE as soon as one of the switches has a value of FALSE. When the protective door closes, both the S_GuardSwitch1 input and the S_GuardSwitch2 input should switch to TRUE.

This function block will monitor the symmetry of both switches' switching behavior. The S_GuardMonitoring output will remain FALSE if only one of the contacts has completed an opening / closing operation.

The behavior of the S_GuardMonitoring output will depend on the time difference between the switch inputs. The discrepancy time will be monitored as soon as the S_GuardSwitch1 and S_GuardSwitch2 inputs have different values from each other. If the discrepancy time has elapsed but the inputs continue to have different values, the S_GuardMonitoring output will remain FALSE.If the second S_GuardSwitch1/S_GuardSwitch2 input switches to TRUE within the value defined for the DiscrepancyTime, the S_GuardMonitoring output will be set to TRUE after resetting.

The S_StartReset and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.10.1 Error detection

External signals: SAFEBOOL inputs provide the error detection functionality. A mechanical assembly combines an NC and an NO in conformity with EN ISO 13849 (protective door with two switches). The discrepancy time for the delay between the two mechanical switches in conformity with EN ISO 13849 is monitored (with "application error detection," i.e., it must be viewed as being generated by the application).

An error will be detected if the time between the first S_GuardSwitch1/S_GuardSwitch2 input and the second is greater than the value of DiscrepancyTime input X. If this is the case, the Error output will be set to TRUE.

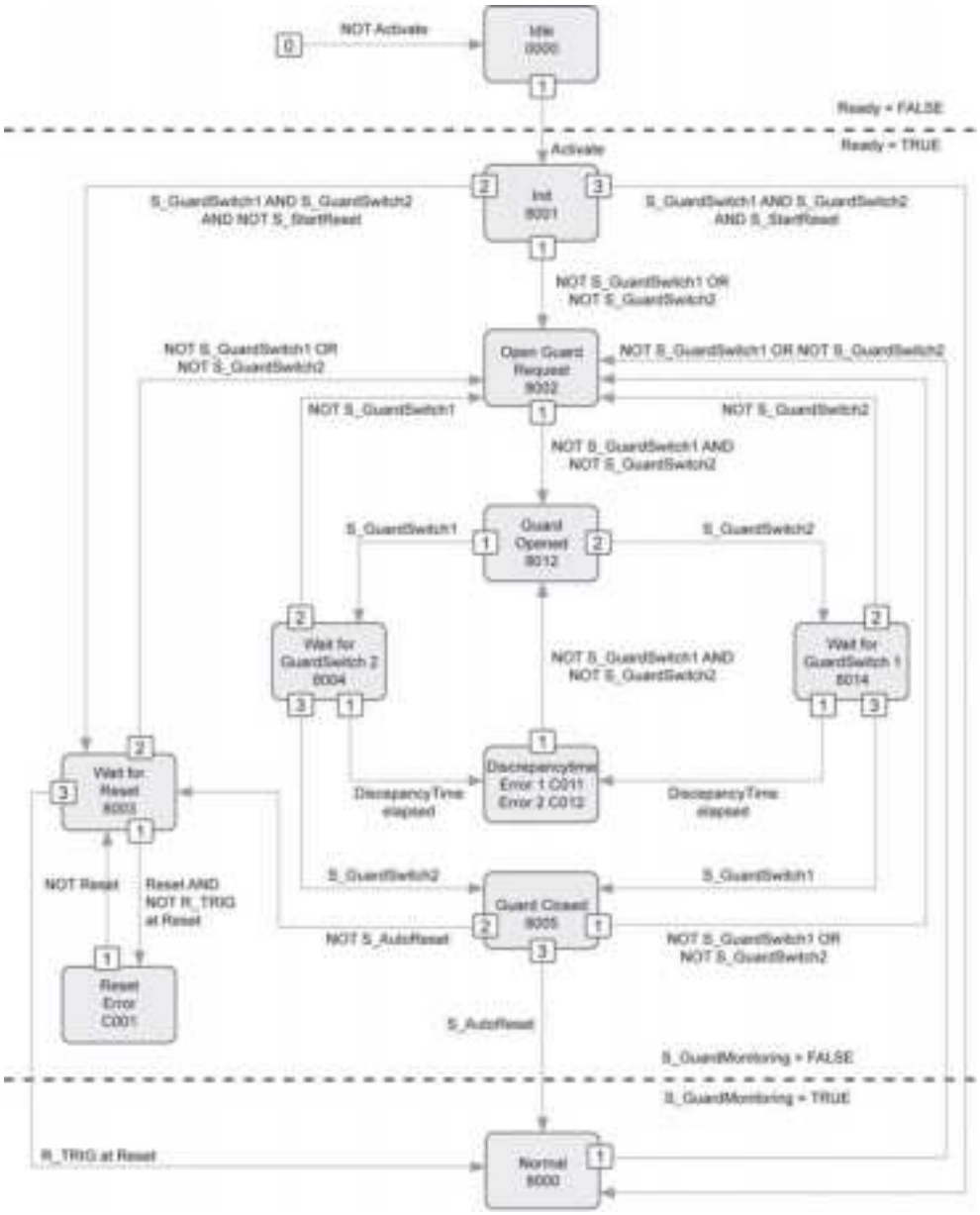
The function block will detect a static TRUE signal at the RESET input.

13.5.10.2 Error and reset behavior

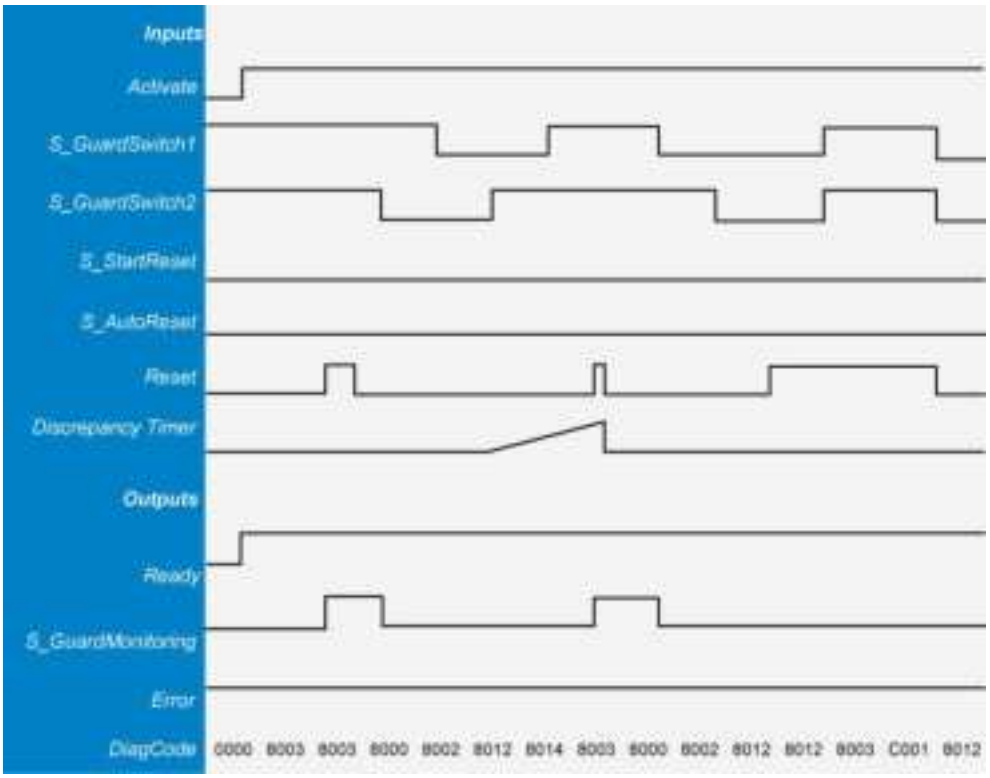
The S_GuardMonitoring output is set to FALSE. If the S_GuardSwitch1 and S_Guardswitch2 inputs are connected to each other, no error will be detected. To exit the reset error condition, the Reset input must be set to FALSE. In order to exit the discrepancy time error, inputs S_GuardSwitch1 and 2 must be set to FALSE.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error	Static reset signal was detected in state 8003. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C011	Discrepancytime Error 1	DiscrepancyTime elapsed in state 8004. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
C012	Discrepancytime Error 2	DiscrepancyTime elapsed in state 8014. Ready = TRUE S_GuardMonitoring = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_GuardMonitoring = FALSE Error = FALSE
8000	Regular	Protective door closed and safe condition detected. Ready = TRUE S_GuardMonitoring = TRUE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8002	Open Guard Request	Complete operating sequence required. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8003	Wait for Reset	Waiting for rising trigger at Reset. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8012	Guard Opened	Guard fully opened. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8004	Wait for GuardSwitch2	S_GuardSwitch1 was switched to TRUE - wait for S_ GuardSwitch2; discrepancy timer was started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8014	Wait for GuardSwitch1	S_GuardSwitch2 was switched to TRUE - wait for S_ GuardSwitch1; discrepancy timer was started. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE
8005	Guard Closed	Protective door closed. Waiting for reset if S_AutoReset = FALSE. Ready = TRUE S_GuardMonitoring = FALSE Error = FALSE

13.5.10.3 Status Diagram SF_GuardMonitoring

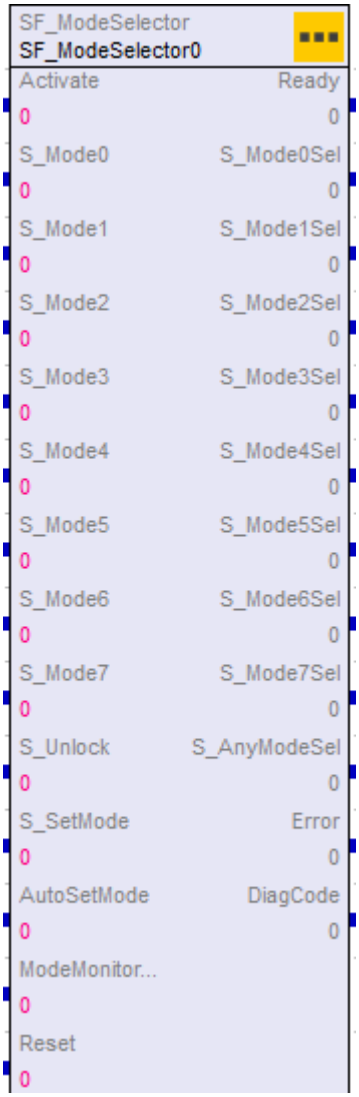


13.5.10.4 Time Diagram SF_GuardMonitoring



13.5.11 SF_ModeSelector function

This function block selects the system's operating mode, such as Manual, Automatic, Semi-Automatic, etc.



Input variables	
Activate	Activates the function block (BOOL)
S_Mode0 S_Mode1 S_Mode2 S_Mode3 S_Mode4 S_Mode5 S_Mode6 S_Mode7	Applies to all: Input X from operating mode switch FALSE: Mode X not selected. TRUE: Mode X selected by the operator.
S_Unlock	FALSE: Locks the operating mode. A change to the S_ModeX inputs will not result in a

	change at the S_ModeXSel outputs.
S_SetMode	TRUE: Confirmation for setting a new operating mode if there are valid S_ModeX inputs.
AutoSetMode	TRUE: Automatic confirmation for setting a new operating mode if there are valid S_ ModeX inputs.
ModeMonitorTime	Maximum monitoring time in which all S_ModeX inputs have a value of FALSE.
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL)
S_Mode0Sel S_Mode1Sel S_Mode2Sel S_Mode3Sel S_Mode4Sel S_Mode5Sel S_Mode6Sel S_Mode7Sel	Applies to all. TRUE: Indicates the current operating mode.
S_AnyModeSel	TRUE: An operating mode has been selected.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

This function block selects the system's operating mode (e.g., manual, automatic, semi-automatic). When the function block starts, it is necessary to make sure that machine X is in a safe state. In addition, when the machine starts, the transition to the mode set with the mode switch must be triggered with a function block input (machine START button, for example).

The default state after the function block is activated will be the ModeChanged state. This state is also the function block's safe state, in which all S_ModeXSel and S_ AnyModeSel are set to FALSE.

If the function block is in the ModeChanged state:

- The new S_ModeX input value must be confirmed with a rising edge at the S_ SetMode input, which will result in a new S_ModeXSel output value (if AutoSetMode = FALSE).
- The new S_ModeX input value will automatically result in a new S_ModeXSel output (if AutoSetMode = TRUE).
- A transition from 8005 to 8000 will only be valid if an S_ModeX input has a value of TRUE. As long as all S_ModeX have a value of FALSE, the function block will remain in state 8005, even if a rising edge is triggered at S_SetMode.

The transition from the ModeChanged state to the ModeSelected state that must be confirmed with S_SetMode by the operator is not monitored with a timer.

If the function block is in the ModeSelected state and a new S_ModeX input value (higher priority) and the NOT S_Unlock signal (lower priority) occur at the same time, this will result in the ModeChanged state.

The S_ModeX input parameters that are not used to select the module should be connected to a constant FALSE value in order to simplify the program check.

The AutoSetMode input should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.11.1 Error detection

The function block will detect if none of the input modes are selected. This invalid condition will be detected after the elapsing of ModeMonitorTime:

- That restarts with every falling edge at an S_ModeX switch mode input
- That is then brought to the ModeChanged state after the function block is activated

In contrast, the function block will detect whether more than one S_ModeX input is selected at the same time.

A static reset state will be detected if the function block is either in the C001 or C002 error state.

13.5.11.2 Error behavior

In the event of an error, the S_ModeXSel and S_AnyModeSel outputs will be set to their safe state of FALSE. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE.

Errors must be reset with a rising edge at the Reset input. The function block will switch from its error condition to the ModeChanged state.

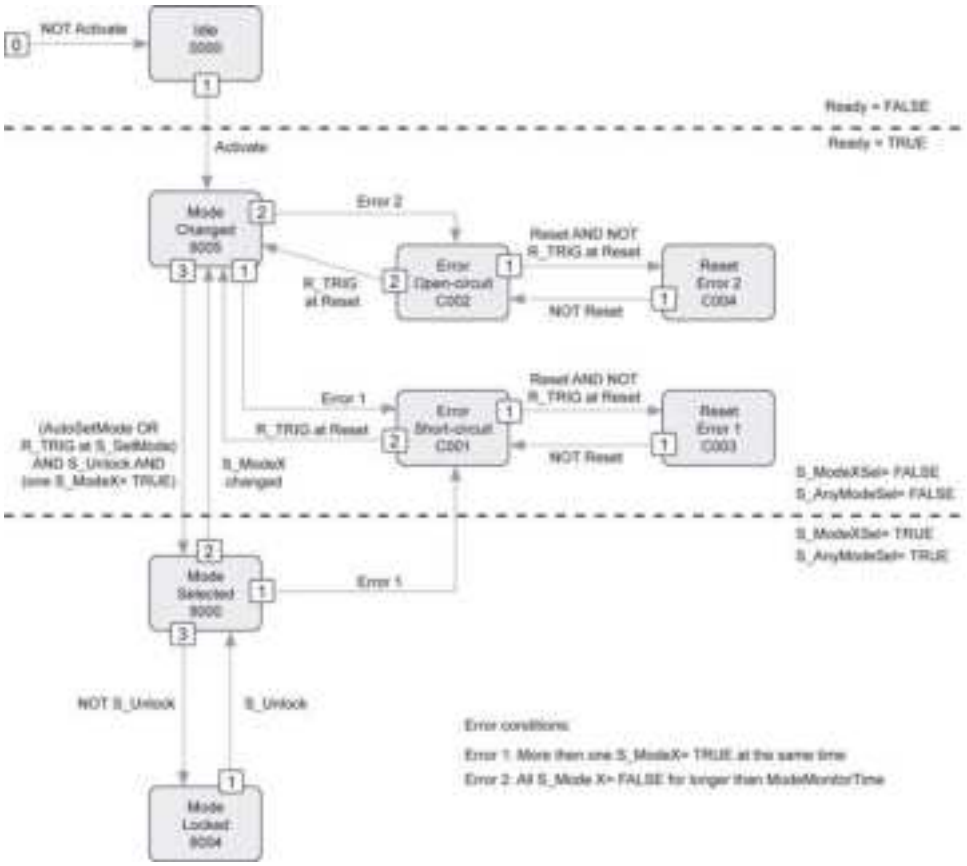
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error Short-circuit	The function block detected that two or more S_ModeX inputs have a value of TRUE (e.g., due to a cable short-circuit). Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C002	Error Open-circuit	The function block detected that all S_ModeX inputs have a value of FALSE: The time after a falling S_ModeX edge exceeds the ModeMonitorTime, e.g., due to an open wire. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C003	Reset Error 1	Static reset signal was detected in state C001. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE

DiagCode	Status name	Status description and output setting
C004	Reset Error 2	Static reset signal was detected in state C002. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8005	ModeChanged	State after activation or if S_ModeX has been changed (if not locked) or after an error condition is reset. Ready = TRUE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8000	ModeSelected	Valid mode selection, not locked yet. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, the others are FALSE.
8004	ModeLocked	Valid mode selection is locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = Selected X is TRUE, the others are FALSE.

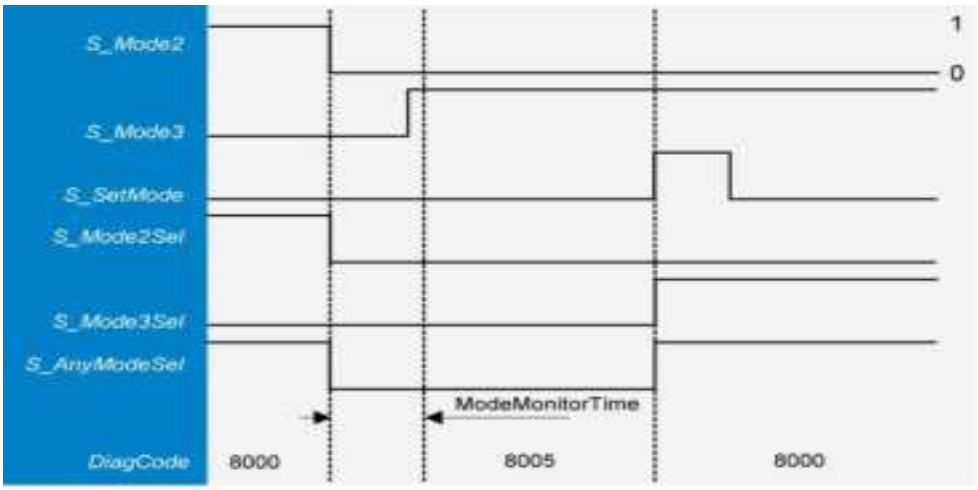
13.5.11.3 Characteristics

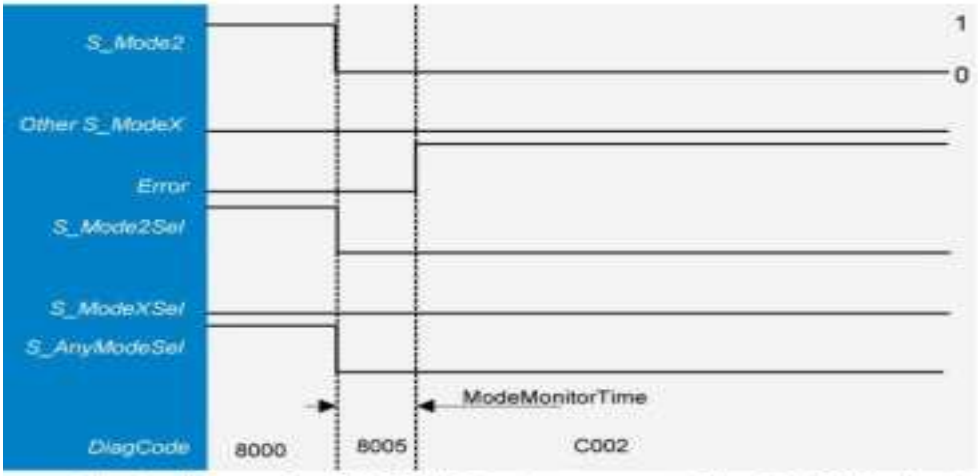
The monitoring time in state 8005 will not be started until after all S_ModeX inputs have a value of SAFE_FALSE and an input was set in the previous pass.

13.5.11.4 Status Diagram SF_ModeSelector

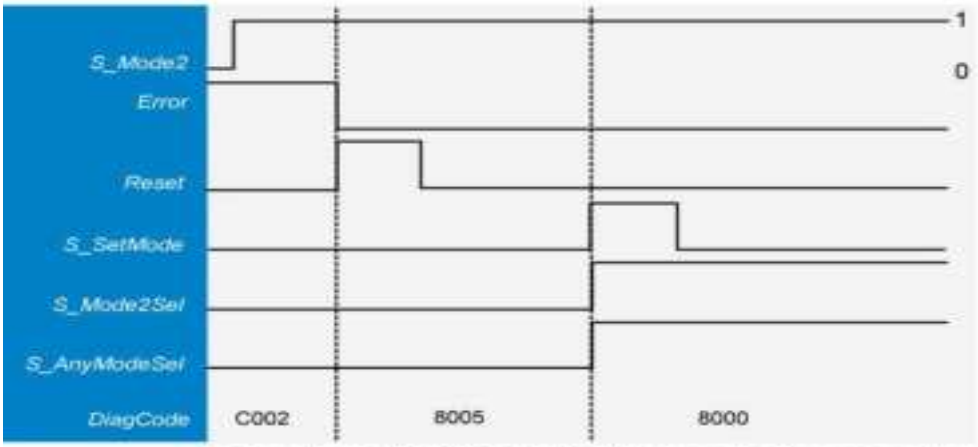


13.5.11.5 Time Diagram SF_ModeSelector





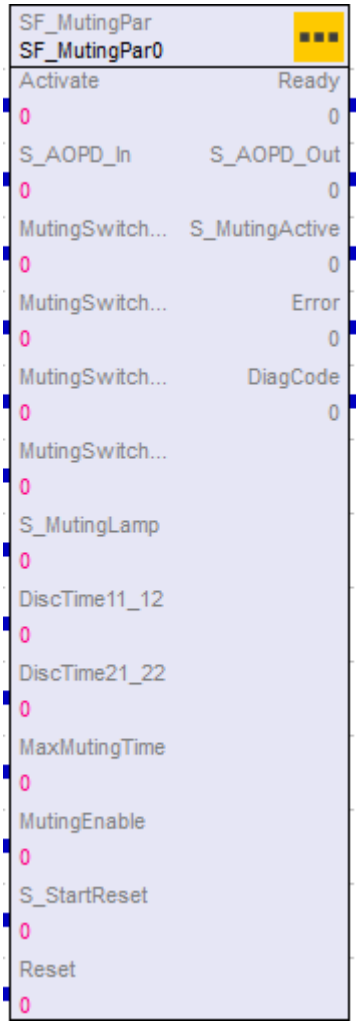
Timing diagram for SF_ModeSelector, error condition 2 at Mode inputs



Timing diagram for SF_ModeSelector, reset of error condition

13.5.12 SF_MutingPar function

"Muting" refers to intentionally suppressing safety functions. You can use this function block to implement parallel muting with four muting sensors.



Input variables	
Activate	Activates the function block (BOOL).
S_AOPD_In	OSSD signal (Output Signal Switching Device) of AOPD (Active Opto-electronic Protective Device). FALSE: Protected area disrupted. TRUE: Protected area not disrupted.
MutingSwitch11 MutingSwitch12 MutingSwitch21 MutingSwitch22	Muting sensor status. Applies to all: FALSE: Muting sensor not activated. TRUE: Workpiece activating the muting sensor. Please note that a SAFEBOOL will need to be connected instead of a BOOL depending on the safety requirements in question. (Safebool to Bool FB)
S_MutingLamp	Indicates the muting lamp's operating state. FALSE: Faulty muting lamp.

	TRUE: No muting lamp faults
DiscTime11_12	Constant, 0–4 s; maximum discrepancy time for MutingSwitch11 and MutingSwitch12.
DiscTime21_22	Constant, 0–4 s; maximum discrepancy time for MutingSwitch21 and MutingSwitch22.
MaxMutingTime	Constant, 0–10 s; maximum time for a full muting sequence; the timer will be started when the first muting sensor is activated.
MutingEnable	Command of the control system that activates the start of the muting function when the machine cycle is required. Once the muting function has started, the signal can be switched off. FALSE: Muting not activated TRUE: Start of the muting function activated.
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL).
S_AOPD_Out	The safety-relevant output indicates the protective door's state. FALSE: AOPD protected area is disrupted and muting is not active. TRUE: AOPD protected area is not disrupted or muting is active.
S_MutingActive	FALSE: Muting not active. TRUE: Muting active.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

"Muting" refers to intentionally suppressing safety functions. This can be necessary, for example, in order to transport material to a danger zone without stopping the corresponding machine. This muting is monitored by muting sensors, and the use of two or four muting sensors and their correct integration into the production sequence must ensure that no one enters the danger zone while the light curtain is off.Muting sensors include proximity switches, light barriers, limit switches, etc. that do not have to be failsafe devices.

When muting mode is active, indicator lights must signal this.

There are sequential and parallel muting methods. This function block implements parallel muting with four muting sensors. The function block can be used for muting in both directions – forward and backward. Muting must be activated by the process control system with the MutingEnable signal so as to prevent tampering.

The function block inputs include the signals for the four muting sensors (MutingSwitch11–MutingSwitch22), the OSSD signal from the "active opto-electronic protective device," S_AOPD_In, and three configurable times (DiscTime11_12, DiscTime21_22, and MaxMutingTime).

The S_StartReset input should only be enabled if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.12.1 Error detection

The function block will detect the following error conditions:

- DiscTime11_12 and DiscTime21_22 have been set to values less than T#0 s or greater than T#4 s.
- MaxMutingTime has been set to a value that is less than T#0s or greater than T#10 min.
- The discrepancy time for the MutingSwitch11/MutingSwitch12 or MutingSwitch21/MutingSwitch22 sensor pair has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 have been activated in the wrong order.
- The muting sequence starts without having been activated by MutingEnable.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static reset signal is in state 8001 or 8003.

13.5.12.2 Error behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs will be set to FALSE. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. Restarting will be prevented until the error conditions are fixed and the operator has reset the safe state with Reset.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal was detected after function block activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting Lamp	Fault identified in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

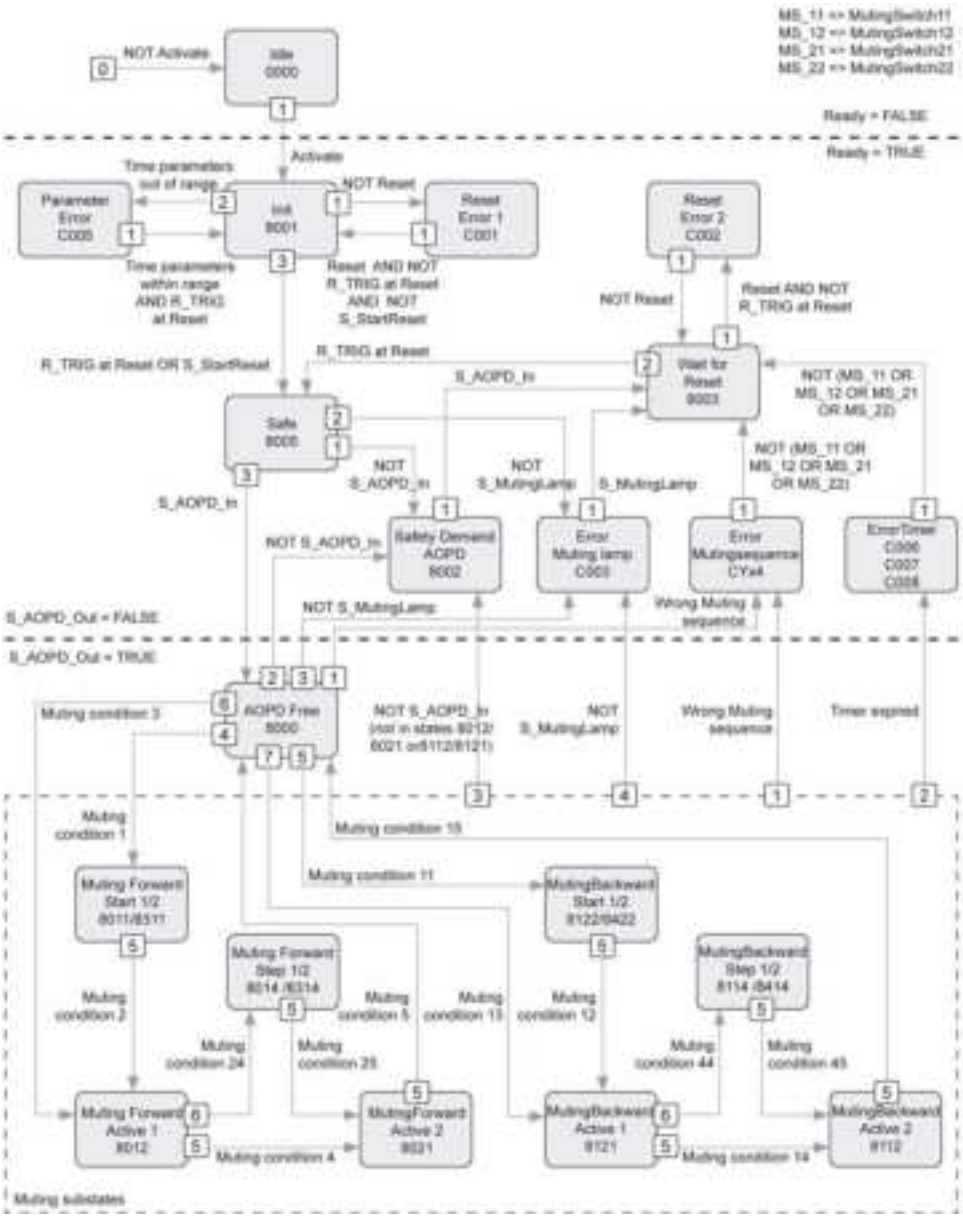
DiagCode	Status name	Status description and output setting
CYx4	Error Muting sequence	Incorrect muting sequence in status 8000, 8011, 8311, 8012, 8021, 8014, 8314, 8122, 8422, 8121, 8112, 8114 oder 8414. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (six states for forward and six states for backwards). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in forward state 8011 C2x4 = Error occurred in forward state 8311 C3x4 = Error occurred in forward state 8012 C4x4 = Error occurred in forward state 8014 C5x4 = Error occurred in forward state 8314 C6x4 = Error occurred in forward state 8021 C7x4 = Error occurred in backward state 8012 C8x4 = Error occurred in backward state 8422 C9x4 = Error occurred in backward state 8121 CAx4 = Error occurred in backward state 8114 CBx4 = Error occurred in backward state 8414 CCx4 = Error occurred in backward state 8112 CFx4 = Muting Enable missing x = Status of the sensors when the error occurs. (4 Bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).
C005	Parameter Error	The value of DiscTime11_12, DiscTime21_22, or MaxMutingTime falls outside the permissible range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error Timer MaxMuting	Timing error: Active muting time (if S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C007	Error Timer MS11_12	Timing-Fehler: Discrepancy time of MutingSwitch11 and MutingSwitch12> > DiscTime11_12. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C008	Error Timer MS21_22	Timing-Fehler: Discrepancy time of MutingSwitch21 and MutingSwitch22> > DiscTime21_22. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
FB-specific status codes (no error)		

DiagCode	Status name	Status description and output setting
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety request from AOPD. The timers will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety request from AOPD was detected; muting is not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety requests or errors were detected and are not being cleared. Reset from operator required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start 1	Muting forward sequence is in starting phase after a rising edge at MutingSwitch11. DiscTime11_12 monitoring is activated. MaxMutingTime monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8311	Muting Forward Start 2	Muting forward sequence is in starting phase after a rising edge at MutingSwitch12. DiscTime11_12 monitoring is activated. MaxMutingTime monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

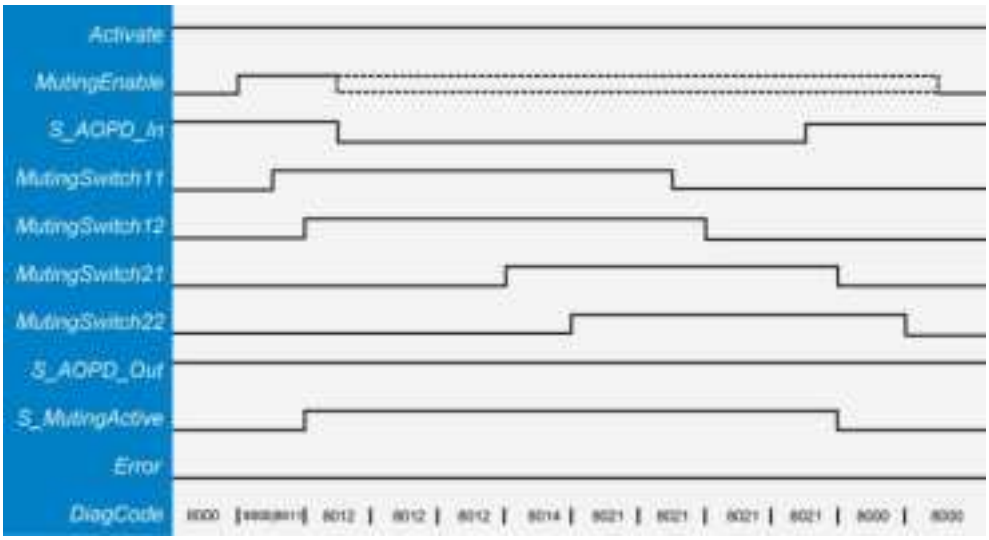
DiagCode	Status name	Status description and output setting
8012	Muting Forward Active 1	Muting forward sequence is active: - After a rising edge at the second MutingSwitch (12 or 11). - If both MutingSwitch 11 and 12 have been actuated in the same cycle. DiscTime11_12 monitoring has been stopped. MaxMuting time monitoring will be activated if the transition is directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8014	Muting Forward Step 1	Muting forward sequence is active. MutingSwitch21 is the first actuated output switch. DiscTime21_22 monitoring will be started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8314	Muting Forward Step 2	Muting forward sequence is active. MutingSwitch22 is the first actuated output switch. DiscTime21_22 monitoring will be started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8021	Muting Forward Active 2	Muting forward sequence is still active. MutingSwitch21 and 22 are actuated; DiscTime21_22 monitoring will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start 1	Muting backward sequence is in starting phase after a rising edge at MutingSwitch21. DiscTime21_22 monitoring is activated. MaxMutingTime monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8422	Muting Backward Start 2	Muting backward sequence is in starting phase after a rising edge at MutingSwitch22. DiscTime21_22 monitoring is activated. MaxMutingTime monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8121	Muting Backward Active 1	Muting backward sequence is active: - After a rising edge at the second MutingSwitch (21 or 22). - If MutingSwitch 21 and 22 have been actuated in the same cycle. DiscTime21_22 monitoring will be stopped. MaxMuting-Time monitoring will be activated if the transition is directly from state 8000. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8114	Muting Backward Step 1	Muting backward sequence is active. MutingSwitch11 is the first actuated output switch. DiscTime11_12 monitoring will be started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8414	Muting Backward Step 2	Muting backward sequence is active. MutingSwitch12 is the first actuated output switch. DiscTime11_12 monitoring will be started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8112	Muting Backward Active 2	Muting backward sequence is still active. The exit switches MutingSwitch11 and 12 are actuated; DiscTime11_12 monitoring will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

13.5.12.3 Status Diagram SF_MutingPar

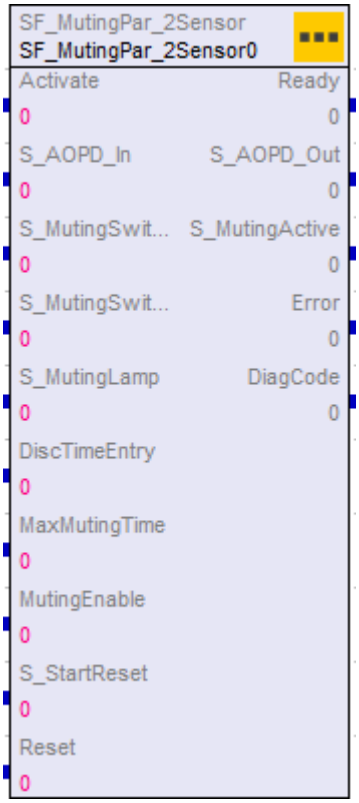


13.5.12.4 Time Diagram SF_MutingPar



13.5.13 SF_MutingPar_2Sensor function

"Muting" refers to intentionally suppressing safety functions. You can use this function block to implement parallel muting with two muting sensors.



Input variables	
Activate	Activates the function block (BOOL)
S_AOPD_In	OSSD signal (Output Signal Switching Device) of AOPD (Active Opto-electronic Protective Device). FALSE: Protected area disrupted. TRUE: Protected area not disrupted.
S_MutingSwitch11 S_MutingSwitch12	Muting sensor status. Applies to all: FALSE: Muting sensor not activated. TRUE: Workpiece activating the muting sensor.
S_MutingLamp	Indicates the muting lamp's operating state. FALSE: Faulty muting lamp. TRUE: No muting lamp fault
DiscTimeEntry	Constant 0-4 s; Max. discrepancy time for activating S_MutingSwitch11 and S_MutingSwitch12.
MaxMutingTime	Constant, 0–10 min; maximum time for a full muting sequence; the timer will be started when the first muting sensor is activated.
MutingEnable	Command from the control system that activates the start of the muting function of the machine cycle if required. Once the muting function has started, the signal can be switched off. FALSE: Muting not activated, TRUE: Start of the muting function activated.

S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU has been started (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the outputs are valid. (BOOL)
S_AOPD_Out	Safety-relevant output; indicates the status of the muting protective door. FALSE: AOPD protected area disrupted and muting not active. TRUE: AOPD protected area not disrupted or muting is active.
S_MutingActive	Indicates the status of the muting process. FALSE: Muting not active. TRUE: Muting active.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

"Muting" refers to intentionally suppressing safety functions. This can be necessary, for example, in order to transport material to a danger zone without stopping the corresponding machine. This muting is monitored by muting sensors, and the use of two or four muting sensors and their correct integration into the production sequence must ensure that no one enters the danger zone while the light curtain is off.

Muting sensors include pushbuttons, proximity switches, light barriers, limit switches, etc. that do not have to be failsafe devices.

When muting mode is active, indicator lights must signal this.

There are sequential and parallel muting methods. This particular function block implements parallel muting with two muting sensors.

The function block can be used for muting in both directions – forward and backward. However, please note that the actual direction cannot be identified.

Muting must be activated by the process control system with the MutingEnable signal so as to prevent tampering.

The function block inputs include the signals from both muting sensors (S_MutingSwitch11 and S_MutingSwitch12), the OSDD signal from the "active opto-electronic protective device," S_AOPD_In, and two configurable times (Disc-TimeEntry and MaxMutingTime).

The S_StartReset input should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.13.1 Error detection

The function block will detect the following error conditions:

- DiscTimeEntry has been set to a value less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value that is less than T#0 s or greater than T#10 min.
- The discrepancy time for the S_MutingSwitch11/S_MutingSwitch12 sensor pair was exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors S_MutingSwitch11, S_MutingSwitch12 were activated in the wrong order.
- Muting sequence starts without having been activated by MutingEnable.
- Static muting sensor signals.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static reset signal is in state 8001 or 8003.

13.5.13.2 Error behavior

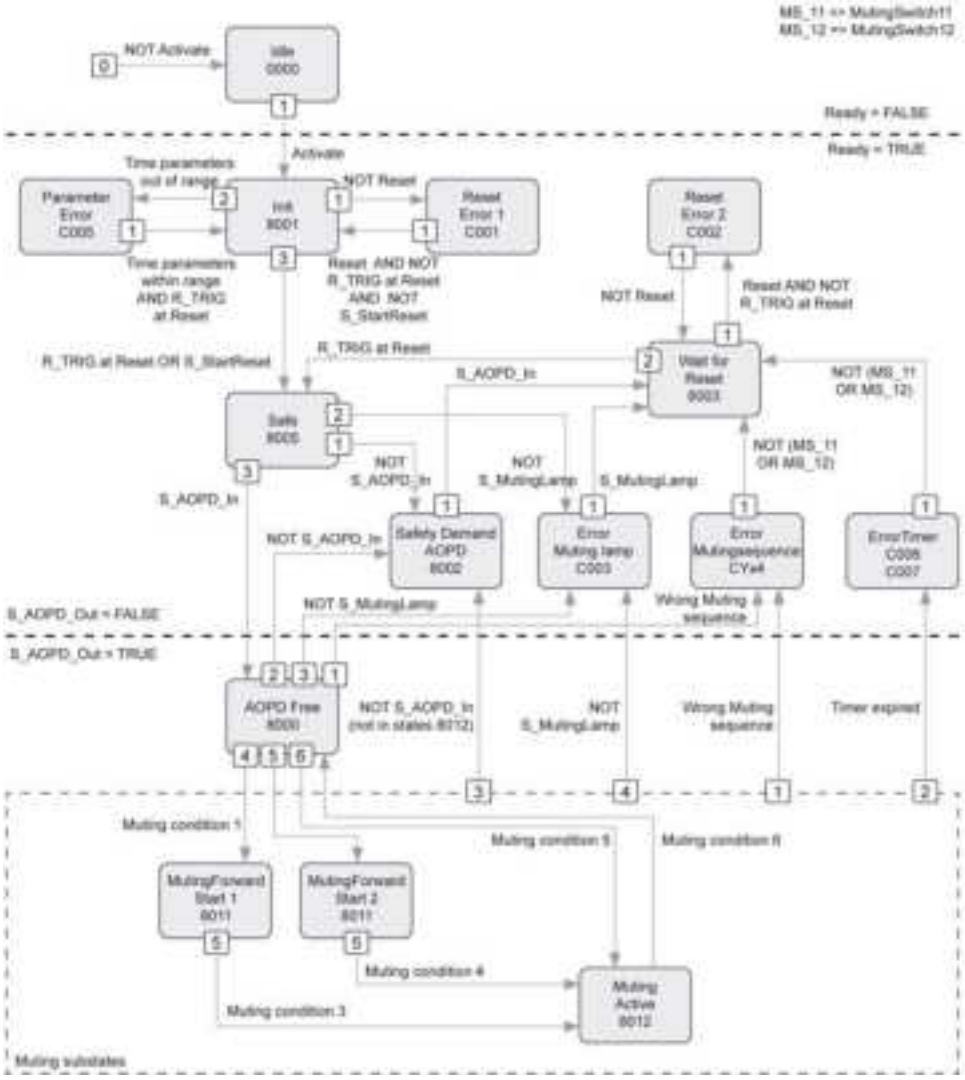
In the event of an error, the S_AOPD_Out and S_MutingActive outputs will be set to FALSE. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. Restarting will be prevented until the error conditions have been fixed and the operator has reset the safe state with Reset.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal was detected after activation in state 8001. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting Lamp	Fault identified in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

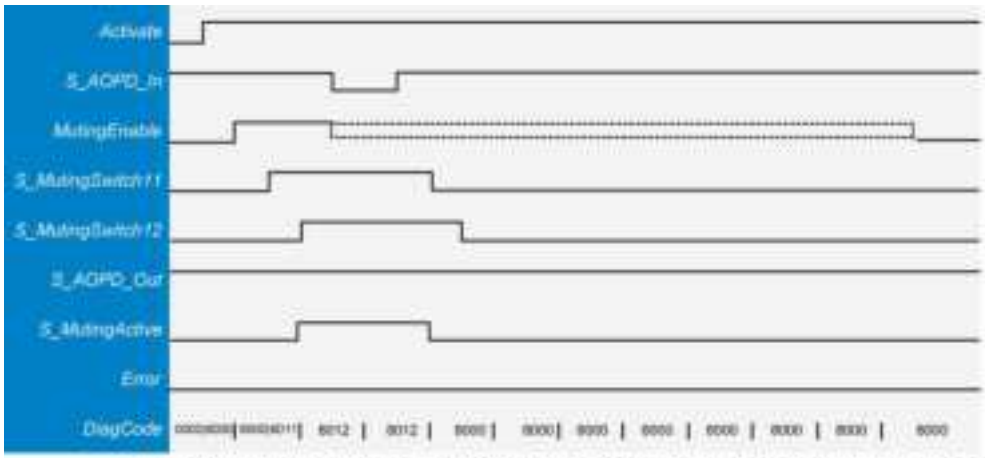
DiagCode	Status name	Status description and output setting
CYx4	Error Muting sequence	CYx4 muting sequence fault was detected in muting sequence state 8000, 8011, 8311. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state 8011 C2x4 = Error occurred in state 8311 CFx4 = Muting Enable missing x = Status of sensors when fault occurred (four bits: LSB = MS_11, next to LSB = MS_12).
C005	Parameter Error	The value in DiscTimeEntry or MaxMutingTime falls outside the permissible range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error timer MaxMuting	Timing error: Active muting time (if S_MutingActive = TRUE) MaxMutingTime exceeded. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C007	Error timer Entry	Timing error: Discrepancy time elapsed for change in S_MutingSwitch11 and S_MutingSwitch12 from FALSE to TRUE. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety request from AOPD. The timers will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8002	Safety Demand AOPD	Safety request from AOPD was detected; muting not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8003	Wait for Reset	Safety request or error was detected and is not reset. Reset by operator is required for Reset. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Start 1	Muting sequence is in starting phase after a rising edge at S_MutingSwitch11. DiscTimeEntry monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8311	Muting Start 2	Muting sequence is in starting phase after a rising edge at S_MutingSwitch12. DiscTimeEntry monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8012	Muting Active	The muting sequence is active. - After a rising edge was detected at the second S_MutingSwitch (12 or 11). - If S_MutingSwitch 11 and 12 were actuated in the same cycle. DiscTimeEntry monitoring will be stopped. MaxMutingTime monitoring is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE

13.5.13.3 Status Diagram SF_MutingPar_2Sensor



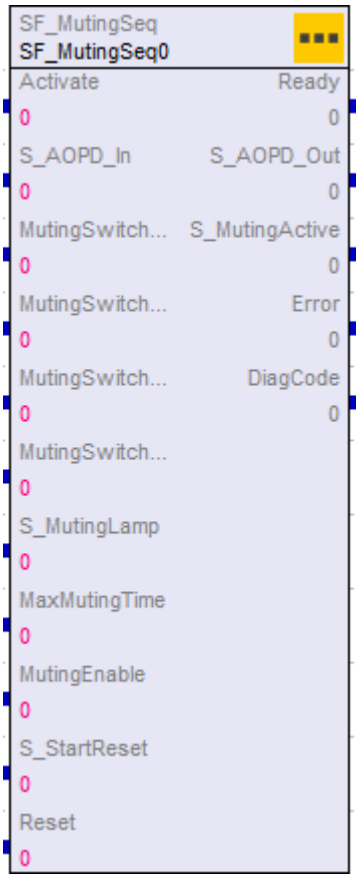
13.5.13.4 Time Diagram SF_MutingPar_2Sensor



Timing diagram for SF_MutingPar_2Sensor (S_StartReset = TRUE, Reset = FALSE, S_MutingLamp = TRUE)

13.5.14 SF_MutingSeq function

"Muting" refers to intentionally suppressing safety functions. You can use this function block to implement sequential muting with four muting sensors (e.g., light barriers).



Input variables	
Activate	Activates the function block (BOOL)
S_AOPD_In	OSSD signal (output signal switching device) of AOPD (active opto-electronic protective device). FALSE: Protected area disrupted. TRUE: Protected area not disrupted.
MutingSwitch11 MutingSwitch12 MutingSwitch21 MutingSwitch22	Muting sensor status. Applies to all: FALSE: Muting sensor not activated. TRUE: Workpiece activating the muting sensor. Please note that a SAFEBOOL will need to be connected instead of a BOOL depending on the safety requirements in question. (Safebool to Bool FB)
S_MutingLamp	Indicates the muting lamp's operating state. FALSE: Faulty muting lamp. TRUE: No muting lamp faults
MaxMutingTime	Constant, 0–10 s; maximum time for a full muting sequence; the timer will be started when the first muting sensor is activated.

MutingEnable	Befehl vom Steuerungssystem, das den Start der Muting-Funktion bei Bedarf des Maschinenzyklus aktiviert. Once the muting function has started, the signal can be switched off. FALSE: Muting not activated TRUE: Start of muting function activated
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL).
S_AOPD_Out	Safety-relevant output; indicates the status of the muting protective door. FALSE: AOPD protected area disrupted and muting is not active. TRUE: AOPD protected area not disrupted or muting is active.
S_MutingActive	Indicates the status of the muting process. FALSE: Muting not active TRUE: Muting active
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

"Muting" refers to intentionally suppressing safety functions. This can be necessary, for example, in order to transport material to a danger zone without stopping the corresponding machine. This muting is monitored by muting sensors, and the use of two or four muting sensors and their correct integration into the production sequence must ensure that no one enters the danger zone while the light curtain is off.Muting sensors include proximity switches, light barriers, limit switches, etc. that do not have to be failsafe devices.

When muting mode is active, indicator lights must signal this.

There are sequential and parallel muting methods. This function block implements parallel muting with four muting sensors. The function block can be used for muting in both directions – forward and backward. Muting must be activated by the process control system with the MutingEnable signal so as to prevent tampering. If the MutingEnable signal is not available, this input must be set to TRUE.

The function block inputs include the signals for the four muting sensors (MutingSwitch11–MutingSwitch22) and the OSSD signal from the "active opto-electronic protective device," S_AOPD_In.

The S_StartReset input should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.14.1 Error detection

The function block will detect the following error conditions:

- Muting sensor MutingSwitch11, MutingSwitch12, MutingSwitch21 and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without having been activated by MutingEnable.
- A faulty muting lamp is indicated by S_MutingLamp = FALSE.
- A static reset condition.
- MaxMutingTime has been set to a value that is less than T#0 s or greater than T#10 min.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.

13.5.14.2 Error behavior

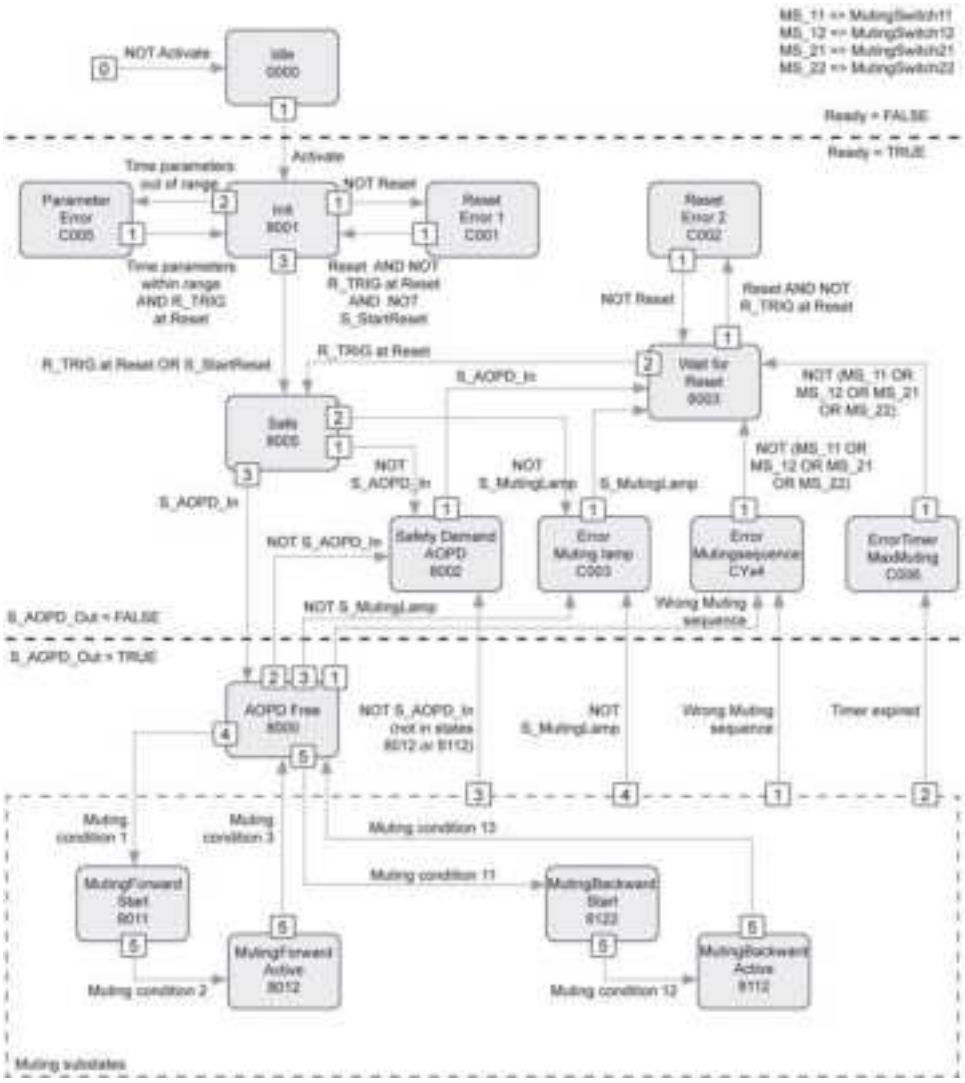
In the event of an error, the S_AOPD_Out and S_MutingActive outputs will be set to FALSE. The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. Restarting will be prevented until the error conditions are fixed and the operator has reset them with Reset.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal was detected after function block activation. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state 8003. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C003	Error Muting lamp	Fault identified in muting lamp. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE

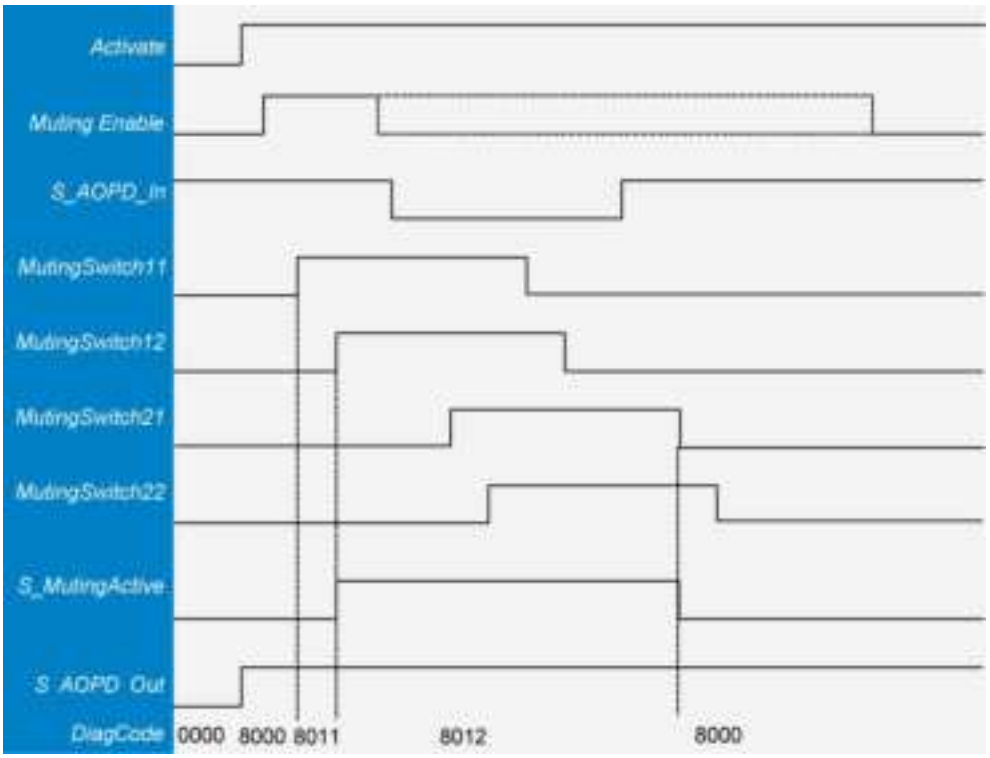
DiagCode	Status name	Status description and output setting
CYx4	Wrong mutig sequence	Wrong muting sequence detected in state 8000, 8011, 8012, 8112, or 8122. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE Y = Status in the sequence (two states for the forward direction and two for the backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in forward state 8011 C2x4 = Error occurred in forward state 8012 C3x4 = Error occurred in backward state 8122 C4x4 = Error occurred in backward state 8112 CFx4 = Muting Enable missing x = Status of the sensors occurred when the error occurred (4 Bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).
C005	Parameter Error	MaxMutingTime falls outside of the permissible range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
C006	Error Timer	Timing error: Active muting time (if S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8000	AOPD Free	Muting not active and no safety request from AOPD. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8001	Init	Function block was activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8002	Safety Demand AOPD	Safety requests from AOPD were detected; muting is not active. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8003	Wait for Reset	Safety request or fault was detected and has now been reset by the operator. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8005	Safe	Safety function activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE Error = FALSE
8011	Muting Forward Start	Muting forward sequence is in its starting phase and no safety request. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE
8012	Muting Forward Active	Muting forward; sequence is active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8112	Muting Backward Active	Muting backward; sequence active. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE Error = FALSE
8122	Muting Backward Start	Muting backward sequence is in its starting phase and no safety request. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE Error = FALSE

13.5.14.3 Status Diagram SF_MutingSeq

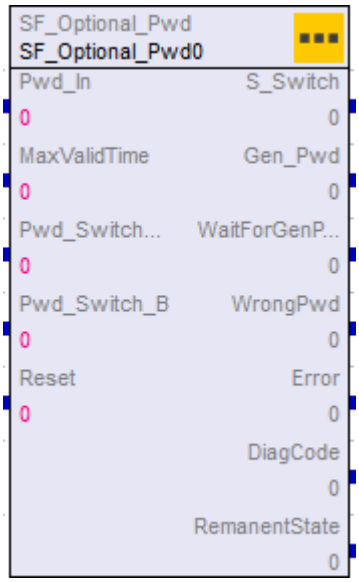


13.5.14.4 Time Diagram SF_MutingSeq



13.5.15 SF_Optional_Pwd function

Used for password requests and password checks for optional modules. This function block should only be used together with the → "SF_Optional_Switch function", page 300 function block.



Input variables	
Pwd_In	Entered password (DINT); a change will trigger the change operation.
MaxValidTime	Maximum time (DINT) that the generated confirmation password will continue to be valid. A value of 0 means that this time is unlimited. A constant must be used to write this value.
Pwd_Switch_A	Password (DINT) for activating the optional module. A constant must be used to write this value.
Pwd_Switch_B	Password (DINT) for deactivating the optional module. A constant must be used to write this value.
Reset	Reset (BOOL)
Output variables	
S_Switch	Switches the connected function block SF_Optional_Switch. (SAFEBOOL) FALSE: Optional module turned on. TRUE: Optional module turned off. In the state transition time (while there is a wait for the confirmation password), this output will be set to ERROR.
Gen_Pwd	Generated confirmation password (DINT).
WaitForGenPwd	Indicates (BOOL) whether the system is waiting for a generated password from Gen_Pwd to be entered. FALSE: There is no valid confirmation password being applied. TRUE: The specified confirmation password can be entered.
WrongPwd	Indicates (BOOL) whether there is an expected password at the Pwd_In input. FALSE: Default state TRUE: An incorrect password has been entered.
Error	Error display (BOOL)

DiagCode	Status display (HDINT)
RemanentState	Retentive value (DINT) stored in the module's flash memory in order to restore the previous state after the system is reset.

The SF_Optional_Pwd-FB function block is responsible for checking the passwords for optional modules and controls the connected → "SF_Optional_Switch function", page 300 function block. This combination is used to separate blocks that would trigger errors under certain circumstances from the rest of the system. These blocks include hardware modules that need to be temporarily removed or that need to be shut down due to maintenance and servicing work.In order to make sure that they do not block the entire system, it is possible to switch to an alternative value / alternative block. This ensures that the block that is switched off/suppressed cannot trigger any more errors.

One unique characteristic of this function block is that a Safety CPU restart will be triggered with every state transition. Moreover, in order to get a safeguarded state transition, the S_Switch output will be set to ERROR during password confirmation. Another unique characteristic is that the state of the function block is stored retentively and will be restored automatically after the Safety CPU is turned off and then back on.

13.5.15.1 Error detection

The function block will detect the following error conditions:

- The wrong password being entered in order to confirm the change operation
- Timeout when entering the confirmation password

If a wrong password is entered in state 8000 or 8010, this will not trigger an error. Instead, this will simply be signaled with a value of TRUE at the WrongPwd output.

13.5.15.2 Error behavior

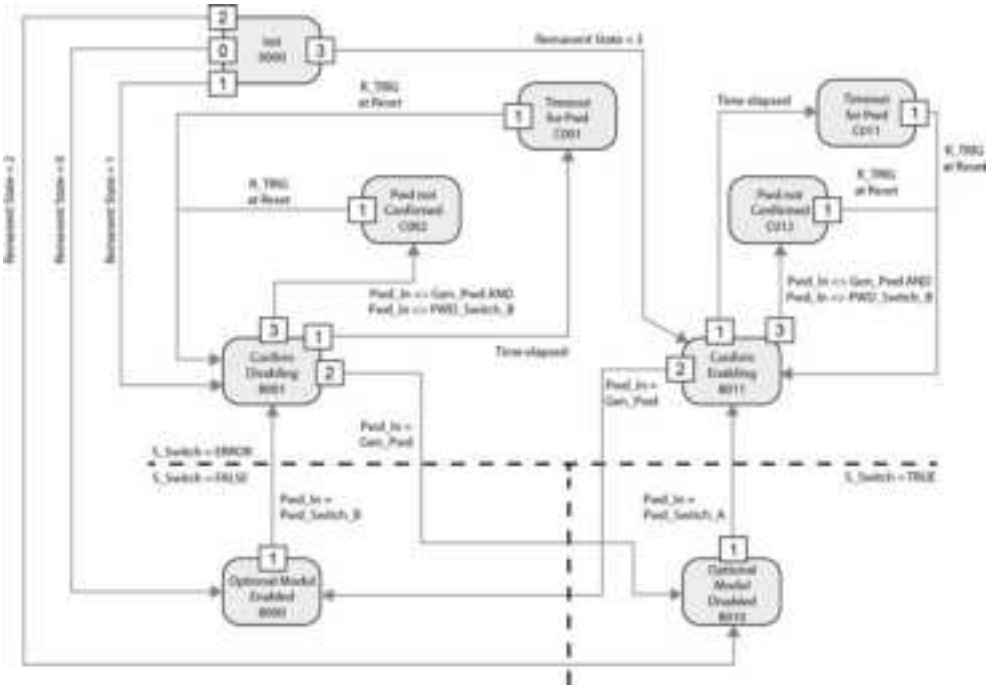
The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. In order to exit the error condition, it must be reset with a rising edge at the Reset input. The Pwd_In input should be set to "0" in this case.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Timeout for Pwd	Time in status 8001 has expired. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE RemanentState = 1

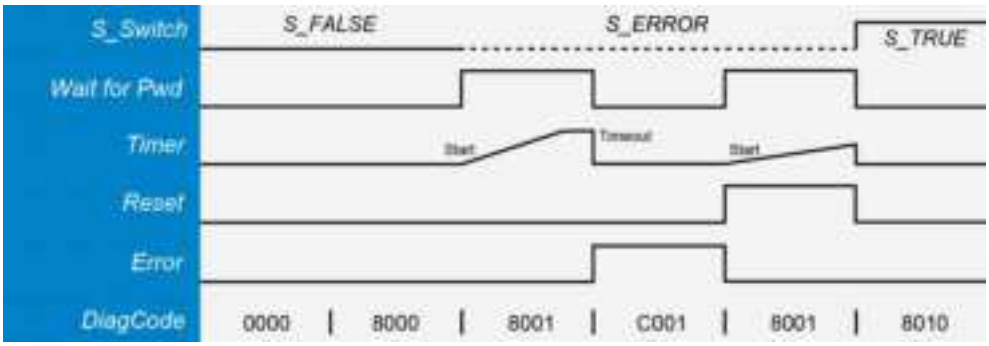
DiagCode	Status name	Status description and output setting
C011	Timeout for Pwd	Time in status 8011 has expired. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE RemanentState = 3
C002	Pwd not Confirmed	Pwd_In was not changed to generated password in state 8001. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE RemanentState = 1
C012	Pwd not Confirmed	Pwd_In was not changed to generated password in state 8011. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE RemanentState = 3
FB-specific status codes (no error)		
0000	Init	The system will automatically switch to the state stored on the retentive server (the retentive server's initial value is 0). S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE
8000	Optional Modul Enabled	Optional module suppression turned off. The downstream switch block will be instructed to connect S_InA through to the output. If the value of Pwd_Switch_B is applied to the Pwd_In input in this state, the function block will switch to state 8001 in order to get confirmation for the change operation. S_Switch = FALSE WaitForGenPwd = FALSE WrongPwd = Pwd_In != Pwd_Switch_A Error = FALSE RemanentState = 0
8010	Optional Modul Disabled	Optional module suppression activated. The downstream switch block will be instructed to connect S_InB through to the output. If the value of Pwd_Switch_A is applied to the Pwd_In input in this state, the function block will switch to state 8011 in order to get confirmation for the change operation. S_Switch = TRUE WaitForGenPwd = FALSE WrongPwd = Pwd_In != Pwd_Switch_B Error = FALSE RemanentState = 2

DiagCode	Status name	Status description and output setting
8001	Confirm Disabling	Waits for the confirmation in the form of entering the Gen_Pwd at Pwd_In in order to complete the switching-off operation with state 8010. S_Switch = ERROR WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE RemanentState = 1
8011	Confirm Enabling	Waits for the confirmation in the form of entering the Gen_Pwd at Pwd_In in order to complete the switching-on operation with state 8000. S_Switch = ERROR WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE RemanentState = 3

13.5.15.3 SF_Optional_Pwd Status Diagram

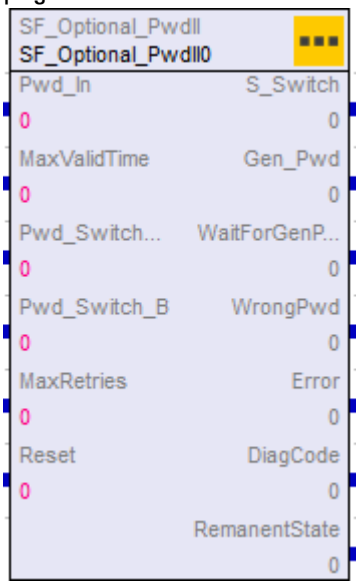


13.5.15.4 Time Diagram SF_Optional_Pwd



13.5.16 SF_Optional_PwdII function

Used for password requests and password checks for optional modules. This function block should only be used together with the → "SF_Optional_Switch function", page 300 function block.



Input variables	
Pwd_In	Entered password (DINT); a change will trigger the change operation.
MaxValidTime	Maximum time (DINT) for completing a change operation (from 8000 to 8010 or from 8010 to 8000). 0 to maximum eight hours. A constant must be used to write this value.
Pwd_Switch_A	Password (DINT) for activating the optional module. A constant must be used to write this value.
Pwd_Switch_B	Password (DINT) for deactivating the optional module. A constant must be used to write this value.
MaxRetries	Maximum number (DINT) of failed attempts for the confirmation password.
Reset	Reset (BOOL)
Output variables	
S_Switch	Switches the connected → "SF_Optional_Switch function", page 300 function block. (SAFEBOOL) FALSE: Optional module turned on. TRUE: Optional module turned off.
Gen_Pwd	Generated confirmation password (DINT).
WaitForGenPwd	Indicates (BOOL) whether the system is waiting for a generated password from Gen_Pwd to be entered. FALSE: There is no valid confirmation password being applied. TRUE: The specified confirmation password can be entered.
WrongPwd	Indicates (BOOL) whether there is an expected password at the Pwd_In input. FALSE: Default state TRUE: An incorrect password has been entered.

Error	Error display (BOOL)
DiagCode	Status display (HDINT)
RemanentState	Retentive value (DINT) stored in the module's flash memory in order to restore the previous state after the system is reset.

The SF_Optional_PwdII function block is responsible for checking the passwords for optional modules and controls the connected → "SF_Optional_Switch function", page 300 function block. This combination is used to separate blocks that would trigger errors under certain circumstances from the rest of the system. These blocks include hardware modules that need to be temporarily removed or that need to be shut down due to maintenance and servicing work.In order to make sure that they do not block the entire system, it is possible to switch to an alternative value / alternative block. This ensures that the block that is switched off/suppressed cannot trigger any more errors.

One unique characteristic of this function block is that the S_Switch output will be set to ERROR during an error condition in order to get a safeguarded state transition. Another unique characteristic is that the state of the function block is stored retentively and will be restored automatically after the Safety CPU is turned off and then back on.

Differences between the → " SF_Optional_Pwd function", page 289 and→ " SF_Optional_PwdII function", page 294 function blocks

- During the return confirmation phase, the S_Switch output will keep its old value and will not switch to SAFEBOOL_ERROR when using the SF_Optional_PwdII function block, unlike with the SF_Optional_Pwd function block.
- With the SF_Optional_PwdII function block, a Safety CPU reboot will not be triggered when there is a retentive storage operation.

13.5.16.1 Error detection

The function block will detect the following error conditions:

- The wrong password being entered in order to confirm the change operation
- Timeout when entering the confirmation password

If a wrong password is entered in state 8000 or 8010, this will not trigger an error. Instead, this will simply be signaled with a value of TRUE at the WrongPwd output.

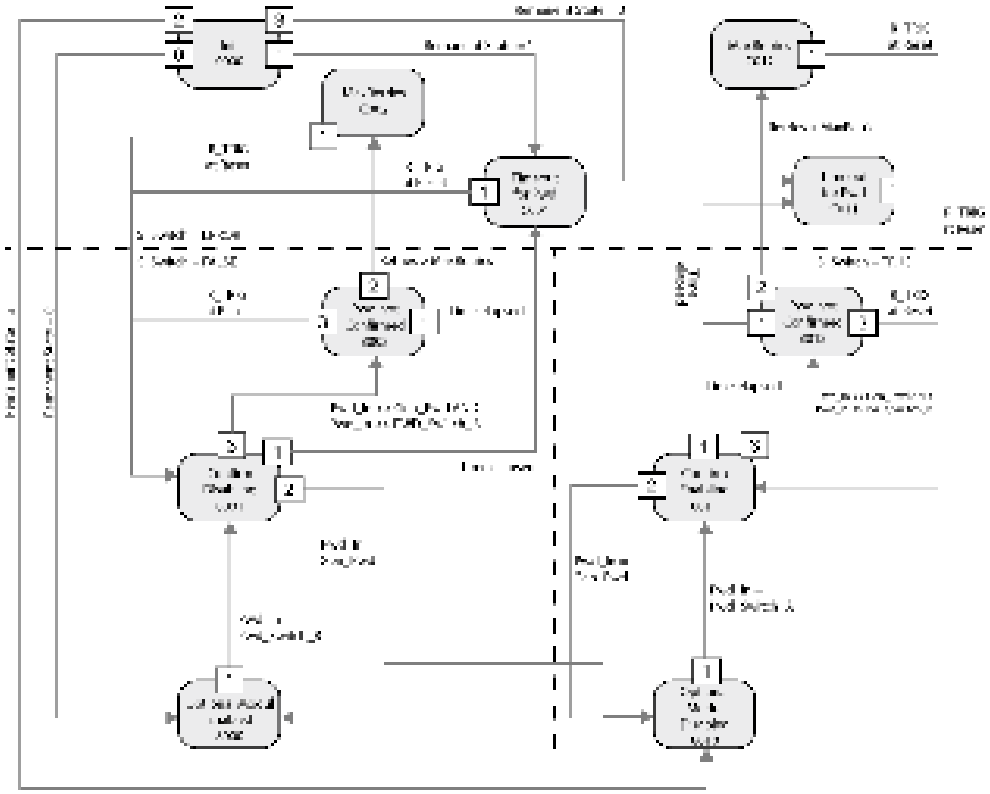
13.5.16.2 Error behavior

The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. In order to exit the error condition, it must be reset with a rising edge at the Reset input. The Pwd_In input should be set to "0" in this case.

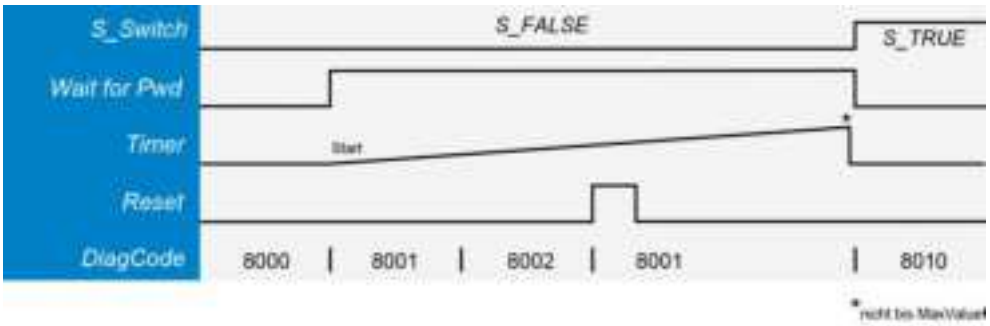
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Timeout for Pwd	Time in state 8001 or 8002 has elapsed. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE RemanentState = 1
C011	Timeout for Pwd	Time in state 8011 or 8012 has elapsed. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE RemanentState = 3
C002	MaxRetries	Number of attempts was exceeded in state 8001. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE RemanentState = 1
C012	MaxRetries	Number of attempts was exceeded in state 8011. S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE RemanentState = 3
FB-specific status codes (no error)		
0000	Init	The system will automatically switch to the state stored on the retentive server (the retentive server's initial value is 0). S_Switch = ERROR WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE
8000	Optional Modul Enabled	Optional module suppression turned off. The downstream switch block will be instructed to connect S_InA through to the output. If the value of Pwd_Switch_B is applied to the Pwd_In input in this state, the function block will switch to state 8001 in order to get confirmation for the change operation. S_Switch = FALSE WaitForGenPwd = FALSE WrongPwd = Pwd_In != Pwd_Switch_A Error = FALSE RemanentState = 0

DiagCode	Status name	Status description and output setting
8010	Optional Modul Disabled	Optional module suppression activated. The downstream switch block will be instructed to connect S_InB through to the output. If the value of Pwd_Switch_A is applied to the Pwd_In input in this state, the function block will switch to state 8011 in order to get confirmation for the change operation. S_Switch = TRUE WaitForGenPwd = FALSE WrongPwd = Pwd_In != Pwd_Switch_B Error = FALSE RemanentState = 2
8001	Confirm Disabling	Waits for the confirmation in the form of entering the Gen_Pwd at Pwd_In in order to complete the switching-off operation with state 8010. S_Switch = FALSE WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE RemanentState = 1
8011	Confirm Enabling	Waits for the confirmation in the form of entering the Gen_Pwd at Pwd_In in order to complete the switching-on operation with state 8000. S_Switch = TRUE WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE RemanentState = 3
8002	Pwd not Confirmed	Pwd_In was not changed to generated password in state 8001. S_Switch = FALSE WaitForGenPwd = FALSE WrongPwd = TRUE Error = FALSE RemanentState = 1
8012	Pwd not Confirmed	Pwd_In was not changed to generated password in state 8011. S_Switch = TRUE WaitForGenPwd = FALSE WrongPwd = TRUE Error = FALSE RemanentState = 3

13.5.16.3 Status Diagram SF_Optional_PwdII



13.5.16.4 Time Diagram SF_Optional_PwdII



13.5.16.5 Safety measures and instructions

The use of the "SF_OPTIONAL_PWDII" function block affects all safety functions in general, which is why it is absolutely necessary to observe the following measures and hazard information. Failure to observe these warnings can result in serious injury or death!

DANGER

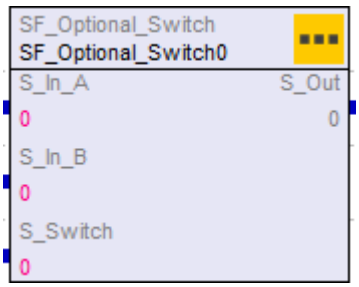
- The machine's user must conduct a risk assessment regarding the potential hazards that can be posed by the use of the "SF_

OPTIONAL_PWDII" function block while taking into account the standard-based requirements that apply to the specific application in question (e.g., hydraulic press brakes) and take all required risk mitigation measures.

- The time during which the code calculated by the Safety CPU continues to be valid (login time) must always be as short as possible. In particular, it must always be ensured that this time does not cut across shifts. It is absolutely necessary to ensure that the authorized operator that enters the password is exactly the same one that confirms the automatically calculated code.
- The machine must be within sight of the machine's operator at all times while the automatically calculated code continues to be valid.
- With the exception of the machine's operator, no one else should be within the machine's limits as defined in EN 12100. In particular, suitable measures must be used in order to ensure that the machine cannot be freely accessed.
- Actuators without an emergency stop function must be installed in such a way that an emergency stop can be triggered within hand's reach from any position at all times.

13.5.17 SF_Optional_Switch function

A switch input can be used to select the input that should be connected through to the output.



Input variables	
S_In_A	Input A (SAFEBOOL)
S_In_B	Input B (SAFEBOOL)
S_ Switch	Switch input used to select which input (S_In_A or S_In_B) should be connected through to the output. If this input assumes an invalid SAFEBOOL value or ERROR, then the output will be switched to ERROR. (SAFEBOOL)
Output variables	
S_Out	Value of the input connected through or ERROR if there is no valid value at the S_Switch switch input (SAFEBOOL).

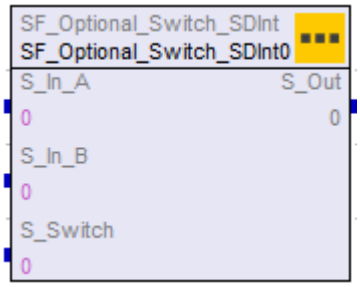
A special characteristic is that a check for errors is not run on the inputs, but on the output instead. This means that there can be an undefined value at the input that is not being connected through and this will not cause the output to switch to ERROR.

13.5.17.1 Table of values

S_Switch	S_Out
FALSE	S_In_A
TRUE	S_In_B
ERROR	ERROR
undefined	ERROR

13.5.18 SF_Optional_Switch_SDint

A switch input can be used to select which input should be connected through to the output.

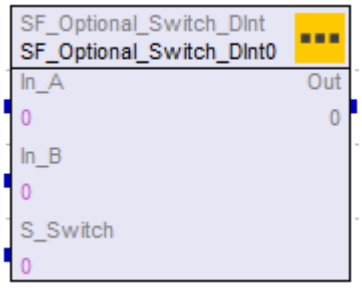


Input variables	
S_In_A	Input A (SAFEDINT)
S_In_B	Input B (SAFEDINT)
S_ Switch	Switch input used to select which input (S_In_A or S_In_B) should be connected through to the output. If this input assumes a value of SAFEBOOL_ERROR, then the output will switch to SAFEDINT_ ERROR. (SAFEBOOL)
Output variables	
S_Out	Value of the input connected through or ERROR if there is an ERROR at the S_Switch switch input. (SAFEDINT).

A special characteristic is that a check for errors is not run on the inputs, but on the output instead. This means that there can be an undefined value at the input that is not being connected through and this will not cause the output to switch to ERROR.

13.5.19 SF_Optional_Switch_Dint

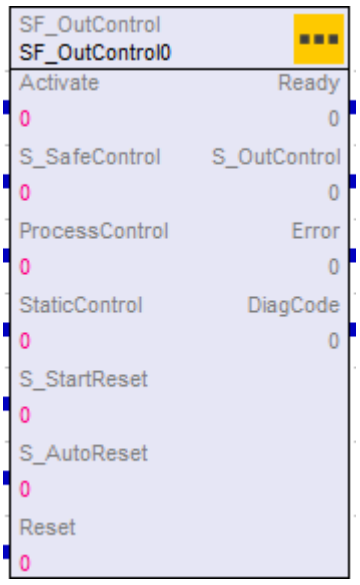
A switch input can be used to select which input should be connected through to the output.



Input variables	
In_A	Input A (DINT)
In_B	Input B (DINT)
S_ Switch	Switch input used to select which input (In_A or In_B) should be connected through to the output. If this input assumes a value of SAFEBOOL_ERROR, then IN_A will be connected through to the output. (SAFEBOOL)
Output variables	
Out	Value of input connected through or In_A if there is an ERROR at the S_Switch switch input. (DINT)

13.5.20 SF_OutControl function

Controls a Safety output with a signal from the functional application and a Safety signal with an optional start inhibit.



Input variables	
Activate	Activates the function block (BOOL)
S_SafeControl	Control signal from a preceding safety function block (e.g., SF_Estop, SF_GuardMonitoring, SF_TwoHandControlTypeII, and others). FALSE: The preceding safety function blocks are in their safe state TRUE: The preceding safety function blocks are activating the safety PLC.
ProcessControl	Control signal from functional application. (BOOL) FALSE: Request for setting S_OutControl to FALSE. TRUE: Request for setting S_OutControl to TRUE.
StaticControl	Optional prerequisites for process control. (BOOL) FALSE: Dynamic change at ProcessControl (FALSE => TRUE) required after the block activation or a triggered safety function. Additional function start required. TRUE: No dynamic change required at ProcessControl (FALSE => TRUE) after block activation or a triggered safety function.
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released (SAFEBOOL)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the outputs are valid. (BOOL)
S_OutControl	Controls the connected actuator. (SAFEBOOL) FALSE: Disable for the connected actuators TRUE: Enable for the connected actuators
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

13.5.20.1 General

The SF_OutControl function block is an output driver for a safety output. This safety output is controlled with S_OutControl with the use of a signal from the functional application (ProcessControl/BOOL, for controlling the process) and a signal from the safety application (S_SafeControl/SAFEBOOL, for controlling the safety function).

13.5.20.2 Optional conditions for ProcessControl

- An additional function start (ProcessControl FALSE => TRUE) is required after the block's activation or feedback from the safe signal (S_SafeControl). A static TRUE signal at ProcessControl will not set S_OutControl to TRUE.
- An additional function start (ProcessControl FALSE => TRUE) is not required after block activation or feedback from the safe signal (S_SafeControl). A static TRUE signal at ProcessControl will set S_OutControl to TRUE if the other prerequisites are met.

13.5.20.3 Optional startup inhibits

- Start inhibit after function block activation.
- Start inhibit after the safety device is disrupted.

The StaticControl, S_StartReset, and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.20.4 Error detection

The following conditions will force a switch to the error state:

- Invalid static reset signal in the process
- Invalid static ProcessControl signal.
- ProcessControl and Reset are connected incorrectly due to a programming mistake.

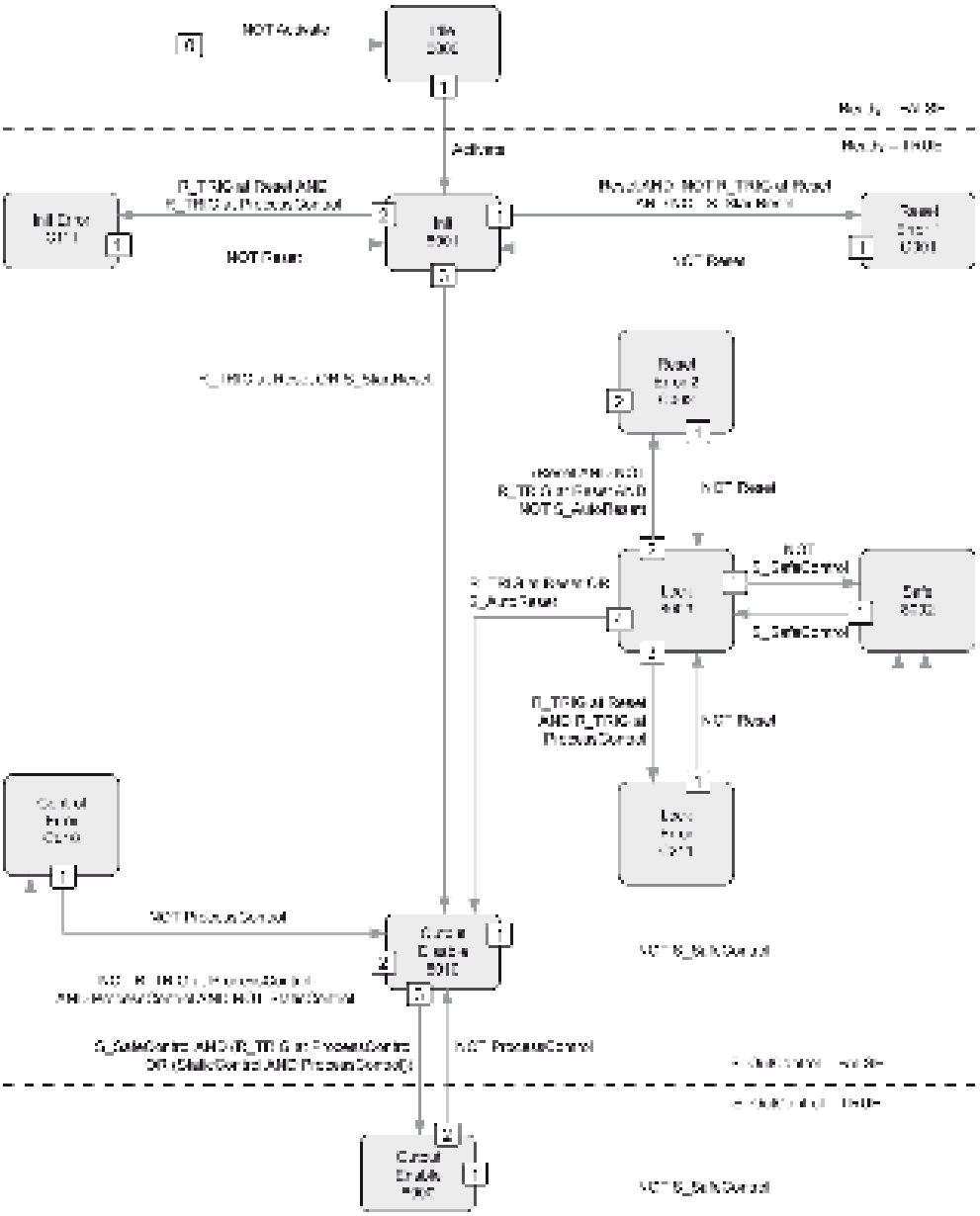
13.5.20.5 Error behavior

In the event of an error, the S_OutControl output will be set to FALSE and will remain in its safe state. To exit the reset, init, or lock error states, the Reset input must be set to FALSE. To exit the control error state, the ProcessControl input must be set to FALSE.

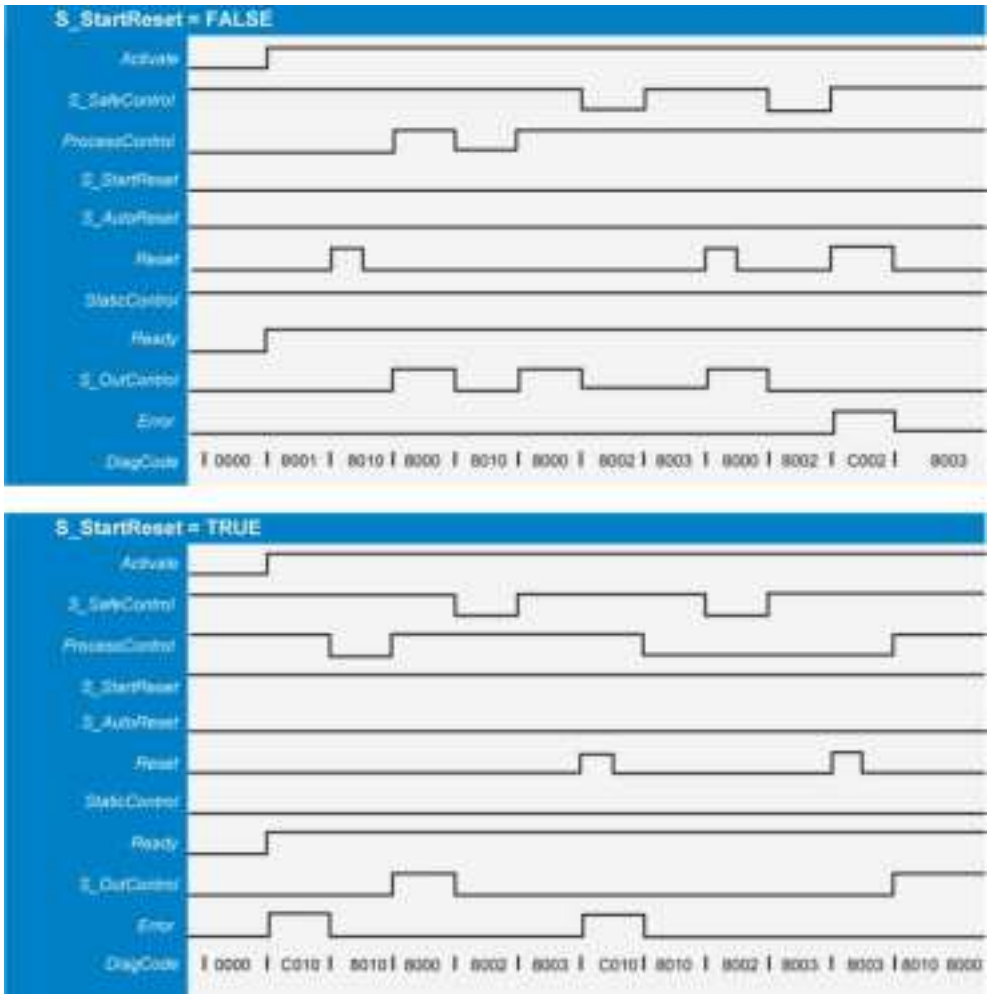
After S_SafeControl transitions to TRUE, the optional startup inhibit can be reset with a rising edge at the Reset input. After block activation, the optional start inhibit can be reset with a rising edge at the Reset input.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE
C010	Control Error	Static signal at ProcessControl in state 8010. Ready = TRUE S_OutControl = FALSE Error = TRUE
C111	Init Error	Simultaneous rising edges at Reset and ProcessControl in state 8001. Ready = TRUE S_OutControl = FALSE Error = TRUE
C211	Lock Error	Simultaneous rising edges at Reset and ProcessControl in state 8003. Ready = TRUE S_OutControl = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OutControl = FALSE Error = FALSE
8001	Init	Block activation start inhibit is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8002	Safe	Triggered safety function. Ready = TRUE S_OutControl = FALSE Error = FALSE
8003	Lock	Start inhibit for safety function is active. Reset required. Ready = TRUE S_OutControl = FALSE Error = FALSE
8010	Output Disable	Process control is not active. Ready = TRUE S_OutControl = FALSE Error = FALSE
8000	Output	Process control activation is active and safety is established. Ready = TRUE S_OutControl = TRUE Error = FALSE

13.5.20.6 Status Diagram SF_OutControl

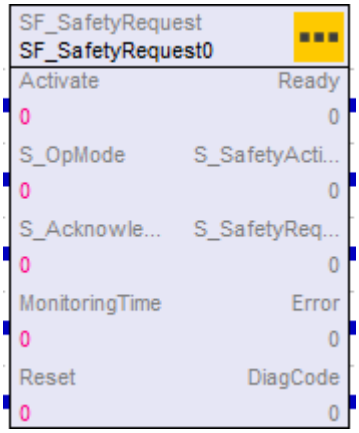


13.5.20.7 Time Diagram SF_OutControl



13.5.21 SF_SafetyRequest function

This function block serves as an interface to a generic actuator (a safety drive or a safety valve, for example) and can be used to bring this actuator to a safe state.



Input variables	
Activate	Activates the function block (BOOL)
S_OpMode	Requested safe mode for an actuator (SAFEBOOL). FALSE: Safe mode requested TRUE: Operating mode requested
S_Acknowledge	Actuator confirmation if the actuator is in a safe state. (SAFEBOOL) FALSE: Operating mode (non-safe) TRUE: Safe mode
MonitoringTime	Monitoring of response time between the requested safe mode (S_OpMode with a value of FALSE) and the actuator confirmation (S_Acknowledge with a value of TRUE). (DINT)
Reset	Reset (BOOL)
Output variables	
Ready	TRUE: If the function block is activated and the output results are valid. (BOOL)
S_SafetyActive	Safe mode confirmation. FALSE: Non-safe Modus TRUE: Safe Modus (SAFEBOOL)
S_SafetyRequest	Requested mode for an actuator FALSE: Safe mode requested TRUE: Non-safe mode (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

This function block acts as an interface for the safety-related system of a generic actuator. In other words, the actuator's safety-relevant functions will be available in the application program. However, please note that there are only two binary signals for controlling the generic actuator's safe state – one for queries and one for receiving a confirmation.

The safety function is provided by the actuator. Accordingly, the function block only initiates the request, monitors it, and sets the output if the actuator detects a safe state. This is indicated with the "S_SafetyActive" output.

This function block does not define any generic actuator-specific parameters (you will instead need to define these parameters directly in the generic actuator). Instead, it switches the generic actuator in question from operating mode to a safe state.

13.5.21.1 Error detection

The function block will detect if the actuator is not in a safe state within the monitoring time.

The function block will detect if the confirmation signal is lost while the request is still in place.

The function block will detect a static reset signal

External FB error:

There are no external errors, since there is no error bit / information provided by the generic actuator.

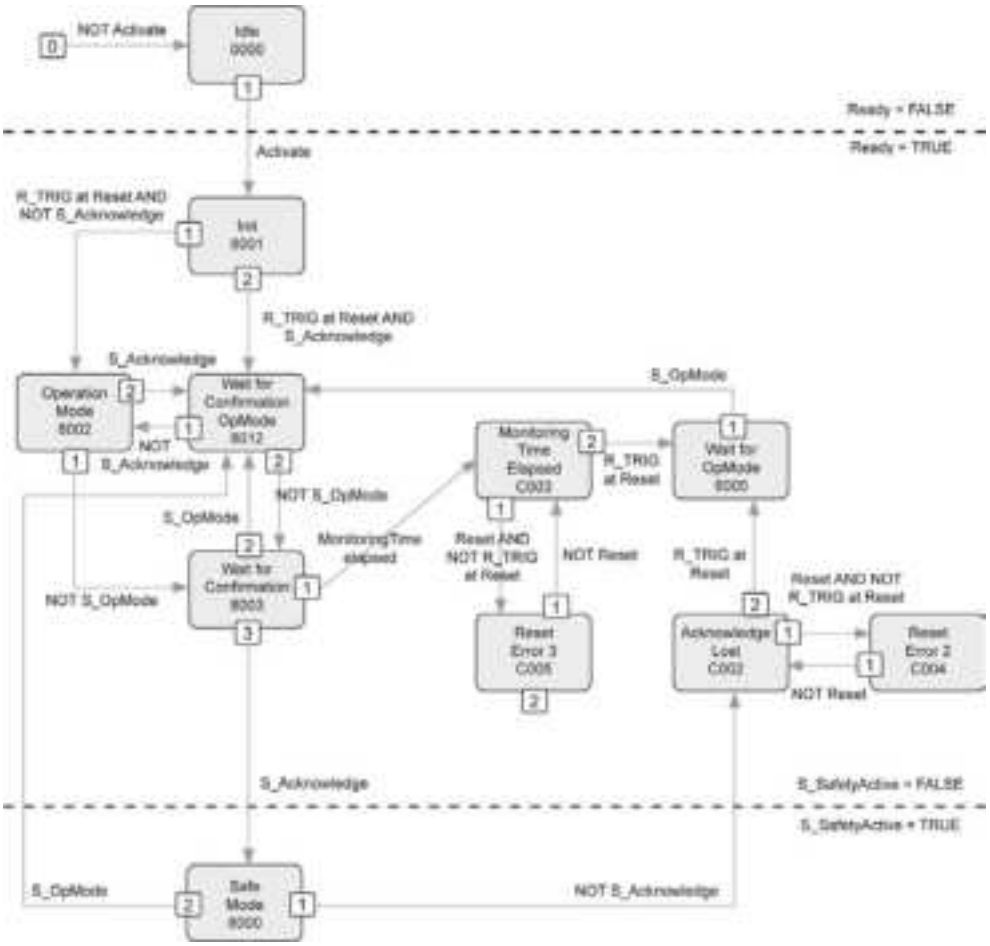
13.5.21.2 Error behavior

In the event of a fault, the S_SafetyActive output will be set to FALSE. An error must be reset with a rising edge at the Reset input. To resume function block operation after a reset, the S_OpMode request must be set to TRUE.

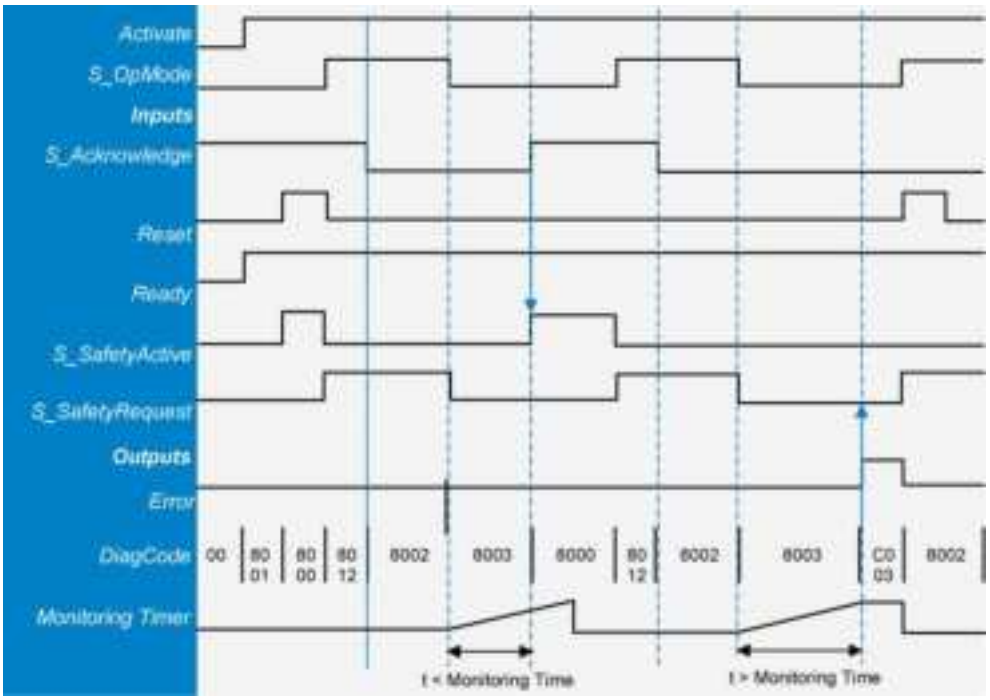
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C002	Acknowledge Lost	No response to confirmation in safe state. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C003	MonitoringTime Elapsed	No response to S_OpMode request within monitoring time. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
C004	Reset Error 2	Static reset was detected in state C002. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE

DiagCode	Status name	Status description and output setting
C005	Reset Error 3	Static reset was detected in state C003 (MonitoringTime elapsed). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8000	Safe Mode	Actuator is in safe state. Ready = TRUE S_SafetyActive = TRUE S_SafetyRequest = FALSE Error = FALSE
8001	Init	State after activation Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8002	Operation Mode	Operating mode without safe state confirmation. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8012	Wait for Confirmation OpMode	Operating mode with safe state confirmation. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE Error = FALSE
8003	Wait for Confirmation	Wait for actuator confirmation. (System Interface.) Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE
8005	Wait for OpMode	Error has been acknowledged. Wait until S_OpMode is set to TRUE. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE Error = FALSE

13.5.21.3 Status Diagram SF_SafetyRequest

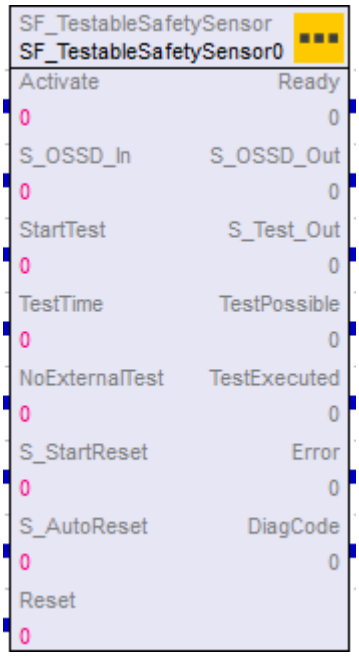


13.5.21.4 Time Diagram SF_SafetyRequest



13.5.22 SF_TestableSafetySensor function

This function block detects, for example, the loss of sensor detection, longer response times than specified, and static ON signals in single-channel sensor systems. It can be used for externally testable safety sensors (ESPE).



Input variables	
Activate	Activates the function block (BOOL).
S_OSSD_In	OSSD = Output Signal Switching Device Sensor status (e.g., light curtain) (SAFEBOOL) FALSE: Safety sensor in test off mode or in the state corresponding to a safety-relevant response TRUE: Safety sensor in normal operating condition status.
StartTest	Input for starting the sensor test. Sets "S_TestOut" and starts the time function with the internal monitoring. (BOOL) FALSE: No test required. TRUE: Test requested.
TestTime	Constant. Range: 0–150 ms. Test time for Safety sensor. (DINT)
NoExternalTest	Constant. Indicates if external manual sensor testing is supported. (BOOL) FALSE: External manual sensor testing is supported. Automatic testing will not be possible until after a complete manual sensor switching sequence. TRUE: External manual sensor testing is not supported. Automatic testing will be possible without a manual sensor switching sequence after a faulty automatic sensor test.
S_StartReset	FALSE: Manual reset TRUE: Automatic reset after the CPU is started (SAFEBOOL)
S_AutoReset	FALSE: Manual reset TRUE: Automatic reset after the emergency stop button is released (SAFEBOOL).

Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid (BOOL).
S_OSSD_Out	Safety-relevant output for the ESPE state. (SAFEBOOL) FALSE: Safety-relevant request for the sensor or test error. TRUE: No safety-relevant request for the sensor and no test error.
S_TestOut	Connected to the sensor's test input. Even though this output is implemented as a SAFEBOOL output, the signal is often output with a BOOL value in practice. FALSE: Test requested TRUE: Test not requested
TestPossible	Feedback signal for the process. (BOOL) FALSE: Automatic sensor testing is not possible. TRUE: Automatic sensor testing is possible
TestExecuted	A positive signal edge indicates that the automatic sensor test was carried out successfully. (BOOL) FALSE: - An automatic sensor test has not been carried out yet. - An automatic sensor test is active. - An automatic sensor test was faulty. TRUE: - A sensor test was carried out successfully.
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

Type 2 ESPEs must include a way to carry out regular testing in order to detect dangerous faults (e.g., loss of sensor detection capability, response time exceeding the specified value). The test signal must simulate the actuation of the sensing device, and the test must not last more than 150 ms. For light curtains, this test must ensure that every single beam works in conformity with supplier specifications.If the test is initiated by an external safety-relevant control system (such as a machine), suitable input options (such as terminals) must be incorporated into the ESPE. Please note that the ESPE must be selected in conformity with product standards EN IEC 61496-1, 61496-2, and 61496-3 and the required categories in EN ISO 13849. It must be monitored with a separate functionality so that the test will be initiated at appropriate intervals.The S_StartReset and S_AutoReset inputs should only be activated if it has been ensured that no potential hazardous situations can occur if the PES is started.

13.5.22.1 Test mode

1. StartTest = TRUE: S_TestOut = FALSE. Start monitoring time
2. S_TestOut signal will stop the transmitter (monitoring of TestTime started for the first time)
3. S_OSSD_In will change from TRUE to FALSE (monitoring of TestTime started for the second time)
4. S_TestOut will change from FALSE to TRUE
5. Start the transmitter
6. Sensor S_OSSD_In will change from FALSE to TRUE
7. Stop the monitoring time
8. S_OSSD_Out set to TRUE during the test

13.5.22.2 Optional Startup inhibits

- Start inhibit after function block activation.
- Start inhibit after the safety device is disrupted.

13.5.22.3 Error detection

The following conditions will force a switch to the error state:

- Test time exceeded without delayed sensor feedback.
- Test without sensor signal feedback.
- Invalid static reset signal in process.
- Validation check for time setting for monitoring.

13.5.22.4 Error behavior

In the event of a fault, the S_OSSD_Out output will be set to FALSE. As soon as the error has been fixed and the sensor has a value of (S_OSSD_In = TRUE), a reset will reset the error condition and set the S_OSSD_Out output to TRUE. If S_AutoReset = FALSE, a rising edge will be required at the Reset input. After S_OSSD_In transitions to TRUE, the optional start inhibit can be reset with a rising edge at the Reset input.After block activation, the optional start inhibit can be reset with a rising edge at the Reset input.

DiagCode	Status name	Status description and output setting
FB-specific error codes		

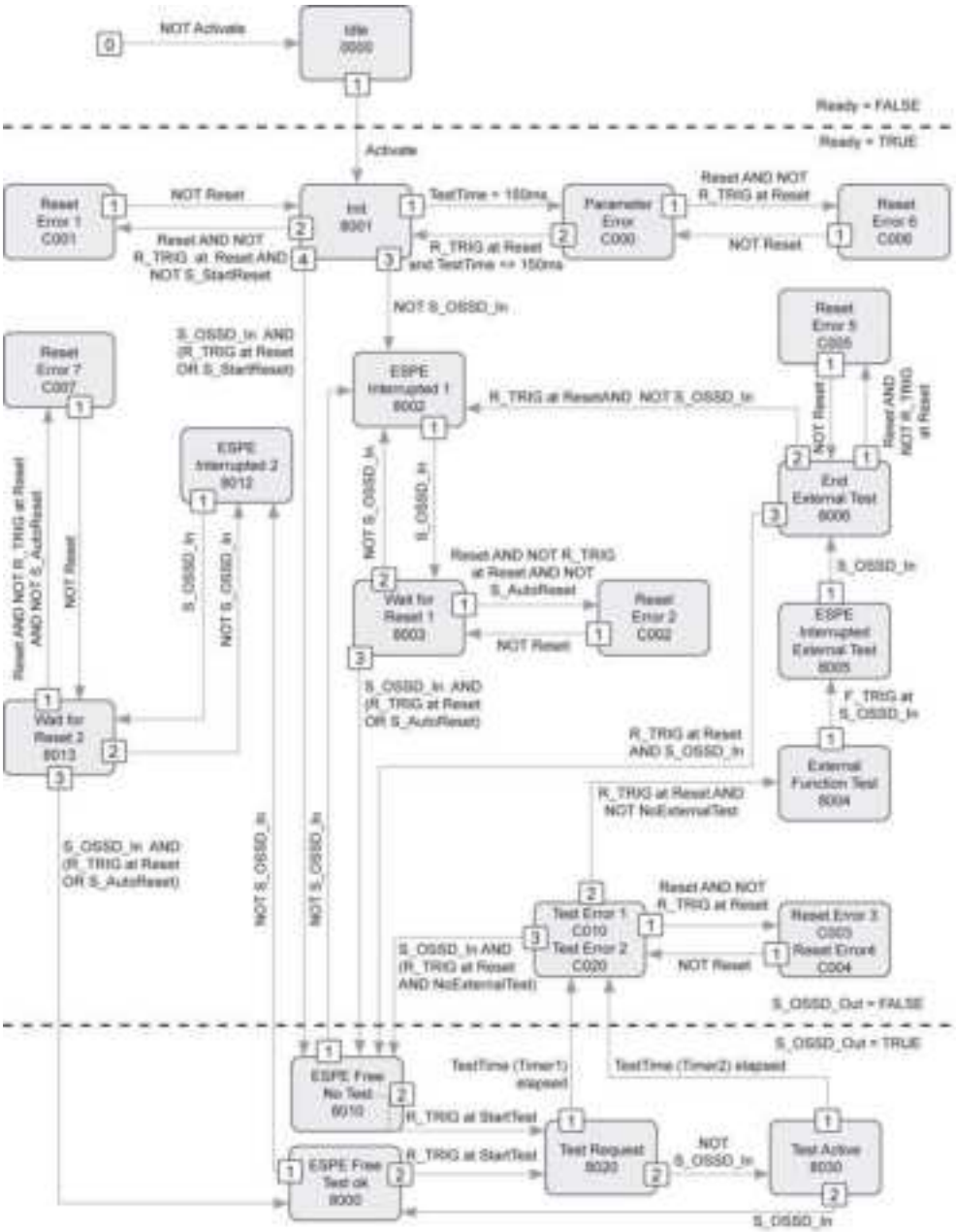
DiagCode	Status name	Status description and output setting
C000	Parameter Error	Invalid value at the TestTime input. Values between 0 ms and 150 ms are allowed. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C001	Reset Error 1	Static reset signal was detected after FB activation. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C002	Reset Error 2	Static reset signal was detected in state 8003. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C003	Reset Error 3	Static reset signal was detected in state C010. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C004	Reset Error 4	Static reset signal was detected in state C020. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C005	Reset Error 5	Static reset signal was detected in state 8006. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C006	Reset Error 6	Static reset signal was detected in state C000. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE

DiagCode	Status name	Status description and output setting
C007	Reset Error 7	Static reset signal was detected in state 8013. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = TRUE
C010	Test Error 1	Test time in status 8020 has expired. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
C020	Test Error 2	Test time in status 8030 has expired. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8001	Init	The function block has been activated. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8002	ESPE Interrupted 1	The function block detected a safety request. The sensor had not yet been tested automatically. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE

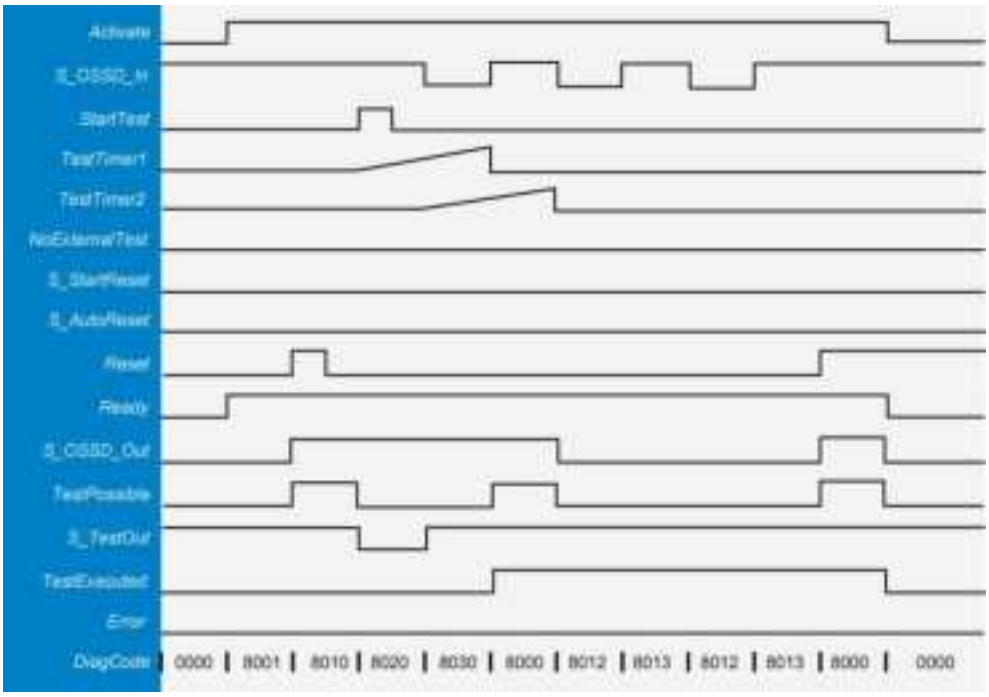
DiagCode	Status name	Status description and output setting
8003	Wait for Reset 1	Waiting for a rising Reset edge after state 8002. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8004	External Function Test	The automatic sensor test was faulty. An external manual sensor test is required. Support for the required external manual sensor test has been activated on the function block (NoExternalTest = FALSE). A negative edge at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8005	ESPE Interrupted External Test	The automatic sensor test was faulty. An external manual sensor test is required. Support for the required external manual sensor test has been activated on the function block (NoExternalTest = FALSE). A TRUE signal at the sensor is required. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8006	End External Test	The automatic sensor test was faulty. An external manual sensor test is required. Support for the required external manual sensor test has been activated on the function block (NoExternalTest = FALSE). The external manual test is completed. The function block has a complete sensor switching cycle (externally controlled). Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8010	ESPE Free No Test	The function block did not detect a safety request. The sensor was not tested automatically. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = FALSE Error = FALSE

DiagCode	Status name	Status description and output setting
8020	Test Request	The automatic sensor test is active. The test timer will be started for the first time. The sensor's transmitter signal will be turned off by the function block. The receiver's signal must follow the transmitter's signal. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = FALSE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8030	Test Active	The automatic sensor test is active. The test timer will be started for the second time. The sensor's transmitter will be turned on by the function block. The receiver's signal must follow the transmitter's signal. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE Error = FALSE
8000	ESPE Free Test ok	The function block did not detect a safety request. The sensor was tested automatically. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = TRUE Error = FALSE
8012	ESPE Interrupted 2	The function block detected a safety request. The switch was tested automatically. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE
8013	Wait for Reset 2	Waits for a rising edge at the Reset input after state 8012. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE Error = FALSE

13.5.22.5 Status Diagram SF_TestableSafetySensor

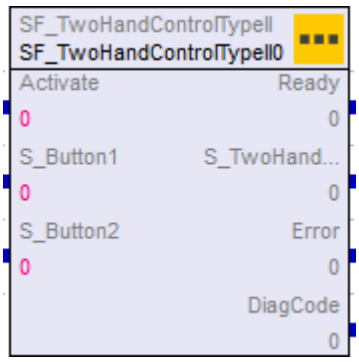


13.5.22.6 Time Diagram SF_TestableSafetySensor



13.5.23 SF_TwoHandControlTypeII function

This function block offers two-hand control functionality (see EN 574, section 4 type II).



Input variables	
Activate	Activates the function block (BOOL)
S_Button1	Input 1 (SAFEBOOL)
S_Button2	Input 2 (SAFEBOOL)
Output variables	
Ready	If TRUE: Function block is activated and the outputs are valid. (BOOL)
S_TwoHandOut	Safety-relevant output signal (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

This function block offers the two-hand control functionality in accordance with EN 574, section 4 type II. If S_Button1 and S_Button2 are both set to TRUE, the output S_TwoHandOut is also set to TRUE. If one of the S_Buttons is then set to FALSE, S_TwoHandOut is also set to FALSE. Before S_TwoHandOut can be set to TRUE again, both S_Button1 and S_Button2 must first be set to FALSE.If one of the S_Button inputs is set from TRUE to FALSE and the other is set from FALSE to TRUE at the same time, an error will be triggered.

13.5.23.1 Error detection

If one or both buttons are already activated when the function block is activated, this will result in a function block error condition.

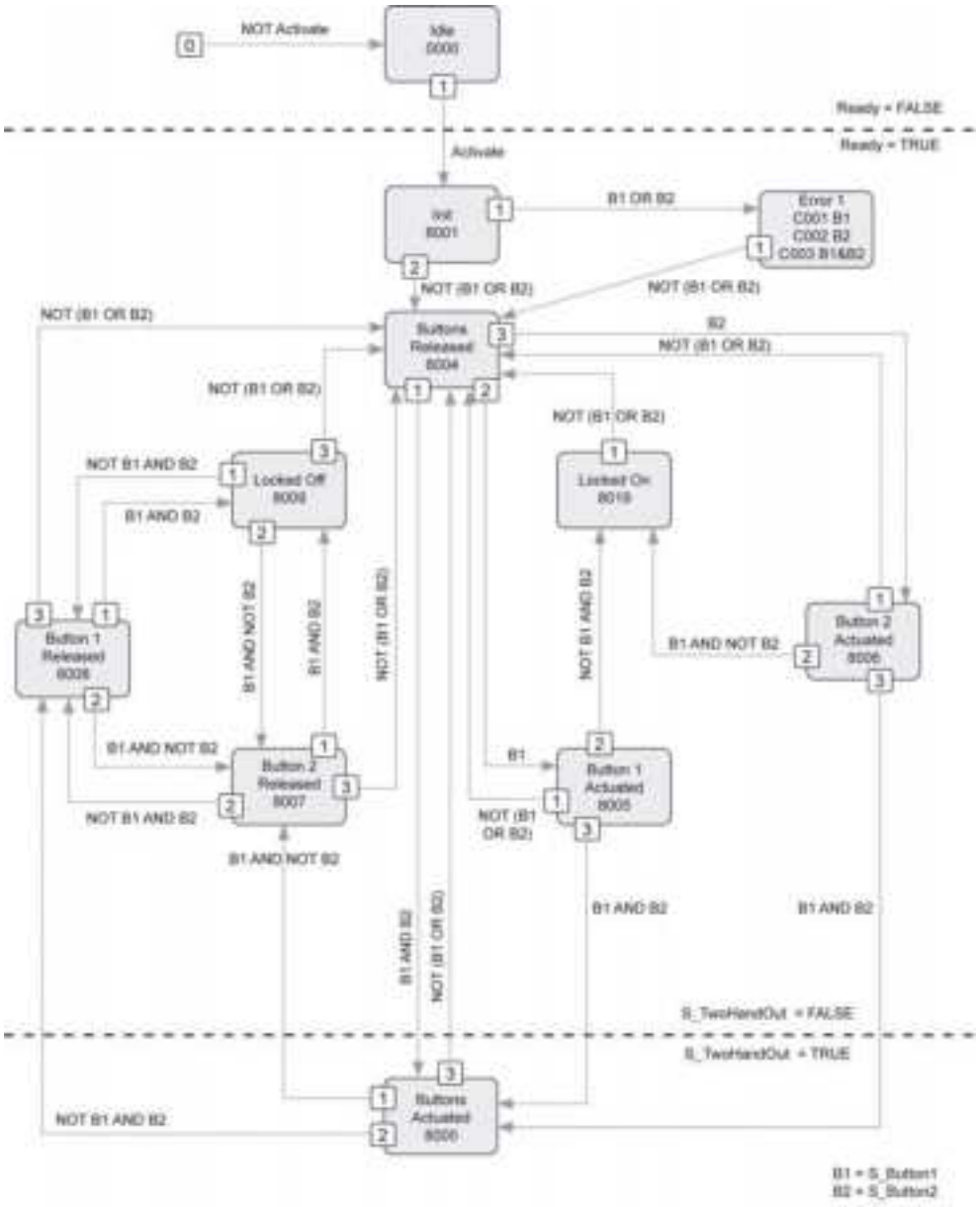
13.5.23.2 Error behavior

In the event of a fault, the S_TwoHandOut output will be set to FALSE and will remain in its safe state. The error state will be exited if both buttons are released (set to FALSE).

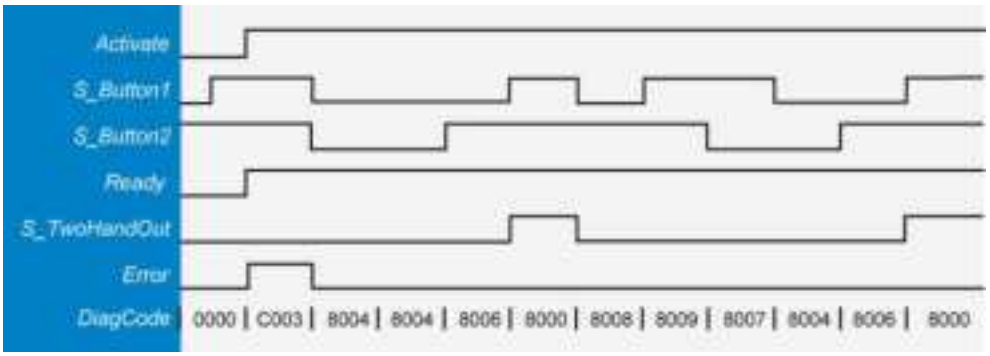
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error B1	S_Button1 has a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error B2	S_Button2 has a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error B1&B2	The S_Button1 and S_Button2 signals have a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons were actuated correctly. The safety-relevant output will be activated. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active but in the init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No buttons are being pressed. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8005	Button 1 Actuated	Button 1 is pressed. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8006	Button 2 Actuated	Button 2 is pressed. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of TRUE and S_Button2 has a value of FALSE. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	Status name	Status description and output setting
8008	Button 1 Released	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of FALSE and S_Button2 has a value of TRUE. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8009	Locked Off	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of TRUE and S_Button2 will have a value of TRUE again after the safety-relevant output is deactivated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect button actuation. Wait for both buttons to be released. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

13.5.23.3 Status Diagram SF_TwoHandControlTypeII

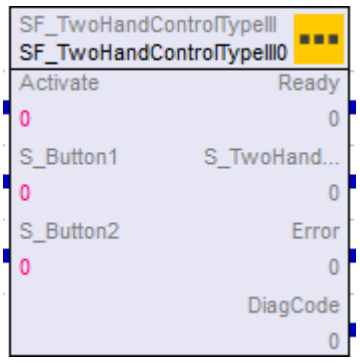


13.5.23.4 Time Diagram SF_TwoHandControlTypeII



13.5.24 SF_TwoHandControlTypeIII function

This function block offers two-hand control functionality (see EN 574, section 4 type III). A fixed time difference of 500 ms is specified.



Input variables	
Activate	Activates the function block (BOOL)
S_Button1	Input 1 (SAFEBOOL)
S_Button2	Input 2 (SAFEBOOL)
Output variables	
Ready	If TRUE: The function block will be activated and the outputs will be valid. (BOOL)
S_TwoHandOut	Safety-relevant output signal (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

This function block offers the two-hand control functionality in accordance with EN 574, section 4 type II. If S_Button1 and S_Button2 are set to TRUE within 500 ms, the output S_TwoHandOut is also set to TRUE. If one of the S_Buttons is subsequently set to FALSE, S_TwoHandOut is also set to FALSE. Before S_TwoHandOut can be set to TRUE again, both S_Button1 and S_Button2 must first be set to FALSE.If one of the S_Button inputs is set from TRUE to FALSE and the other is set from FALSE to TRUE at the same time, an error will be triggered.

13.5.24.1 Error detection

If one or both buttons are already activated when the function block is activated, this will result in a function block error condition. The function block will detect instances in which the deviation between the input signals exceeds 500 ms.

13.5.24.2 Error behavior

In the event of a fault, the S_TwoHandOut output will be set to FALSE and will remain in its safe state. The error state will be exited if both buttons are released (set to FALSE).

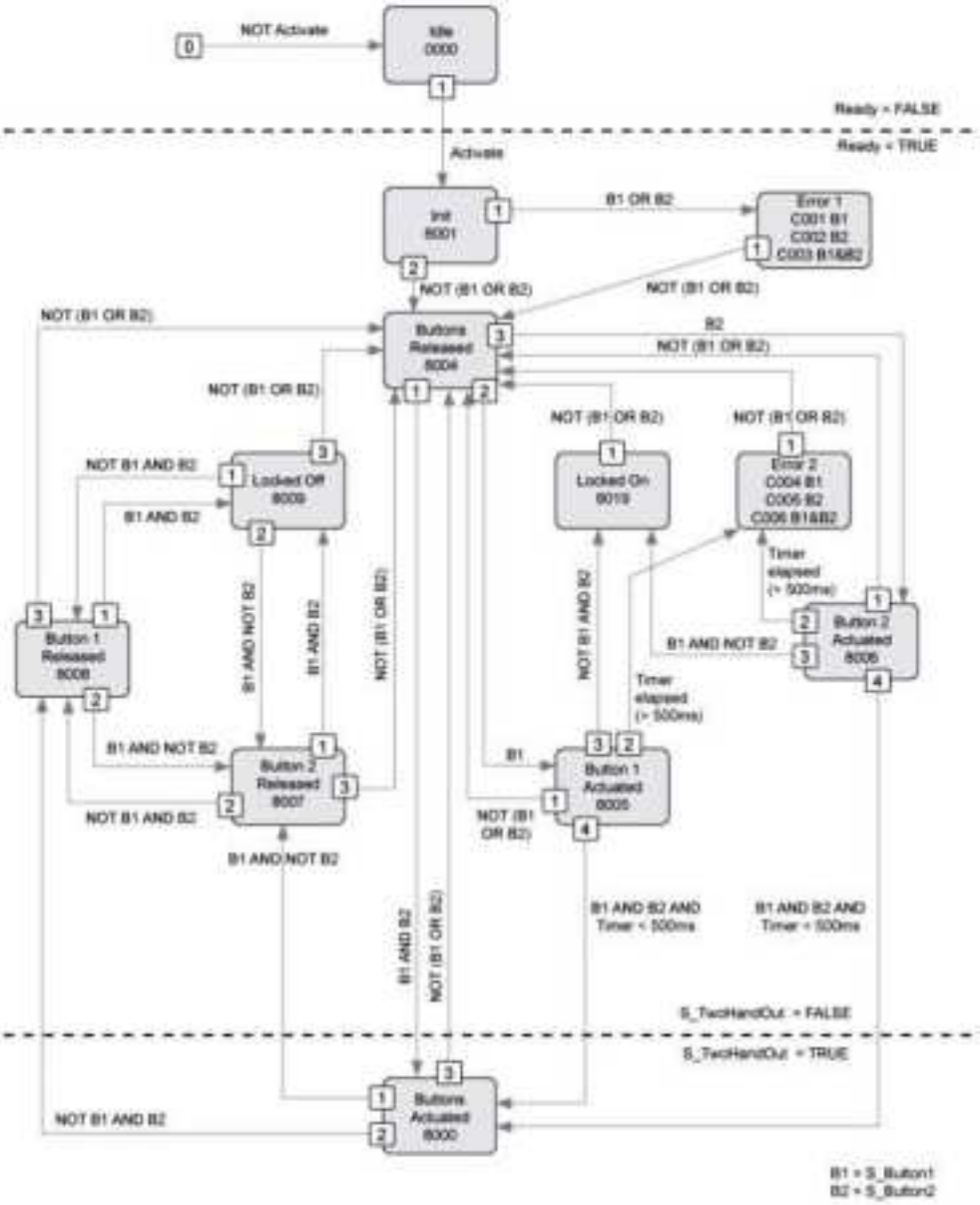
DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error 1 B1	S_Button1 has a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C002	Error 1 B2	S_Button2 has a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C003	Error 1 B1&B2	Signals S_Button1 and S_Button2 have a value of TRUE when the function block is activated. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C004	Error 2 B1	S_Button1 = FALSE, S_Button 2 = TRUE and 500 ms elapsed. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C005	Error 2 B2	S_Button1 = TRUE, S_Button 2 = FALSE and 500 ms elapsed. Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
C006	Error 2 B1&B2	S_Button1 = TRUE, S_Button 2 = TRUE and 500 ms elapsed Ready = TRUE Error = TRUE S_TwoHandOut = FALSE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_TwoHandOut = FALSE
8000	Buttons Actuated	Both buttons were actuated correctly. The safety-relevant output will be activated. Ready = TRUE Error = FALSE S_TwoHandOut = TRUE
8001	Init	Function block is active but in the init state. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8004	Buttons Released	No buttons actuated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8005	Button 1 Actuated	Button 1 is actuated. Start monitoring time. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

DiagCode	Status name	Status description and output setting
8006	Button 2 Actuated	Button 2 is actuated. Start monitoring time. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8007	Button 2 Released	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of TRUE and S_Button2 has a value of FALSE. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8008	Button 1 Released	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of FALSE and S_Button2 has a value of TRUE. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8009	Locked Off	The safety-relevant output was activated and is deactivated again. In this state, S_Button1 has a value of TRUE and S_Button2 will have a value of TRUE again after the safety-relevant output is deactivated. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE
8019	Locked On	Incorrect button actuation. Wait for both buttons to be released. Ready = TRUE Error = FALSE S_TwoHandOut = FALSE

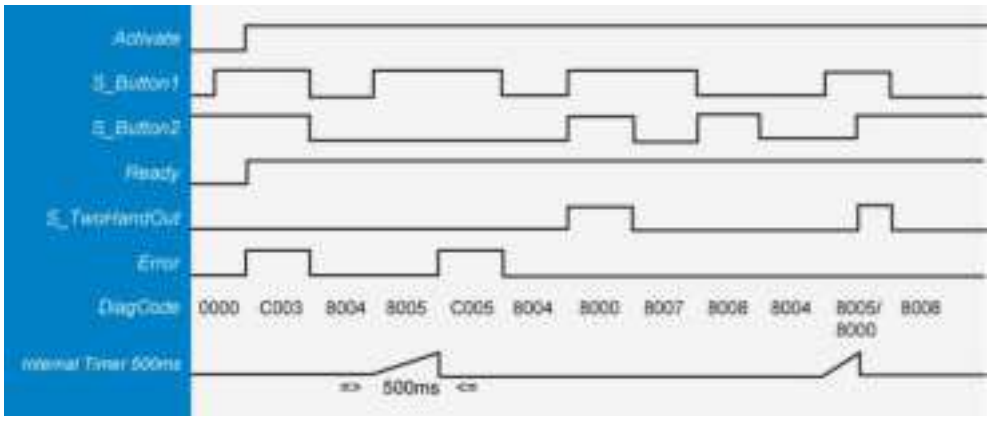
13.5.24.3 Characteristics

In state 8005 and state 8006, the first thing to be checked will be whether both S_Button inputs have a value of SAFE_FALSE, in which case the system will branch to state 8004. The reason for this is that an error state is not defined if the monitoring time has elapsed and both S_Button inputs have a value of SAFE_FALSE.

13.5.24.4 Status Diagram SF_TwoHandControlTypeIII



13.5.24.5 Time Diagram SF_TwoHandControlTypeIII

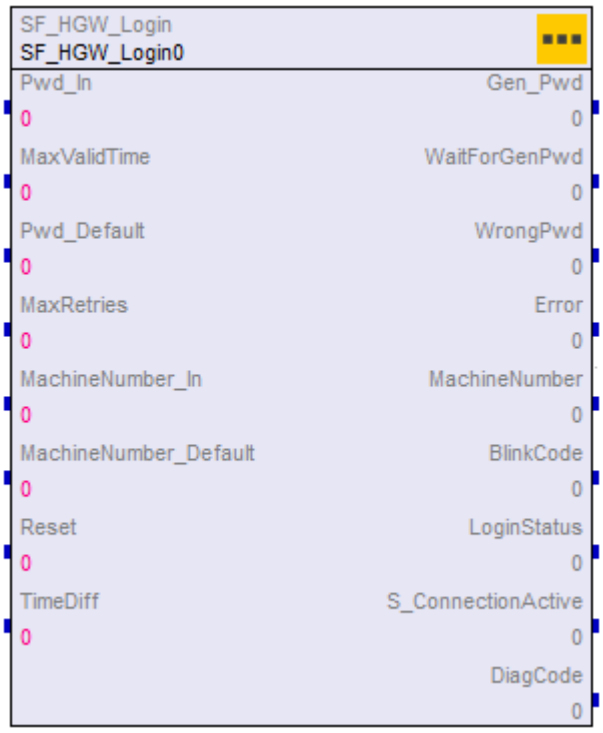


13.5.25 SF_HGW_Login

You can use this function block to connect an HGW module to a Safety CPU (please note that this function block can only be used together with an HGW module). The function block contains two separate mechanisms for connecting the module.

In the first area, you can assign a unique machine number for the system. This machine number can either be set with a constant or with a password-protected login mechanism from the visualization.

The second area is used to set up a safe connection between an HGW module and a Safety CPU.



Input variables	
Pwd_In	Entered password (DINT); a change will trigger the change operation for changing the machine number.
MaxValidTime	Maximum time difference (DINT) for completing the login operation used to change the machine number (from 8001 to 8004). 0 to a maximum of eight hours. 0 means that the time check is disabled. A constant must be used to write this value.
Pwd_Default	Specified password (DINT) for starting the login operation for changing the machine number. A constant must be used to write this value.
MaxRetries	Maximum number (DINT) of failed attempts for the confirmation password.
MachineNumber_In	Entered machine number (DINT); the value will be used to change the machine number. State 8004 is required for this purpose
MachineNumber_	Default value for the machine number

Default	1 to 99 A constant must be used to write this value.
Reset	Used to reset (BOOL) the change operation for the machine number
Time_Diff	Maximum time (DINT) for completing the login operation between the HGW module and Safety CPU (from 8001 to 8012). > 0 ms to a maximum of eight hours. A constant must be used to write this value.
Output variables	
Gen_Pwd	Generated confirmation password (DINT).
WaitForGenPwd	Indicates (BOOL) whether the block will wait for a generated password from Gen_Pwd to be entered or for the flashing code that is shown during the HGW module login to be entered. FALSE: There is no valid confirmation password or no valid flashing code. TRUE: The specified confirmation password or the flashing code can be entered.
Wrong_Pwd	Indicates (BOOL) whether there is an expected password at the Pwd_In input. FALSE: Default state TRUE: An incorrect password has been entered.
Error	Error display (BOOL)
MachineNumber	(Retentive) Set machine number (DINT)
BlinkCode	Generated flashing code (DINT) required in order to confirm the HGW module login.
LoginState	Login status (DINT), indicates the current HGW module login status in the Safety CPU. 0 = Not logged in: 1 = Wait for flashing code confirmation; 2 = Logged in, no FSoE connection established yet; 3 = Error; 4 = FSoE connection running; 5 = Default login successful
S_ConnectionActive	Status flag (SAFEBOOL), indicates whether an FSoE connection has been established
DiagCode	Status display (HDINT)

This function block is required in order to assign a unique machine number to the Safety CPU. You will be responsible for ensuring that this machine number is globally unique in the system. This machine number can be used in the HGW module in order to select the Safety CPU to which a connection should be established.

→ As mentioned already, the assigned machine number must be globally unique throughout the whole system.

The first time the Safety CPU is booted up, the value at the MachineNumber_Default input will be stored in the retentive MachineNumber output. This stored machine number can be changed at runtime. To do this, the machine operator will need to change the machine number by using the login mechanism. This login is started by entering the specified password. The function block will then generate a password, at the Gen_Pwd output, that the user will need to confirm.After a successful confirmation, the MachineNumber_In input can be used to change the machine number.

→ Please note that the function block will not apply a machine number of 0 at the MachineNumber_In input, since this is an invalid value.

If the HGW module sends a machine number to the Safety CPU and the function block recognizes it as being correct, preparations will be made for establishing an FSoE connection between the HGW module and the Safety CPU. The function block

will generate a flashing code at the intended output, and this flashing code must be confirmed on the HGW module. As soon as the flashing code has been confirmed, an FSoE connection will be established between the HGW module and the Safety CPU.When a valid FSoE connection is established, this will be indicated at the ConnectionActive output.

The inputs of the HGW module will only be transmitted to the Safety CPU if there is a valid FSoE connection. The HGW module's inputs must be connected with an SF_Optional_Switch function block. This function block must in turn be switched with the ConnectionActive output on the SF_HGW_Login function block.

13.5.25.1 Error detection

The function block will detect the following error conditions:

- The wrong password being entered in order to confirm the change operation
- Timeout when entering the confirmation password
- A wrong machine number being entered when changing the number
- The HGW module logging in with a wrong machine number
- A wrong flashing code being entered
- HGW module login timeout
- The FSoE connection being terminated

If a wrong password is entered in state 8000 or 8010, this will not trigger an error. Instead, this will simply be signaled with a value of TRUE at the WrongPwd output.

13.5.25.2 Error behavior

The DiagCode output will signal the corresponding error code and the Error output will be set to TRUE. In order to exit the error condition, it must be reset with a rising edge at the Reset input. The Pwd_In input should be set to "0" in this case.

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	MaxValidTime_exceed	A MaxValidTime timeout was detected. Can occur from states 8002 & 8003. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE

DiagCode	Status name	Status description and output setting
C002	Too_many_retries	The generated password was entered incorrectly too many times. Can occur from states 8002 & 8003. WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
C003	Invalid_ MachineNumber_ entered	An invalid machine number has been applied to MachineNumber_In. Can occur from state 8004. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
C010	Wrong_MachNum_ for_login	The machine number for the login and the set machine number do not match. Can occur from state 8001. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 3 ConnectionActive = SAFEBOOL_FALSE
C011	TimeDiff_exceed	A TimeDiff timeout was detected. Can occur from states 8010 & 8011. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 3 ConnectionActive = SAFEBOOL_FALSE
C012	Wrong_blinkcode_ entered	A wrong flashing code was entered. Can occur from state 8010. WaitForGenPwd = FALSE WrongPwd = TRUE Error = TRUE LoginState = 3 ConnectionActive = SAFEBOOL_FALSE
C013	FSoE_connection_ broken	The FSoE connection was interrupted. Can occur from state 8012. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 3 ConnectionActive = SAFEBOOL_FALSE
C020	Wrong_ MachineNumber_ Detected	An invalid machine number was detected at the output. WaitForGenPwd = FALSE WrongPwd = FALSE Error = TRUE LoginState = 3 ConnectionActive = SAFEBOOL_FALSE
FB-specific status codes (no error):		

DiagCode	Status name	Status description and output setting
0000	Init	The function block was activated (initial state). WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
8000	Init	The function block was activated. Check whether a machine number has been set. If no machine number has been set, the MachineNumber output will be set to the value at the MachineNumber_Default input in this state. WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
8001	Wait_for_Operation	Check whether a password that is the same as Pwd_Default at Pwd_In is present if Reset has a value of FALSE. Check whether an FSoE connection should be established. WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
8002	Login_Request_ Default	The default password was applied at Pwd_In. The FUB will generate a confirmation password that will be applied at Gen_Pwd. To confirm the login operation, the user will need to apply this password at Pwd_In. WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
8003	Login_Request_ Default_Retry	The confirmation password was entered incorrectly. Users can try to enter the password as many times as defined with the value at Max_Retry. WaitForGenPwd = TRUE WrongPwd = TRUE Error = FALSE LoginState = 0 ConnectionActive = SAFEBOOL_FALSE
8004	Logged_In_Default	The generated password was confirmed. The user will now be able to set the value of the MachineNumber output with MachineNumber_In. WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 5 ConnectionActive = SAFEBOOL_FALSE

DiagCode	Status name	Status description and output setting
8010	Login_Request_ Machine	The entered machine number was recognized on the HGW and the comparison with the MachineNumber output was successful. A flashing code that the user needs to confirm will be generated. WaitForGenPwd = TRUE WrongPwd = FALSE Error = FALSE LoginState = 1 ConnectionActive = SAFEBOOL_FALSE
8011	Login_Request_ Blinkcode_ACK	The user has confirmed the flashing code. A successful login has been carried out. The HGW will establish an FSoE connection to the Safety CPU. WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 2 ConnectionActive = SAFEBOOL_FALSE
8012	Logged_In	The FSoE connection was established successfully. The safe data will now be transferred from the HGW to the Safety CPU. WaitForGenPwd = FALSE WrongPwd = FALSE Error = FALSE LoginState = 4 ConnectionActive = SAFEBOOL_TRUE

States 8002, 8003, and 8004 can be aborted by setting Reset to TRUE. The function block will switch to state 8001.

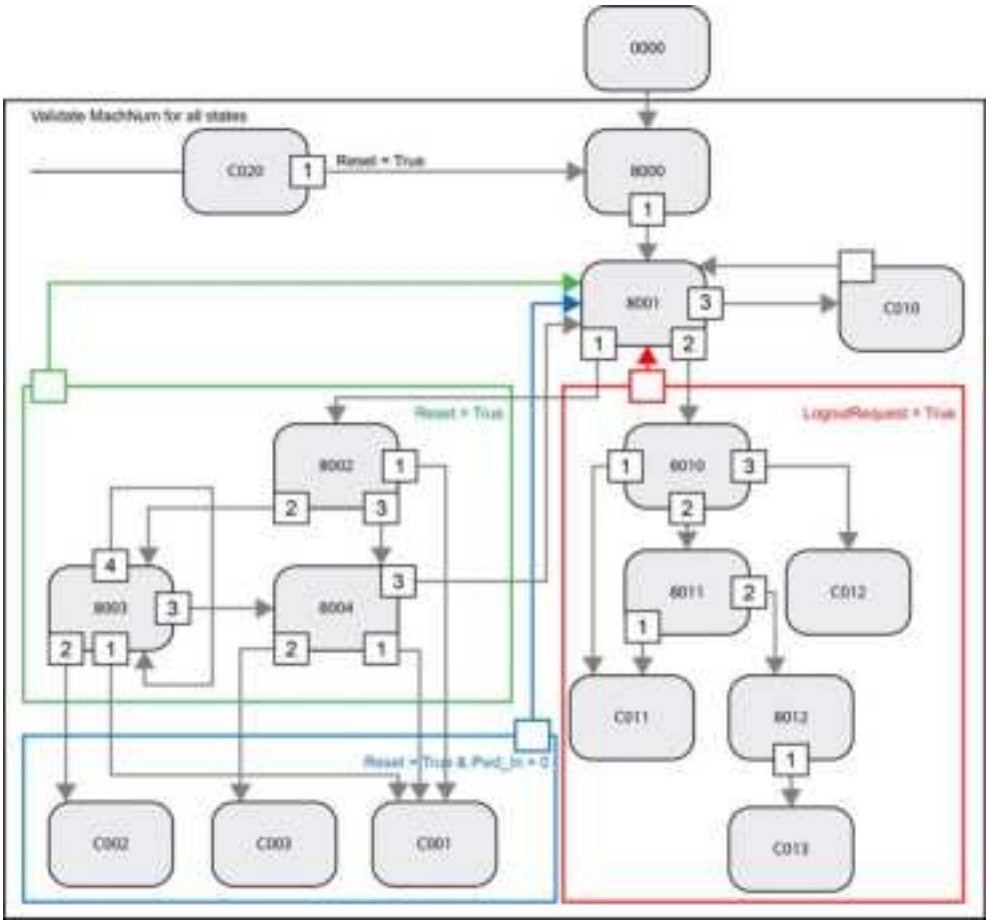
Error conditions C001, C002, and C003 can be reset by setting reset to TRUE and Pwd_In to 0. The function block will switch to state 8001.

States 8010, 8011, and 8012, as well as error conditions C010, C011, C012, and C013, can be aborted with the HGW module's Logout_Request. The function block will switch to state 8001.

Error condition C020 can be generated from any state, with the exception of the init states. The error can be reset with Reset and Pwd_In = 0.

Resetting the error will result in a machine number change in which the machine number will be set to the MachineNumber_Default value!

13.5.25.3 SF_HGW_Login Status Diagram



13.5.26 Complex numeric function blocks

→ " SF_LimitComparator", page 338
→ " SF_Numeric_Selector", page 343
→ " SF_RampMonitor", page 347
→ " SF_DirectionMonitor", page 351
→ " SF_SkewMonitor", page 355

See also

→ "Standard numerical function blocks", page 172

13.5.26.1 SF_LimitComparator

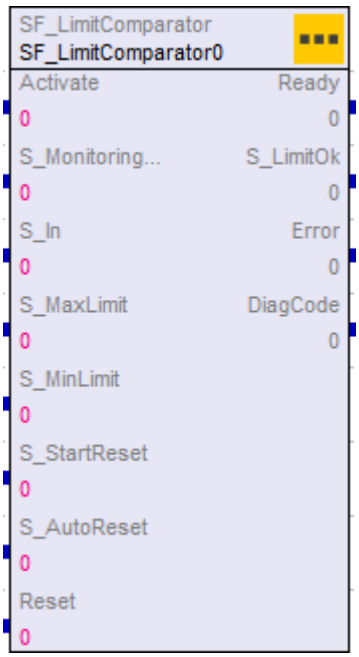
Checks the input signal for upper and lower limit. The upper limit must be greater than the lower limit.

The function block's monitoring function can be enabled or disabled with S_MonitoringOn. If S_MonitoringOn has a value of FALSE, the S_LimitOk output will be set to

TRUE. If S_MonitoringOn has a value of TRUE and S_In is higher than S_MaxLimit or lower than S_MinLimit, the S_LimitOk output will be set to FALSE.

After the upper limit is exceeded or the lower limit is fallen below, you will only be able to switch back to the monitoring state by carrying out a reset (condition: Eingang S_In within the limits). The reset behavior will depend on the S_AutoReset and Reset inputs.

If S_AutoReset has a value of TRUE, then resetting will be done automatically. If S_AutoReset has a value of FALSE, there must be a rising edge at the Reset input in order for the activation to be reset.



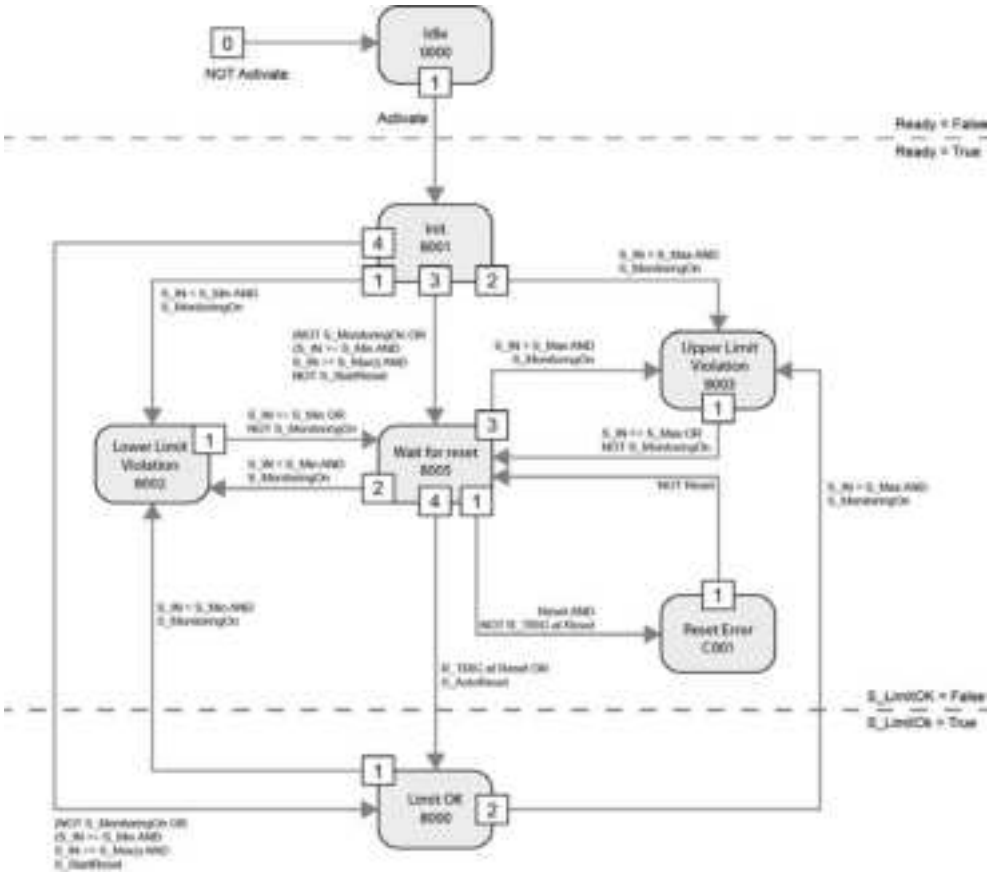
Input variables	
Activate	Activates the function block. (BOOL)
S_MonitoringOn	Turns monitoring on. FALSE: No monitoring. No rule violations will be evaluated. TRUE: Monitoring function is on. (SAFEBOOL)
S_In	Input signal that should be monitored. (SAFEDINT)
S_MaxLimit	Upper limit (SAFEDINT)
S_MinLimit	Lower limit (SAFEDINT)
S_StartReset	Determines the switch-on behavior: FALSE: Manual reset TRUE: Automatic reset after the function block is started. (SAFEBOOL)
S_AutoReset	Determines the reset behavior: FALSE: Manual reset TRUE: Automatic reset (SAFEBOOL)

Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)
S_LimitOk	Whether the input value is within the limits when monitoring is on. FALSE: Input value outside of limits. TRUE: Input within limits or monitoring is off (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

Error behavior

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal in state 8005 Ready = TRUE S_LimitOk = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_LimitOk = FALSE Error = FALSE
8001	Init	The function block has detected an activation and the status is now activated. Ready = TRUE S_LimitOk = FALSE Error = FALSE
8000	Limit Ok	Monitoring status. If S_MonitoringOn has a value of TRUE, the input will be checked in terms of both limits. If S_MonitoringOn has a value of FALSE, there will be no limit check. Ready = TRUE S_LimitOk = TRUE Error = FALSE
8002	Lower Limit Violation	The input value has fallen below the lower limit. Ready = TRUE S_LimitOk = FALSE Error = FALSE
8003	Upper Limit Violation	The input value has exceeded the upper limit. Ready = TRUE S_LimitOk = FALSE Error = FALSE
8005	Wait for Reset	The activation is TRUE. The input value is within the two limits. Check whether S_AutoReset has a value of TRUE or wait for a rising edge at Reset. Ready = TRUE S_LimitOk = FALSE Error = FALSE

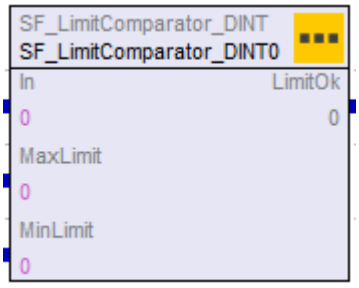
Status Diagram SF_LimitComparator



13.5.26.2 SF_LimitComparator_Dint

Checks the input signal for upper and lower limit.

If the "In" input > "MaxLimit" or "In" < "MinLimit", the "LimitOk" output will be set to FALSE.

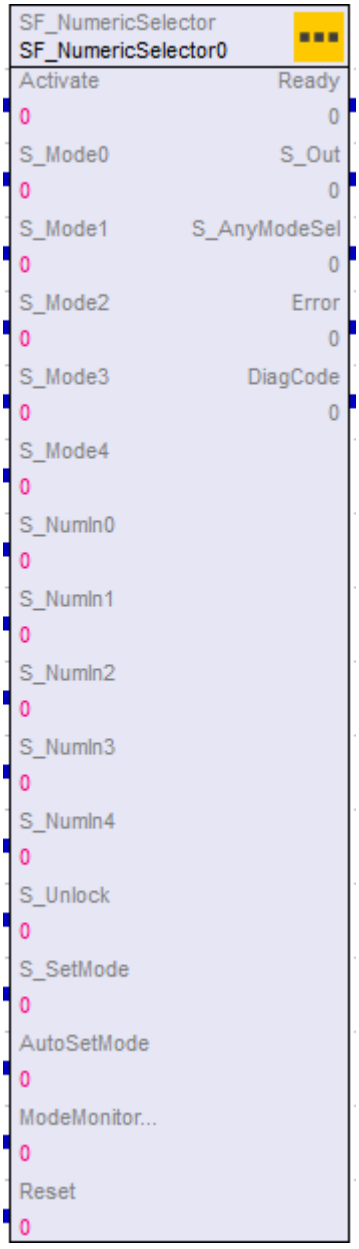


Input variables	
In	Input (DINT)
MaxLimit	Upper limit for the input value
MinLimit	Lower limit for the input value
Output variables	
LimitOk	TRUE if the input value falls within the specified limits (BOOL)

13.5.26.3 SF_Numeric_Selector

Like→ " SF_ModeSelector function", page 261, but with SafeDint values. You can use the S_ModeX SafeBool inputs to select a SafeDint input that will be connected through to the S_Out output.

In the event of an error (no inputs or multiple inputs selected), the S_Out output will always be set to 0. In this case, you must additionally incorporate the S_AnyModeSel output into the user logic in order to determine whether S_Out is valid.



Input variables	
Activate	Activates the function block (BOOL)

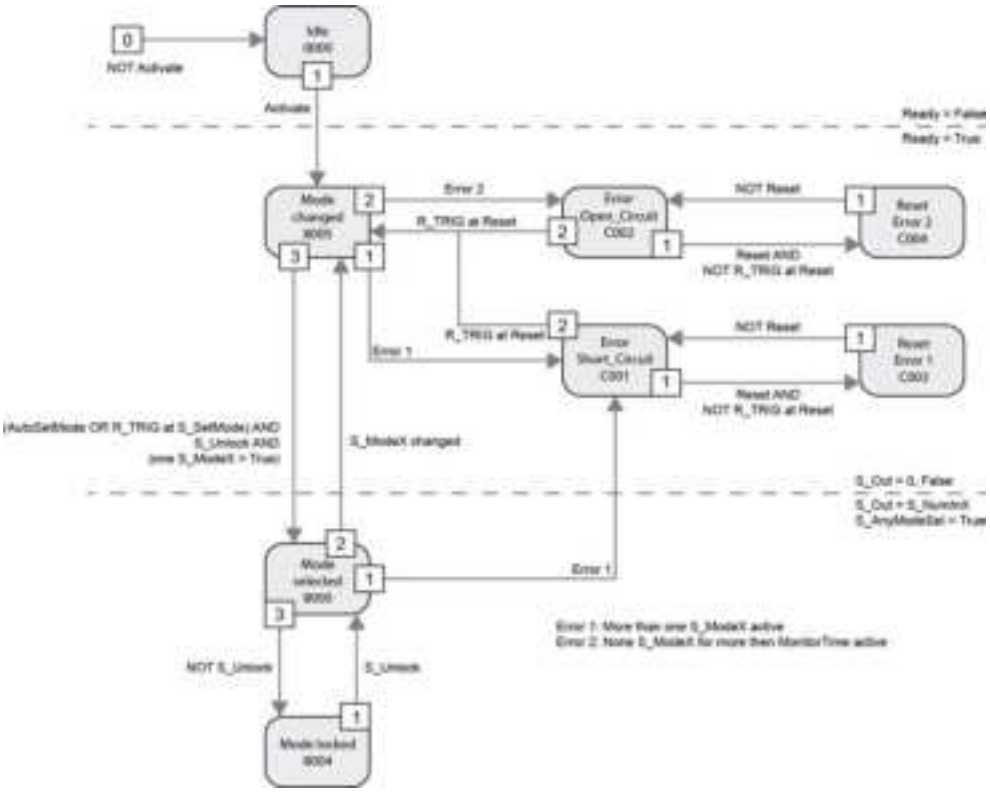
S_ Mode0 S_ Mode1 S_ Mode2 S_ Mode3 S_ Mode4	Applies to all: Input X from operating mode switch FALSE: Mode X not selected. TRUE: Mode X selected by the operator. (SAFEBOOL)
S_NumIn0 S_NumIn1 S_NumIn2 S_NumIn3 S_NumIn4	Applies to all: If S_NumInX is active, S_Out will be set to its value. (SAFEDINT)
S_Unlock	FALSE: Locks the operating mode. A change at the S_ModeX inputs will not result in a change at the S_ModeXSel outputs. (SAFEBOOL)
S_SetMode	TRUE: Confirmation for setting a new operating mode if there are valid S_ModeX inputs. (SAFEBOOL)
AutoSetMode	TRUE: Automatic confirmation for setting a new operating mode if there are valid S_ ModeX inputs. (BOOL)
ModeMonitorTime	Maximum operating time in which all S_ModeX inputs are FALSE. (DINT)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)
S_Out	S_NumInX value of the selected mode. (SAFEDINT)
S_AnyModeSel	TRUE: An operating mode has been selected. (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

Error behavior

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Error Short-circuit	The function block has detected that two or more S_ModeX inputs have a value of TRUE. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE S_Out = 0
C002	Error Open-circuit	The function block has detected that all S_ModeX inputs have a value of FALSE: The time after a falling S_ModeX edge exceeds the ModeMonitorTime. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE S_Out = 0
C003	Reset Error 1	Static reset signal was detected in state C001. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE S_Out = 0

DiagCode	Status name	Status description and output setting
C004	Reset Error 2	Static reset signal was detected in state C002. Ready = TRUE Error = TRUE S_AnyModeSel = FALSE S_Out = 0
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE Error = FALSE S_AnyModeSel = FALSE S_Out = 0
8005	ModeChanged	Ready = TRUE Error = FALSE S_AnyModeSel = FALSE S_Out = 0
8000	ModeSelected	Valid mode selection not yet locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_Out = Selected Input Signal.
8004	ModeLocked	Valid mode selection is locked. Ready = TRUE Error = FALSE S_AnyModeSel = TRUE S_Out = Selected Input Signal.

Status Diagram SF_NumericSelector



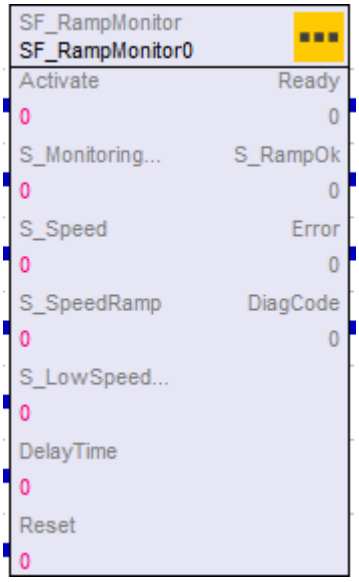
13.5.26.4 SF_RampMonitor

RampMonitor monitors the speed reduction in the event of an emergency stop. A rising edge at S_MonitoringOn will cause the current speed to be applied as the initial speed for the monitoring function.

If S_MonitoringOn TRUE and the delay has elapsed, the speed must decrease by a value of S_SpeedRamp (the speed must be below the computed ramp). The delay is required for the mechanical components used to trigger the deceleration operation.

Monitoring will be activated if S_MonitoringOn has a value of TRUE and will not be terminated if the speed is lower than S_LowSpeedThreshold (S_LowSpeedThreshold is used as a lower value for the computer ramp). Speeds lower than S_LowSpeedThreshold mean that the machine has stopped.

Errors can be reset with a rising edge at the Reset input.



Input variables	
Activate	Activates the function block. (BOOL)
S_MonitoringOn	Activates monitoring for the S_Speed input. FALSE: No monitoring. Rule violations will not be evaluated. TRUE: Monitoring function is on. (SAFEBOOL)
S_Speed	Current speed from the XS-322-2INC module (SAFEDINT)
S_SpeedRamp	Factor for calculating the curve for the computed speed (speed/[ms]). (SAFEDINT)
S_LowSpeedThreshold	Limit for computed speed. (SAFEDINT)
DelayTime	Delay until monitoring starts after S_MonitoringOn = TRUE. (DINT)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)

S_RampOk	Result of the monitoring function: FALSE: If S_MonitoringOn TRUE and S_Speed higher than calculated value vx. TRUE: If the speed is lower than the calculated value or S_MonitoringOn has a value of FALSE (monitoring deactivated). S_RampOk becomes invalid if one of the Safety inputs is invalid. (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

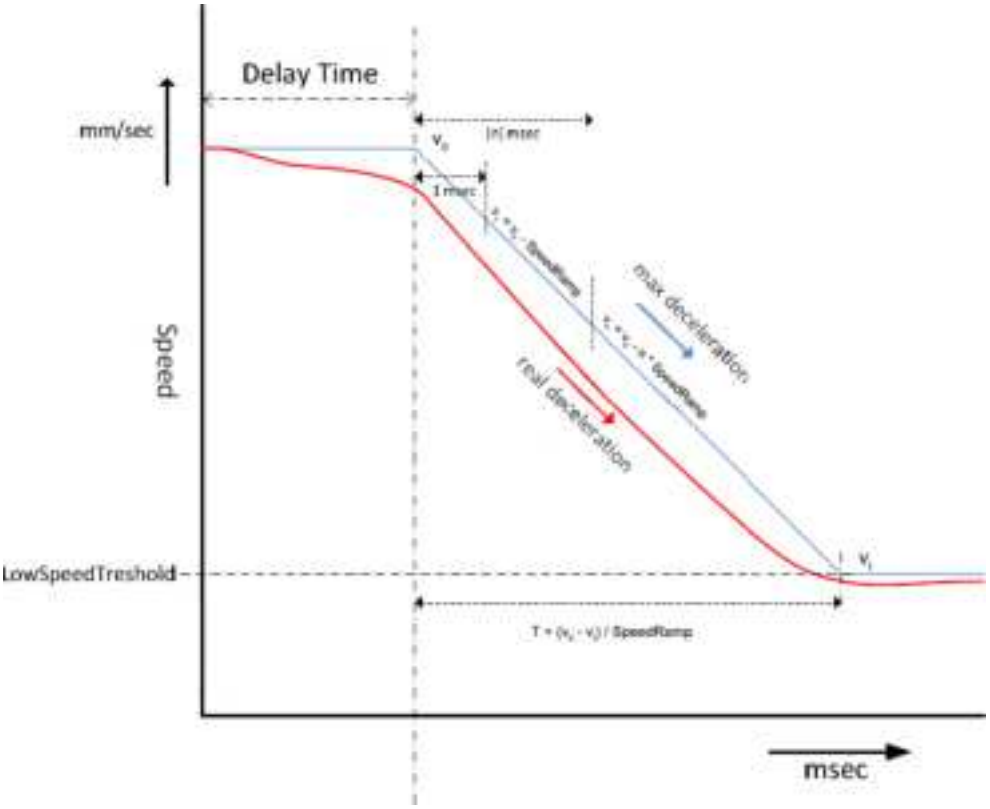
Formula for calculating the speed curve

$$v_x = v_0 - SpeedRamp * (x - T_d)$$

$$T_d = T_0 + DelayTime$$

$$x > DelayTime$$

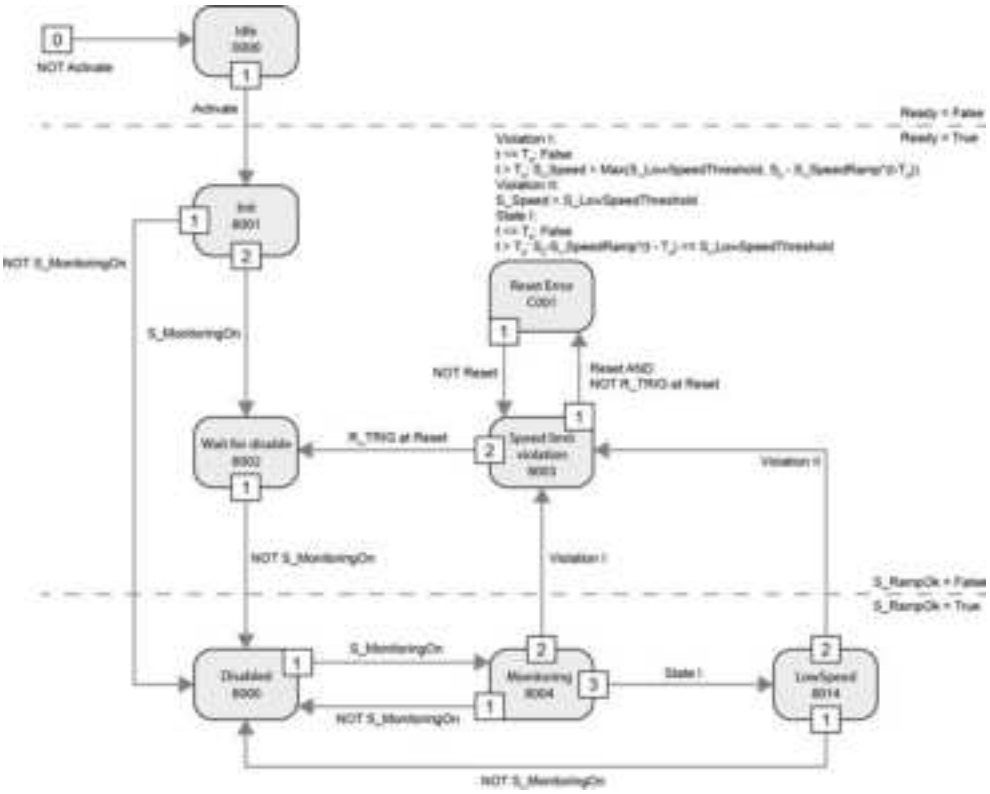
$$v_x > S_LowSpeedThreshold$$



Error behavior

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal in state 8003 Ready = TRUE S_RampOk = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_RampOk = FALSE Error = FALSE
8001	Init	The function block has detected an activation and the status is now activated. Ready = TRUE S_RampOk = FALSE Error = FALSE
8002	Wait for Disable	Waiting for monitoring to be disabled. Ready = TRUE S_RampOk = FALSE Error = FALSE
8003	Speed Limit Violation	Speed at the input is higher than the computed speed. Ready = TRUE S_RampOk = FALSE Error = FALSE
8000	Disabled	Monitoring is deactivated. The input value is not being checked. Ready = TRUE S_RampOk = TRUE Error = FALSE
8004	Monitoring	Monitoring is activated. After the delay time elapses, the input will be monitored based on computed values. Ready = TRUE S_RampOk = TRUE Error = FALSE
8014	Low Speed	Monitoring is activated. The computed values have fallen below the value of S_LowSpeedThreshold. Input monitoring is now only being carried out in regard to this limit. Ready = TRUE S_RampOk = TRUE Error = FALSE

Status Diagram SF_RampMonitor



13.5.26.5 SF_DirectionMonitor

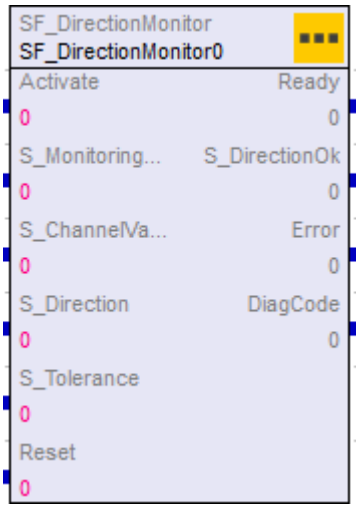
The S_Direction input determines the direction in which the input value should be checked (positive travel: S_Direction = FALSE; negative travel: S_Direction = TRUE).

Monitoring starts when there is a rising edge at S_MonitoringOn and sets the current value as the maximum value (or minimum value if S_Direction = TRUE).

During monitoring, the highest value (or the lowest value if S_Direction = TRUE) will be stored.

S_Tolerance is the maximum difference between the current value and the stored maximum value (minimum value if S_Direction = TRUE) in the wrong direction. S_DirectionOk will become FALSE if S_MonitoringOn is TRUE and the tolerance is fallen below or exceeded.

Errors can be reset with a rising edge at the Reset input.



Input variables	
Activate	Activates the function block. (BOOL)
S_MonitoringOn	Activates the monitoring function. FALSE: No monitoring. Rule violations will not be evaluated. TRUE: Monitoring function is on. (SAFEBOOL)
S_ChannelValue	Current value from the XS-322-2INC module. (SAFEDINT)
S_Direction	Monitoring direction in which the values are allowed to change. (SAFEDINT) FALSE: positive motion TRUE: negative motion
S_Tolerance	Maximum value in wrong direction. (SAFEDINT)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)
S_DirectionOk	Result of the monitoring function: FALSE: If S_MonitoringOn is TRUE and the input value in the wrong direction of movement is greater than the tolerance when compared to the last determined value in the correct dir-

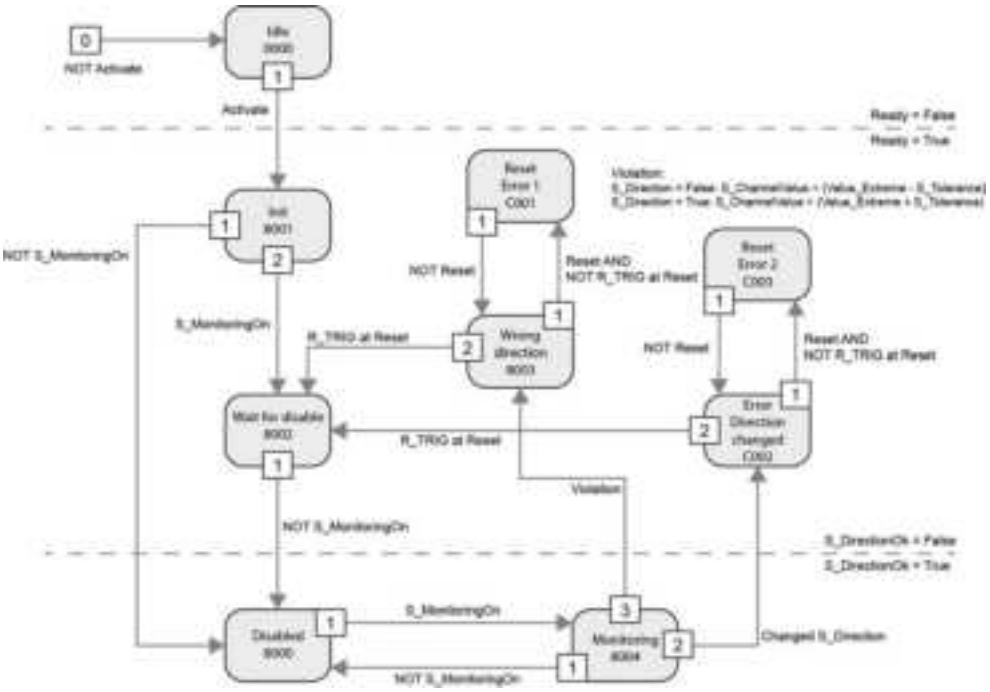
	ection. TRUE: If the direction is correct or not activated. S_DirectionOk will become invalid if one of the Safety inputs is not valid. (SAFEBOOL)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)

S_ChannelValue will be interpreted as an unsigned value. If the value falls outside the value range, this will be ignored. Accordingly, the function block can also be used for incremental encoders.

Error behavior

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Reset Error 1	Static reset signal in state 8003 Ready = TRUE S_DirectionOk = FALSE Error = TRUE
C002	DirectionChanged	Direction signal S_Direction changed during monitoring. Ready = TRUE S_DirectionOk = FALSE Error = TRUE
C003	Reset Error 2	Static reset signal in state C002 Ready = TRUE S_DirectionOk = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_DirectionOk = FALSE Error = FALSE
8001	Init	The function block has detected an activation and the status is now activated. Ready = TRUE S_DirectionOk = FALSE Error = FALSE
8002	Wait for Disable	Waiting for monitoring to be disabled. Ready = TRUE S_DirectionOk = FALSE Error = FALSE
8003	Wrong Direction	The input value has exceeded the specified S_Tolerance limit in the wrong direction. Ready = TRUE S_DirectionOk = FALSE Error = FALSE
8000	Disabled	The input value is not being checked. Ready = TRUE S_DirectionOk = TRUE Error = FALSE
8004	Monitoring	Monitoring is enabled. Input value S_ChannelValue is being monitored, so that only S_Tolerance in the opposite direction is allowed. Ready = TRUE S_DirectionOk = TRUE Error = FALSE

Status Diagram SF_DirectionMonitor



13.5.26.6 SF_SkewMonitor

A rising edge at the S_Calibrate input will start the calibration process and set S_CalibrationOk to FALSE. For the calibration, the S_Monitoring input must be set to FALSE. During this calibration, the difference between input signals S_ChnValue1 and S_ChnValue2 will be checked with a comparison to S_MaxOffset. If the difference is greater than S_MaxOffset, the S_SkewOk output will be set to FALSE.

A falling edge at S_Calibrate will apply the current difference and store it retentively so that the S_CalibrationOk output is set to TRUE as long as the value does not exceed S_MaxOffset. If it does, S_CalibrationOk will keep its value of FALSE.

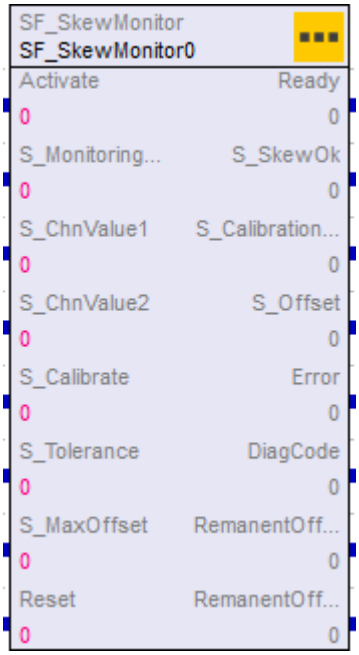
S_CalibrationOk will only have a value of TRUE if the stored offset is valid after an error-free calibration.

If S_Calibrate = FALSE and S_MonitoringOn = TRUE, monitoring will be carried out in conformity with the following rule:

ABS(S_ChnValue1 - S_ChnValue2 – S_Offset) > ABS(S_Tolerance)

If this monitoring rule is met, S_SkewOk will be set to FALSE.

Errors can be reset with a rising edge at the Reset input.



Input variables	
Activate	Activates the function block. (BOOL)
S_MonitoringOn	Activates the monitoring function. FALSE: No monitoring. Rule violations will not be evaluated. TRUE: Monitoring function is on. (SAFEBOOL).
S_ChnValue1	Filtered value of channel 1 of the SSI module. (SAFEDINT)

S_ChnValue2	Filtered value of channel 2 of the SSI module. (SAFEDINT)
S_Calibrate	TRUE: Starts the calibration and sets the S_CalibrationOk output to FALSE (SAFEBOOL)
S_Tolerance	Maximum deviation between channel values during monitoring. (SAFEDINT)
S_MaxOffset	Maximum deviation between channel values during calibration. (SAFEDINT)
Reset	Reset (BOOL)
Output variables	
Ready	If TRUE: Indicates that the function block is activated and the output results are valid. (BOOL)
S_SkewOk	Result of the monitoring function: FALSE: The difference between S_ChnValue1 and S_ChnValue2 exceeds the tolerance value (absolute values). TRUE: Both values fall within the current tolerance or are inactive. S_SkewOk will become invalid if one of the Safety inputs is invalid. (SAFEBOOL)
S_CalibrationOk	Indicates whether the function block is calibrated. FALSE: The calibration value is invalid or the calibration has been started. TRUE: The function block is calibrated (and the retentive values are valid) (SAFEBOOL)
S_Offset	Offset value If the S_CalibrationOk output has a value of FALSE, the offset is invalid. During calibration, the value will be the current difference. (SAFEDINT)
Error	Error display (BOOL)
DiagCode	Status display (HDINT)
RemanentOffset1	Retentively stored offset value (DINT)
RemanentOffset2	Retentively stored inverted offset value (DINT)

S_MaxOffset is used for the calibration. S_Tolerance is used for regular monitoring.

The calibration value (offset) will be retentively stored at the RemanentOffset1 and RemanentOffset2 outputs. In order to determine whether the stored value is valid (memory not initialized), the storage operation will be carried out twice: once with the same value and the second time with the ones' complement. The S_CalibrationOk output will only be TRUE if both values match. Please note that the retentive outputs will not be written to during the calibration process.They are instead only written to at the beginning (in order to set the value to invalid) and at the end of the calibration in order to store the new offset value.

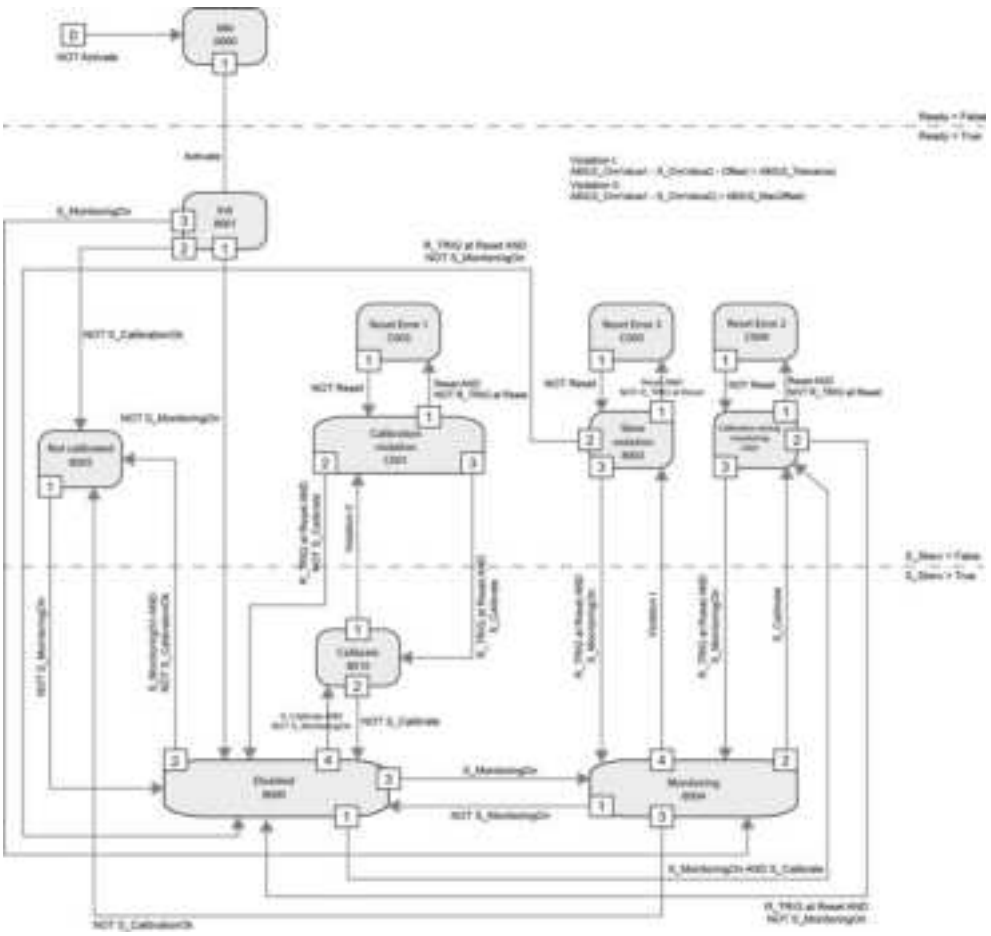
Regular monitoring is turned off during the calibration.

Error behavior

DiagCode	Status name	Status description and output setting
FB-specific error codes		
C001	Calibration Violation	The difference between the two input values exceeded the maximum value of S_MAXOFFSET during the calibration. Ready = TRUE S_SkewOk = FALSE Error = TRUE
C002	Calibration during Monitoring	The calibration was activated during monitoring (S_MonitoringOn = TRUE and S_Calibrate = TRUE). Ready = TRUE S_SkewOk = FALSE Error = TRUE
C003	Reset Error 1	Static reset signal in state C001 Ready = TRUE S_SkewOk = FALSE Error = TRUE
C004	Reset Error 2	Static reset signal in state C002 Ready = TRUE S_SkewOk = FALSE Error = TRUE
C005	Reset Error 3	Static reset signal in state 8003 Ready = TRUE S_SkewOk = FALSE Error = TRUE
FB-specific status codes (no error)		
0000	Idle	The function block is not active (initial state). Ready = FALSE S_SkewOk = FALSE Error = FALSE
8001	Init	The function block has detected an activation and the status is now activated. Ready = TRUE S_SkewOk = FALSE Error = FALSE
8003	Skew Violation	The difference between the input values (incl. S_Offset) exceeded tolerance S_Tolerance. Ready = TRUE S_SkewOk = FALSE Error = FALSE
8005	Not Calibrated	Monitoring should be activated, but function block is not calibrated yet. Ready = TRUE S_SkewOk = FALSE Error = FALSE
8000	Disabled	The input value is not being checked. Ready = TRUE S_SkewOk = TRUE Error = FALSE

DiagCode	Status name	Status description and output setting
8004	Monitoring	Monitoring is activated. The difference between the input values including the S_Offset calibration value will be compared to S_Tolerance. Ready = TRUE S_SkewOk = TRUE Error = FALSE
8010	Calibrate	Determining the difference between the two input values. If this state is exited, the value will be stored retentively (falling edge at S_Calibrate). Ready = TRUE S_SkewOk = TRUE S_CalibrationOk = FALSE Error = FALSE

Status Diagram SF_SkewMonitor



13.6 Function block macros

13.6.1 Description

Function block macros are complete networks that can be used like a function block as an instance in a network. You can drag and drop macros to a network and then use them like function blocks in the network-.

13.6.2 Terms

Local macro	A function block macro that belongs to the current project and is saved with it.
Reference network	Function block macro network. You can open this network by double-clicking on the icon in the macro tree view or with the "Open Macro Reference Network" option in the context menu for a macro instance (when a macro is used in a network, an instance of it is actually used).
Object network	Copy of the reference network that is stored in an instance of the macro. You can open this network by double-clicking on the right upper corner of a macro instance or by clicking on the "Open Macro Object Network" option in the instance's context menu.
Macro library	A collection of function block macros, under a library icon, that can be saved and used.
Embedded macro	A macro embedded in another macro (macro-in-macro).
Derived macro	A derived macro.

Changes made in the reference network or in an object network will be copied to all object networks and the reference network.

13.6.3 Creating macros

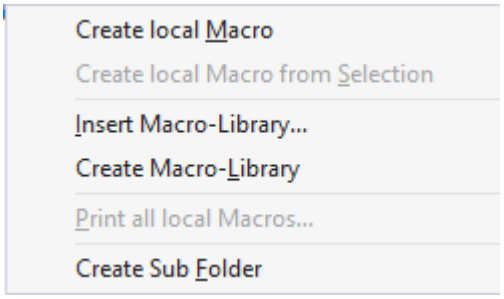
→ "Assembling a macro", page 360
→ "Create a macro from existing elements", page 361
→ "Inserting connections into the macro network", page 362
→ "Creating embedded macros", page 364
→ "Creating a copy of a macro", page 364

See also

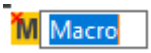
→ " Error codes", page 382
→ "Standard function blocks", page 167
→ " Complex function blocks", page 210

13.6.3.1 Assembling a macro

You can right-click on the "Macro Libraries" in the Function Block Macro Tree pane and then click on the "Create local Macro" option to create a local macro. If you instead right-click on an existing macro library and then click on "Create Macro," you will be able to create a macro in the macro library.

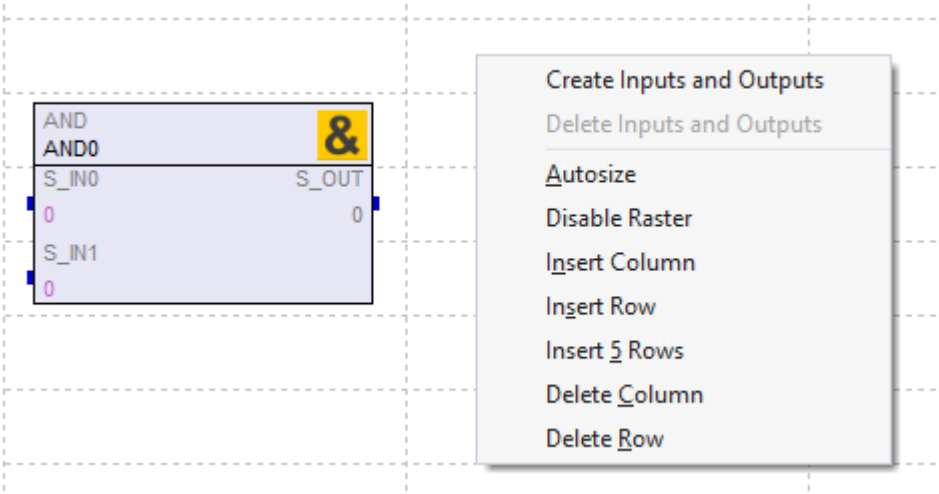


To change the name of a macro, press the [F2] key or use the corresponding context menu. If you enter an invalid name, the text will be shown in red.

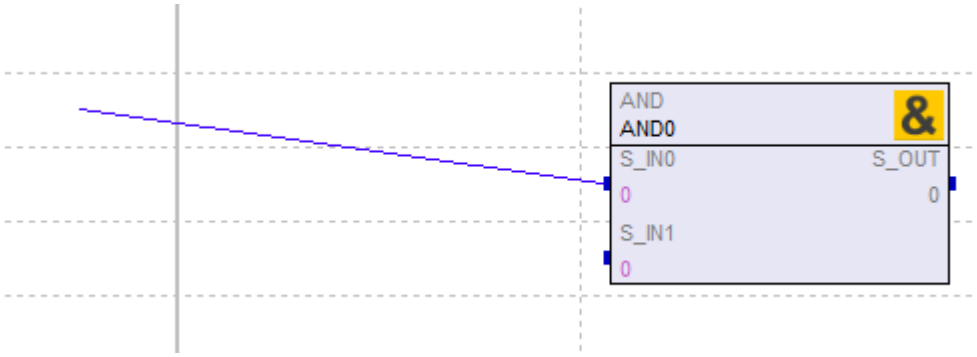


After you create a macro, you will be able to expand its tree and double-click on the network icon (which will have the same name as the macro) to open the corresponding reference network, which will be empty at this point.

You can drag and drop function blocks, constants, and macros in and connect them to each other. With regard to constants, you will be able to assign values to them as usual. You can also create the macro connections automatically as elements in the input and output columns with the "Create Inputs and Outputs“ network context menu option, in which case all inputs and outputs will be automatically drawn from instances outward.

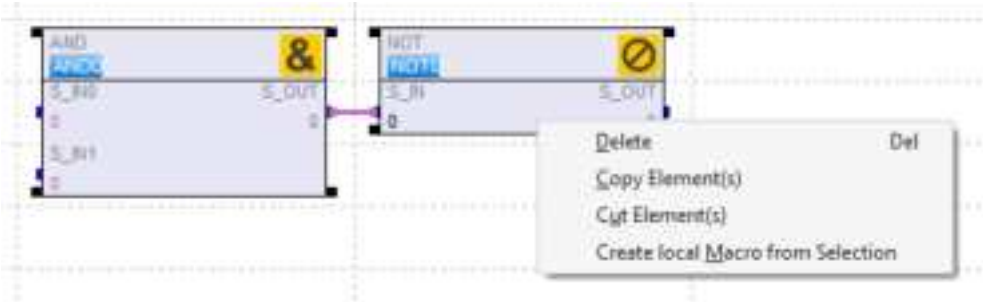


If you instead only want individual inputs and outputs to be drawn outward, you can do this by manually dragging the inputs or outputs of the corresponding instances to the network's input or output column.

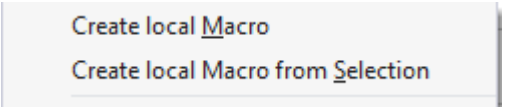


13.6.3.2 Create a macro from existing elements

If you select a group with instances, constants, macros, and comments in a network and then click on the "Create local Macro from Selection" context menu option, a macro will be generated.



In addition, you can select the elements with "Cut" or "Copy" and then select them in the Function Block Macro Tree pane with the "Create local Macro from Selection" context menu option.

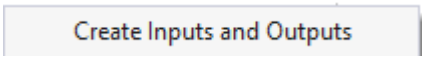


You can select the macro you just generated in the Function Block Macro Tree pane. When the macro is created, it will be automatically be named "Macro" with a number (Macro0, for example). As described previously, you can then edit this macro.

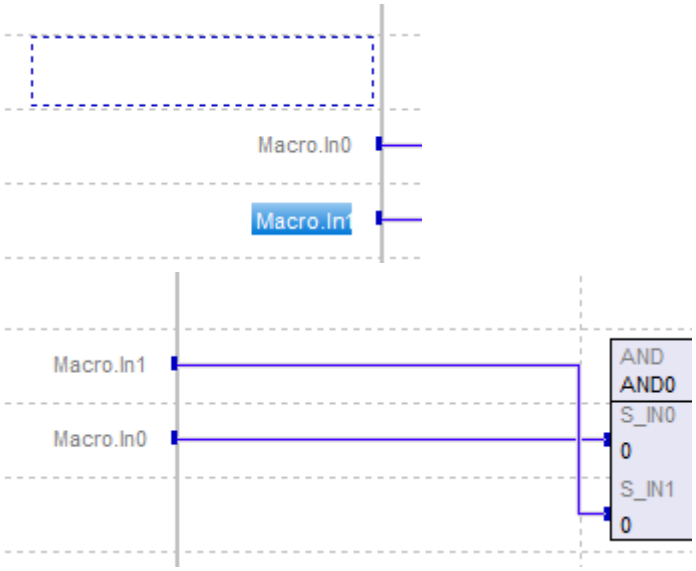
→ Please note that function block macros can only contain the following elements: function blocks, constants, connections, comments, and embedded macros.

13.6.3.3 Inserting connections into the macro network

The "Create Macro Inputs and Outputs“ context menu option will appear after you right-click inside a reference or object network.

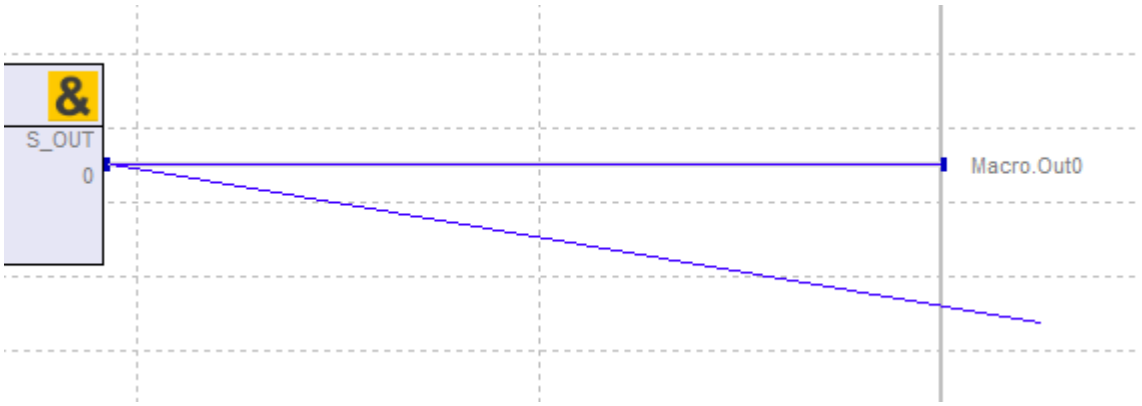


This option will automatically create an element in the input or output column for every input and output in the macro and connect these elements to the corresponding instance. If you want to connect an instance to a different existing input, you will need to delete the connection and draw it to the different input. The macro connection order will be based on the elements in the input and output columns, meaning that you can change it by moving these elements accordingly.

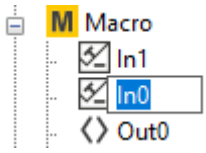


Die Elemente in der Eingangs– und Ausgangsspalte können auch durch das Ziehen von Verbindungen erstellt werden.You can also create elements in the input and output columns by drawing connections. To do this, drag a connection from an unconnected input on an instance to the input column. Then click in the input column to create a new macro input.

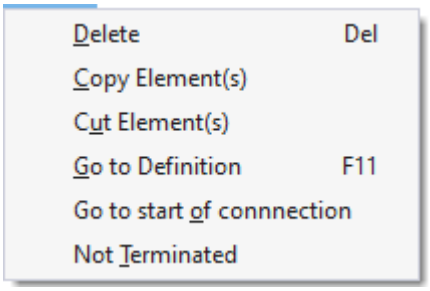
This also works for outputs, and can also be used to create an additional output even if the outputs are already connected.



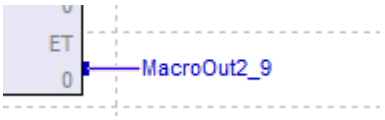
You can rename the macro connections in the input and output columns in the tree.



You can set outputs that are not required to "Not Terminated" in the output column.



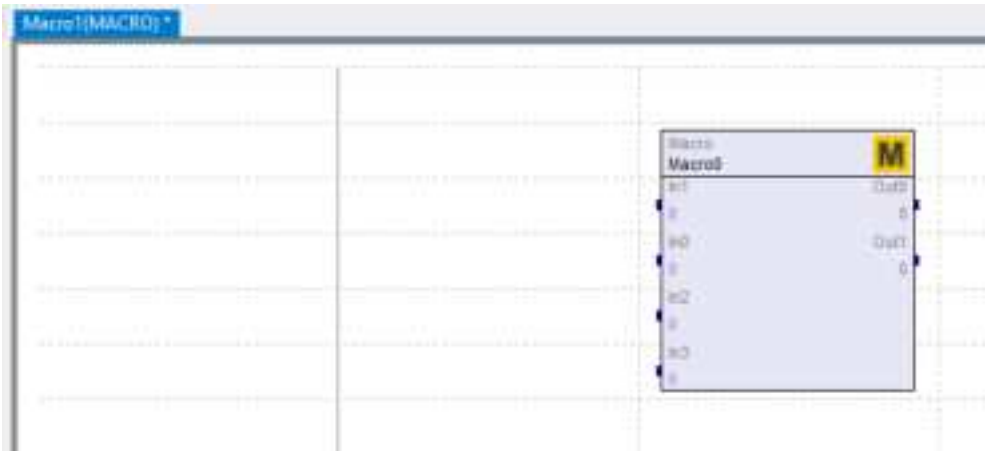
The macro output will be deleted and a temporary global connection will be created.



As long as the macro's connecting elements have not been created, the instance will not show any inputs or outputs.

13.6.3.4 Creating embedded macros

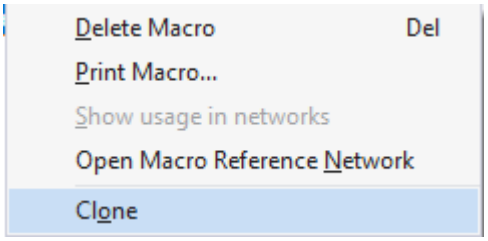
You can add macros as instances in other macros. To add them, simply drag and drop them in the network for a macro. Please note that for this to work, the incorporated macro must have already been fully created along with its input and output elements.



Make sure that when you are placing embedded macros, you are not inserting macros into their own network (including recursively).

13.6.3.5 Creating a copy of a macro

After you select a macro, you can click on the "Clone" context menu option to create a copy of the macro.



13.6.4 Macro libraries

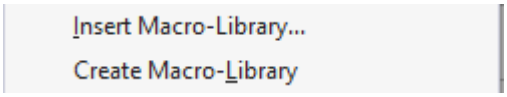
→ "Creating macro libraries", page 365
→ "Saving macro libraries", page 366
→ "Displaying macro libraries", page 367
→ "Using macros from libraries", page 367
→ "Local macro library", page 368
→ "Locking macro libraries", page 369
→ "Comparison of macros", page 371

See also

→ "Creating macros", page 359

13.6.4.1 Creating macro libraries

Select the "Create Macro Library" option in the context menu for the "Macro Libraries" element in the Function Block Macro Tree pane.

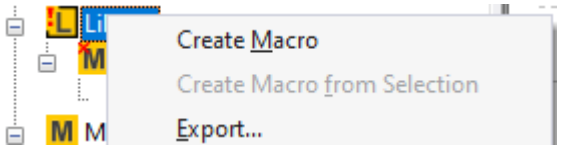


A library for macros will be created. You can change the name of this macro library by pressing the [F2] key or with the context menu.

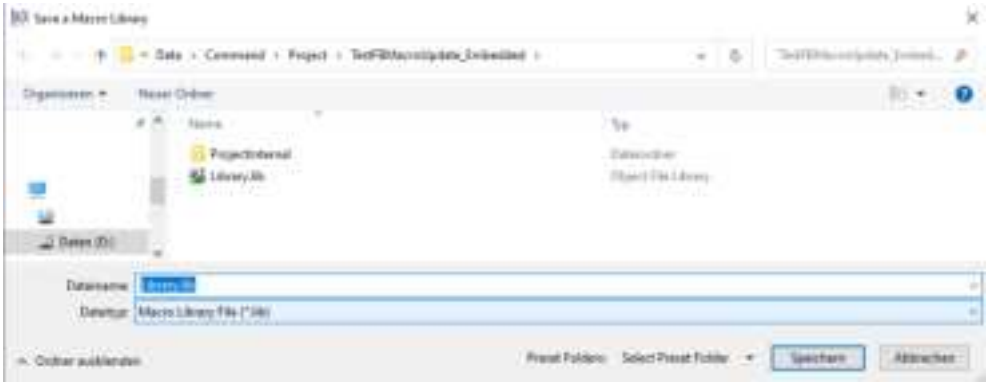
If you try to enter an invalid name, the text will be shown in red. You can use the context menu to insert macros within a macro library.

13.6.4.2 Saving macro libraries

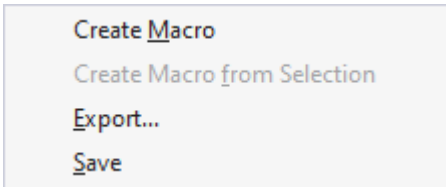
Every newly created library needs to be exported once first. Start by clicking on the "Export..." option in the context menu.



Then select the location where you want to save the library in the "Save" dialog box and change the suggested filename if you want.



After doing this, you will be able to click on the "Save" context menu option to save changes in the library without having to export it again.

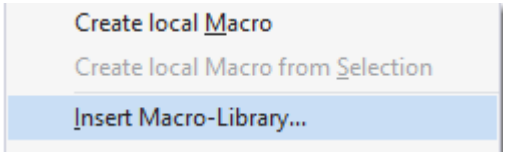


If the library has not been locked, you can export it again (i.e., selecting a different storage location and filename). The previously created file will remain unchanged.

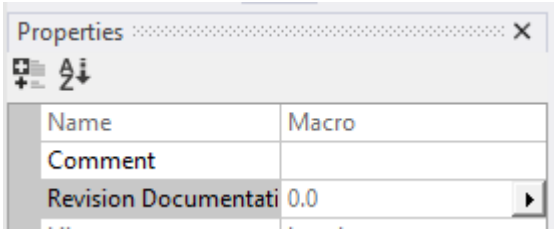
When you click on the "Save" menu option for a macro library, the file that was last created with an export or from which the library was loaded will be used.

13.6.4.3 Displaying macro libraries

You can click on the "Insert Macro Library" context menu option for "Macro Libraries" in the Function Block Macro Tree pane to load a library from a file.

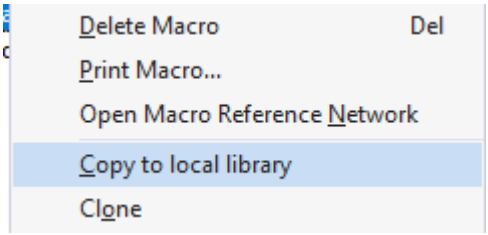


By expanding the tree elements, you can view the corresponding content and open the reference networks for the macros to see what they do. In addition, every library and every macro can include a comment and their own revision documentation.



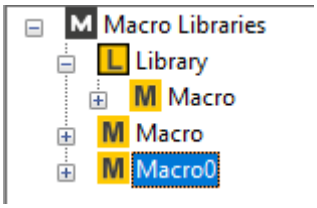
13.6.4.4 Using macros from libraries

Both macros from libraries and local macros can be dragged and dropped from the tree and into a network. When you do this with macros from libraries, a copy will be automatically created in the project (i.e., under the local macros) and embedded macros will be copied as well. If the name has not yet been used, these embedded macros will have the same code as in the library; otherwise, a number will be appended to the name.You can also copy macros in libraries to the project with the "Copy to local library" context menu option.



13.6.4.5 Local macro library

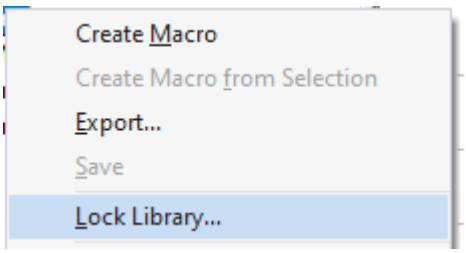
Local macros are macros that belong to a project and are saved together with it. These macros are listed under the libraries in the Function Block Macro Tree pane, since they do not belong to any external libraries.



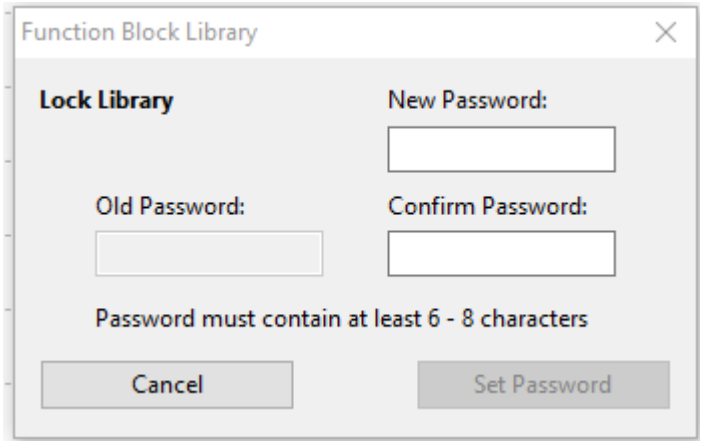
All the macros used in the project will be listed here. In addition, they will be shown with a gray font and cannot be deleted. Local macro libraries from project libraries will not be shown in the Library Tree pane, but in the Function Block Library Tree pane instead. This will be the same as for a locked external library with the name "Local" and the project library name in parentheses.You will not be able to edit any of this, and instead will only be able to copy whole macros to the local macro library of the current project.

13.6.4.6 Locking macro libraries

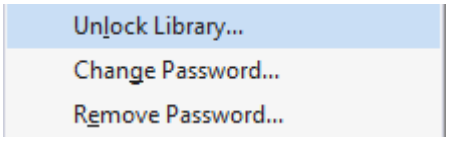
You can lock macro libraries with the context menu.



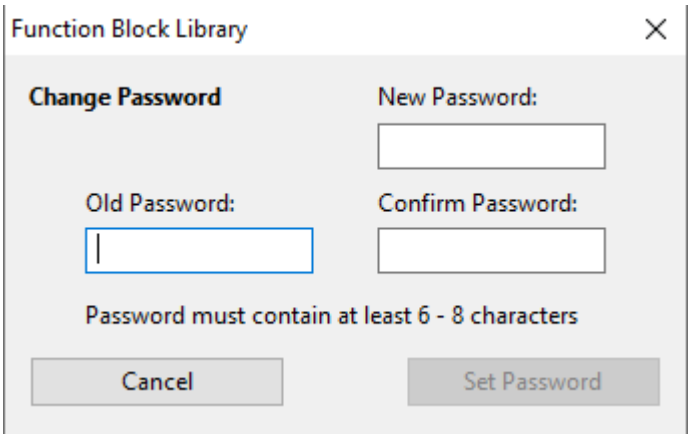
When you click on this option, a dialog box will appear and you will be able to enter a password for locking the library. When you create a password, you will need to enter it a second time in order to confirm it.



If a library has been locked, you will only be able to open it by entering the corresponding password. You can do this by clicking on the "Unlock Library" option in the library context menu, which also includes options for deleting and changing the password.

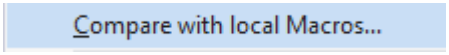


When you click on any of these three options, a dialog box with a corresponding input prompt will appear.

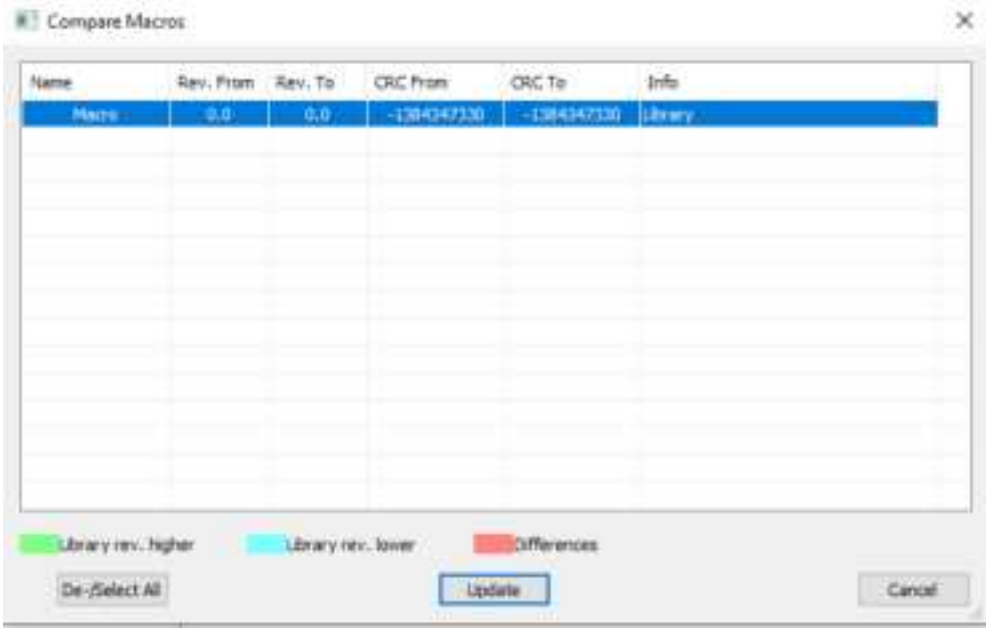

Function Block Library
Change Password
New Password:
Old Password:
Confirm Password:
Password must contain at least 6 - 8 characters
Cancel
Set Password

13.6.4.7 Comparison of macros

You can compare local macros with the macros in a library. Click on the the "Compare with local Macros" library context menu option for this purpose.



When you click on this option, a dialog box showing the data will appear.



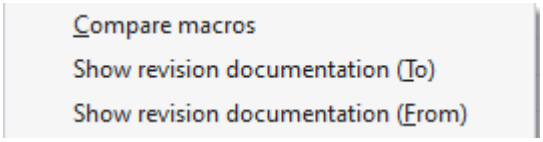
As a first step, the dialog box will show whether the revisions are different. Color highlighting will be used to indicate whether the revision of the macro you are using is lower or higher than the revision in the libraries.

Example view:

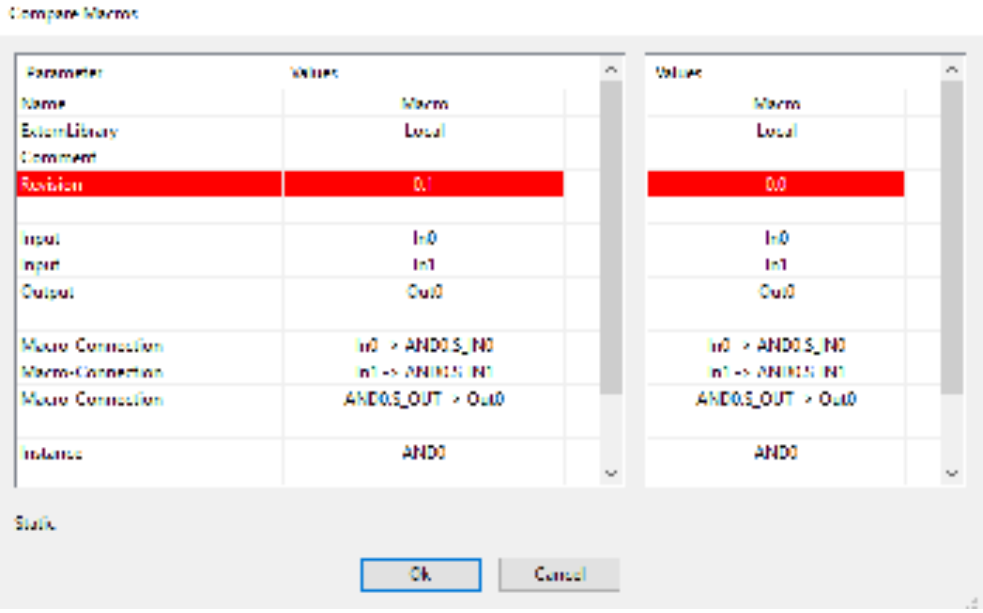
Name	Rev. Lib	Rev. Prj	Info
Macro	0.0	0.1	Library
Macro0	1.0	0.0	Library

You can click on the relevant context menu option to see the differences.

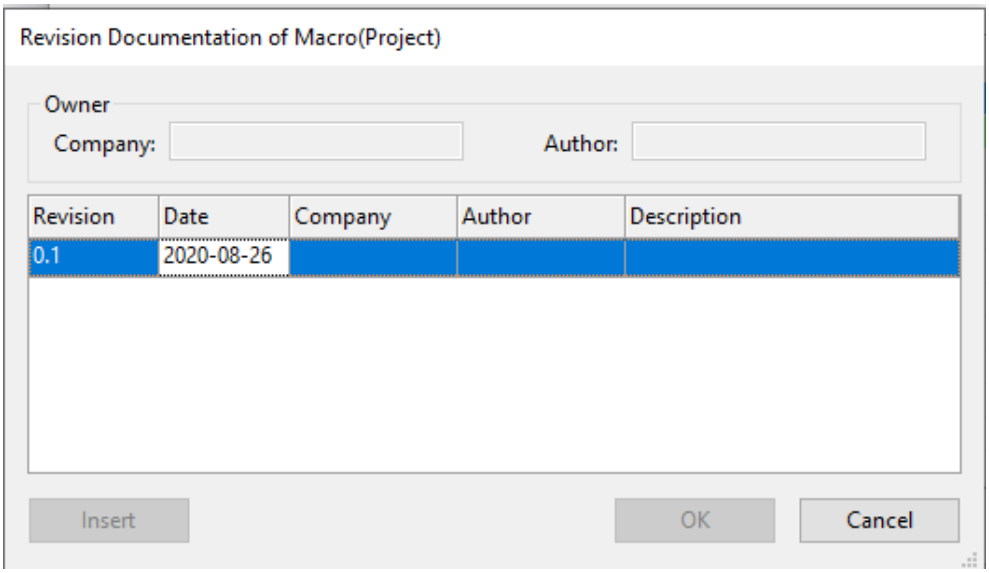
Example view:



Clicking on "Compare macros" will open a dialog box showing the differences.



Clicking on "Show revision documentation" will show the documentation in the project and in the library.

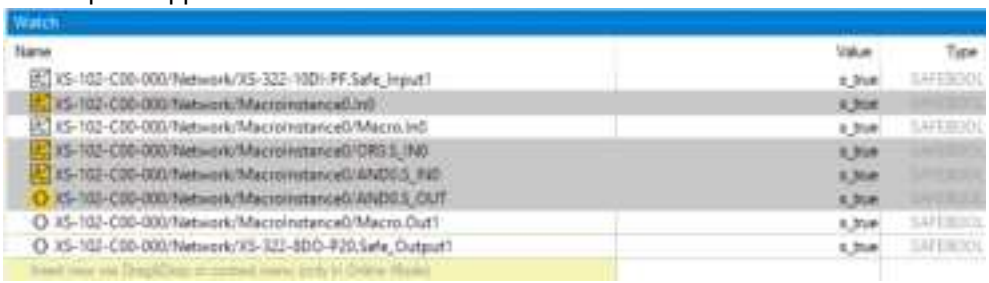


Clicking on the Update button will replace the macro in the project with the macro from the library. XN Safety Designer will check whether this replacement is allowed. In addition, it is possible that the update will lead to new inputs/outputs being added, which can result in the macro not having enough space in the network and requiring for other elements to be moved. If this is not possible, the update can be prohibited, in which case you will first need to create enough space in the network.

13.6.5 Debugging function block macros

When in debug mode, you can check what is going on within each individual macro instance. To do this, you will need to open the object network of that instance by double-clicking on the instance or with the context menu.

The name and order of the elements in the object network's input and output columns will correspond to the macro instance's connections. Please note that it is not possible to force the values of the connection elements – otherwise, everything described in the "Graphic editor" section in regard to debugging, forcing, and the Watch pane applies here as well.



14. Safety modules

14.1 Description of XS-102-C00-000 Safety CPU module

The XN Safety CPU module XS-102-C00-000 supports up to 16 Safety I/O modules.
In addition, the Safety CPU module is able to handle hand-held terminals with emergency stop or enable buttons.
The Safety CPU module is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).

The Safety CPU module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual, → "Further usage information", page 441
The Safety CPU module alone already represents a minimum system for a safety controller.

XS-102-C00-000

In addition, the Safety CPU module uses secure bus frames to control communications with remote safety modules while ensuring the correct timing is used.
Its tasks include:

- Processing the safety application

And

- Distributing the configuration data to remote safety modules

Equipment supplied

1x XS-102-C00-000
1x plug connector PLUG-FMC-3S (1.5/3-ST-3.5)
1x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

14.2 Description of XS-322-4AI-I2 Safety Analog input module

The XS-322-4AI-I2 Safety Analog input module provides the values of four analog current inputs to the safe CPU (XN Safety CPU module).

The four current inputs feature an input range of 4 to 20 mA with a resolution of 16 bits.

A +24 V sensor supply voltage is provided for the analog inputs. This voltage must be fed externally. The external +24 V power supply for the analog inputs will be monitored for overvoltage and undervoltage.

The safe evaluation of the analog inputs is ensured with two independent diverse-redundance evaluations with mutual monitoring.

XS-322-4AI-I2

The module's safety functions meet:

- **With dual-channel use**
  The requirements for SIL3 in conformity with EN IEC 62061 and performance level e (PL e), Cat. 4 in conformity with EN ISO 13849.

as well as

- **With single-channel use**
  The requirements for SIL3 in conformity with EN IEC 62061 and performance level e (PL e), Cat. 3 in conformity with EN ISO 13849.

Equipment supplied

1x XS-322-4AI-I2

5x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

Manual Safety Analog input module XS-322-4AI-I2 MN050025

14.3 Description of XS-322-10DI-PF Safety Digital input module

The Safety Digital input module XS-322-10DI-PF is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).

XS-322-10DI-PF

The Safety Digital input module features:

- Ten (10) safe inputs (EN 61131-2; EN IEC 62061, and EN ISO 13849)
- Redundant clock signal output (short-circuit proof)

The safe inputs are used for the failsafe reading of ten (10) sensor signals (emergency stop, enabling switch, etc.)

In order to make it possible to test inputs and detect cross-fault short circuits (e.g., emergency stop, the Safety Digital input module features two non-failsafe clock signal outputs TA and TB.

The Safety Digital input module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual,
→ section "Further usage information", page 441

In order to be used in an application, the Safety Digital input modulealso requires at least an XS-102-C00-000 XN Safety CPU module that uses secure bus frames to control communications with safety modules while ensuring the correct timing is used. Moreover, this includes:

- Processing the safety application

And

- Deploying the configuration data to remote safety modules

Equipment supplied

1x XS-322-10DI-PF

5x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

ManualXN Safety module XS-322-10DI-PF MN050021

14.4 Description of XS-322-8DO-P20 Safety Digital output module

The XS-322-8DIO-PF20 Safety Digital output module is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).

The Safety Digital output module features:

- Eight (8) safe outputs (EN 61131-2; EN IEC 62061, and EN ISO 13849)
- Redundant clock signal output (short-circuit proof)

The safe outputs are used for the failsafe outputting of eight sensor signals, such as those used to drive relays or valves.

The Safety Digital output module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual,
→ "Further usage information", page 441

XS-322-8DO-P20

In order to be used in an application, the Safety Digital output module also requires at least an XS-102-C00-000 XN Safety CPU module that uses secure bus frames to control communications with safety modules while ensuring the correct timing is used. Moreover, this includes:

- Processing the safety application

And

- Deploying the configuration data to remote safety modules

Equipment supplied

1x XS-322-8DO-P20
5x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

Manual Safety Digital output module XS-322-8DO-P20 MN050022

14.5 Description of XS-322-8DIO-PF20 Safety Digital input/output module

The XS-322-8DIO-PF20 Safety Digital input/output module is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).
The Safety Digital input/output module features:

- Two (2) safe +24 V outputs with a maximum of 2 A per output
  (EN 61131-2; EN IEC 62061, and EN ISO 13849)
- Six (6) safe +24 VDC/0.5 ms inputs
  (EN 61131-2; EN IEC 62061, and EN ISO 13849)
- Redundant clock signal output (short-circuit proof)

The safe outputs are used for the failsafe outputting of two sensor signals to, for example, drive relays or valves.
The safe inputs are used for the failsafe reading of six sensor signals (emergency stop, enable button, etc.).
In order to make it possible to test inputs and detect cross-fault short circuits
(e.g., emergency stop, the Safety Digital input/output module features two non-failsafe clock signal outputs TA and TB.

XS-322-8DIO-PF20

The Safety Digital input/output module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual, → section "Further usage information", page 441

In order to be used in an application, the Safety Digital input/output module also requires at least an XS-102-C00-000 XN Safety CPU module that uses secure bus frames to control communications with safety modules while ensuring the correct timing is used. Moreover, this includes:

- Processing the safety application

And

- Deploying the configuration data to remote safety modules

Equipment supplied

1x XS-322-8DIO-PF20
5x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

Manual Safety Digital input/output moduleXS-322-8DIO-PF20 MN050023

14.6 Description of XS-322-2DO-RNODC Safety DC Relay output module

The XS-322-2DO-RNODC Safety DC Relay output module is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).

XS-322-2DO-RNODC

The Safety DC Relay output module features:

- Two (2) safe outputs (EN 61131-2; EN IEC 62061, and EN ISO 13849)

The two outputs are used for the failsafe closing (NO) of a circuit at a permissible rated voltage of +24 VDC and a maximum continuous current of 6 A.

The Safety DC Relay output module is intended for switching high power levels and electrically isolated circuits – including those found in hydraulic valves and in relaying emergency stop circuit signals.

The Safety DC Relay output module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual, → "Further usage information", page 441

In order to be used in an application, the Safety DC Relay output module also requires at least an XS-102-C00-000 XN Safety CPU module that uses secure bus frames to control communications with safety modules while ensuring the correct timing is used. Moreover, this includes:

- Processing the safety application

And

- Deploying the configuration data to remote safety modules

Equipment supplied

1x XS-322-2DO-RNODC
2x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

ManualSafety DC Relay output module XS-322-2DO-RNODC MN050027

14.7 Description of XS-322-2DO-RNOAC Safety Relay output module

The XS-322-2DO-RNOAC Safety Relay output module is characterized by safety integrity level SIL3 (EN IEC 62061) and performance level e (PL e) (EN ISO 13849).

The Safety Relay output module features:

- Two (2) safe outputs (EN 61131-2; EN IEC 62061, and
  EN ISO 13849)

The two outputs are used for the failsafe closing (NO) of a circuit with a permissible rated voltage of +24 VDC/230 VAC and a maximum continuous current of 6 A.

The Safety Relay output module is intended for switching high power levels and electrically isolated circuits – including those found in hydraulic valves and in relaying emergency stop circuit signals.

XS-322-2DO-RNOAC

The Safety Relay output module for safety-related applications is suitable for use in systems with optional modules and interface tags as set forth in XN Safety system manual, → "Further usage information", page 441

In order to be used in an application, the XS-322-2DO-RNOAC also requires at least an XS-102-C00-000 XN Safety CPU module that uses secure bus frames to control communications with safety modules while ensuring the correct timing is used. Moreover, this includes:

- Processing the safety application

And

- Deploying the configuration data to remote safety modules

Equipment supplied

1x XS-322-2DO-RNOAC
2x plug connector PLUG-FKC-4L (2.5/4-ST-5.08)

See also

ManualSafety Relay output module XS-322-2DO-RNOAC MN050026

14.8 Description of XS-322-2INC Safety Encoder module

The XS-322-2INC Safety Encoder module provides the values from two incremental encoders to the safe CPU (XN Safety CPU module).

The dual-channel safety function is implemented by "listening in on" the increments at the incremental encoder interfaces and processing them in two microcontrollers –the "safety core" – with cross-communications.
The incremental encoder function is monitored with a dual-channel evaluation of the encoder signals with dual-channel error detection for each data cable and with measurements of the encoder supply voltages and currents.
Safe speed, position, direction, and acceleration monitoring is carried out in the safe application in the Safety CPU.
The module can detect a variety of fault types, including open wires, cross-fault short circuits, and input signal swapping.

XS-322-2INC

The module's safety functions meet:

- **With dual-channel use**
  The requirements for SIL 3, HFT 1 in conformity with EN IEC 62061 and PL e / Cat. 4 in conformity with EN ISO 13849.

as well as

- **With single-channel use**
  The requirements for SIL 3, HFT 1 in conformity with EN IEC 62061 and PL d, Cat. 2 in conformity with EN ISO 13849.

Equipment supplied

1x XS-322-2INC
2x plug connector PLUG-FMC-8S (1.5/8-ST-3.5)
1x plug connector PLUG-FMC-4S (1.5/4-ST-3.5)

See also

ManualSafety Encoder moduleXS-322-2INC MN050024

15. Error codes

If none of the measures listed below solve the problem, please contact Eaton.

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(0)	No Error			Normal mode	
ERRVAL(1000)	No Error			Normal mode	
ERRVAL(1001)	Cyclical processing has not been completed within the permitted time. Possible cause(s): ■ Incorrect application	Current cycle time in microseconds (the maximum of the two values from controller 1 and 2)	Cycle time limit in microseconds	Configuration must be changed, e.g. increasing the max. cyclic time for the safety CPU Troubleshooting with "Quit Error".	Error LED of the safety module lights up red
ERRVAL(1002)	A buffer was passed as a parameter in a function call. However, the size of the buffer is too small. Possible cause(s): ■ Program error	Actual buffer size or with a value ≥ 0x1000 Identification of the program location of the error (possible values 0x1000-0x100B)	Required buffer size	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1003)	A global variable has been overwritten. Possible cause(s): ■ Program error ■ Defective RAM	Affected variable that was overwritten Possible values: 0: Current error code	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1004)	Program code is executed which, according to the program logic, should never be executed. Possible cause(s): ■ Program error ■ Faulty CPU	is used to identify the code location that was executed Possible values: 0 up to 104	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1005)	When accessing an array, the index lies outside the array boundaries. Possible cause(s): ■ Program error	is used to identify the cause of the error Possible values: 0 up to 57	contains the maximum possible array index	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1006)	Invalid parameter value for a function call (NULL pointer). Possible cause(s): ■ Program error	is used to identify the cause of the error Possible values: 0 up to 28	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1007)	A read error or CRC-32 error was detected during the RAM test, algorithm March C. This is a permanent error that cannot be acknowledged. Possible cause(s): ■ Hardware fault ■ RAM, flash, address- or data line Program error ■ A RAM test was run when there were unmasked interrupts	relevant test step	affected area in RAM	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1008)	A read error or CRC-32 error was detected during the RAM test, GALPAT algorithm. This is a permanent error that cannot be reset. Possible cause(s): ■ Hardware fault ■ RAM, flash, address- or data line Program error ■ A RAM test was run when there were unmasked interrupts	relevant test step	affected areas in RAM (CRC32 check and GalPat check)	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1009)	The configuration of the required modules is not correct. Possible cause(s): ■ Exchange of several modules ■ Modules configured incorrectly ■ Incompatible Interface Frame (incorrect CRC)	0: More than one module is missing	if reason code 0 is between 0 and 1: always 0	- If more than one module has been replaced, the application must be downloaded again. - If the configuration is incorrect, the application must be downloaded again. - Check whether the hardware structure matches the application; modify the application accordingly. - If the interface CRC is incorrect, the writing interface must be re-exported and imported on the reading side. The application must be downloaded again. -Troubleshooting with Quit Error	Error LED of the safety module lights up red
		1: A module is missing and a module update has already taken place beforehand			
		2: Nothing can be distributed to the missing module (S-CPU)	if reason code 0 is 2: Security number of the removed module		
		3: Maximum number of retries of the configuration distribution or configuration check has been exceeded	if reason code 0 is 3: number of retries		
		4: when trying to send the configuration, it was determined that the target module is in error state	if reason code 0 4 is runtime state of the removed module		

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
		5: At least one local SDI module is missing	if reason code 0 is between 5 and 10: always 0		
		6: At least one local safety interface module is missing			
		7: At least one local STO module is missing			
		8: At least one local SRO module is missing			
		9: a module (master module) tried to write to the configuration while its own configuration was not deleted and was set to master			
		10: A slave module has detected that the remote modules are not configured correctly			

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
		11: Local input module in the wrong slot (counting starts at 1)	if reason code 0 is between 11 and 13: Slot number of the module		
		12: Local output module in the wrong slot (counting starts at 1)			
		13: DINT input module in wrong slot (count starts from 1)			
		14: Wrong number of local inputs			
		15: Wrong number of local outputs			
		16: An attempt was made to address a local module that is missing			
		17: An expansion module could not be configured (count starts from 1)	If reason code 0 is 17: Module slot number		
		18: Too many connected expansion modules			
		19: Too few connected expansion modules			
		20: Wrong Safety interface module connected			

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1010)	The RAM memory used for data structures with a configuration-dependent size is too small. Possible cause(s): ■ Too many modules ■ Excessively large application	0: Data for I/O lists and remote modules 1: FSoE connection status 2: FSoE buffer	always 0	■ Check the size of the application and reduce it accordingly. Download the application again, reset the error with Quit Error. ■ Check the number of modules and remove modules accordingly; reset the error with Quit Error.	Error LED of the safety module lights up red
ERRVAL(1011)	The max. number of errors was exceeded while deploying the configuration data to a remote module. Possible cause(s): ■ Transmission error ■ Another module is faulty	Error code for last deployment attempt	Return code of last SSDO command run	Trigger the deployment multiple times If there are faulty modules: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1012)	An error occurred during exception handling. Possible cause(s): ■ Software error	0: The maximum number of nested TRY-CATCH-RETRY blocks was exceeded 1: End of a TRY-CATCH-RETRY block reached without starting TRY 2: Exception not thrown in normal program mode 3: No TRY block for catching the exception present	Number of open TRY-CATCH-RETRY blocks	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1013)	After the configuration data was deployed to a remote module, a difference was detected on readback.	Module index of affected remote module	always 0	Download and start the application multiple times If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1014)	The interpreter code in the configuration data was not found. Possible cause(s): ■ Faulty configuration ■ Software error	always 0	always 0	Download and start the application multiple times If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1015)	The configuration interpreter list specifies an FSB for which there is no existing FSB code. Possible cause(s): ■ A faulty configuration has been imported ■ Flash defective	FSB ID of the FSB code that was not found	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1016)	An error was detected during the ROM test in a read-only flash memory area. This is a permanent error that cannot be reset. Possible cause(s): ■ Flash defective	Information regarding the cause of the error	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1017)	The called FSB has returned an error code.	The FSB's error code (never 0)	FSB ID and object index of affected FSB	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1018)	The remote safe input of a missing optional module (remote module or Safety interface module) has exceeded its maximum age.	always 0	always 0	Check whether standard communications with the required safety module are intact. In addition, make sure that the required safety module is not in its error condition. Moreover, it is possible that the configured maximum transmission time is too short. Check all these. If the error is fixed, you can reset it with Quit Error or by restarting the system.	The safety module's Error LED will flash
ERRVAL(1019)	After the FSB was called, an illegal value was detected in the application memory. Possible cause(s): ■ Faulty FSB	Actual pattern at the end of the object memory	Target pattern at the end of the object memory	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1020)	The called FSB did not increment the counter variable. Possible cause(s): ■ Faulty FSB	Actual counter variable value	Target counter variable value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1021)	An error was detected when managing the RAM memory for data structures with a configuration-dependent size. Possible cause(s): ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1022)	The response to an SSDO request is not yet present. Possible cause(s): ■ Running time via the bus ■ Excessively long processing time in target module	always 0	always 0	Increase the time on the bus; if the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1023)	An error was signaled in the return code field in an SSDO response. Possible cause(s): ■ Error in remote module	always 0	always 0	Check remote modules for errors If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1024)	An SSDO response contains less data than expected. Possible cause(s): ■ Error in remote module	Actual length of the data in the SSDO response	Target length of data in the SSDO response	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1025)	An error was detected when testing the checksums for the constant elements of the configuration tables in RAM. Possible cause(s): ■ Software error ■ RAM faulty	Table identification Possible values: 0 up to 11	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1026)	Could not access the application memory. Possible cause(s): ■ Software error ■ Incorrect XN Safety Designer request	0	If reason code 0 = 0: The two least significant bytes will contain the determined invalid cell index; the two most significant bytes will contain the number of existing cells	Please contact Eaton	Error LED of the safety module lights up red
		1	If reason code 0 = 1: invalid cell index		
		2	If reason code 0 = 2: invalid address offset		
		3	If reason code 0 = 3: The two least significant bytes will contain the address offset; the two most significant bytes will contain the length		
		4	If reason code 0 = 4: invalid length (not divisible by 4)		
ERRVAL(1027)	An invalid block header was detected in a flash memory area. Possible cause(s): ■ Hardware fault (faulty flash memory) ■ The programming operation was interrupted	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1028)	An invalid password was entered when attempting to log in. The error will be shown in the console pane in XN Safety Designer. Possible cause(s): ■ Typing error ■ Cyberattack	always 0	always 1	The password needs to be entered again.	Normal operation
ERRVAL(1029)	There is an invalid parameter in an SSDO frame. Possible cause(s): ■ Software error	0	always 0	Please contact Eaton	Error LED of the safety module lights up red
		1			
		2			
		3			
		4	If reason code 0 = 4: The two least significant bytes will contain the size of the passed buffer; the two most significant bytes will contain the max. size		
ERRVAL(1030)	A parameter is missing in an SSDO frame. Possible cause(s): ■ Software error	Used only in the firmware for the hardware test.		Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1031)	Internal error when processing the SSDOs. Possible cause(s): ■ Software error	Length of SSDO being written	Maximum length of an SSDO	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1032)	The SSDO package contains an invalid command byte. Possible cause(s): ■ Software error	Invalid command	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1033)	An invalid new password was entered when attempting to change the password. Possible cause(s): ■ Software error ■ Faulty Safety Designer	always 0	always 0	If you enter the wrong password, Safety Designer will show a warning. If error 33 occurs, it is an internal operating system error. Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1034)	The SSDO package contains an invalid virtual address. Possible cause(s): ■ Software error	Error cause identification Possible values: 0 up to 1	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1035)	While processing the SSDOs, the system detected that an internal buffer was too small for the requested data volume. Possible cause(s): ■ Software error ■ Faulty Safety Designer	Error cause identification Possible values: 0 up to 1	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1036)	An error was detected during the Safety interface controller's self-test. Possible cause(s): ■ Hardware fault	Error cause identification Possible values: 0: Test message could not be sent 1: No response to the test message was received 2: The data in the received test message is incorrect	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1037)	The retentive variables are inconsistent. Possible cause(s): ■ Hardware fault ■ Software error	0: During POST, the system detected that the checksum for the retentive variables in the EEPROM is incorrect	always 0	Please contact Eaton	Error LED of the safety module lights up red
		1: Error in the consistency of the retentive variables detected during cyclical processing			
ERRVAL(1038)	A watchdog error occurred Possible cause(s): ■ Software error	Code address	In the event of an exception outside of the FSB call: ■ always 0 In the event of an exception during the FSB call: ■ FSB ID in the least significant two bytes ■ Object index in the most significant two bytes	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1039)	The bootloader version is incorrect Possible cause(s): ■ Wrong firmware	Bootloader version that is present	Minimum bootloader version required	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1040)	An error occurred when attempting to access the internal I/O bus. Possible cause(s): ■ Hardware fault ■ The module was disconnected	0: Error attempting a read access operation for a byte	If reason code 0 is between 0 and 1: Bytes 0 and 1 are the error's address Bytes 2 and 3 are the module's address; if you divide this value by	If a module has been disconnected, reconnect the module and restart If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
		1: Error attempting a read access operation for a two-byte value			
		2: Error attempting a write access operation for a byte	If reason code 0 is between 2 and 3: Bytes 0 and 1 are the error's address Bytes 2 and 3 are the value being written		
		3: Error attempting a write access operation for a two-byte value			
		4: Different controller values for the module ID			
		5: Timeout error while waiting for the SPI master's Grant	If reason code 0 is between 5 and 8: Position of module in which the error occurred (starting from 0)		
		6: Timeout error while waiting for the SPI master's Ready			
		7: Wrong SPI flash memory length			
		8: Wrong SPI flash memory CRC			
		9: Different safety numbers in the two controllers			

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
		10: No unique safety numbers in the expansion modules	If reason code 0 = 10: Value of safety number for which there is a duplicate		
		11: When determining the serial numbers, the two controllers return different values	If reason code 0 = 11: Position of the module in which the error occurred (starting from 0)		
		12: Different number of expansion modules in the two controllers	If reason code 0 = 12: Byte 0: Number of detected I/O modules on controller Byte 1: Number of detected I/O modules on other controller		
		13: Missing safety number in an expansion module	If reason code 0 = 13: Position of the module in which the error occurred (starting from 0)		
ERRVAL(1041)	The values of the readback outputs that are already in an error state are not 0. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External error (e.g., short-circuit) ■ Software error	Outputs that are not enabled (bit 0 = 1st output)	Actual value (bit 0 = 1st output)	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1042)	Not all outputs assumed a value of 0 when testing the OUTRESET signal. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External error (e.g., short-circuit) ■ Software error	Tested outputs (bit 0 = 1st output)	Actual value (bit 0 = 1st output)	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1043)	A local expansion module is in its error condition. Possible cause(s): ■ Error in the expansion module	0: Expansion module error	Reason code 0 of expansion module	Replace the expansion module If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
		1: 0-based position of expansion module			
ERRVAL(1044)	An error occurred during the init phase for a local expansion module. Possible cause(s): ■ Error in the expansion module ■ Wrong configuration	0: The current values of the INC module cannot be read from the module	0-based position of expansion module	Check the current values, switch Safety CPU to master mode; replace the INC module If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
		1: There are no available current values that should be written to the INC module			
		2: There is an INC module on a slave Safety CPU			
		3: There is an INC module with an unsupported version number			
ERRVAL(1045)	The expansion module corresponding to the input or output is not present. Possible cause(s): ■ An optional module is not connected	0: Digital output	always 0	Check optional modules	Error LED of the safety module lights up red
		1: Digital input			
		2: Incremental encoder module			

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1046)	There are too few resources available for the requested function Possible cause(s): ■ Incorrect configuration	always 0	always 0	Check configuration. If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1047)	An error with the parameter list has occurred. Possible cause(s): ■ Hardware fault (faulty flash memory) ■ Software error	serves to identify the cause of the error Possible values: 0 up to 8	serves to identify the cause of the error	Please contact Eaton	Safety module error LED
ERRVAL(1048)	An error occurred during the OUTRESET watchdog test. Possible cause(s): ■ Hardware fault	Bit mask for identifying the test steps in which the error was detected	Current test step	Please contact Eaton	Safety module error LED
ERRVAL(1049)	When checking the LOG memory during POST, the system detected that both sectors were full. Possible cause(s): ■ Hardware fault ■ Program error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1050)	When checking the LOG memory during POST, the system detected that both sectors contained data but were not full. Possible cause(s): ■ Hardware fault ■ Program error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1051)	The messages in the LOG memory are not continuous (gaps in LOG memory). Possible cause(s): ■ Program error	Start address of first empty entry found in the LOG memory	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1052)	The information regarding the stored safety numbers for the remote modules is inconsistent. Possible cause(s): ■ Program error	always 0xFFFFFFFF	always 0xFFFFFFFF	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1053)	The SSDO command cannot be run in the current state. Possible cause(s): ■ Module is running boot code	always 0	always 0	Restart If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1054)	An attempt was made to access a memory area (flash or SD card) that cannot be accessed or to which access is not allowed. Possible cause(s): ■ Program error ■ Error in Safety Designer.	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1055)	An error occurred during flash programming. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1056)	During the status change in the STM module, the system detected that a change is already active. Possible cause(s): ■ Program error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1057)	While attempting to activate the timer in the STM module, the system detected that a timer is active already. Possible cause(s): ■ Program error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1058)	Not all the required flags were set in a query in the STM module. Possible cause(s): ■ Program error	Flags that should be set	Flags that are currently set	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1059)	The diagnostic code or the login status in function block HGW-Login has a wrong value Possible cause(s): ■ Error in Safety Designer. ■ Programming error	serves to identify the cause of the error 1: The diagnostic code has assumed an invalid value 2: The login status has assumed an invalid value	Value of diagnostic code or login status	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1060)	The flag for deactivating Safety interface communications (HBG/SIB061) is set, but deactivation is not allowed. Possible cause(s): ■ Programming error	1: At least one HBG/SIB061 is configured 2: There is a connection to an HBG/SIB061	always 0	Check the configuration; either set the flag to false or remove the HBG/SIB061	Error LED of the safety module lights up red
ERRVAL(1061)	An error occurred when programming the external flash memory. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1062)	One of the ADC channels for voltage and temperature monitoring did not return a valid reading.	serves to identify the cause of the error 0: Temperature 1: 1V8 2: 5V0 4-6: ADC configuration failed 7: ADC did not return a reading despite a retry	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1063)	One of the ADC channels for voltage and temperature monitoring was outside the valid value range.	serves to identify the cause of the error 0: Temperature 1: 1V8 2: 5V0 3: 24V	Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1064)	An error occurred during the cyclical test for the OUTRESET watchdog. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error 1: Test was not ended 2: Response to toggle signal being turned off did not occur on time 3: Watchdog output was already active during startup	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1065)	An error occurred during an I2C access operation (e.g., EEPROM/EERAM). Possible cause(s): ■ Hardware fault	serves to identify the cause of the error	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1066)	Not Used				
ERRVAL(1067)	The FSoE connection between the HGW module and the Safety CPU dropped out in an uncontrolled manner. Possible cause(s): ■ Connection faulty ■ Excessive distance between the modules	always 0	always 0	Reset the error and log in again	Error LED of the safety module lights up red
ERRVAL(1068)	Invalid parameter value in a function call. Possible cause(s): ■ Program error	serves to identify the cause of the error Possible values: 0 up to 95	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1069)	The space in the SPDO for the safety values to be transmitted is too small. Possible cause(s): ■ Software error	Index of most significant byte of the safety values to be transmitted.	Maximum size of the safety values to be transmitted.	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1070)	Cross-communications not working (transmission error). Possible cause(s): ■ Hardware fault	1: Error during self-test in POST 2: Fault detected in UART	Error flags	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1071)	Not Used				
ERRVAL(1072)	An attempt was made to divide by 0. Possible cause(s): ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1073)	An unexpected value was received through cross-communications. Possible cause(s): ■ Software error	Received value	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1074)	The logical program run monitoring detected that a program section that had to be run in Op/OpTemp mode was not run. Possible cause(s): ■ Software error	Program section ID	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1075)	The local input values read in the two controllers have different values for an impermissibly long time. Possible cause(s): ■ Hardware fault ■ Input circuit	Bit mask with the inputs of your own controller	Bit mask with the inputs of the other controller	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1076)	Cross-communications not working (timeout). Possible cause(s): ■ Hardware fault ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1077)	The content of an STDO response is different in the two controllers. Possible cause(s): ■ Software error	0 or 1 depending on the type of error	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1078)	Runtime monitoring was deactivated in an Operational runtime state despite this not being allowed. Possible cause(s): ■ Software error	ID of the caller that deactivated runtime monitoring.	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1079)-ERRVAL(1081)	Cross-communications not working (unexpected value received). Possible cause(s): ■ Hardware fault ■ Software error	Value received through cross-communications	Expected value (= target value)	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1082)	The computed output image is different in the two controllers. Possible cause(s): ■ Software error ■ Hardware fault	Index of the cell in the application memory with content that does not match the content in the cell of the other controller.	Content of the cell specified in reason code 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1083)	The other controller is no longer synchronized for cross-communications. This situation is not yet an error that will bring about a safe state. As a response to this error, the controllers will resynchronize themselves with each other and exchange information, and only then will the system decide whether to bring about the safe state. Possible cause(s): ■ A controller has detected an error, while the other one has not (yet)	always 0	always 0	Reset the error and, if applicable, restart the system If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1084)	The SD card content could not be loaded, since the configuration is not found in the flash memory and has not been deleted either. Possible cause(s): ■ User error	always 0	always 0	Clear the Safety CPU's memory and try again.	Error LED of the safety module lights up red
ERRVAL(1085)	The SD card could not be initialized Possible cause(s): ■ Hardware fault	always 0	always 0	Use a different SD card If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1086)	The content on the SD card does not match the expected format. Possible cause(s): ■ Hardware fault ■ Software error ■ User error	always 0	always 0	Write the content to the SD card again If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1087)	The configuration data on the SD card does not match the configuration data in the flash memory. Possible cause(s): ■ User error	CRC of configuration data on the SD card	CRC of configuration data in the flash memory	If you want to apply the configuration from the SD card, delete the configuration in the flash memory and restart the system while the SD card is inserted.	Error LED of the safety module lights up red
ERRVAL(1088)	The SD card was inserted while the module was on and not in Service Mode. A dongle was connected when the application started; Service Mode cannot be exited. Possible cause(s): ■ User error ■ Dongle plugged in	always 0	always 0	Remove the SD card again, acknowledge the error and switch to service mode, then insert the SD card. Remove the dongle, acknowledge the error and restart.	Error LED of the safety module lights up red
ERRVAL(1089)	Internal error in a flow table.	serves to identify the cause of the error Possible values: 0 up to 1	Invalid status or index	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1090)	The measured value of the FPGA (REFCLK) reference clock is too low. This is a permanent error that cannot be reset. Possible cause(s): ■ Hardware fault ■ CPU ■ FPGA	Number of measured edges	Minimum number of edges to be measured	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1091)	The measured value of the FPGA (REFCLK) reference clock is too high. This is a permanent error that cannot be reset. Possible cause(s): ■ Hardware fault ■ CPU ■ FPGA	Number of measured edges	Maximum number of edges to be measured	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1092)	A timeout occurred while waiting for the SSDO response. Possible cause(s): ■ Running time via the bus ■ Processing time in the target module	always 0	always 0	Reduce running times	Error LED of the safety module lights up red
ERRVAL(1093)	The other controller has detected an error. Possible cause(s): ■ Cause as per the error code from the other controller	Error code from the other controller	always 0	Check the error code from the other controller	Error LED of the safety module lights up red
ERRVAL(1094)	The configuration state of the other controller does not match the configuration state of your own controller Possible cause(s): ■ Module was shut down during configuration	Configuration status of your own controller	Configuration status of the other controller	Reconfigure module	Error LED of the safety module lights up red
ERRVAL1095)	A constant in the application memory was changed and does not match the value stored in the flash memory. Possible cause(s): ■ Hardware fault (faulty flash memory, faulty RAM) ■ Software error	Index of memory cell corresponding to the faulty constant in the application memory	Index of constant in the flash memory	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1096)-ERRVAL(1097)	Not Used				

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1098)	The response to the internal test could not be determined. Possible cause(s): ■ Software error	Bit mask of the digital inputs	0: Error on a local input 1: Error on a Safety interface module input	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1099)	An error was detected during the input test (internal test). Possible cause(s): ■ Hardware fault	Digital input states	Bit mask of the digital inputs	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1100)	An internal test is active during the external test. Possible cause(s): ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1101)	The response to the external test could not be determined. Possible cause(s): ■ Software error	Bit mask of the digital inputs	0: Error on a local input 1: Error on a Safety interface module input	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1102)	External test: The clock pulse was not detected at an input connected to the clock signal. Possible cause(s): ■ Cross-circuit	always 0	always 0	Compare the cross-circuit wiring to the application	The safety module's Error LED will flash
ERRVAL(1103)	External test: The clock pulse of a different clock signal output was detected at an input connected to the clock signal. Possible cause(s): ■ Cross-circuit	always 0	always 0	Compare the cross-circuit wiring to the application	The safety module's Error LED will flash
ERRVAL(1104)	The lo-resolution timer is not working correctly (the measured values are too small). Possible cause(s): ■ Hardware fault	Measured value of lo-resolution timer	Lower limit for the measurement	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1105)	The lo-resolution timer is not working correctly (the measured values are too large). Possible cause(s): ■ Hardware fault	Measured value of lo-resolution timer	Upper limit for the measurement	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1106)	The millisecond timer is not working correctly (the measured values are too small). Possible cause(s): ■ Hardware fault ■ Software error	Measured value of millisecond timer	Lower limit for the measurement	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1107)	The millisecond timer is not working correctly (the measured values are too large). Possible cause(s): ■ Hardware fault ■ Software error	Measured value of millisecond timer	Upper limit for the measurement	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1108)	Not Used				
ERRVAL(1109)	The format of the info sector in the flash memory is incorrect. Possible cause(s): ■ Module was never initialized (with the Clear All command) ■ Hardware fault in flash memory	always 0	always 0	Configure module If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1110)-ERRVAL(1114)	The format of the configuration data in the flash memory is incorrect. Possible cause(s): ■ Module was configured incorrectly ■ Hardware fault in flash memory	Detailed information on the cause of the error	Detailed information on the cause of the error	Configure module If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1115)	The format of the configuration data in the flash memory is incorrect. Possible cause(s): ■ Module was configured incorrectly ■ The configuration data specifies an unsupported expansion module Hardware fault in flash memory	1 Unsupported expansion module 2–39 Error information 40 More global flags than allowed in the configuration 41 More than five unsafe CAN receive variables	If reason code 0 = 1: Module type If reason code 0 = 2–32: Detailed information	Configure the module, check whether an unsupported expansion module is connected, and, if applicable, use a Safety CPU with up-to-date firmware; check the number of CAN variables If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1116)-ERRVAL(1128)	The format of the configuration data in the flash memory is incorrect. Possible cause(s): ■ Module was configured incorrectly ■ Hardware fault in flash memory	Detailed information on the cause of the error	Detailed information on the cause of the error	Configure module If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1129)	An error occurred during communications with the FPGA. Possible cause(s): ■ Hardware fault (FPGA)	Value that provides more detailed information regarding the error 0: SPI access 0 failed 1: SPI access 1 failed 2: DMA no long working 3: Invalid FPGA version 4: FPGA not ready	Value that provides more detailed information regarding the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1130)	A buffer was passed as a parameter in a function call. However, the size of the buffer is too large. Possible cause(s): ■ Program error	Actual size of the buffer	Maximum size of the buffer	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1131)	A wrong frame type was used for the SSDO command. Possible cause(s): ■ Error in the safety tool	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1132)	A duplicate value is found in the module safety numbers for the remote modules. Possible cause(s): ■ An error occurred while determining the module safety number based on the topology path.	Reason-Code 0: < 1024: Index of module in the table of remote modules Reason-Code 0: = 1024: A safety number of a remote module is the same as the system's own safety number Reason-Code 0: = 1025: When retrieving the system's own safety number, the system's own safety number was not received	Duplicate module safety number	- Restart the application Check all safety numbers to make sure they are unique	Error LED of the safety module lights up red
ERRVAL(1133)	The microcontroller is not in the correct processor mode. Either the flags for the processor mode or for activating the interrupts are not set correctly in the program status register or the wrong stack is being used. Possible cause(s): ■ The processor mode change is not working correctly (due to a hardware fault or software error)	Set flags in the program status register or stack pointer value	Target value for the flags in the program status register or lower-/upper limit of the stack range that must be used	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1134)	The computed output image is different in the two controllers.	States of the local outputs of your own controller	States of the local outputs of the other controller	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1135)	The supply voltage for an expansion module is not OK. Possible cause(s): ■ External power supply not connected ■ Hardware fault	0: If the CPU module's supply voltage is not OK 1: If, during the watchdog test, an error that occurred due to a missing supply voltage was detected	Module position (0 = 1st expansion module)	Make sure that the external power supply for the clock signal outputs for safety modules with digital inputs and the supply voltage of the outputs for safety modules with digital outputs are correct. After applying the correct voltage, you can reset the error with "Quit Error" or by restarting the system.	Error LED of the safety module lights up red DCOK LED does not light up.
ERRVAL(1136)	The Boolean representation of an output has an invalid value. Possible cause(s): ■ Software error (memory was overwritten at illegal point) ■ Hardware error (RAM defective)	0x00000000 means that the error was detected when decoding a cell in the application memory. 0xFFFFFFFF means that the error was detected when coding the SAFE BOOL value for the SPDO frame.	Invalid value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1137)	The values of the readback outputs do not match the values of the outputs' target values. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External fault (e.g., short-circuit, output supply voltage missing) ■ Software error	Setpoint values (bit 0 = 1st output)	Actual values (bit 0 = 1st output)	Check power supply If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1138)	An error occurred during the watchdog test. Possible cause(s): ■ Hardware fault	Bit mask for identifying the test steps in which the error was detected	Current test step	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1139)-ERRVAL(1141)	Invalid parameters were specified for the COPY command in the configuration. Possible cause(s): ■ An incorrect configuration has been imported ■ Flash defective	Index of the source or destination	Maximum number of existing cells or lower or upper index limit	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1142)	After the high-side output switch (HSS) was switched on, at least one output switched to a value of 1 even though none of the low-side output switches (FET) were switched on. This is a permanent error that cannot be acknowledged. Possible cause(s): ■ Output semiconductor (FET) defective	Setpoint value (bit 0 = 1st output) always 0	Actual value, affected outputs (bit 0 = 1st output)	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1143)	After the HSS output switch was switched off, none of the previously set outputs switched to 0. Possible cause(s): ■ Output semiconductor (HSS) defective ■ Impermissibly large capacitive load	Output values after the HSS was switched off (bit 0 = 1st output)	Previously set outputs (bit 0 = 1st output)	Please contact Eaton	The safety module's Error LED will flash
ERRVAL(1144)	After the FET output switch was switched off, the corresponding output did not switch to 0. Possible cause(s): ■ Output semiconductor (FET) defective ■ Impermissibly large capacitive load	Affected output (bit 0 = 1st output)	always 0	Please contact Eaton	The safety module's Error LED will flash

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1145)	After the FET output switch was switched off, an output that was not connected through this FET changed its value. Possible cause(s): ■ Internal cross-fault short-circuit in output circuit	Status of outputs (bit 0 = 1st output)	Target status of outputs (bit 0 = 1st output)	Please contact Eaton	The safety module's Error LED will flash
ERRVAL(1146)	Cyclical processing has not been completed within the permitted time. Possible cause(s): ■ Incorrect application	Caller ID with which runtime triggering occurred the last time	Code address that will be used to resume after the interrupt function is ended	Increase the Safety CPU's cycle time	Error LED of the safety module lights up red
ERRVAL(1147)	Logical program run monitoring detected that a program section has not been called for an impermissibly long time. Possible cause(s): ■ Software error	Safety-CPU XS-102: < 1000 Program section index ≥1000 ID of the program section where the error was detected 1000–1009	Safety-CPU XS-102: Reason code 0: < 1000: Program section time that must be adhered to	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1148)	HS transistor readback is not OK. Possible cause(s): ■ Hardware fault ■ No supply voltage	Expansion module position (0 = 1st module)	Value of HS signal	Check power supply If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1149)	An SSI expansion module detected an incorrectly encoded value at an input. Possible cause(s): ■ Hardware fault	Module position and incorrectly encoded value	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1150)	The input value has not been refreshed for an impermissibly long time. Possible cause(s): ■ Fault in input hardware ■ Continuous change in input signal	Affected input (bit 0 = 1st input)	Maximum age	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1151)	The reserved stack areas were overwritten (lower limit). Possible cause(s): ■ Software error ■ Stack reserve sized too small	ID of affected stack area	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1152)	The reserved stack areas were overwritten (upper limit). Possible cause(s): ■ Software error ■ Stack reserve sized too small	ID of affected stack area	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1153)	The computer output image is different in the controllers. Possible cause(s): ■ Software error	CRC of transmission SPDO frames and the app memory of your own controller	CRC of transmission SPDO frames and the app memory of the other controller	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1154)	The information regarding the stored safety numbers for the remote modules is inconsistent. Possible cause(s): ■ Program error	Index of searched module	Number of elements in the table of the remote modules	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1155)	The CPU test detected an error. Possible cause(s): ■ CPU error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1156)	The encoder (incremental encoder) did not read any encoder values, since the CPLD did not start any read operations. Possible cause(s): ■ The standard application was not started ■ Hardware class for the incremental encoder module has not been placed.	always 0	always 0	Restart the application Check whether the hardware layout matches the application; modify the application accordingly.	

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1157)	The maximum jitter between two read operations for the encoder value was exceeded. Possible cause(s): ■ Invalid configuration with hardware classes ■ Hardware fault	always 0	always 0	Check the versions of the FPGA and hardware class	
ERRVAL(1158)	The CPLD for the encoder reported a fault, which meant that no encoder values could be read. Possible cause(s): ■ Incorrect settings for the encoder ■ Encoder is not connected ■ Encoder is not supported ■ Encoder is faulty	always 0	always 0	Check the encoder data sheet to see whether the parameters for the encoders are correct in the standard application and have been applied in the safe application Check the cable connection to the encoders Check whether the encoder used is supported as per the data sheet Check whether there is a faulty encoder	
ERRVAL(1159)	The data read by the CPLD is invalid. Possible cause(s): ■ Incorrect settings for the encoder ■ The settings for the encoder were changed at runtime. ■ Encoder is faulty	always 0	always 0	Check the encoder data sheet to see whether the parameters for the encoders are correct in the standard application and have been applied in the safe application Check the cable connection to the encoders Check whether there is a faulty encoder	
ERRVAL(1160)	The supply voltage for the encoder is not OK. Possible cause(s): ■ The supply voltage for the encoders is not present ■ Hardware fault	always 0	always 0	Check the wiring for the encoders' supply voltage. The error message will be generated in the event of an undervoltage or overvoltage in the 24 V encoder power supply, which must be externally connected, at the SNC 021/SSI 021(X3). The supply voltage tolerances of the power supply can be found in the HW documentation for the SNC 021/SSI 021.	
ERRVAL(1161)	There is an error in the configuration for the module. Possible cause(s): ■ Encoder connected even though it is not used as per the Safety application	always 0	always 0	Check whether the parameters were set correctly for the connected encoder in the Safety application.	

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1162)	When testing the incremental encoder's signals, the first RS-422 driver detected a signal error. Possible cause(s): ■ Signal cable cross-fault short-circuit. ■ Wire break ■ Short-circuit	always 0	always 0	Check the encoder's wiring and the encoder itself.	Safe DATA OK LED of affected encoder channel does not light up Encoder status LED of affected encoder flashes
ERRVAL(1163)	When testing the incremental encoder's signals, the second RS-422 driver detected a signal error. Possible cause(s): ■ Signal cable cross-fault short-circuit. ■ Wire break ■ Short-circuit	always 0	always 0	Check the encoder's wiring and the encoder itself.	Safe DATA OK LED of affected encoder channel does not light up Encoder status LED of affected encoder flashes
ERRVAL(1164)	A signal error was detected when testing the signals from the incremental encoder's Z channel. Possible cause(s): ■ Signal cable cross-fault short-circuit. ■ Wire break ■ Short-circuit	always 0	always 0	Check the encoder's wiring and the encoder itself.	Safe DATA OK LED of affected encoder channel does not light up Encoder status LED of affected encoder flashes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1165)	When testing the signals from the incremental encoder's Z channel, the system detected that signal cables Z+ and Z- were swapped. Possible cause(s): ■ Swap signal cables Z+ and Z-	always 0	always 0	Check the encoder's wiring	Safe DATA OK LED of affected encoder channel does not light up The affected encoder's encoder status LED flashes
ERRVAL(1166)	No Z pulse was detected even though the motion in question should have resulted in a Z pulse. Possible cause(s): ■ The configured encoder resolution does not match the encoder's actual resolution ■ Faulty encoder	always 0	always 0	Check the encoder's resolution in the parameters for the SNC 021 in XN Safety Designer Check the encoder's wiring and the encoder itself.	Safe DATA OK LED of affected encoder channel does not light up
ERRVAL(1167)	Too few edges were counted on the A and B signals between two Z pulses. Possible cause(s): ■ The configured encoder resolution does not match the encoder's actual resolution ■ Faulty encoder	always 0	always 0	Check the encoder's resolution in the parameters for the SNC 021 in XN Safety Designer Check the encoder's wiring and the encoder itself.	Safe DATA OK LED of affected encoder channel does not light up
ERRVAL(1168)-ERRVAL(1171)	An input or output of an expansion module is in its error condition. Possible cause(s): ■ Error in the expansion module	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1172)	Not Used				

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1173)	The values of the readback outputs do not match the target values. The target value is 1, but the actual value is 0. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External fault (e.g., short-circuit, output supply voltage missing) ■ Software error	always 0	always 0	Check power supply If the error continues to occur: Please contact Eaton	The LED of the safe output flashes
ERRVAL(1174)	The values of the readback outputs do not match the target values. The target value is 0, but the actual value is 1. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External fault (e.g., short-circuit, output supply voltage missing) ■ Software error	always 0	always 0	Check power supply If the error continues to occur: Please contact Eaton	The LED of the safe output flashes
ERRVAL(1175)	The values of the readback outputs do not match the target values. Possible cause(s): ■ Hardware fault (e.g., faulty output transistor) ■ External fault (e.g., short-circuit, output supply voltage missing) ■ Software error	always 0	always 0	Check power supply If the error continues to occur: Please contact Eaton	The LED of the safe output flashes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1176)	The firmware update for a local expansion module has failed. Possible cause(s): ■ Incorrect update file ■ Flashing the image in the expansion module is not possible due to a hardware defect. ■ Error in the expansion module ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1177)-ERRVAL(1180)	Not Used				
ERRVAL(1181)	The input's safe measurement range was fallen below. Possible cause(s): ■ Hardware fault ■ Wiring faults	always 0	always 0	Check the wiring If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1182)	The input's safe measurement range was exceeded. Possible cause(s): ■ Hardware fault ■ Wiring faults	always 0	always 0	Check the wiring If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1183)	The tolerance for the end value between the controllers was exceeded. Possible cause(s): ■ Hardware fault	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1184)-ERRVAL(1188)	Not Used				

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1189)	The input's safe measurement range was fallen below. The error will be reset automatically if the reading becomes valid again. Possible cause(s): ■ Hardware fault ■ Wiring faults	always 0	always 0	Check the wiring If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1190)	The input's safe measurement range was exceeded. The error will be reset automatically if the reading becomes valid again. Possible cause(s): ■ Hardware fault ■ Wiring faults	always 0	always 0	Check the wiring If the error continues to occur: Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1191)-ERRVAL(1196)	Not Used				
ERRVAL(1197)	There is an incorrect CRC2 in the configuration. The module's status cannot be changed to "configured." Possible cause(s): ■ Hardware fault ■ Incorrect configuration	always 0	always 0	Check the configuration to make sure it is valid If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1198)	The internal counter for the cycle time is working incorrectly. Possible cause(s): ■ Software error	Last time difference	Maximum time difference	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1199)	The remote safe input value has exceeded the maximum age. Possible cause(s): ■ Bus transmission error ■ Fault in the remote module's hardware ■ Software error in remote module	always 0	always 0	Check whether standard communications with every required safety module are intact. In addition, make sure that none of the required safety modules are in their error condition. Moreover, it is possible that the configured maximum transmission time is too short. Check all these. If the error is fixed, you can reset it with Quit Error or by restarting the system.	The safety module's Error LED will flash

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1200)	Not Used				
ERRVAL(1201)-ERRVAL(1205)	An unexpected exception has occurred. Possible cause(s): ■ Software error	Address of interrupted code	In the event of an exception outside of the FSB call: - always 0 In the event of an exception during the FSB call: - FSB ID in the two least significant bytes - Object index in the two most significant bytes	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1206)	The controllers' system times have an excessive large deviation between them. This is a permanent error that cannot be acknowledged. Possible cause(s): ■ Hardware fault	Calculated drift between the two controller system times	Maximum permissible drift	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1207)	The module type and module version in the firmware do not match the values stored in the non-editable module parameters. Possible cause(s): ■ Wrong firmware	The bootloader's module type and version	The firmware's module type and version	Please contact Eaton	Error LED of corresponding safety module lights up red
ERRVAL(1208)-ERRVAL(1209)	The non-editable module parameters are different in the controllers. Possible cause(s): ■ Hardware fault ■ Flash memory programmed incorrectly	Your own controller's module parameters	The other controller's module parameters	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1210)	The firmware versions are different in the controllers. Possible cause(s): ■ Wrong firmware	Firmware version in your own controller	Firmware version in the other controller	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1211)-ERRVAL(1212)	An error was detected during the ROM test in a read-only flash memory area. This is a permanent error that cannot be reset. Possible cause(s): ■ Flash defective	Contains more detailed information regarding the error's cause	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1213)-ERRVAL(1217)	An error was detected during the ROM test in an editable flash memory area. Possible cause(s): ■ Flash defective ■ The module was turned off during flash programming	Contains more detailed information regarding the error's cause	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1218)	After the low-side output switch (FET) was switched on, at least one output switched to a value of 1 even though none of the high-side output switches (HSS) were switched on. Possible cause(s): ■ Output semiconductor (HSS) defective	Setpoint value (bit 0 = 1st output) always 0	Actual value, affected outputs (bit 0 = 1st output)	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1219)	The content of an SSDO response is different in the two controllers. Possible cause(s): ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1220)	An attempt was made to write to the flash memory with an invalid length. Possible cause(s): ■ Program error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1221)	An attempt was made to read from a data buffer that contained less than the requested data volume. Possible cause(s): ■ Software error ■ Faulty Safety tool (SSDO command)	Number of bytes that should be read from the buffer	Size of buffer	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1222)	The time for test mode or the maximum idle time has elapsed. Possible cause(s): ■ Test mode time has elapsed; PLC is currently being reset	0: The time for test mode has elapsed	always 0	If test mode has elapsed, download the configuration again and restart or reset the error and restart	Error LED of the safety module lights up red
		1: The max. idle time has elapsed 2: The max. check config time has elapsed	always 0	If the IDLE time or check config time has elapsed, reset the error and restart. Start the PLC if it is being reset	
ERRVAL(1223)	The timing in the internal test is incorrect. An attempt was made to start a test while the previous test had not yet ended. Possible cause(s): ■ Software error	Bit mask for the inputs or number of faulty test step	More detailed information regarding the error's cause	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1224)	The timing in the internal test is incorrect. An attempt was made to start a test while the previous test had not yet ended. Possible cause(s): ■ Software error	Bit mask for the inputs being tested	Bit mask for the inputs previously tested	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1225)	The configuration state cannot be set to "configured," since the CRC for the areas belonging to the configuration are invalid. Possible cause(s): ■ A firmware update has been carried out ■ Module was configured incorrectly ■ Hardware fault in flash memory	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1226)	The configuration state cannot be set to "configured," since the CRC passed by the caller does not match the actual CRC. The module will not switch to its error condition. The error will be shown in the console pane in XN Safety Designer. Possible cause(s): ■ Module was configured incorrectly ■ Hardware fault in flash memory	always 0	always 0	The safety module must be reconfigured If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1227)	The format of the configuration data in the flash memory is incorrect. Possible cause(s): ■ Module was configured incorrectly ■ Hardware fault in flash memory	Actual configuration state	Target configuration state	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1228)	The configuration state cannot be set to "configured," since the configuration has been deleted. Possible cause(s): ■ Module was configured incorrectly ■ Hardware fault in flash memory	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1229)	The configuration is not in the state in which the verified flag can be set. Possible cause(s): ■ The configuration has never been deployed ■ The configuration was written to the Safety CPU in Developer Mode	always 0	always 0	Do not start the program in Developer Mode Restart the program to trigger the deployment	Error LED of the safety module lights up red
ERRVAL(1230)	The SSDO command cannot be run due to missing permissions. Possible cause(s): ■ Error in XN Safety Designer	always 0	always 0	Go back online with XN Safety Designer and run the commands again	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1231)	It was not possible to open a session when going online. Possible cause(s): ■ Another instance of XN Safety Designer is already online	always 0	always 0	Check whether there are multiple instances of XN Safety Designer online; only one instance of XN Safety Designer should be trying to connect	Error LED of the safety module lights up red
ERRVAL(1232)	The max. number of remote modules has been exceeded.	Number of remote modules	Maximum number of possible remote modules	Check the number of remote modules and change it accordingly	Error LED of the safety module lights up red
ERRVAL(1233)	The module type coded in the CFG pins does not match the firmware, or an unsupported I/O expansion card is connected. Possible cause(s): ■ Wrong firmware	Bit 31 = 0: CFG pins as per hardware Bit 31 = 1: ID of unsupported expansion card Bit 30 = 1: Vendor ID of unsupported expansion card Bit 29 = 1: ID of unsupported Safety interface module	If reason code 0 bit 31 = 0: CFG pins as per hardware If reason code 0 bit 31 = 1: Always 0 If reason code 0 bit 30 = 1: Device ID of unsupported expansion card	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1234)	The SSDO command cannot be run in this runtime state. Possible cause(s): ■ Error in the safety tool	always 0	always 0	Go back online with XN Safety Designer and run commands again	Error LED of the safety module lights up red
ERRVAL(1235)	The parameters in the firmware header of the new firmware do not match this module. Possible cause(s): ■ Wrong firmware	Parameters in the firmware	Parameters in the boot code	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1236)	An error occurred while attempting to access the SD card. Possible cause(s): ■ SD card was removed ■ Faulty or incompatible SD card	Error cause identification	Error that last occurred	Check whether the SD card has been removed. If the SD card has not been removed, replace the SD card. You can reset the error by removing the SD card and running a "Quit Error" command If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1237)	Write/read error in configuration interpreter code. One of the following errors was detected when checking the configuration: a) An attempt was made to write to a cell to which writing was already carried out from a different source (e.g., if there are two COPY commands with the same target or if a COPY command specifies a read-only cell as a target). b) An attempt was made to read from an unreadable cell. A cell is unreadable if there is no corresponding source from which it was written to (not initialized) or if it is not allowed to be used as a source for a COPY command (e.g., the object memory header). Possible cause(s): ■ A faulty configuration has been imported ■ Flash defective	Index of faulty cell	0 Write error 1 Read error 2 For FSoE: A remote input is being written to by more than one source	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1238)	The POST (power-on self-test) was not completed. Possible cause(s): ■ Software error	Bit mask of pending POST test steps.	always 0	An error was detected during the boot test. XN Safety Designer will show which error was detected. The error can be fixed after this. If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1239)	The sector buffer that should be written to the SD card is different in the two controllers. Possible cause(s): ■ Software error	Number of sector buffer that should be written	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1240)	Force mode was ended impermissibly. Possible cause(s): ■ A session timeout or login timeout, e.g., due to an interruption in online communications with XN Safety Designer ■ Change in login level (e.g., logout) ■ Session closed in XN Safety Designer	always 0	always 0	Reset the error if there was an interruption in communications between XN Safety Designer and the module	Error LED of the safety module lights up red
ERRVAL(1241)	The format of the passwords on the SD card is incorrect. Possible cause(s): ■ Faulty SD card ■ Content was changed afterwards	always 0	always 0	Try to write the content to the SD card again. If the error keeps happening, replace the SD card If the error continues to occur: Contact Eaton	Error LED of the safety module lights up red
ERRVAL(1242)	Internal error when processing the STDOs. Possible cause(s): ■ Software error	Length of STDO being written	Maximum length of an STDO	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1243)	The information regarding the module safety number of a remote module is required, but is not available or is invalid. Possible cause(s): ■ Software error	Value of invalid module safety number (0 or 0xFFFFffff)	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(1244)	The configuration interpreter code specifies an FSB ID for which no corresponding code exists. Possible cause(s): ■ Error in the safety tool	FSB ID	always 0	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1245)	A value specified in the configuration data is too large. Possible cause(s): ■ An excessively large value was entered in XN Safety Designer	Specifies which parameter is too large 0 Test time in temporary Operational status 1 Cycle time for runtime monitoring 2 Maximum age of a remote input 3 Filter time of a local input 4 Watchdog time of a Safety interface input 5 Optional flag of a Safety interface input 6 Cfg major version 7 Unknown data after the container header 8 There is no 0 at the end of the function block area 9 Invalid length for the FSoE slave parameters 10 Invalid value in the FSoE configuration 11 Invalid FSoE connection ID	Limit	Set the value in the configuration correctly (enter a smaller value)	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
		12 Invalid FSoE slave address 13 Saved FSoE addresses in the controllers different or required number of addresses does not match 14 Slave address used to respond to the C_GET_ SLAVEADDR command could not be unambiguously determined			
ERRVAL(1246)	Error in the function blocks. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1247)	Error in the function blocks; overflow in internal calculations. Possible cause(s): ■ Software error ■ Wrong parameters ■ SafeDINT values outside the permissible range			Check the parameters and the configured value range. If the error continues to occur: Please contact Eaton	
ERRVAL(1248)	Error in the function blocks, wrong SafeDINT values. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1249)	Error in the function blocks, wrong internal status. Possible cause(s): ■ Software error			Please contact Eaton	

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(1250)	Error in the function blocks, wrong parameters. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1251)	Error in the function blocks, wrong SafeBool values. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1252)	Error in the function blocks, wrong function block ID. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1253)	Error in the function blocks, wrong terminator. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1254)	Error in the function blocks, wrong start delimiter. Possible cause(s): ■ Software error			Please contact Eaton	
ERRVAL(1255)	The function block has returned an error code. Possible cause(s): ■ Software error	always 0	always 0	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10000)	The encoder's external supply voltage is not OK. Possible cause(s): ■ External supply voltage not connected ■ Hardware fault	Byte 0 = 0; On boot, the system detected that the supply voltage was not OK Byte 1: Position of module downstream of Safety CPU	always 0	Check the supply voltage. If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10001)	The maximum permissible current of an encoder was exceeded (please refer to "Technical data" in the product data sheet for the SNC 021 module). Possible cause(s): ■ Wiring faults ■ Cross-circuit ■ Short-circuit ■ Use of an incorrect encoder	Byte 0: 0: Max. current of first encoder exceeded 1: Max. current of second encoder exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Max. permissible current in 10 µA Byte 2 and 3: Measured encoder current in in 10 µA	Check the encoder's current Check the encoder's wiring and the encoder itself. If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red
ERRVAL(10002)	The current measured for an encoder is lower than the allowed limit. Possible cause(s): ■ Wiring faults ■ Cable break ■ Faulty encoder ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been fallen below 1: The limit for encoder 2 has been fallen below Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Lower limit for allowed current in 10 µA Byte 2 and 3: Measured encoder current in in 10 µA	Check the encoder's wiring and the encoder itself. If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red
ERRVAL(10003)	The current measured for an encoder is higher that the allowed limit. Possible cause(s): ■ Wiring faults ■ Cross-circuit ■ Short-circuit ■ Faulty encoder ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been exceeded 1: The limit for encoder 2 has been exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Upper limit for allowed current in 10 µA Byte 2 and 3: Measured encoder current in in 10 µA	Check the encoder's wiring and the encoder itself. If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10004)	The voltage offset for current monitoring measured for an encoder is lower than the allowed limit when the supply voltage is cut off. Possible cause(s): ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been fallen below 1: The limit for encoder 2 has been fallen below Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Lower limit for the allowed voltage offset in mV Byte 2 and 3: Measured voltage offset of encoder in mV	Replace the SNC module	Error LED of Safety CPU lights up red
ERRVAL(10005)	The voltage offset for current monitoring measured for an encoder is higher than the allowed limit when the supply voltage is cut off. Possible cause(s): ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been exceeded 1: The limit for encoder 2 has been exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Upper limit for the allowed voltage offset in mV Byte 2 and 3: Measured voltage offset of encoder in mV	Replace the SNC module	Error LED of Safety CPU lights up red
ERRVAL(10006)	The supply voltage measured for an encoder is lower than the allowed limit. Possible cause(s): ■ Wiring faults ■ Cable break ■ Cross-circuit ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been fallen below 1: The limit for encoder 2 has been fallen below Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Lower limit for the allowed voltage in mV Byte 2 and 3: Measured encoder voltage in mV	Check the external power supply's and the encoders' wiring If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10007)	The supply voltage measured for an en encoder is higher than the allowed limit. Possible cause(s): ■ Wiring faults ■ Hardware fault	Byte 0: 0: The limit for encoder 1 has been exceeded 1: The limit for encoder 2 has been exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Upper limit for the allowed voltage in mV Byte 2 and 3: Measured encoder voltage in mV	Check the external power supply's and the encoders' wiring If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red
ERRVAL(10008)	The parameters for the current setpoint are invalid. Possible cause(s): ■ Software error ■ Incompatible Safety CPU (see product data sheet for SNC 021 module)	Byte 0: Error information Byte 1: Position of module downstream of Safety CPU	always 0	Check Safety CPU If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red
ERRVAL(10009)	An error was detected when checking the checksum for the constant elements for current monitoring in RAM. Possible cause(s): ■ Software error ■ RAM faulty	Byte 0: Error information Byte 1: Position of module downstream of Safety CPU	always 0	Replace the SNC module	Error LED of Safety CPU lights up red
ERRVAL(10010)	The two RS-422 drivers are returning different results for at least one encoder signal. Possible cause(s): ■ Hardware fault	Byte 0: Information indicating which signal is affected (bit 0 = signal A; bit 1 = signal B, bit 2 = signal Z) Byte 1: Position of module downstream of Safety CPU	always 0	Replace the SNC module	Error LED of Safety CPU lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10011)	The max. permissible input or counter frequency was exceeded (please refer to the product data sheet for the SNC 021 module). Possible cause(s): ■ The encoder has an excessively high resolution ■ The encoder is being operated at an excessive speed	Byte 0: 0: The maximum permissible input or counter frequency for encoder 1 was exceeded 1: The maximum permissible input or counter frequency for encoder 2 was exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 1: Maximum permissible counter frequency in increments/ms Byte 2 and 3: Measured counter frequency in increments/ms	Check the encoder in terms of its resolution and the speed at which it is being operated. If the error continues to occur: Please contact Eaton	Error LED of Safety CPU lights up red
ERRVAL(10012)	The configured encoder resolution falls outside the defined limits (please refer to the product data sheet for the SNC 021 module). Possible cause(s): ■ The configured encoder resolution is too high	Byte 0: 0 The maximum configured resolution for encoder 1 was exceeded 1 The maximum configured resolution for encoder 2 was exceeded Byte 1: Position of module downstream of Safety CPU	Byte 0 and 3: Configured encoder resolution	Check the encoder's resolution. If the error continues to occur: Please contact Eaton	

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10013)	The 3V3 supply voltage is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10014)	The 3V3 voltage is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10015)	The 5V0 supply voltage is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10016)	The 5V0 voltage is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10017)	The 5V1 supply voltage is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10018)	The 5V1 voltage is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10019)	The 15V0 supply voltage is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10020)	The 15V0 voltage is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10021)	The 24V1 supply voltage is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10022)	The 24V1 voltage is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10023)	The analog voltage monitoring has reported an excessively low value. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10024)	The analog voltage monitoring has reported an excessively high value. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: Index of the ADC channel Byte 1: Module position	Permitted limit value	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10025)	The calibration data is incorrect or could not be read. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error 0 Read error 1 Wrong length 2 Wrong CRC 3 Wrong version 4 Wrong module ID 5 Invalid controller ID	serves to identify the cause of the error	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10026)	The voltage supplied by the 24 V sensor power supply is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: 0 = Error triggered by measurement; 1 = Error triggered by DCOK pin Byte 1: Module position	serves to identify the cause of the error Bytes 0–1: Allowed voltage limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10027)	The voltage supplied by the 24 V sensor power supply is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 1: Module position	serves to identify the cause of the error Bytes 0–1: Allowed voltage limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10028)	The common-mode voltage (CMVR) is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: CMVR index (0–3) Byte 1: Module position	serves to identify the cause of the error Bytes 0–1: Allowed voltage limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10029)	The common-mode voltage (CMVR) is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: CMVR index (0–3) Byte 1: Module position	serves to identify the cause of the error Bytes 0–1: Allowed voltage limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10030)	The temperature or an ADC reference is higher than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: 0 = ADC reference; 1 = Temperature Byte 1: Module position	serves to identify the cause of the error Byte 0-1: Permitted limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10031)	The temperature or an ADC reference is lower than the allowed limit. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 0: 0 = ADC reference; 1 = Temperature Byte 1: Module position	serves to identify the cause of the error Byte 0-1: Permitted limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red

15. Error codes

Error variable	Description	Reason-Code 0	Reason-Code 1	Measures	LED status indication
ERRVAL(10032)	The calibration or offset voltage is lower than the allowed limit	serves to identify the cause of the error Byte 0: Bits 0–3 channel index Byte 1: Module position Byte 2–3: Extended error code	serves to identify the cause of the error Byte 0-1: Permitted limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10033)	The calibration or offset voltage is lower than the allowed limit	serves to identify the cause of the error Byte 0: Bits 0–3 channel index Byte 1: Module position Byte 2–3: Extended error code	serves to identify the cause of the error Byte 0-1: Permitted limit Byte 2-3: Measured value	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10034)	The ADC received unexpected data. Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 1: Module position	serves to identify the cause of the error Bytes 0–3: Received data	Please contact Eaton	Error LED of the safety module lights up red
ERRVAL(10035)	A brownout fault occurred Possible cause(s): ■ Hardware fault	serves to identify the cause of the error Byte 1: Module position Byte 2–3: Extended error code	always 0	Please contact Eaton	Error LED of the safety module lights up red

Appendix

A.1 Further usage information 441
A.2 Shortcuts - Keyboard shortcuts and special keys 442
Alphabetical index 443

A.1 Further usage information

Hardware

For more information and information on additional XN Safety module, please refer to the following documentation:

XN Safety system manual	MN050020
Installation instructions XS-102-C00-000, XS-322-...	IL050036ZU
Manual XN Safety CPU module XS-102-C00-000	MN050019
Manual XN Safety module XS-322-10DI-PF	MN050021
Manual XN Safety module XS-322-8DO-P20	MN050022
Manual XN Safety module XS-322-8DIO-PF20	MN050023
Manual XN Safety module XS-322-2INC	MN050024
Manual XN Safety module XS-322-4AI-I2	MN050025
Manual XN Safety module XS-322-2DO-RNOAC	MN050026
Manual XN Safety module XS-322-2DO-RNODC	MN050027

Download Center, Eaton Online Catalog

Enter "xnsafety" into the search box and the catalog will take you directly to the corresponding product group in the Automation, Control and visualization section.

Eaton.com/documentation

Eaton.com/ecat

Product information

For up-to-date information, please consult the product page on the Internet.

Eaton.com/xnsafety

A.2 Shortcuts - Keyboard shortcuts and special keys

Frequently required tasks can be carried out quickly using function keys and key combinations. Following is an overview:

A.2.1 Function keys and key combinations

A.2.1.1 Keyboard shortcuts in the FILE menu

Purpose	Keyboard shortcuts
Open new project	Ctrl + N
Open existing project	Ctrl + O
Print file	Ctrl + P
Save project	Ctrl + S

A.2.1.2 Keyboard shortcuts in the EDIT menu

Purpose	Keyboard shortcuts
Undo operation	Ctrl + Z
Restore operation	Ctrl + Y

A.2.1.3 Function key in BUILD menu

Purpose	Key/key combination
Generate project	F9

A.2.1.4 Function keys and keyboard shortcut in DEBUG menu

Purpose	Key/key combination
Online	Alt + F6
Reset forced values	F8
Online Status display	F6

Alphabetical index

A

ABS (Absolute value) ... 173
ABS_NS (absolute value, not safe) ... 173
ADD (Addition) ... 173
ADD_NS (Addition, not safe) ... 174
Adding comments ... 73
Adding network elements ... 68
After Sales Service ... 2
Analog-input module;Description ... 375
AND function ... 198
AND_5I function ... 199
AND_BW (AND, bitwise) ... 179
AND_NS function ... 199
Archiving ... 122

B

BOOL_TO_SAFEBOOL ... 169
Brand names
    Product names ... 2

C

Calculating the monitoring settings ... 97
Changing the column width ... 78
Collision_Detection ... 211
Column for inputs ... 59
Column for outputs ... 62
Company information ... 2
Comparative function blocks ... 189
Complex function blocks ... 210
Configuring network elements ... 69
Copying, cutting, and pasting elements ... 74
Copy-protected ... 2
Copyright ... 2
Counter function blocks ... 206
Cross Circuit Detection ... 81
CTD ... 207
CTU ... 207
CTUD ... 208
Cycle time estimate for user program ... 100

D

Debugging ... 101
Deleting comments ... 73
Deleting connections ... 71
Deleting network elements ... 69
Description ... 374
Digital input module;description ... 376
Digital input/output module;Description ... 378
Digital output module;Description ... 377
DINT_Compare ... 197
DINT_TO_SAFEDINT ... 170
Directly jumping to temp. variables ... 77
Directly jumping to the definition of an element 75
Directly jumping to the start points / end points of a connection ... 76
Download Center ... 441
Download Wizard ... 154
Drawing connections ... 70

E

ecat ... 441
Editor ... 58
Enable signal for safe outputs ... 85
Entering comments ... 74
EQ (equal) ... 192
EQ_NS (equal, not safe) ... 192
Error codes ... 382
Error messages when trying to add an element 79

to a network
Export of download data for a safety CPU 160

F

F_TRIG 206
Filling the input column 67
Filling the output column 67
Filter Time 81
FlipFlop function blocks 208
Forcing 102
Function block-specific errors and status codes 220
Further reading 441

G

GE (greater than or equal to) 190
GE_NS (greater than or equal to, not safe) 190
General usage instructions 67
Get Module State 159
GT (greater than) 189
GT_NS (greater than, not safe) 190

H

Hardware Revision 82

I

Incremental encoder module;Description 381

K

Keyboard shortcuts 442
Programming 442

L

LE (less than or equal to) 191
LE_NS (less than or equal to, not safe) 192
LIMIT (within limits) 195
LIMIT_NS (within limits, not safe) 195
Logical function blocks 198
LT (less than) 191
LT_NS (less than, not safe) 191

M

Manuals 441
MAX (Maximum) 194
Max. Cycle Time 81
MAX_NS (Maximum, not safe) 194
Maximum Transmission Time 81
Maximum Transmission Time (Output as Input) 81
MIN (Minimum) 193
MIN_NS (Minimum, not safe) 193
Moving comments 73
Moving network elements 69
MULDIV (multiplication and division) 176
MULDIV_NS (Multiplication and Division, not safe) 175

N

Network editor 63
Network elements 64
NOT function 200
NOT_BW (NOT, bitwise) 180

O

Online actions 152
Online actions and download 149
Online Catalog 441
Optional modules 87
OR function 200
OR_5I function 201
OR_BW (OR, bitwise) 179
Original operating manual 2
Overview of actions allowed by each password level 158

P

Parameter Revision ... 82
Password mechanism ... 123
Passwords ... 157
Path Speed ... 183
PDO frame length ... 100
Printing ... 114

R

R_TRIG ... 206
Relay DC output module;Description ... 379
Relay output module;Description ... 380
Replacing existing elements ... 75
Replacing placeholders ... 75
Resizing comments ... 74
Resource consumption ... 99
Resource consumption and response time ... 98
Restart_Interlock ... 181
RS_Flipflop ... 209

S

SAFEBOOL_TO_BOOL ... 170
SAFEDINT_TO_DINT ... 170
SAFEDINTERROR_TO_SAFEDINT ... 171
Saving as an XML file ... 122
SDINT_Compare ... 196
Selecting connections ... 71
Selecting network elements ... 68
SERROR_TO_SFALSE ... 171
Settings in the project ... 81
SF_Antivalent function ... 219
SF_Antivalent function Error behavior ... 220
SF_Antivalent function Error detection ... 220
SF_Antivalent function Function block-specific errors and status codes ... 220
SF_Antivalent function Status Diagram ... 221
SF_Antivalent function Time Diagram ... 222
SF_DirectionMonitor ... 351
SF_DirectionMonitor - Error behavior ... 353
SF_DirectionMonitor - Status Diagram ... 354
SF_EDM Characteristics ... 227
SF_EDM Error behavior ... 224
SF_EDM Error detection ... 224
SF_EDM function ... 223
SF_EDM Status Diagram ... 228
SF_EDM Time Diagram ... 229
SF_EmergencyStop Error behavior ... 232
SF_EmergencyStop Error detection ... 231
SF_EmergencyStop function ... 230
SF_EmergencyStop Status Diagram ... 233
SF_EmergencyStop Time Diagram ... 234
SF_EnableSwitch Error behavior ... 237
SF_EnableSwitch Error detection ... 237
SF_EnableSwitch function ... 235
SF_EnableSwitch Status Diagram ... 239
SF_EnableSwitch Time Diagram ... 240
SF_Equivalent Error behavior ... 242
SF_Equivalent Error detection ... 242
SF_Equivalent function ... 241
SF_Equivalent Status Diagram ... 243
SF_Equivalent Time Diagram ... 244
SF_ESPE Error behavior ... 247
SF_ESPE Error detection ... 246
SF_ESPE function ... 245
SF_ESPE Status Diagram ... 248
SF_ESPE Time Diagram ... 249
SF_GuardLocking Error behavior ... 252
SF_GuardLocking Error detection ... 251
SF_GuardLocking function ... 250
SF_GuardLocking operating sequence ... 251
SF_GuardLocking Status Diagram ... 254
SF_GuardLocking Time Diagram ... 255

SF_GuardMonitoring error and reset behavior 257
SF_GuardMonitoring Error detection 257
SF_GuardMonitoring function 256
SF_GuardMonitoring Status Diagram 259
SF_GuardMonitoring Time Diagram 260
SF_LimitComparator 338
SF_LimitComparator - Error behavior 340
SF_LimitComparator Status Diagram 341
SF_ModeSelector Characteristics 264
SF_ModeSelector Error behavior 263
SF_ModeSelector Error detection 263
SF_ModeSelector function 261
SF_ModeSelector Status Diagram 265
SF_ModeSelector Time Diagram 265
SF_MutingPar Error behavior 269
SF_MutingPar Error detection 269
SF_MutingPar function 267
SF_MutingPar Status Diagram 274
SF_MutingPar Time Diagram 275
SF_MutingPar_2Sensor 276
SF_MutingPar_2Sensor Error behavior 278
SF_MutingPar_2Sensor Error detection 278
SF_MutingPar_2Sensor Status Diagram 281
SF_MutingPar_2Sensor Time Diagram 281
SF_MutingSeq Error behavior 284
SF_MutingSeq Error detection 284
SF_MutingSeq function 282
SF_MutingSeq Status Diagram 287
SF_MutingSeq Time Diagram 288
SF_NumericSelector 343
SF_NumericSelector - Error behavior 344
SF_NumericSelector - Status Diagram 346
SF_Optional_Pwd Error behavior 290
SF_Optional_Pwd Error detection 290
SF_Optional_Pwd function 289
SF_Optional_Pwd Status Diagram 292
SF_Optional_PwdII Error behavior 295
SF_Optional_PwdII Error detection 295
SF_Optional_PwdII function 294
SF_OutControl Error behavior 304
SF_OutControl Error detection 304
SF_OutControl function 303
SF_OutControl Status Diagram 306
SF_OutControl Time Diagram 307
SF_RampMonitor 347
SF_RampMonitor - Error behavior 349
SF_RampMonitor – Formula for calculating the speed curve 348
SF_RampMonitor - Status Diagram 350
SF_SafetyRequest Error behavior 309
SF_SafetyRequest Error detection 309
SF_SafetyRequest function 308
SF_SafetyRequest Status Diagram 311
SF_SafetyRequest Time Diagram 312
SF_SampleHold 176
SF_SkewMonitor 355
SF_SkewMonitor - Error behavior 357
SF_SkewMonitor - Status Diagram 358
SF_TestableSafetySensor Error behavior 315
SF_TestableSafetySensor Error detection 315
SF_TestableSafetySensor function 313
SF_TestableSafetySensor Status Diagram 320
SF_TestableSafetySensor Time Diagram 321
SF_TwoHandControlTypeII Error behavior 322
SF_TwoHandControlTypeII Error detection 322
SF_TwoHandControlTypeII function 322
SF_TwoHandControlTypeII Status Diagram 325
SF_TwoHandControlTypeII Time Diagram 326
SF_TwoHandControlTypeIII Characteristics 329
SF_TwoHandControlTypeIII Error behavior 327
SF_TwoHandControlTypeIII Error detection 327
SF_TwoHandControlTypeIII function 327

SF_TwoHandControlTypeIII Status Diagram 330
SF_TwoHandControlTypeIII Time Diagram 331
SHL (Shift-Left) 177
SHL_NS (Shift-Left, not safe) 177
SHR (Shift-Right) 178
SHR_NS (Shift-Right, not safe) 178
Speed 185
SR_Flipflop 209
Status Diagram SF_Optional_PwdII 298
SUB (subtraction) 174
SUB_NS (subtraction, not safe) 175

T

Target/actual comparison of the hardware 151
Temporary Operation Time 81
Time Diagram SF_Optional_Pwd 293
Time Diagram SF_Optional_PwdII 298
Timer function blocks 204
TOFF 204
TON 205
Toolbars 38
Total response time 100
TP 205
Trigger function blocks 205

U

Upload Log-Data 159
User Interface 32

V

Validating and generating download data 89
Version control and archiving 122
Versioning 122

W

Watch window 104
Working in the network editor 68

X

XOR function 201
XOR_BW (XOR, bitwise) 180

Eaton is an intelligent power management company dedicated
to improving the quality of life and protecting the environment for people everywhere. We are guided by our commitment to do business right, to operate sustainably and
to help our customers manage power – today and well into the future.
By capitalizing on the global growth trends of electrification and digitalization, we're accelerating the planet's transition to renewable energy, helping to solve the world's most urgent power management challenges, and doing what's best for our stakeholders and all of society.

For more information, please visit Eaton.com.

Eaton Industries GmbH
Hein-Moeller-Str. 7–11
D-53115 Bonn

© 2025 Eaton
All rights reserved.
06/25 MN050028