

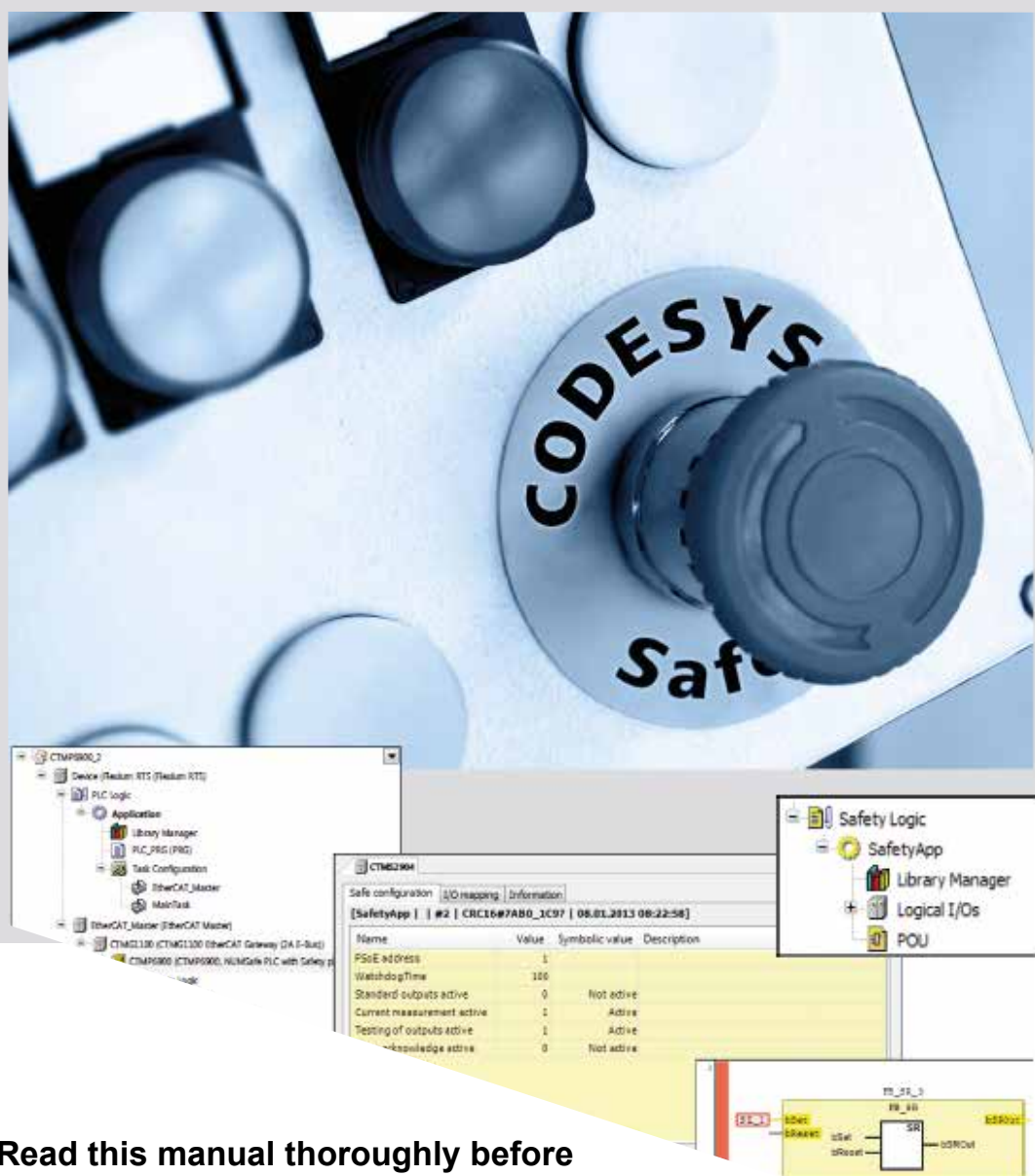
CTMP6900 NUMsafe controller

CTMP1960–2600 NUMsafe compact controller

Programming manual

M00037EN-03

Edition 04.2020



Read this manual thoroughly before you start working with NUMSafe.

The reproduction, distribution and utilization of NUM documents as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages.
All rights reserved in the event of the grant of a patent, utility model or design.

CODESYS® is a trademark of 3S-Smart Software Solutions GmbH.

Despite the care taken in the preparation of this document, NUM ca not guarantee the accuracy of the information it contains and cannot be held responsible for any errors therein, nor for any damage which might result from the use or application of the document.

The physical, technical and functional characteristics of the hardware and software products and the services described in this document are subject to modification and cannot under any circumstances be regarded as contractual.

The programming examples described in this manual are intended for guidance only. They must be specially adapted before they can be used in programs with an industrial application, according to the automated system used and the safety levels required.

Summary

1	Introduction	9
1.1	Document revisions	10
1.2	NUM Agencies.....	10
1.3	Index	10
2	Standards	15
3	Requirements.....	21
3.1	System requirements	21
3.2	Qualified personnel.....	21
3.3	CTMP6900 System limitations	22
3.4	CTMP1960-2600 System limitations	23
4	Project development in "CODESYS Safety for EtherCAT Safety Module".....	27
4.1	General information	27
4.2	Creation of a new project with a safety application	28
4.3	Device administration	37
4.4	Libraries	38
4.5	Project structure	39
4.5.1	Insertion of a NUMsafe controller into the project tree.....	39
4.5.2	NUMsafe controller.....	39
4.5.3	Safety Logic	40
4.5.4	Safety application	40
4.5.4.1	Safety application object	40
4.5.4.2	Logical I/Os.....	47
4.5.4.3	POUs	59
4.5.4.4	Library manager	63
4.6	Access protection	64
4.6.1	Overview of access protection	64
4.6.2	User management in the project	64
4.6.3	User management – NUMsafe controller	65
5	Programming	69
5.1	Programming overview.....	69
5.2	Safe data flow	70
5.3	Data types	71
5.4	Language elements	72
5.4.1	Overview of the language elements	72
5.4.2	Rules for identifiers of safety objects.....	72
5.4.3	POUs	73
5.4.4	Networks.....	74
5.4.4.1	Overview of networks	74
5.4.4.2	Fundamental information about variables	77
5.4.4.3	FB calls	78
5.4.4.4	FBD assignments	80

5.4.4.5	Monitoring	81
5.5	I/O access	82
5.6	Deactivating of Groups	82
5.7	Programming guidelines	82
5.8	Example projects	84
5.8.1	Example project - step by step	84
5.8.2	Application example: two-hand control	99
6	Online operation	105
6.1	Project download	105
6.2	Project upload	106
6.3	Delete Project	107
6.4	Status Display	107
7	Pinning.....	111
8	Verification and acceptance	119
8.1	Verification and acceptance	119
8.2	Archiving	119
8.3	Acceptance documentation printout.....	121
9	Software updates	125
9.1	Overview of versioning	125
9.2	Updating the device version	125
9.3	Changing a library version	126
9.4	Updating the CODESYS version	127
9.5	Change request / Bug fix process	127
10	Function blocks	131
10.1	Introduction	131
10.2	Library " CTMP6900 NUMsafe controller " and " CTMP1960-2600 NUMsafe compact controller"	132
10.2.1	FB_AND	132
10.2.2	FB_CS	134
10.2.3	FB_DECOUPLER	136
10.2.4	FB_EDM	138
10.2.5	FB_ESTOP	140
10.2.6	FB_MON.....	144
10.2.7	FB_MUTING.....	149
10.2.7.1	Example with 4 individual muting sensors.....	153
10.2.7.2	Example with 2 two-channel muting sensors	153
10.2.8	FB_MODE	156
10.2.9	FB_OR.....	159
10.2.10	FB_RS	161
10.2.11	FB_SR	163
10.2.12	FB_TOF	165
10.2.13	FB_TON.....	167
10.2.14	FB_TWOHAND	169
10.3	Library " CTMP1960-2600 NUMsafe compact controller "	172
10.3.1	FB_ADD	172
10.3.2	FB_COMPARE	174
10.3.3	FB_COUNTER	178
10.3.4	FB_DIV	181

10.3.5	FB_ENVELOPE.....	183
10.3.6	FB_LIMIT.....	186
10.3.7	FB_LOADSENSING.....	188
10.3.8	FB_MUL.....	191
10.3.9	FB_SCALING.....	193
10.3.10	FB_SLI.....	196
10.3.11	FB_SPEED.....	198
10.3.12	FB_SUB.....	200
10.3.13	FB_VIOLATIONCNT.....	202
11	Error messages	207
12	Glossary and Certificates	215
12.1	Glossary	215

Page deliberately empty

Summary

1	Introduction	9
1.1	Document revisions	10
1.2	NUM Agencies	10
1.3	Index.....	10

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

1 Introduction

"CODESYS Safety for EtherCAT Safety Module" is a software for programming CTMP6900 NUMsafe controller and CTMP1960-2600 NUMsafe compact controller. This manual is part of the "CODESYS Safety for EtherCAT Safety Module" product package and must be read without fail before creating a safety application.

In addition to a description of the functions of "CODESYS Safety for EtherCAT Safety Module", the manual also contains information which must be observed without fail when creating your application.

The following kinds of safety information are used:



CAUTION!

Disregarding these safety instructions can lead to personal injuries.



Notice!

Disregarding these instructions can lead to errors and problems in the development and verification of the safety project.



This symbol marks information intended to improve understanding.

"CODESYS Safety for EtherCAT Safety Module" is suitable for the development of safety applications in the sense of application software up to SIL3 according to IEC 61508 / 62061 and Category 4PL-e according to ISO 13849.

This manual describes:

- how a safety application is managed
- how the modules are programmed
- how safe I/O modules are to be configured and used
- how to protect an application against unauthorised access
- how to manage the version control of applications
- how the exchange of data between safety application and default application is configured
- how a safety application is loaded to a safety controller
- how a safety application can be verified and accepted.

Part of the manual is the exact specification of the language scope that is available for the creation of a safety application. This contains all data types and all available function blocks.

The language scope corresponds to the scope of the NUMSafe CTMP6900 and CTMP1960-2600 safety controllers.

The data types available are described in the Chapter 5.3 "Data types", while the list of all the function blocks offered by the system is to be found in the Chapter 10 "Function blocks".

The prerequisites for understanding the manual are good knowledge of CODESYS and knowledge of IEC 61131-3.

We hope you will enjoy reading the manual !

1.1 Document revisions

Date	Index	Reason for revision
05 - 2013	00	Document creation
11 - 2013	01	Version 3.0: 4.2 Creating a safety application (Delete Project). 4.5.4.3 POU's. 5.4.4.5 Monitoring. 5.7.1 Example project: Step 7: Settings of the parameters of the logical I/Os. 6.3 Delete Project. 10.7 FB_MON. 10.9 FB_MODE. 11 Error messages.
11 - 2017	02	2 Standards 13849: 2015. 12.2 New Certificate.
04 - 2020	03	Analog functions for CTMP1960-2600

1.2 NUM Agencies

The list of NUM agencies is given on <http://www.num.com>

1.3 Index

The [index](#) at the end of the manual facilitates access to information by keywords.

Page deliberately empty

Page deliberately empty

Summary

2	Standards	15
---	-----------------	----

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

2 Standards



The versions of the standards valid on 1 July 2012 are considered.

The valid safety standards

Machine manufacturers and operators of technical plants are responsible for ensuring that machines or plants meet the requirements for health and safety at work. If the respective relevant standards are adhered to, this ensures that the machine or device manufacturer or the installer of a plant has fulfilled its duty to exercise diligence and that the state of the art has thus been attained.

The machine directive defines a uniform safety level for the prevention of accidents for machines brought onto the market within the European Economic Area (EEA) and in Switzerland and Turkey.

It is the duty of the device manufacturer to furnish proof of adherence to the underlying requirements. The proof can be furnished by adherence to the respective European standards (EN). These standards ("harmonised standards") are listed in the Official Journal of the European Union.

- EN IEC 62061 and EN ISO 13849: for the functional part of the safety of control systems.
- Specific harmonised standards ("product standards"): for the functional part of the safety of individual machine types (machine tools, woodworking machines, printing machines, packaging machines, etc.).

These standards have the absolute higher priority for the machine manufacturer. By adhering to these standards it can be assumed that the fundamental safety requirements of the machine directive are fulfilled.

"CODESYS Safety for EtherCAT Safety Module" is suitable for programming the safety functions of the machine in conformity with these standards.

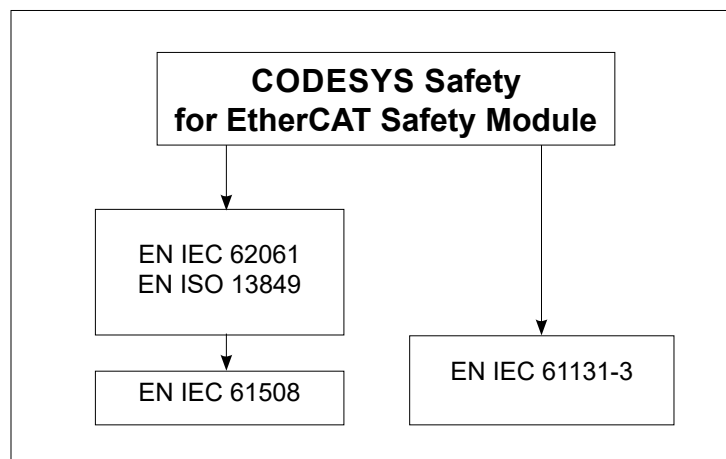


Fig. 1. "CODESYS Safety for EtherCAT Safety Module" Overview of safety standards

Standard	Title
EN IEC 61508:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems
EN IEC 62061:2005	Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems
EN ISO 13849:2015	Safety of machinery – Safety-related parts of control systems
EN IEC 61131-3:2003	Programmable controllers – general information

Safety levels

Depending on the severity of the damage risk, EN IEC 62061 and EN ISO 13849 place different demands on the reliability of the programmed safety functions of the machines.

- EN ISO 13849 defines various reliability levels for this, called PL-a to PL-e.
- EN IEC 62061 refers in addition to the EN IEC 61508 basic safety standard, where the reliability levels SIL1 to SIL4 are defined.
- The IEC61508 standard describes the certification of electrical, electronic and programmable electronic systems (E/E/PES for short). According to this standard integrable software parts can also be certified. It is the basic standard to which EN IEC 62061 and EN ISO 13849 refer.

The two levels systems correspond to each other approximately as follows:

Correspondence of PL to SIL

PL	SIL
a	No correspondence
b	1
c	1
d	2
e	3
No correspondence	4

"CODESYS Safety for EtherCAT Safety Module" is suitable for machines at the levels SIL1 to SIL3 and PL-a to PL-e.

Requirements of the software development

So that the programmed safety function attains the necessary reliability level, the development of the control software must already meet certain requirements.

On the whole EN IEC 61508 defines the cycle of the complete system with 16 phases, starting with Phase1: "Concept" and ending with Phase16: "Decommissioning". "CODESYS Safety for EtherCAT Safety Module" is used as a tool in EN IEC 61508 Phase10 "Realisation, safety software life cycle".

The standard demands among other things the definition of a safety plan, the creation of software specifications, the definition of programming guidelines and the planning of test activities.

The requirements of the standard are reduced by the use of a Limited Variability Language (or LVL for short), because with these languages the program logic can be expressed more clearly and can be more easily understood and verified.

These easier requirements are defined in the EN IEC 62061 and EN ISO 13849 standards. (When using unlimited programming languages, both standards refer to the 61508 basic standard (with general, complex requirements))

The LVLs include in particular the graphic languages of EN IEC 61131-3, such as FBD or LD. They do not include the textual languages such as C, ST, assembler or IL.

Separation of the safe aspects from the non-safe aspects

In order to make a clear distinction between safety-relevant signals and standard signals, new data types were defined with the identification "SAFE". This indicates to the developer that the signals are safety-relevant and need to be treated with special care.

Furthermore the connection of data can be checked automatically by this identification in order to discover all impermissible connections between standard signals and safety-relevant signals.

Although the "SAFE" data types cannot guarantee that the signal status is safe (for example if the peripherals are incorrectly wired), it is nevertheless an organisational tool for the minimisation of errors in the application program. This simplifies and shortens the verification of the signal flow. Safe data types can be used within a safety-relevant environment.

They should be used to distinguish between safe and non-safe signals with the aim of simplifying the validation and certification.

"CODESYS Safety for EtherCAT Safety Module" supports these rules by automatically checking these programming guidelines and the linkage rules. In the event of a violation an application cannot be loaded to the controller. Adherence to the rules is thus automatically proven.

IEC 61131-3

IEC 61131-3 describes the programming languages that are used for the programming of controllers. It defines the FBD language (language with limited language scope).

Page deliberately empty

Summary

3	Requirements	21
3.1	System requirements	21
3.2	Qualified personnel	21
3.3	CTMP6900 System limitations	22
3.4	CTMP1960-2600 System limitations	23

Page deliberately empty

3 Requirements

3.1 System requirements

The following system requirements apply to smaller "CODESYS Safety for EtherCAT Safety Module" projects with a maximum of 100 blocks, a maximum of 10 visualisations and a maximum of 8 fieldbus devices:

- 1 GB RAM
- 1 GHz Pentium
- 1 GB free hard disc space
- Screen resolution 1024 x 768.

The "CODESYS Safety for EtherCAT Safety Module" programming system is approved only for the following operating systems:

- Windows 7 (32/64 bit).
- Windows 10 (32/64 bit).

3.2 Qualified personnel

The prerequisite for the use of safety products is sufficient qualification.

Hence, personnel who plan, create and commission safety applications with this product must:

- be familiar with the applicable standards and regulations.
- be familiar with the relevant security concepts in automation technology.
- have knowledge of accident prevention rules and possible special operating regulations.
- have sufficient knowledge of a language to understand this manual.
- have good knowledge of CODESYS programming.

3.3 CTMP6900 System limitations

During the system configuration, please always take into consideration the following limitations:

Description	Limitations of the CTMP6900 System
NUMSafe enables networks with up to	65535 NUMSafe devices
NUMSafe PLC can establish up to	128 connections (NUMSafe connections) with 1 or 2 bytes safe user data per connection.
	Maximum. 50 connections per NUMSafe group.
	Only 16 connections of an CTMP6900 can be slave connections.
Safe data per connection	Up to 14 bytes safe user data. (correspondingly lower total number of connections)
NUMSafe function blocks	Maximum 255
NUMSafe groups	Maximum 32
Standard PLC inputs	Dynamic up to 255 bit
Standard PLC outputs	Dynamic up to 255 bit



Notice!

Several NUMSafe PLC are cascable in a network.

3.4 CTMP1960-2600 System limitations



Project design limits

The maximum project design size of the CTMP1960-2600 is limited by the available memory. This is managed dynamically. The values specified in the following table are therefore only guide values and may differ from the actual values, depending on the safety project.

Description	Limitations of the CTMP1960-2600 System
Process image size	max. 1486 bytes per data direction (maximum memory size 0x1E00 for 3 buffers, ie with the same size of input and output process data, a maximum size of 1280 bytes per data direction is possible. Only straight start addresses are possible, so fill bytes must be taken into account).
NUMSafe connections	128 max. (up to 255 CRCs in total; 1 CRC is required for a NUMSafe connections with 1 or 2 byte safe data.)
Supported hardware for CTMP1960-2600 NUMsafe compact controller	CTMS1904 CTMS2904 NUM-SAMX
Safe data per NUMSafe connection	Maximum 126 byte (telegram length 255 byte)
NUMSafe blocks	Maximum 512 (ESTOP with complete input and output mapping)
NUMSafe groups	128 max.
NUMSafe user	40 max.
Standard PLC inputs	Dynamic (memory-dependent), max. 1024 byte
Standard PLC outputs	Dynamic (memory-dependent), max. 1024 byte



NUMsafe compact controller

Only one NUMSafe connection is possible between two NUMSafe components.
For communication with a CTMP6900, for example, a connection with up to 14 bytes safe user data can be used.

Page deliberately empty

Summary

4	Project development in "CODESYS Safety for EtherCAT Safety Module"	27
4.1	General information	27
4.2	Creation of a new project with a safety application	28
4.3	Device administration.....	37
4.4	Libraries	38
4.5	Project structure.....	39
4.5.1	Insertion of a NUMsafe controller into the project tree.....	39
4.5.2	NUMsafe controller	39
4.5.3	Safety Logic	40
4.5.4	Safety application.....	40
4.5.4.1	Safety application object.....	40
4.5.4.2	Logical I/Os	47
4.5.4.3	POUs.....	59
4.5.4.4	Library manager.....	63
4.6	Access protection.....	64
4.6.1	Overview of access protection.....	64
4.6.2	User management in the project.....	64
4.6.3	User management – NUMsafe controller	65

Page deliberately empty

4 Project development in "CODESYS Safety for EtherCAT Safety Module"

4.1 General information

The following contains a description of the structure of the project tree and the project tree objects that are available for the creation of a safety application with "CODESYS Safety for EtherCAT Safety Module" in accordance with the guidelines IEC 62061 and ISO13849.

The creation of a project and the use of the user interface take place according to the control concept of Standard CODESYS.

Commands from Standard CODESYS that are not available in "CODESYS Safety for EtherCAT Safety Module" are visible, but cannot be activated.

Yellow safety structures

The safety objects in the project structure and the safety editors are clearly emphasised by yellow structures and thus clearly differentiate themselves visually from the corresponding standard objects and standard editors.

In addition the safe signal flows are also marked yellow. This supports and facilitates the tasks for the development, verification and acceptance of the safety application.

4.2 Creation of a new project with a safety application

1. To create a new project: activate the “New Project...” command in the File menu.
2. Select “Empty project” or “Standard project” in the dialog window. Enter a file name and location, then actuate the “OK” button.

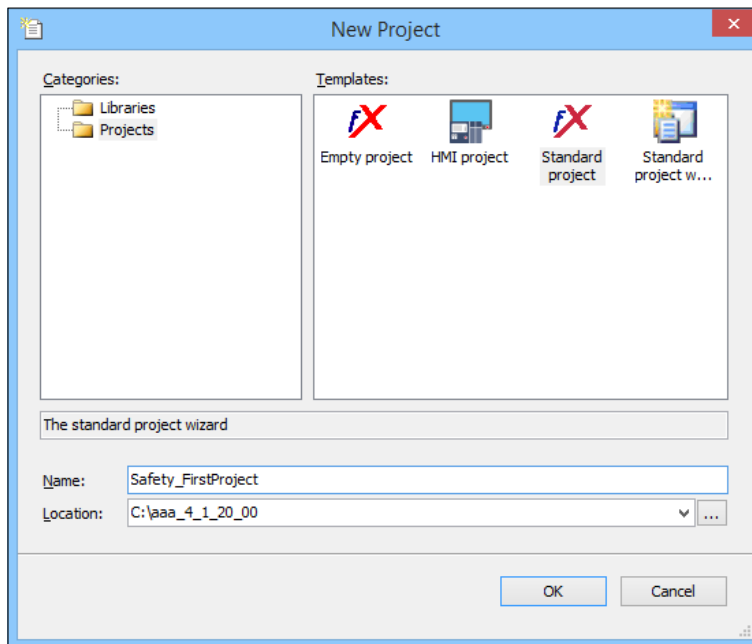


Fig.2 Create a new project

3. In the next dialog window push “Next”

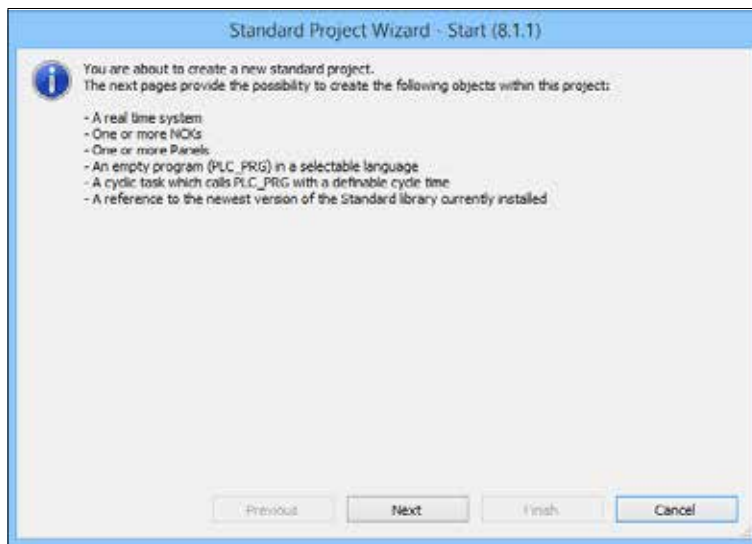


Fig.3 Standard project

4. Choose the right RTS version depending from the Windows operating system where the project will run:
 Flexium RTS (NUM) for 32 bit operating system
 Flexium RTS x64 (NUM) for 64 bit operating system
 Then push "Next"

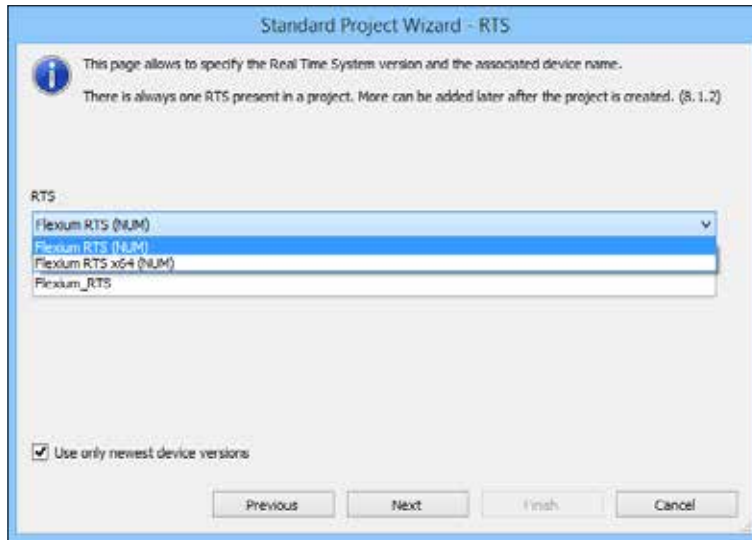


Fig.4 RTS selection

5. Choose the right NCK type depending from your device:
 Flexium NCK
 or
 Flexium+ NCK
 Then push "Next"

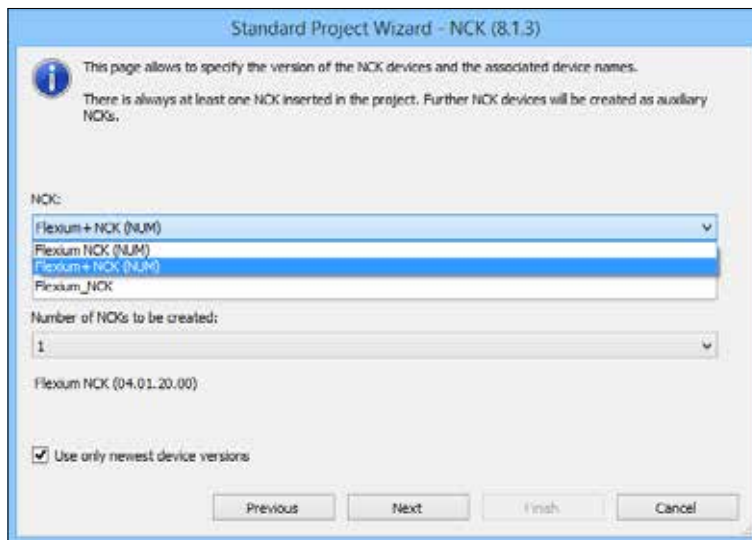


Fig.5 NCK selection

6. In the next dialog window push “Next”

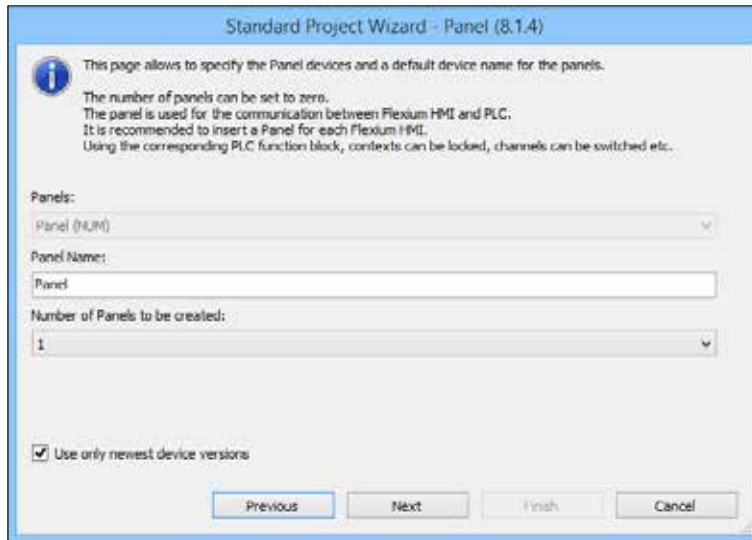


Fig.6 Panel selection

7. Choose “PLC Programming Language”: Structured Text (ST)

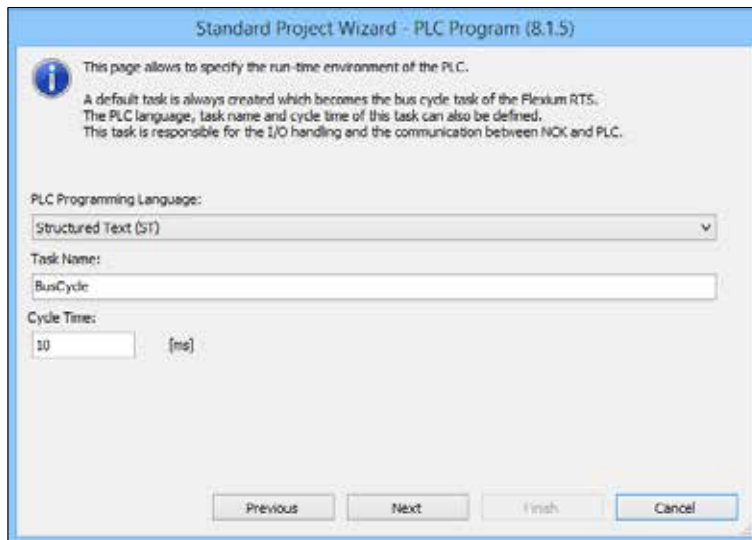


Fig.7 PLC Programming Language selection

8. In the next dialog window push "Next"

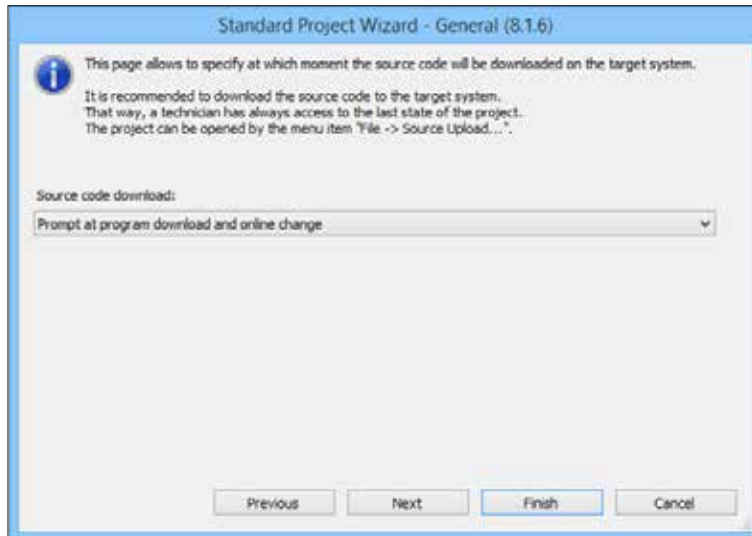


Fig.8 Source code download condition selection

9. Check everything and push "Finish"

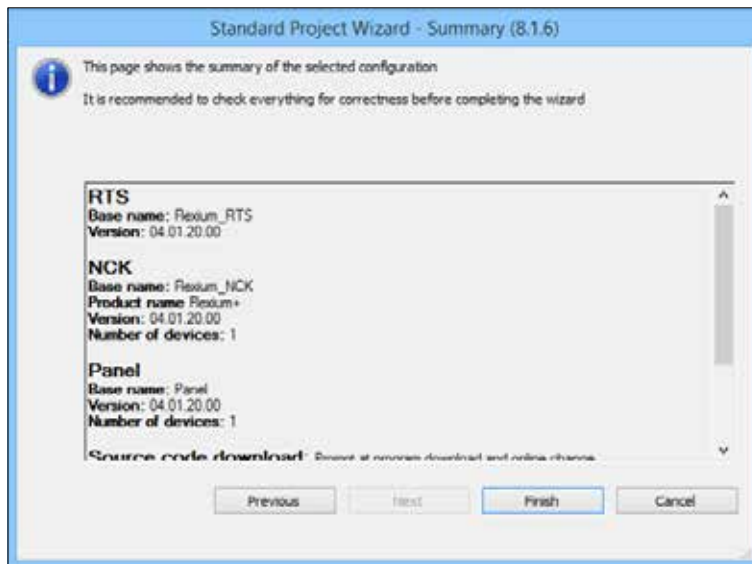


Fig.9 Summary check

After quitting the dialog with "Finish" the new project is created; all components are shown in the device tree.

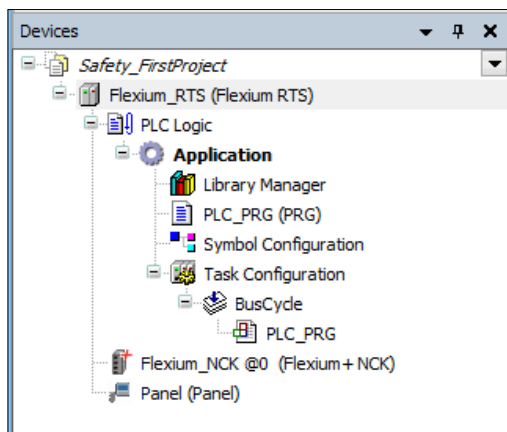


Fig.10 New project device tree

10. Add an EtherCAT Master to the standard controller:

Select the standard controller (e.g. Flexium_RTS), activate the “Add device...” command in the context menu and select EtherCAT Master.

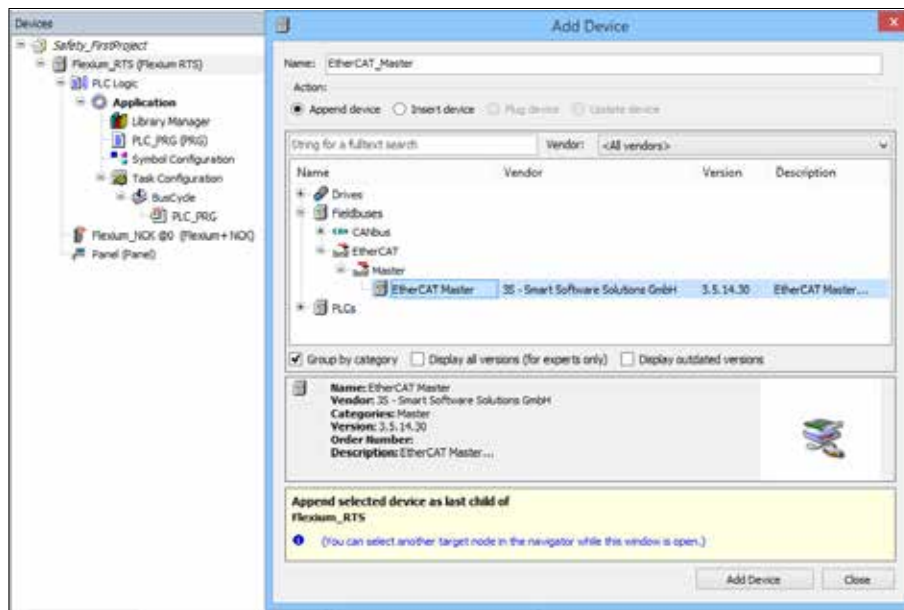


Fig.11 Add EtherCAT master

11. Add a bus gateway (e.g. CTMG1100) under the EtherCAT Master:

Select the EtherCAT Master, activate the “Add device...” command in the context menu and select CTMG1100.

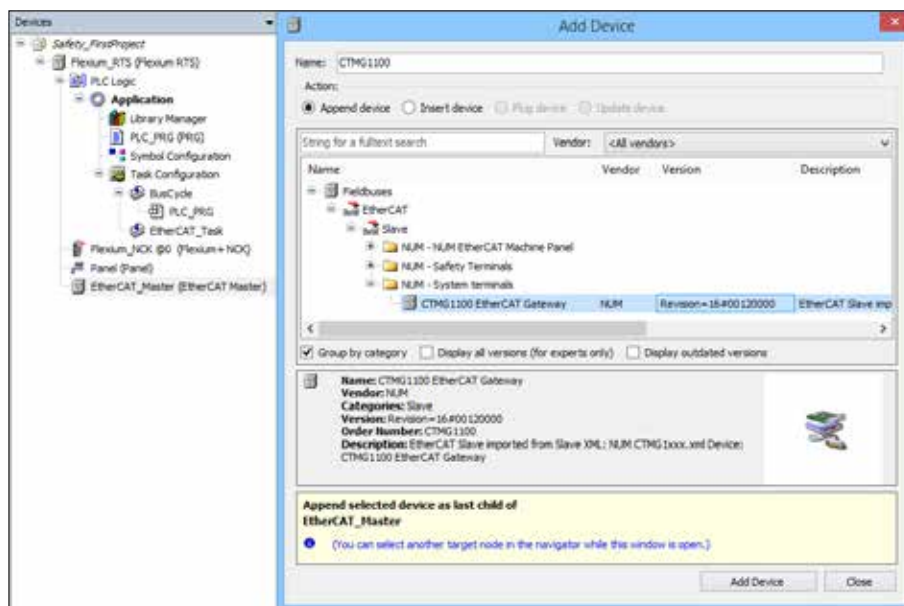


Fig.12 Bus gateway device CTMG1100 add

12. Addition of a NUMsafe controller under the bus gateway:

Select the bus gateway, activate the "Add device..." command in the context menu and select the NUMsafe controller. Alternatively the NUMsafe controller can be added to the device tree using the "Scan For Devices" command

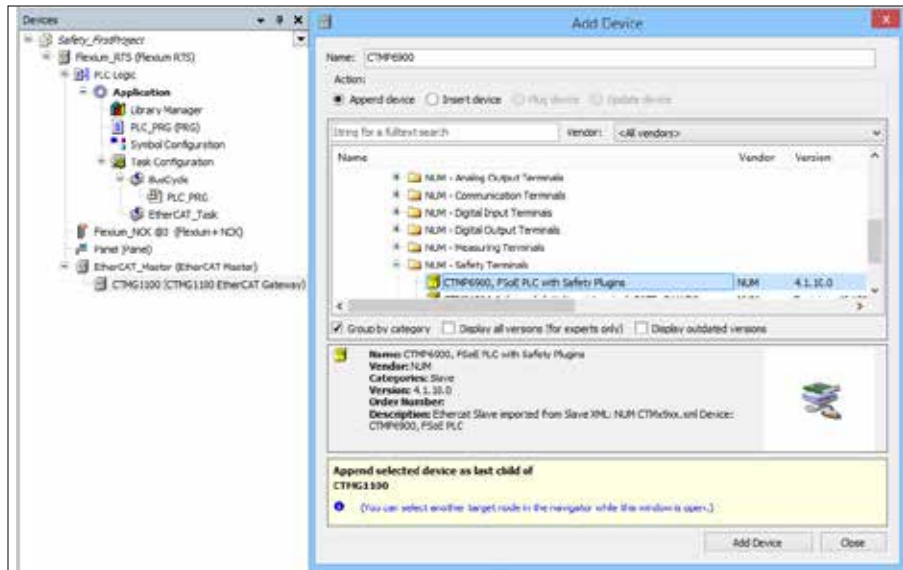


Fig.13 Add Device NUMsafe controller CTMP6900

Finally you will have

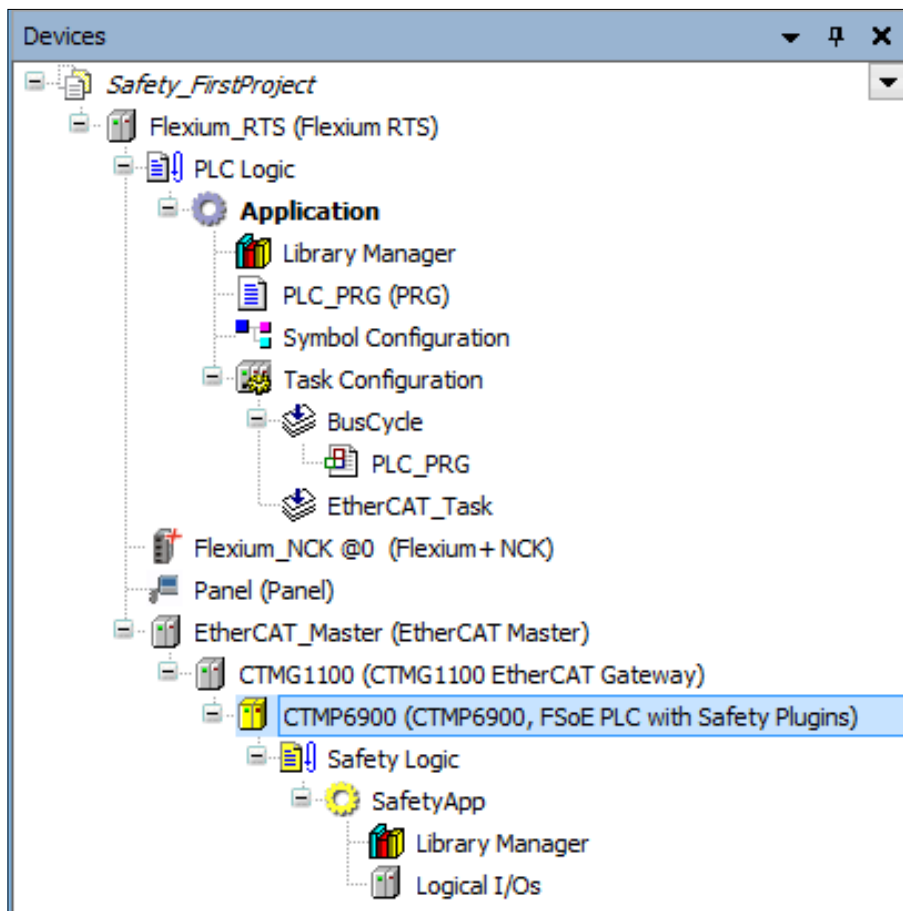


Fig.14 CTMP6900 NUMsafe controller

Objects or node points that are automatically inserted together with the NUMsafe controller and that exist only once:

- **Safety Logic:**

Logical node point of the NUMsafe controller, below which precisely one safety application object can be attached.

- **SafetyApp:**

Node point below which the objects belonging to the safety application object are located.

Object that defines the currently recorded version (pin) of the application.

(see Chapter 7 "Pinning")

The editor of the object manages the list of objects currently belonging to the safety application object.

- **Library manager:**

Contains the libraries available on the inserted NUMsafe controller.

These are:

- TcEL6900FBs
- TcEL69xxAnalogFBs

- **Logical I/Os:**

Node point to which logical I/O objects can be added. These added logical I/Os are used for the exchange of data and I/Os with the standard controller.

Objects that can be manually added to the safety application object (also several times):

- **Safety POU:**

POU (Programm).

Specific settings of the NUMsafe controller

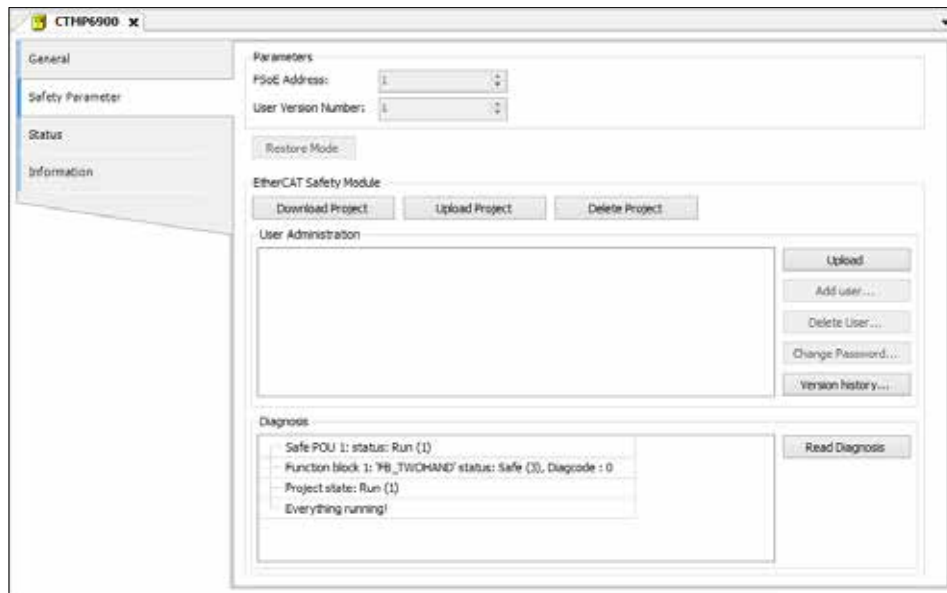


Fig.15 Safety Parameter dialog

"FSOE Address" Unique address for FSoE protocol.

"User version number" This number can be freely selected by the user. It is stored on the NUMsafe controller and displayed in the version history.

“Restore Mode”

The dialog lists all Slaves which support this function and which are used in the program.

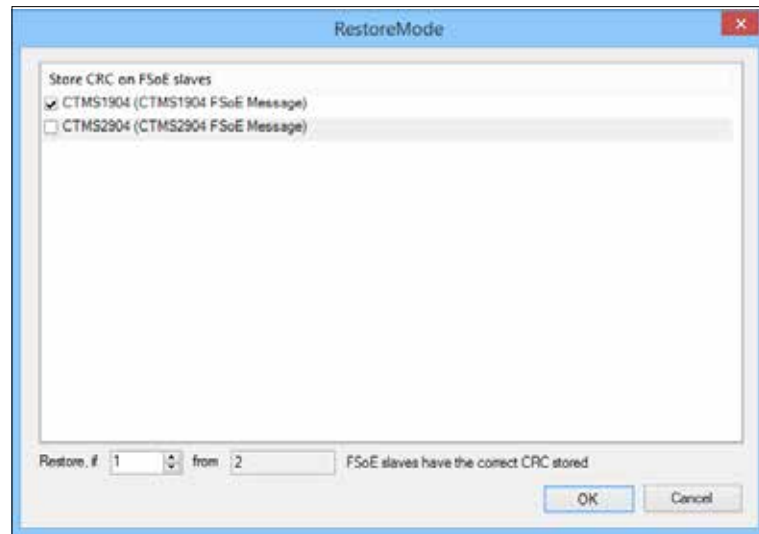


Fig.16 New project with safety application - Restore Mode

Checkboxes allow to define which slaves should save a checksum. In the lower field of the dialog the user can define how many of these slaves must have the right checksum to restore the project automatically after the NUMsafe controller has been replaced.

“Download Project”

Starts the download. Further information can be found in section Project download (6.1)

“Upload Project”

The project stored in the NUMsafe controller is uploaded and compared with the current project. Any differences will be displayed in a dialog.

“Delete Project”

This command deletes the project in the NUMsafe controller. Further information can be found in section Delete Project (6.3).

“User Administration”

These commands enable the management of users on the NUMsafe controller. A precise description can be found in section User management – NUMsafe controller (4.6.3)

“Read diagnosis”

Diagnostic messages can be read from the NUMsafe controller using the “Read diagnosis” command when FlexiumTools is in ONLINE mode.

4.3 Device administration

Safe devices are managed, installed and uninstalled in a similar way to standard devices in the "Device Repository" ("Tools" menu). For detailed information refer to the CODESYS Standard online help.

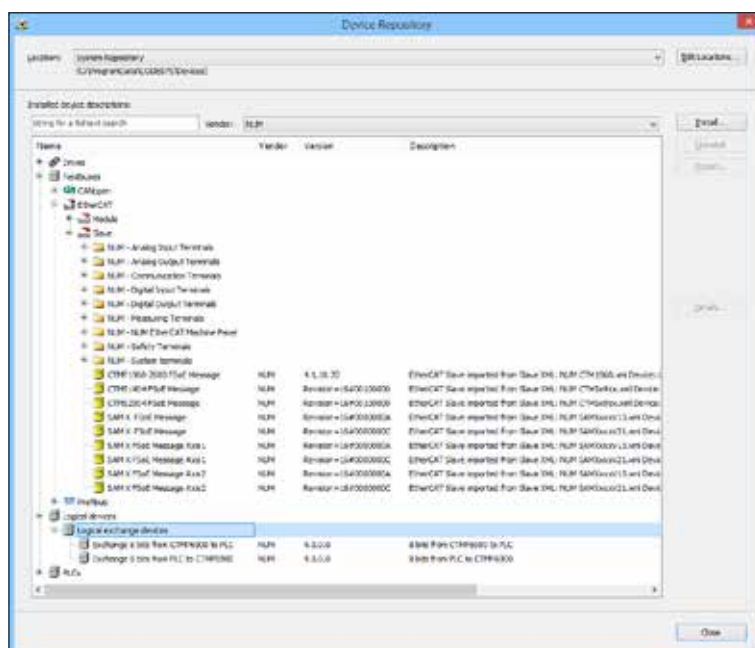


Fig.17 Device Repository

The Device Repository can be selected and opened in the "Tools" menu.

The devices relevant for "CODESYS Safety for EtherCAT Safety Module" are grouped as follows in the Device Repository:

- All field devices for the NUMsafe controller (EtherCAT Bus Gateway, NUMsafe controller and safe modules) are located in the "EtherCAT/Slave" subfolder.
- The logical I/Os of the physical field devices are managed in the "EtherCAT/Slave" subcategory of the "Logical devices" category.
- The "Logical exchange devices" subcategory for the exchange of data with the standard controller is located in the "Logical Devices" category.
- Safe field devices that have been installed in the Device Repository can be inserted in the project tree on the standard side like any installed device.
- The parameters of the fail-safe field devices are edited in the respective logical I/Os (see Chapter 4.5.4.2.3.2 "Safe configuration").
The parameterization of the standard field devices takes place as in CODESYS standard.

Safety libraries with pre-certified function blocks are provided to the developer for programming with "CODESYS Safety for EtherCAT Safety Module". The following libraries are part of the "CODESYS Safety for EtherCAT Safety Module" product:

- "TcEL6900FBs"
- "TcEL69xxAnalogFBs"

These libraries are automatically part of the safety applications. An installation is not necessary.

Fieldbus-specific libraries may be available to the user depending on the safety fieldbus.

These libraries are managed as in CODESYS Standard.

**Notice!**

The library repository and the library manager are not suitable for the verification of the library blocks used in the safety application during the verification and the acceptance. The verification of the execution-relevant statuses must take place via the comparison view, see Chapter 7 "*Pinning*". For acceptance documentation see Chapter 8.3 "*Acceptance documentation printout*" 7.

A description of the TcEL6900FBs library and the TcEL69xxAnalogFBs library can be found under Chapter 10 "*Function blocks*".

4.5 Project structure

4.5.1 Insertion of a NUMsafe controller into the project tree

The NUMsafe controller is displayed in the project tree below an EtherCAT gateway (CTMP6900) or below the EtherCAT master (CTMP1960-2600).

Precisely one application (in this case SafetyApp) is located under the NUMsafe controller.

Below that are all the safety objects belonging to a project.

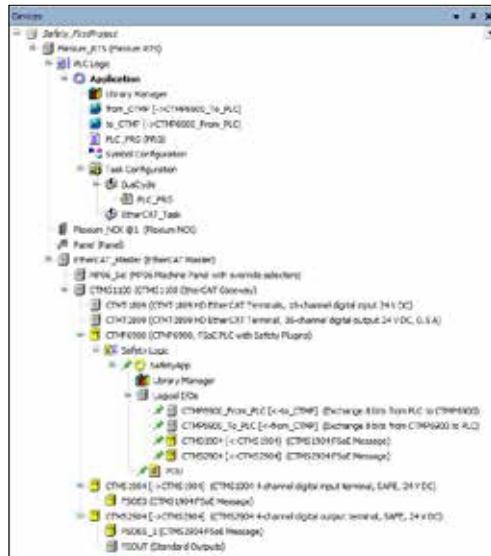


Fig. 18 Example of a project tree with standard and safety application

Refer to the Standard CODESYS online help for the part of the project tree that contains the objects of the default application.

4.5.2 NUMsafe controller

Adding the NUMsafe controller

The NUMsafe controller CTMP6900 is inserted under an EtherCAT Gateway (e.g. CTMG1100). The NUMsafe compact controller CTMP1960-2600 is inserted under the EtherCAT master. This is done by selecting the Gateway or the master and activating the "Add device..." context menu command with selection of the EtherCAT Safety Module. Alternatively the module can be appended to the Gateway or master using the "Scan For Devices" command in the context menu.

After the insertion of the NUMsafe controller, the "Safety Logic" logical node point, the "SafetyApp" safety application object, the "Library Manager" and the "Logical I/Os" node point are always inserted automatically with it.

The NUMsafe controller can be updated to a newer version of the device description with the "Update Device" context menu command. Libraries may possibly be replaced by newer versions.

Object properties of the NUMsafe controller

A NUMsafe controller attached in the project tree has a Properties dialog with the "Common" and "Access Control" dialogs.

The Properties dialog is opened by selecting the NUMsafe controller in the project tree and activating the "Properties..." context menu command.

4.5.3 Safety Logic

The logical node “Safety Logic”  exists for the standard controller for reasons of symmetry. It has no meaning in the NUMsafe controller.

The “Safety Logic” logical node point is inserted automatically into the project tree with the safety controller. There can only be precisely one “Safety Logic” below a NUMsafe controller. Only a “Safety App” can be added to it as an object.

Safety PLC has a Properties dialog with “Common” and “Access Control” dialogs.

The Properties dialog is opened by selecting “Safety Logic” in the project tree and activating the “Properties...” context menu command.

“Safety Logic” cannot be opened by any editor.

4.5.4 Safety application

4.5.4.1 Safety application object

Structure of the project tree of the safety application

The project tree is displayed in the Device window (“Devices” in the “View” menu).

The objects relevant for the NUMsafe controller and its programming are located below the NUMsafe controller. Directly below them is always the “Safety Logic” symbolic node point .

Under each “Safety Logic” there can only ever be **one** safety application object  “SafetyApp” (default name), which can contain the following safety objects:

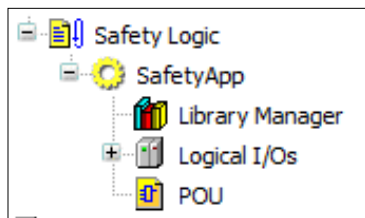


Fig. 19 Example: safety application object with objects

The following objects must be present precisely **once** in a safety application object:

- Library manager
- Logical I/Os.

Adding the safety application object

The safety application object is added to the project tree:

- automatically when the NUMsafe controller is added
- or
- manually by selecting the "Safety Logic" logical node point and the "Add Object" context menu command with selection "Safety Application".

In the Properties dialog (selection of the "Safety Application" object and activation of the "Properties..." command) a name can be edited for the safety application object on the "Common" tab and a comment can be edited for it on the "Safety" tab (see Fig. 18).

If the "Safety Application" object is added manually, then a name and comment for the safety application object can be edited in the dialog which opens when adding it.

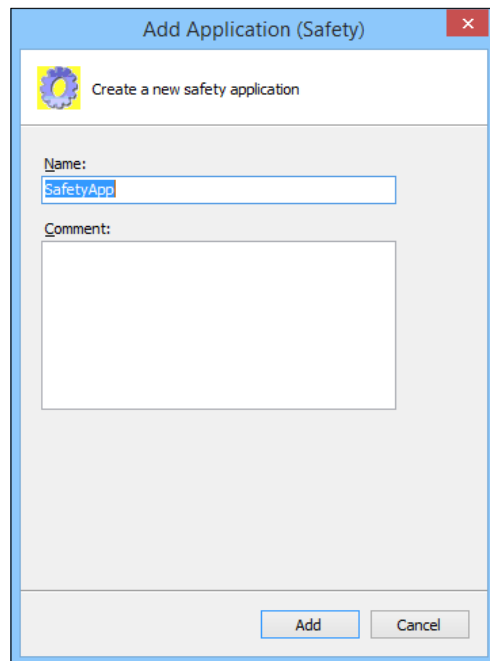


Fig. 20 Dialog when manual adding the safety application object with the default name "SafetyApp"



The Properties dialog of the safety application object and all objects of the safety application object has the “Common” dialog (with name, type of object and editor with which the object is opened) and “Access control” tab (access rights of the user groups for the object).

Additional, object-specific dialogs are written with the individual objects.

The Properties dialog is opened by selecting the object in the project tree and activating the “Properties” context menu command.

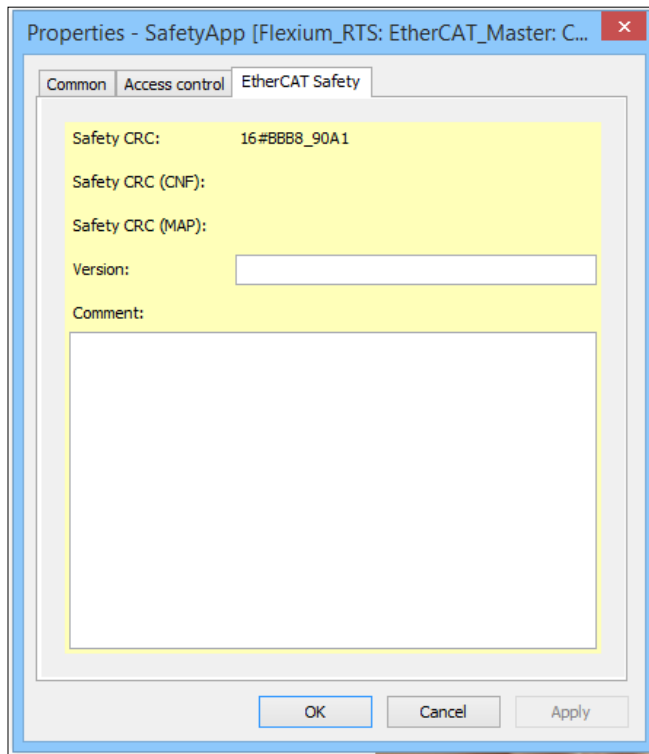


Fig. 21 Dialog: Properties of the safety application object Dialog: Safety



Pin checksum

The Pin checksum is a checksum for all objects of the safety application including the library blocks employed.



Checksum of an object

Contrary to the Pin checksum, the checksum for the individual objects of the safety application identifies the object contents together with the object version. The checksum is of importance if individual objects of a safety application are reused in another application.

The Safety dialog contains:

Information

"Safety CRC"	Information about the checksum of the safety application object.
"Safety CRC (CNF)"	Information about the checksum for the safe configuration (see Chapter 4.5.4.2.3.2 "Safe configuration").
"Safety CRC (PAR)"	Information about the checksum for the safe device parameters (if one exists).
"Version" (editable)	The version can be freely assigned by the developer and ensures that the version of the object is easily recognisable in the comparison view.
"Comment" (editable)	

1

2

3

4

5

6

7

8

9

10

11

12

Editor of the safety application object with comparison view

The editor is opened by selecting the “*Safety Application*” object and activating the “*Edit Object*” context menu command.

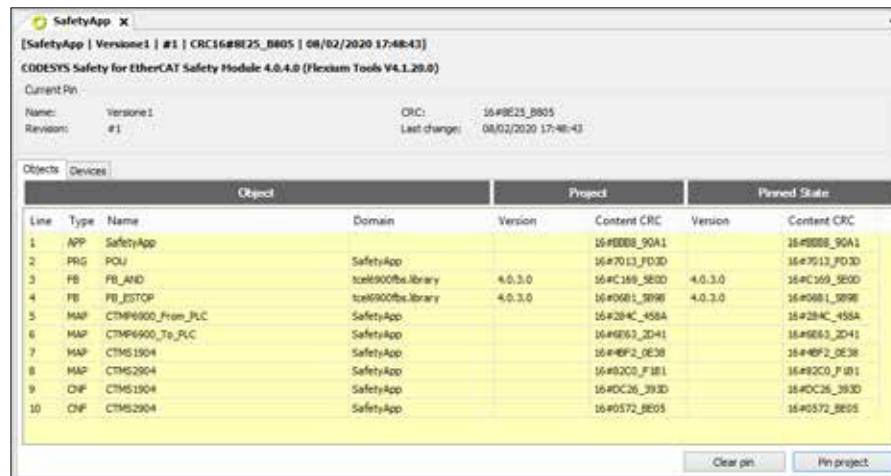


Fig.22 Example, editor of the safety application object with comparison view (object list)

Dialogs of the editor of the safety application object

- Objects
- Devices

Both dialogs contain the following information:

- Pin information or, if the application has been changed: “*InWork*”
- Current version of “CODESYS Safety for EtherCAT Safety Module”
- “*Current Pin*” of the safety application, consisting of:
 - *Name*
 - “*CRC*” (Pin checksum).
 - “*Revision*”
 - “*Last change*”

See “*Display of the pin information and its deviations*” for precise information on the Pin information.

Objects



The "Objects" dialog of the editor of the safety application object is the **comparison view**! It indicates which objects have been changed since the last pinning.



Notice!

The comparison view must be used during the verification and acceptance of a safety application for the comparison of the pin versions *Chapter 7 "Pinning"*.

The comparison view shows a list of the objects belonging to the safety application:

- "Line" Sequential numbers of the entries.
- "Type" Object type (see table).
- "Name" Name of the object.
- "Version" and "Content CRC" Version and checksum of the object in the current project (project status).
- "Version" and "Content CRC" Version and checksum of the object of the most recently pinned project.

Displayed values for type and name in the object list (depending on the type of object)

Object type	Type	Name
Application	APP	Name of the application object
POU program	PRG	Name of the program POU
POU function block	FB	Name of the POU object of the function block
I/O mapping	MAP	Name of the logical device object/application parameter
Safe configuration	CNF	Name of the logical I/O object
Safe parameterization	PAR	Name of the logical device object

In addition, the comparison view contains the "Clear pin" and "Pin project" buttons. "Pin Project" is used to create a new Pinning of the project

Devices

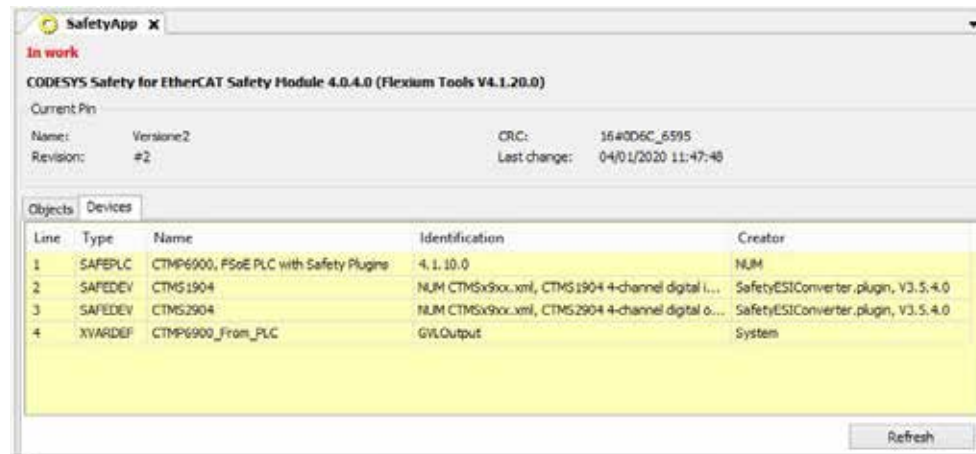


Fig.23 Editor of the safety application object with device list

The Devices dialog shows the following information:

- Pin information or, if the application is not pinned: *"In Work"*.
- Current version of "CODESYS Safety for EtherCAT Safety Module".
- *"Current pin"*, consisting of:
 - *Name*
 - *"CRC"*
 - *"Revision"*
 - *"Last change"*
- List of the field devices belonging to the safety application and the NUMsafe controller to which the safety application belongs, in each case with the associated information about the description files:
 - *"Line"* Sequential numbers of the entries.
 - *"TYPE"* Type of device (see table).
 - *"Name"* Name of the device in the project tree.
 - *"Identification"* Device-specific information for identification.
 - *"Creator"* Information about the creator of the device-specific information.

Displayed values for type in the device list

Type	Description
SAFEPLC	NUMsafe controller to which the application is assigned
SAFEDEV	Safe logical device of the application
XVARDEF	Standard logical device for variable exchange

The *"Refresh list"* button is for updating the device list if the devices (logical I/Os) or the NUMsafe controller in the project tree have changed with the editor open.

4.5.4.2 Logical I/Os

4.5.4.2.1 Overview of logical I/Os

No further hardware modules can be inserted below the NUMsafe controller. Safe modules as well as standard modules are inserted in the device tree underneath a gateway (e.g. CTMG1100).

The connection to the NUMsafe controller or the exchange of data with it takes place via so-called "logical I/Os". These objects are inserted underneath the "logical I/Os" node.

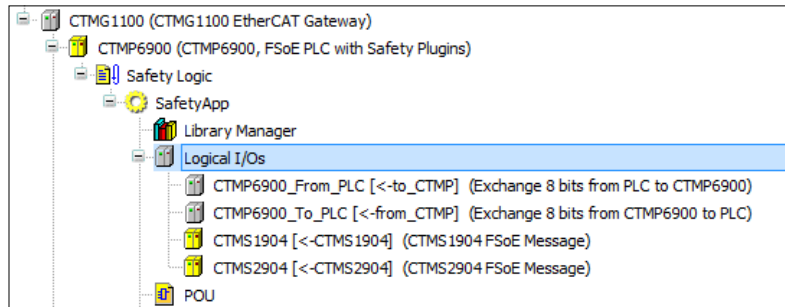


Fig. 24 Logical I/Os

Principle of the logical I/Os

CODESYS distinguishes between two different kinds of logical I/Os.

Types of logical I/Os:

- Logical I/Os of safe field devices
If safe field devices are inserted in the device tree underneath a coupler (e.g. CTMG1100), then a matching logical I/O is automatically inserted (same name as the I/O module). In case of more than one NUMsafe controller the matching logical I/O must be added manually. Afterwards this logical I/O must be selected in the I/O mapping dialog of the corresponding field device.
The logical I/O contains all the safety parameters of the module, so that all safety-relevant information can be found under the safety application. (see Chapter 4.5.4.2.2.1 "Logical I/O of a safe physical device").
- Logical I/Os of global variables:
Additional data can be exchanged between NUMsafe controller and standard controller via these objects. To this end global variables (GVL for logical exchange) are defined in the standard project and are then connected with the corresponding logical I/O on the NUMsafe controller ("logical exchange devices").
(see Chapter 4.5.4.2.2.2 "Logical I/O for data exchange with the main controller (standard controller)").

The logical I/Os of the safety application are linked to the default application with physical devices or "GVLs for logical exchange" (special object on standard side). This means that there is precisely one logical I/O in the safety application for each physical device whose input/output signals are processed in the safety application.

Likewise, precisely one logical I/O exists under the safety application for each GVL for the logical exchange of the standard controller.



The fail-safe parameters of a device whose inputs/outputs are processed in the safety application can be edited only in the appropriate logical I/O in the safety application.

For a better overview, folders can be added to the project tree under "Logical I/O" in order to group the logical I/Os.

Advantages of the logical I/Os

The concept of the logical I/Os gives rise to the following advantages for the development and verification of a "CODESYS Safety for EtherCAT Safety Module" safety application:

- The parameterization of the safe parameters of field devices takes place only into the logical I/Os of the safety application. If safety user management is set up, this parameterization can only be done by members of the Safety user group.
- Changed assignments of physical field devices and GVLs for logical exchange do not change the safety application, since the changes of assignments take place under the main application (default application) and become effective by downloading to the main controller.
- An already verified and accepted "CODESYS Safety for EtherCAT Safety Module" safety application can be detached from the original project and integrated completely in a different project without this new safety application having to be verified again. When doing this suitable field devices and GVLs for logical exchange must be reassigned to the logical I/Os of the safety application.

Notes on the logical I/Os

The logical I/Os are assigned depending on the name of the application and the logical I/O object. This gives rise to the following notes:



Notice!

If the name of a logical I/O is changed, then the assignment in the physical device or in the GVL for logical exchange must be reset and reassigned to the logical I/O.

Notice!

The name of the safety application object must be unique project-wide and the name of the logical I/O object must be unique within the safety application in order to be able to map the logical I/O object unambiguously to the field device or to the GVL for logical exchange.

Notice!

The renaming (except in the case of "default assignment") or relocation of field devices in the device tree or of GVLs for logical exchange in the main application does not change the assignment to the logical I/O in any way. A modification of the safety application is made automatically at the renaming: the assigned logical I/O is renamed analogously to the field device.

Notice!

The deletion of the field device from the device tree or of the GVL for logical exchange from the default application means that the assigned logical I/O will no longer be mapped to anything.

Notice!

The deletion of the logical I/O object means that fail-safe settings are no longer mapped to the field device or that the "from Safety" variables in the GVL for logical exchange are no longer supplied.

Notice!

If a different logical I/O object is given the old name of a renamed or deleted logical I/O object, then from now on its settings are mapped to the assigned field device or its variable values are exchanged with the exchange GVL.

The assignment is thus indirectly changed. If the logical I/O is deleted and a new logical I/O is reinserted, then the connection is reactivated.

Object properties

The Properties dialog of all logical I/Os contains the "EtherCAT Safety" dialog in addition to the "Common" and "Access control" dialogs.

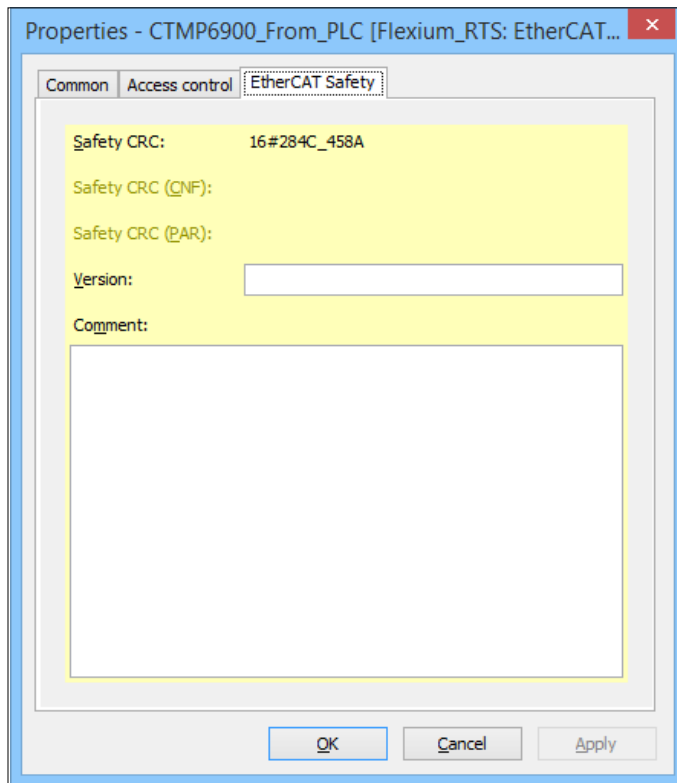


Fig.25 Dialog: Properties of logical I/Os, dialog: Safety

The "Properties" dialog of logical I/Os of a safety device additionally contains the dialog "Connection Settings". If the option "Module Fault (Fail Safe Data) is COM ERR" is enabled in this dialog, then receiving the FailSafeData command will be handled like a communication error.

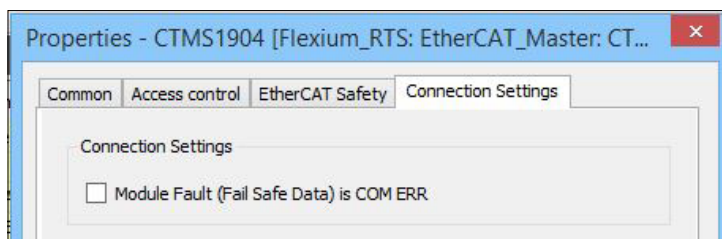


Fig.26 Dialog: Properties of logical I/Os of a safety device

4.5.4.2.2 Usage types of the logical I/Os

4.5.4.2.2.1 Logical I/O of a safe physical device

These logical I/Os serve the exchange of fail-safe I/Os between the NUMsafe controller and the standard controllers.

For this exchange the safe field device must first be inserted under the standard controller as in CODESYS.

Addition of a safe field device under the standard controller

1. Select the appropriate CTMG1100 fieldbus gateway under the EtherCAT Master in the project tree.
2. Activate the “Add Device...” context menu command.
3. In the dialog that opens, select the desired safe field device from the “EtherCAT/Slave” subcategory under the “Fieldbuses” category.
4. Actuate the “Close” button.



When inserting a safe field device under the EtherCAT gateway (e.g. CTMG1100), the corresponding logical I/O is automatically inserted under the “Logical I/Os” node point of the safety application.



Notice!

The prerequisite for this automatic generation of the logical I/O of a safe field device is that there should be only one NUMsafe controller (and hence only one safety application) below the standard controller under which the safe field device is inserted.

Type consistency of the I/O channels

The type consistency of the I/O channels is only ensured:

- If the application revision levels on the NUMsafe controller and on the main controller correspond to the revision level of the same translatable project.

and

- if the field devices in the project correspond to the field devices in the machine.
(A mismatch is detected automatically here, depending on the bus system).

4.5.4.2.2.2 Logical I/O for data exchange with the main controller (standard controller)

The exchange of data between the NUMsafe controller and standard controller takes place via variables, which are defined in the logical I/Os “Logical exchange devices”. In addition, a “GVL for logical exchange” is created on the standard side and connected with the corresponding “logical exchange device”.

These data are used as inputs/outputs in the NUMsafe controller; in the default application they are available as global variables.



The data flow between two variables is clear. This means that the same variable from one application cannot be exchanged with two variables of the other application.

Addition of the "Logical Exchange GVL"

1. Select the default application object in the project tree.
2. Activate the "Add Object" context menu command with selection "Logical Exchange GVL ..."
3. A name can be edited for the GVL in the "Add Logical Exchange GVL" dialog. The default name is "Logical_GVL".
4. Actuate the "Add" button.

Addition of the "logical exchange device"

1. Select the "Logical I/Os" node point of the safety application object in the project tree.
2. Activate the "Add Object..." context menu command with selection "Logical Device...".
3. In the "Add Logical Device" dialog (see Fig. 23) in the "Logical devices" category, select the desired logical I/O in the "Logical exchange devices" subcategory.
4. Actuate the "Add" button.

For editing the exchange variables, see Chapter 4.5.4.2.3.1 "Information and I/O mapping".

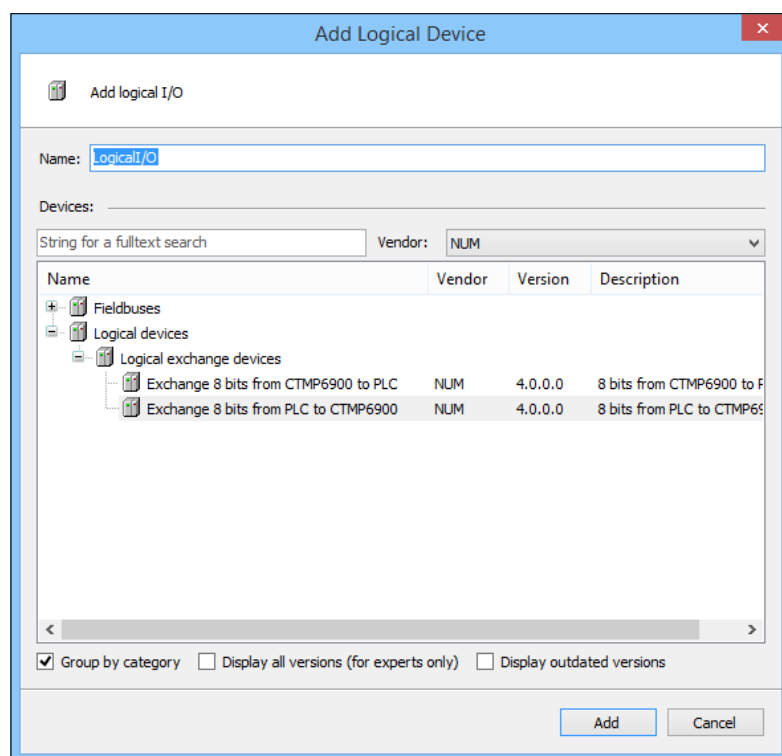


Fig. 27 Dialog: Add Logical Device

The following items are defined when selecting the “logical exchange device” :

- The exchange direction: IN or OUT:
 - IN: from the default application to the safety application.
 - OUT: from the safety application to the default application.



*The exchange variables can be edited only in the logical I/O.
Variables cannot be entered or changed in the Logical Exchange GVL.*

Connection of the "GVL for logical exchange" to the "logical exchange device"

1. Select Logical Exchange GVL ... of the default application in the project tree.
2. Activate the “Edit Object” context menu command.
3. Click on the “Logical Exchange mapping” combo box.
4. Select the appropriate logical exchange object from the list that opens (see Fig. 24).

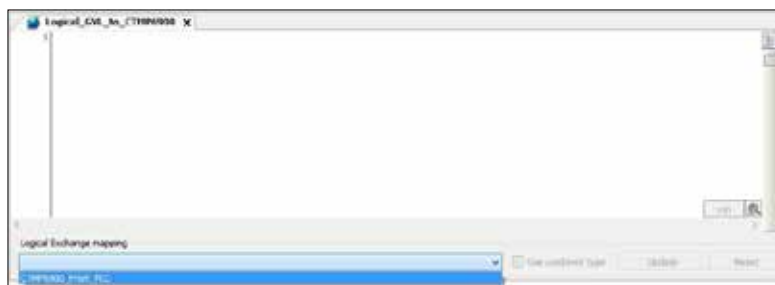


Fig.28 Editor of the GVL for logical exchange with opened combo box

All logical exchange objects that have not yet been mapped are available in the combo box.
If a logical exchange device is selected, then the variables are implicitly updated.

If a logical exchange device is selected, then the variables are implicitly updated.

If the “Use combined type” option in this editor is activated, then the single Boolean variables of the connected logical exchange device (type BYTE) of the safety application are not mapped to Boolean variables. Instead, they are mapped to the first variable with the combined data type according to the logical I/O of the safety application.



Notice!

If changes are made in the connected logical exchange device, then the “Update” button in the GVL for logical exchange must be actuated in order to update the variable list.

An existing connection is terminated by the actuation of the “Reset” button.

Notes on the exchange of data between the standard and NUMsafe controllers



Notice!

The type-consistency of the variable exchange is only guaranteed if the application versions on the NUMsafe controller and on the main controller correspond to the version of the same translatable project.



Notice!

Only data whose variables possess a standard data type can be exchanged. Variables of a data type with the prefix SAFE cannot be exchanged between a NUMsafe controller and a standard controller.



Notice!

A change of the variable exchange can only become active by downloading the safety application and the default application again.

1

2

3

4

5

6

7

8

9

10

11

12

4.5.4.2.3 Editor of the logical I/Os

4.5.4.2.3.1 Information and I/O mapping

Information

The editor of all logical I/Os has the “*Information*” dialog, which contains precise information and, if necessary, a picture of the respective logical I/O.

The following information is listed: name, vendor, categories, type, ID, version, order number and description.

Possible categories:

- Safe modules.
- Generated logical devices.
- Logical exchange device.



Fig.29 Example: Information dialog of a logical I/O

I/O mapping

The variables that serve the access of the safety application to the I/Os are defined in the "I/O mapping".



In "CODESYS Safety for EtherCAT Safety Module", I/O channels can only be mapped to new variables, i.e. not to already existing variables.

Variables mapped to input channels contain the corresponding input signals from field devices and can thus be read. Variables mapped to output channels can be written and set output signals in field devices.

An implicit global variable with the corresponding name and the data type specified in the "Type" column is created in the safety application for each input or output channel of an I/O module to which a variable has been assigned.

Sections of the I/O mapping dialog:

- List of the variables of the I/O mapping with: variable (name), channel (input or output), type, unit and description. The information in the "Channel", "Type" (IEC data type), "Unit" and "Description" columns is defined in the SDD file and cannot be changed.
- Physical I/O mapping: display of the object of the default application connected to this logical I/O.
- Instances: List of the implicit instances. These are available to the safety application as global variables. (see Chapter 4.5.4.2.4 "Use of logical I/Os in the project").

Editing the variables

The mapping variables are edited and displayed in the "I/O mapping" dialog of the logical I/O. In order to edit a variable, the respective line must be opened with a double click.

All mapping variables entered in the table are deleted on actuating the "Reset mapping" button, i.e. the mapping of the physical device on the logical I/O is reset.

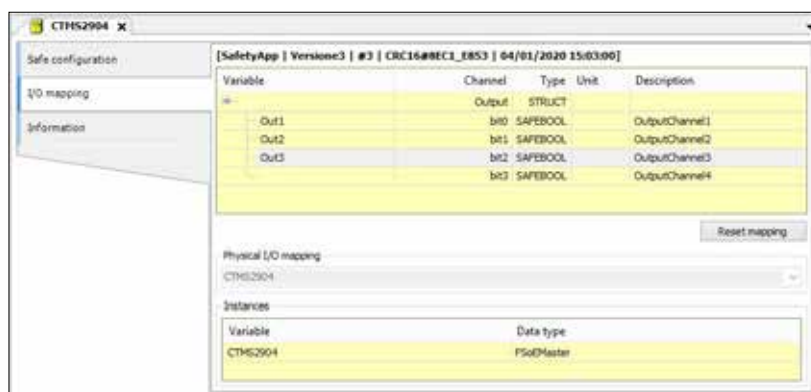


Fig. 30 Example: I/O mapping dialog of the logical I/O of a safe field device

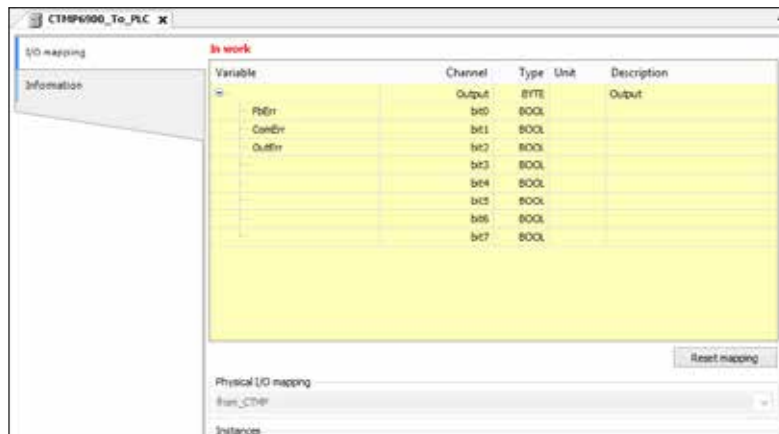


Fig.31 Example: I/O mapping dialog of a logical I/O, type Byte with bit access



Changes made in the “I/O mapping” dialog are marked red. Only the last-made change is marked. All change markings are removed on closing the editor.

A description of the use of the variables defined in the “I/O mapping” in the project can be found under Chapter 4.5.4.2.4 “Use of logical I/Os in the project”.



The editor of the logical I/Os of standard devices does not possess a “Safe Configuration” dialog. The configuration and the device parameterization take place under the default application as in CODESYS Standard.



Notice!

If a device is assigned to the NUMsafe controller, then the mapping dialog of the device editor shows only this assignment and no longer the channel mapping to the variables of the main controller.

4.5.4.2.3.2 Safe configuration

The "Safe Configuration" dialog of the editor of the logical I/Os exists only for the fail-safe I/Os.

The Pin information or "In Work" appears in the upper section. The parameters are listed in tabular form in each case in the adjoining section.

"Safe Configuration" contains the configuration parameters for safe communication.

The default values for the parameters originate from the respective device description file (SDD).



As an option, integer parameters can also be displayed in hexadecimal. This optional setting must be made in the device description file.

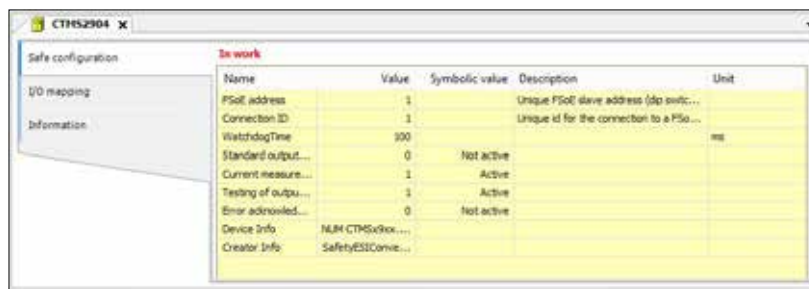


Fig.32 Example: "Safe configuration"



Changes made in the "Safe Configuration" dialog are marked red. Only the last-made change is marked. All change markings are removed on closing the editor.

Description of the bus-specific safe parameters

"FSoE address"	Unique EtherCAT address of the module. An FSoE connection corresponds in each case to a safe fieldbus device (or its logical I/O).
"WatchdogTime"	an error is output if the module does not answer within this time.
"Connection ID"	Unique connection number in the entire EtherCAT network.

4.5.4.2.4 Use of logical I/Os in the project

Every mapping variable (channel variable) declared in the "I/O mapping" of a logical I/O is available to the developer as a global variable during the programming of the safety application.

As an alternative to the explicit declaration, these variables and instances can be selected in the implementation section of POU's either in the Input Assistant or in the automatically displayed "Intellisense" selection list.

In the Input Assistant of the variable declaration they are located in the "Global variables" category in the "IoConfigRemote_Globals" GVL.

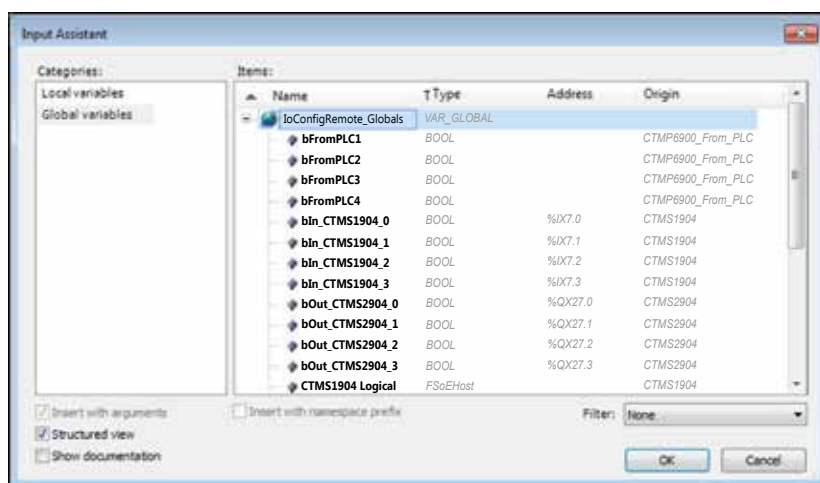


Fig.33 Input Assistant of the variable declaration with "Global variables"

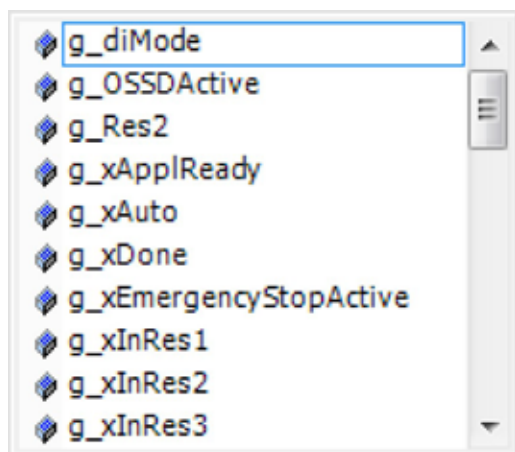


Fig.34 Example: Intellisense with global variables of "IoConfigRemote_Globals"



Variables mapped to an I/O channel can be written and forced when debugging the program!

4.5.4.3 POU

POUs (Programmable Organisation Units) are the programming objects of "CODESYS Safety for EtherCAT Safety Module", which are declared as programs.

In the project tree of the Safety application a maximum of 32 POU (CTMP6900) or 64 POU (CTMP1960-2600) can be added. If this value is exceeded a compiler error is created.

Features of program and function blocks:

- A program cannot be called by other programs, but can call instances of function blocks from libraries.
Programs are called directly by the system.

Addition of a POU

1. Select "SafetyApp" in the project tree.
2. Context menu command "Add object" - "Safety POU".
3. Enter the name and comment for the POU in the "Add Safety POU" dialog window.

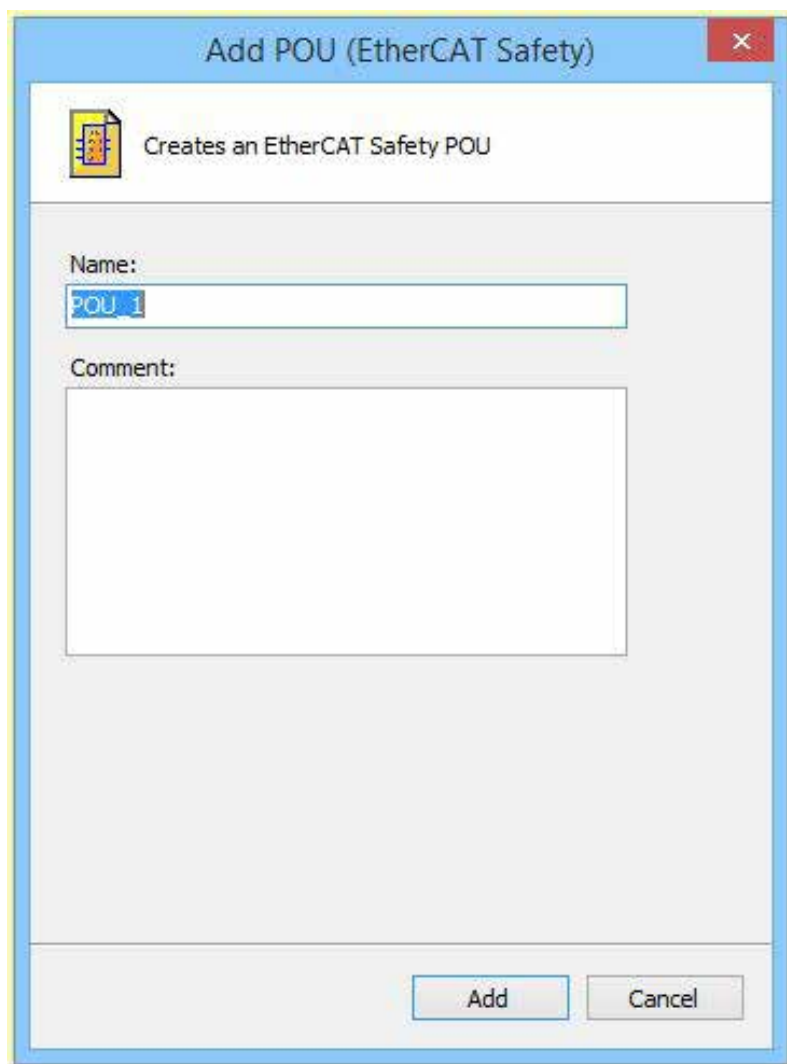


Fig.35 Example: Dialog when creating a POU

A description of the editor of a POU and the creation of the program code in POU can be found under *Chapter 5.4.3 "POUs"*.

Object properties of a POU

Each POU of the safety application has a Properties dialog with “Common”, “EtherCAT Safety”, “Group IO’s” and “Access Control” dialogs.

The Properties dialog is opened by selecting the respective POU in the project tree and activating the “Properties...” context menu command.

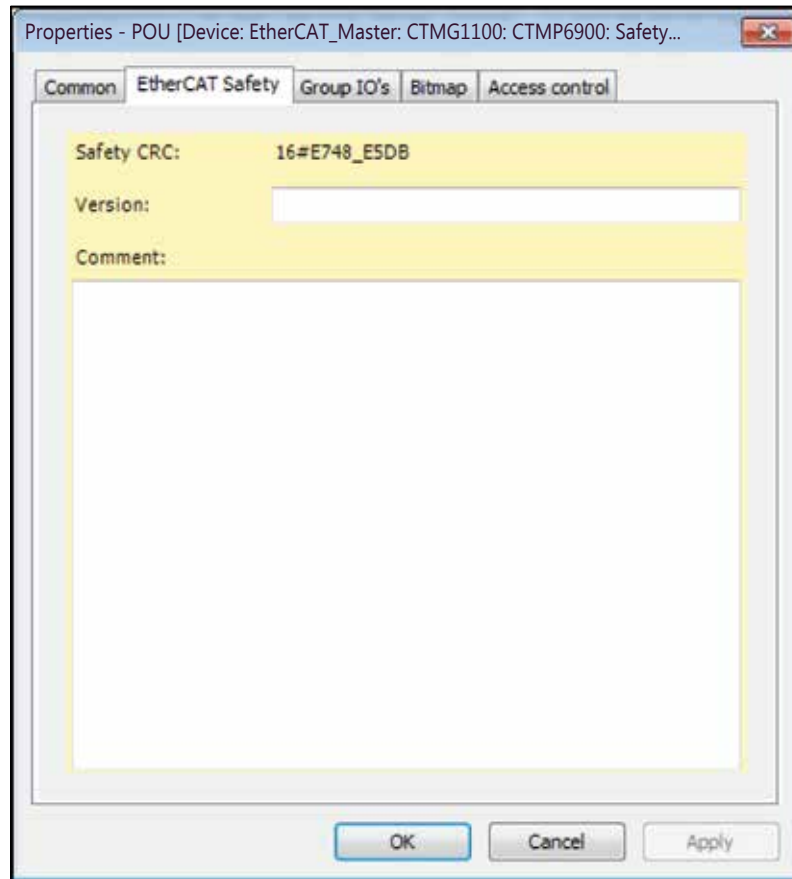


Fig.36 Dialog: Properties of a POU

- “CRC” Checksum for this POU.
- “Version” (editable) The developer can freely assign the version.
The version of the object can be quickly recognised in the comparison view of the safety application object with the aid of the version.
- “Comment” (editable)

Properties - Deactivation

This tab is only available for the CTMP1960-2600 NUMsafe compact controller

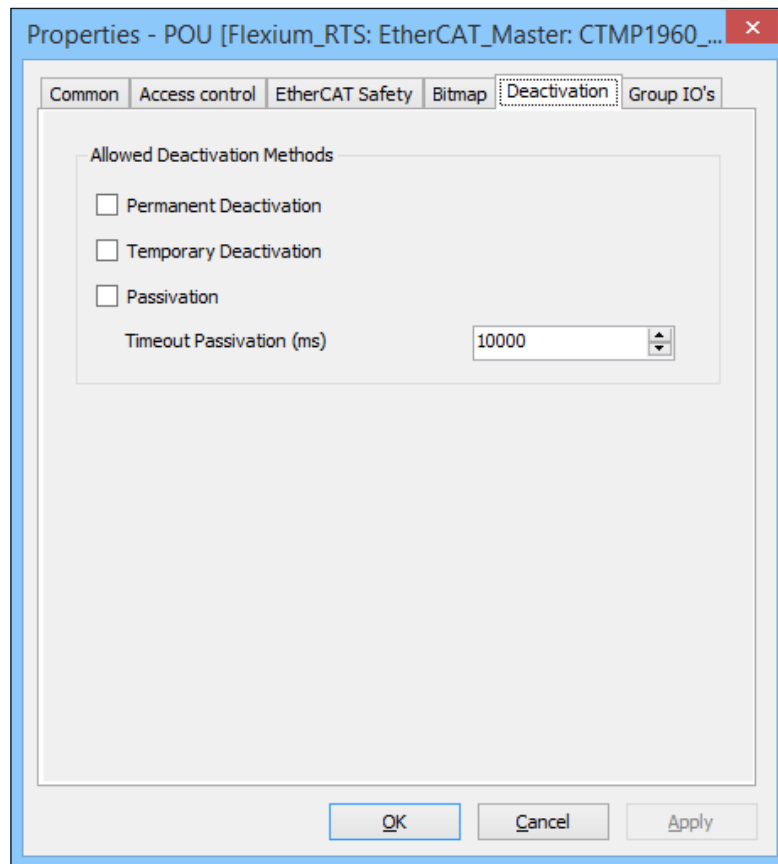


Fig.37 Dialog: Properties of a POU

This tab defines the deactivation methods for the group of this POU. A "group" is defined as a POU running in the safety module with its associated logical devices. For more information, see the online help.

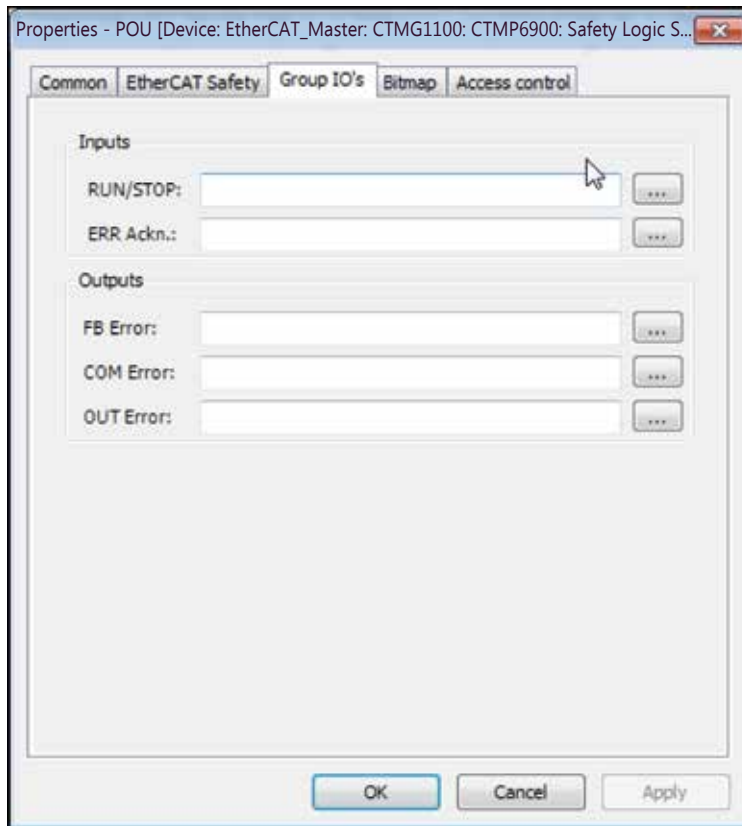


Fig.38 Dialog: Properties of a POU - Group I/Os

Inputs:

- “*RUN/STOP:*” Variable with which the processing of the POU's can be started or stopped (optional).
- “*ERR Ackn.:*” Variable with which an error can be acknowledged (must be occupied).

For inputs you can use:

- Input variables of logical I/O for data exchange with the main controller (See chapter 4.5.4.2.2.2 "Logical I/O for data exchange with the main controller (standard controller)").
- Output variables of function blocks which are used in this or in other EtherCAT Safety POU's. The variables must be defined with POU name, name of the FB instance and name of the output. Example: `POU_1.FB_AND_1.bAndOut`.

Input variables of logical I/Os for data exchange with safe field devices cannot be used.

Outputs:

- “*FB Error:*” Variable with which a function block error can be displayed (optional).
- “*COM Error:*” Variable with which a communication error is displayed. (optional).
- “*OUT Error:*” Variable with which an error of the local outputs is displayed.

For outputs you can use:

- Output variables of logical I/O for data exchange with the main controller (See chapter 4.5.4.2.2.2 "Logical I/O for data exchange with the main controller (standard controller)").
- Output variables of logical I/Os for data exchange with safe field device.

Input variables of the used function blocks cannot be used.

4.5.4.4 Library manager

The libraries that are usable by the safety application are managed in the library manager. The list of the libraries originates from the device description of the NUMsafe controller, which contains the safety application with the library manager.

All libraries available for this NUMsafe controller are automatically inserted.



The library manager of "CODESYS Safety for EtherCAT Safety Module" corresponds to the library manager of CODESYS. See the CODESYS online help for details.

For more precise descriptions of the safety libraries, see *Function blocks (10)*.

Addition of the library manager

The "Library Manager" object is usually inserted automatically into the project tree together with the NUMsafe controller. This object can exist only once below a Safety Application object.

It can be added manually by selecting the "Safety App" safety application object in the project tree and executing the "Add Object –" "Library Manager" context menu command.

Object properties

As in CODESYS Standard, the Properties dialog contains the "Common", "Access control" and "Build" dialogs.

Library manager editor

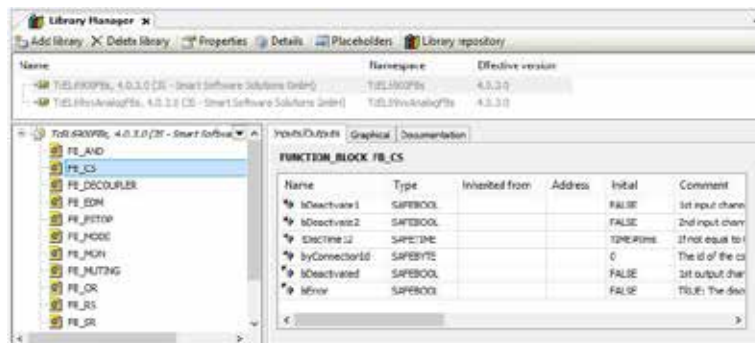


Fig.39 Example: Library manager editor

The editor consists of the following three sections:

- Library list.
- Block list (of a selected library; "TcEL6900FBs" in the example)
- FB description (of a function block selected in the function block list, "FB_CS" in the example) with the dialogs: "Inputs/ Outputs", "Graphical" and "Documentation".

See the CODESYS Standard online help for further details of this editor.

Notes on the Library Manager



Notice!

The Library Manager is not suitable for the verification of the library blocks (IEC blocks and external blocks) used in the safety application during the verification and the acceptance. The verification of the execution-relevant statuses must take place via the comparison view, see *Pinning (7)*.

The list of the libraries can change if the NUMsafe controller is updated..

4.6 Access protection

4.6.1 Overview of access protection

Overview of safety access protection

Rights can be assigned in such a way in the project user management that safety objects of a project can be created or modified only by certain user groups ("Safety Developer").

The entire access protection of "CODESYS Safety for EtherCAT Safety Module" safety applications is realised by the user management in the project.



Notice!

It is urgently recommended to configure a user management with safe access protection for each project!

Without user management in the project, each person who collaborates in the project possesses all rights to the standard and safety applications in the project!.

4.6.2 User management in the project



Notice!

In a project without user management each user possesses **all rights**. The project manager must ensure that the safety application is worked on only by persons qualified to do so.

Creation of a new project without a user management template

If a new project is created as an "*empty project*", then the project manager must create a suitable user management for the safety application of the project. Refer to the Standard CODESYS online help for the exact procedure and detailed information on the creation of this user management in the project.

In order to facilitate the selection of the safety specific commands for the developer, "*Safe*" or "*Safety*" is placed in front of the commands that are relevant for the safety developer when assigning rights. In the rights objects the safety objects are easily recognisable by the suffix "*(Safety)*" after the object.



If the developer does not have the rights for a particular operation, he can logon during the operation using a user name with more rights.

4.6.3 User management – NUMsafe controller

In addition to the standard user management of CODESYS, the NUMsafe controller offers its own user management. This allows users to be setup with the associated passwords. These data must be entered during "Download Project" and "Delete Project". Before changes are made to this user management, the current version of the user management must be loaded from the NUMsafe controller into the CODESYS project ("Upload" button in the "User Administration" section).



Fig. 40 User management - NUMsafe controller

- **"Upload"** : Transfer of the current user management of the NUMsafe controller into CODESYS for display and editing.
- **"Add user..."** : The creation of a new user.
The input of the administrator password is necessary for this.
- **"Delete user..."** : The deletion of the selected user.
The input of the administrator password is necessary for this.
- **"Change password..."** : The current password must be entered before a new password can be defined.
- **"Version history"** : After the execution of the command the version history is read from the NUMsafe controller.

User name	Time stamp	Version number	Project CRC
Administrator	18.09.2012 14:46:21	1	16#B013
Administrator	25.09.2012 11:40:30	1	16#12CB
Administrator	26.09.2012 10:47:31	1	16#12CB
Administrator	26.09.2012 11:52:10	1	16#7536
Administrator	27.09.2012 12:09:08	1	16#12CB
User1	28.09.2012 09:51:20	1	16#12CB
User1	28.09.2012 09:52:22	2	16#12CB
Administrator	28.09.2012 09:54:31	2	16#12CB

Fig. 41 NUMsafe controller version history

Access protection with link to source control



Notice!

If a source control is used, then a user management corresponding to the user management in the project must also be set up in the source control.

Page deliberately empty

Summary

5	Programming.....	69
5.1	Programming overview	69
5.2	Safe data flow	70
5.3	Data types.....	71
5.4	Language elements	72
5.4.1	Overview of the language elements	72
5.4.2	Rules for identifiers of safety objects	72
5.4.3	POUs.....	73
5.4.4	Networks	74
5.4.4.1	Overview of networks.....	74
5.4.4.2	Fundamental information about variables.....	77
5.4.4.3	FB calls	78
5.4.4.4	FBD assignments.....	80
5.4.4.5	Monitoring	81
5.5	I/O access	82
5.6	Deactivating of Groups	82
5.7	Programming guidelines	82
5.8	Example projects	84
5.8.1	Example project - step by step.....	84
5.8.2	Application example: two-hand control.....	99

Page deliberately empty

5 Programming

5.1 Programming overview

"CODESYS Safety for EtherCAT Safety Module" supports the developer in the creation of a standard-compliant safety application through:

- The promotion of good programming techniques.
- The prohibition of non-safe language features.
- The promotion of code comprehensibility.
- Facilitated testability.
- Code documentation procedure.

The programming of a safety application takes place in POU (Program Organisation Units).

In the POU the program code is implemented in the IEC 61131-3 FBD language (function block diagram). FBD is characterised by clarity, ease of recognition of programming errors and clear data flow.

The user interface and handling of the "CODESYS Safety for EtherCAT Safety Module" FBD editor correspond to CODESYS Standard.



The FBD-specific commands in "CODESYS Safety for EtherCAT Safety Module" are described with the respective language elements.



Frequently used symbols:

Input Assistant: the  symbol, also called ellipse, marks a button that opens the "Input Assistant" dialog window when actuated.

Language scope of the safety programming with "CODESYS Safety for EtherCAT Safety Module"

The language scope of FBD corresponds to the function blocks of the CTMP6900 NUMsafe controller and CTMP1960-2600 NUMsafe compact controller.

5.2 Safe data flow

The safe data flow of FBD programming is highlighted as follows in "CODESYS Safety for EtherCAT Safety Module":

- Constants and variables declared as constants have a yellow background.
- SAFExxx variables have a yellow background.
- The data flow of SAFE values into SAFE variables and into inputs is represented by thick yellow lines.
- Function blocks are illustrated in yellow if they have at least one SAFE output.

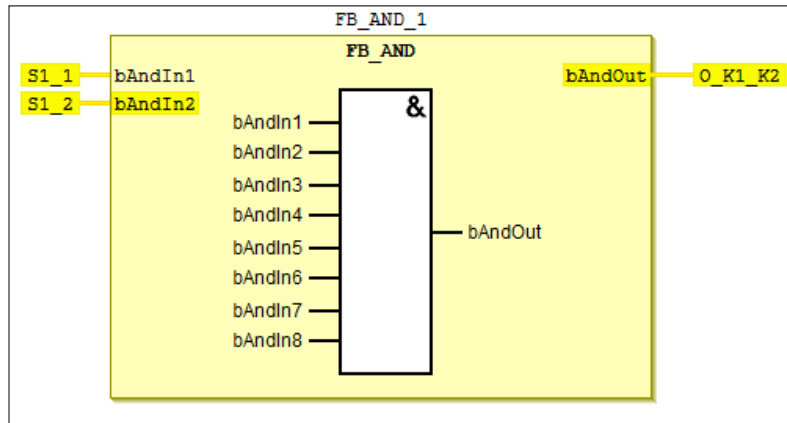


Fig. 42 Example of safe data flow: FB_AND with SAFE variables: S1_1, S1_2 and O_K1_K2

5.3 Data types

The following data types are available for the programming of NUMsafe controllers.

Data type	Bit length	Value range
BOOL	1	0 (FALSE), 1 (TRUE)
SINT	8	-128 ... 127
USINT	8	0 ... 255
INT	16	-32.768 ... 32.767
UINT	16	0 ... 65.535
DINT	32	- 2.147.483.648 ... 2.147.483.647
UDINT	32	0 ... 4.294.967.295
LINT	64	$-2^{63} \dots 2^{63}-1$

Table 1: IEC standard data types

Data type	Bit length	Value range
SAFEBOOL	1	0 (FALSE), 1 (TRUE)
SAFEBYTE	8	0 ... 255
SAFESINT	8	-128 ... 127
SAFEUSINT	8	0 ... 255
SAFEINT	16	-32.768 ... 32.767
SAFEUINT	16	0 ... 65.535
SAFEDINT	32	- 2.147.483.648 ... 2.147.483.647
SAFEUDINT	32	0 ... 4.294.967.295
SAFELINT	64	$-2^{63} \dots 2^{63}-1$
SAFETIME	32	0 ... 2.147.483,647 s

Table 2: Safety-related data types

5.4 Language elements

5.4.1 Overview of the language elements

The following objects and their elements generally serve the implementation of the program code of a safety application:

- POU:
 - Networks
 - FB calls.
- Logical I/Os:
 - Mapping variables.
- Function blocks of the safety library which is referred to in the library manager.

5.4.2 Rules for identifiers of safety objects

- Names for POU, I/O modules, variables, networks (labels), etc. are identifiers as defined in IEC 61131-3. Hence, they must take the following form:
they are a sequence of letters, numbers and underscores, wherein the first character may not be a number and the last character may not be an underscore and two underscores may not directly follow each other.
- Identifiers beginning with an underscore are reserved for implicit code and external FBs and may not occur in POU.
- The IEC keywords listed in IEC 61131-3 (2003-1, Programmable controllers, Part 3: Programming languages - Annex C) may not be used as identifiers.
- The keywords extending beyond the IEC, which are reserved by CODESYS Standard, may not be used as identifiers.
- The following are reserved as "CODESYS Safety for EtherCAT Safety Module"-specific keywords and may not be used as identifiers:
 - IOAPI, IOIN, IOOUT, SYSONLY
 - SAFEXXX for all elementary standard data type names XXX
 - FB_AND, FB_CS, FB_DECOUPLER, FB_EDM, FB_ESTOP, FB_MON, FB_MUTING, FB_MODE, FB_OR, FB_RS, FB_SR, FB_TOF, FB_TON, FB_TWOHAND, FB_ADD, FB_CAMMONITOR, FB_COMPARE, FB_COUNTER, FB_DIV, FB_ENVELOPE, FB_LIMIT, FB_LOADSENSING, FB_MUL, FB_SCALING, FB_SLI, FB_SPEED, FB_SUB, FB_VIOLATIONCNT for all standard function block types FB. FBs with such names are allowed only for external FBs.
- Among the objects that belong to the safety application – POU, I/O modules, library function blocks – no two may have the same name.

5.4.3 POU

The program code of a safety application is created in a POU (Programmable Organisation Unit). (For the addition of a new POU, see *Chapter 4.5.4.3 "POUs"*).

The editor of a POU consists of an implementation window in which the program code is created.

In the implementation window the program code is created in successive networks, which are processed in chronological order.

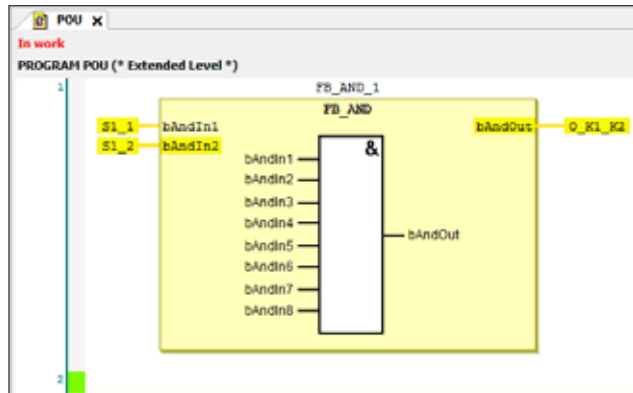


Fig. 43 Example POU editor: Implementation section with a network

For details of the POU editor, see *Chapter 5.4.4.2 "Fundamental information about variables"*.



Various operating functions are available to the developer for the activation of commands and programming in the FBD. Operation via the context menu is usually described for the methods and concrete examples in this manual.

Commands that are either not listed or cannot be activated in the current context menu can also not be executed at the momentary cursor position.

Standard commands

The following standard CODESYS commands can be executed in the implementation section of the FBD editor:

- Copy
- Delete
- Cut
- Past
- Undo
- Redo

Furthermore, the following CODESYS commands are supported:

- Input Assistant
- Find & replace
- Select All
- Bookmark
- Go To Definition
- Browse Cross References

The commands behave as in Standard CODESYS.



Refer to the CODESYS Standard online help for detailed and fundamental information.

5.4.4 Networks

5.4.4.1 Overview of networks

Programming units in FBD are subdivided into networks which are chronologically numbered and which can contain graphically illustrated elements such as FB calls, assignments or labels. The networks are processed chronologically.

The program code must have a tree structure within a network; this means: no parallel interconnection, no splitting and no explicit feedback loops.

The logical elements of a network are joined by lines to make a tree-like formation (tree for short) with the root to the right; the boxes in this tree function as nodes. Exception: multiple assignments fan this tree out in opposite directions. The elements are connected automatically by the editor according to their insertion position; free placement is not possible.

Fundamental functions of the FBD editor

The following FBD special operators supported in CODESYS Standard are not supported in "CODESYS Safety for EtherCAT Safety Module":

- Edge recognition with assignments (to inputs and variables)
- Set/Reset
- EN/ENO parameters.



Refer to the CODESYS Standard online help for detailed and fundamental information on the use of the FBD editor.

Commands for FBD networks

Commands for networks (can be activated in the context menu):

- *"Insert Network"*
- *"Insert network (below)"*
- *"Toggle network comment state"*
- *"Insert Box"* see Chapter 5.4.4.3 *"FB calls"*.
- *"Insert Empty Box"* see Chapter 5.4.4.3 *"FB calls"*.
- *"Insert Assignment"* see Chapter 5.4.4.4 *"FBD assignments"*.
- *"Remove unused FB call parameters"* see Chapter 5.4.4.4 *"FBD assignments"*.
- *"Update parameters"* see *"Update Parameters"*.
- *"Negation"* see *Negation*.
- *"Define output connection"* see *"Define output connection"*.
- *"Data Type"* (*"INT"* , *"UINT"* , *"DINT"* , *"UDINT"* , *"SAFEINT"* , *"SAFEUINT"* , *"SAFEDINT"* , *"SAFEUDINT"*).

Insert networks

The activation of the “*Insert Network*” context menu command inserts a network in the implementation section of the editor. If networks already exist, the network will be inserted before the current network.

The “*Insert Network Below*” context menu command causes a network to be inserted below the current network.

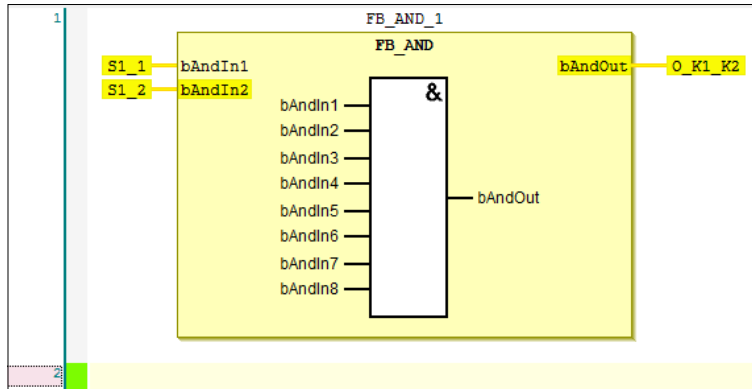


Fig. 44 Example: Network 2 inserted below Network 1

Toggle network comment state

A network is commented out or placed in the normal state by the “*Toggle network comment state*” command. When commented out, the elements contained in the network are ignored and shown as inactive.

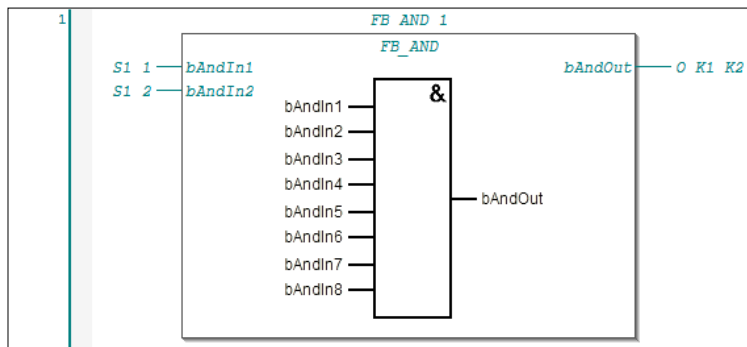


Fig. 45 Example: Commented out network

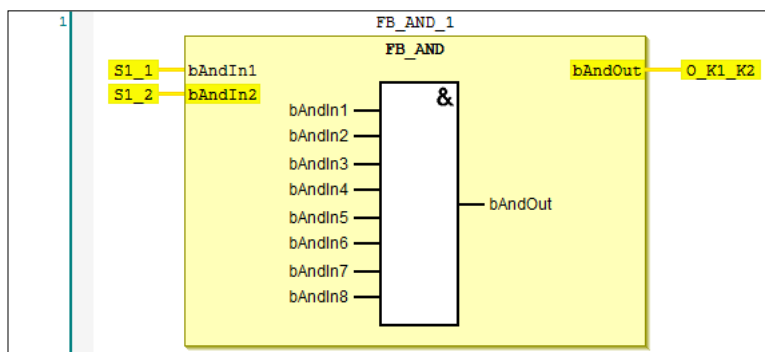


Fig. 46 Example: normal state of the network

Network title and network comment

A title and a comment can be added to each network if the corresponding FBD options are activated.

Activation of the EtherCAT Safety FBD options

1. In the “Tools” menu, open the “Options...” dialog.
2. Activate the “EtherCAT Safety FBD Editor” dialog in the “Options” dialog window.
3. Select the “Show Network Title” and “Show Network Comment” options.
4. Actuate the “OK” button.

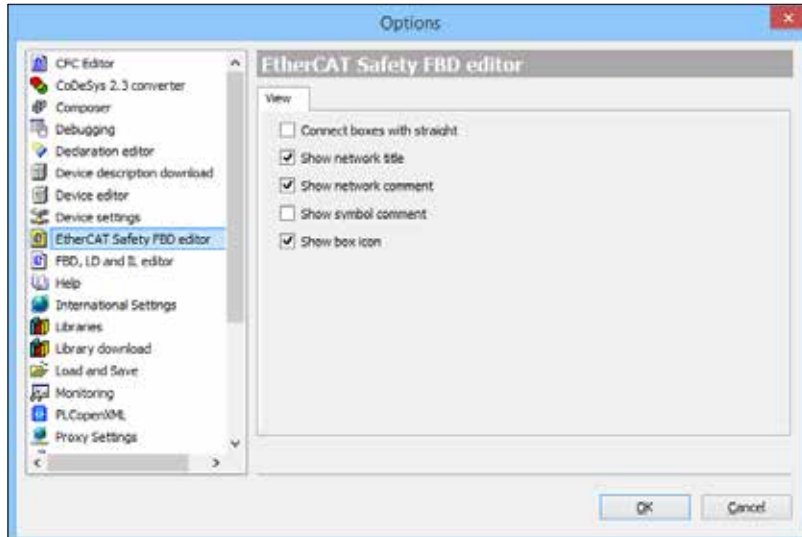


Fig. 47 Dialog: EtherCAT Safety FBD Options

A network title can be edited directly in the first line of the network; the network comment can be edited in the second line of the network. In both cases the respective line must be selected first.

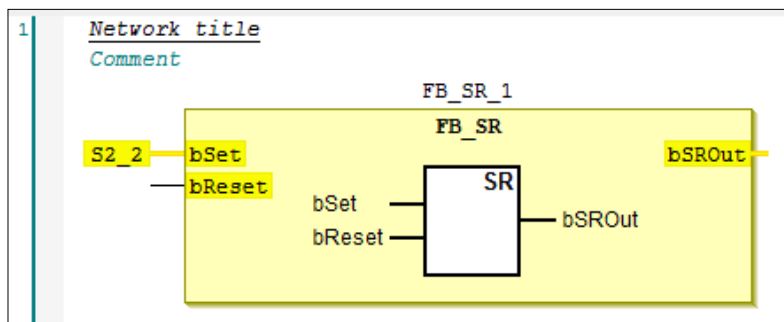


Fig. 48 Network with title and comment

5.4.4.2 Fundamental information about variables

Programming with "CODESYS Safety for EtherCAT Safety Module" takes place in accordance with IEC 61131-3 with the aid of variables. Variables are created by mapped I/Os.

Change markings in the implementation section of the editor

The differences to the previous version are marked in colour after each editing operation. The marking of the last-performed action is always visible. All markings are removed on closing the POU:

- Green: newly added networks or elements
- Red: changes to an existing network/element
- Blue: deletion mark for deleted network or element.
- The network with the change is marked red

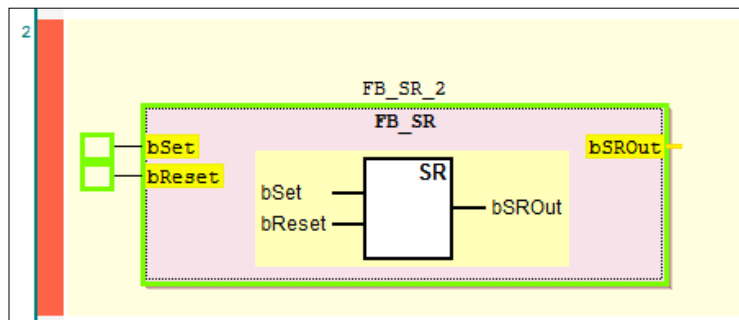


Fig. 49 Example of change marking: newly inserted FB call

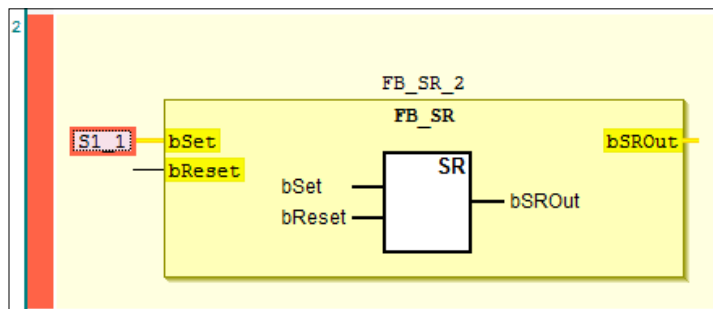


Fig. 50 Example of change marking: input assigned to S1_1

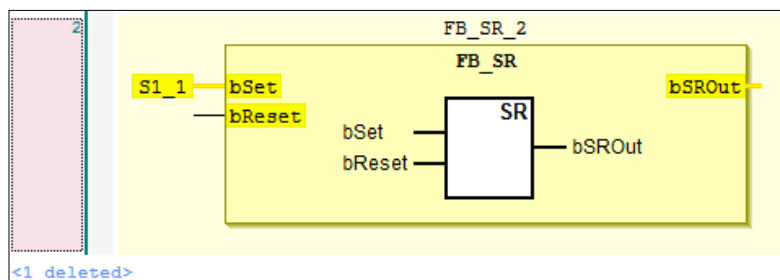


Fig. 51 Example of change marking: deleted network

Automatically checked programming guidelines

Linkage rules for SAFEBOOL data and analogous linkage rules for the other SAFE-xxx data types:

- Only SAFE-typed values may be assigned to safe outputs.



When assigning non-SAFE values to SAFE variables or SAFE inputs, the variables or the input are marked dark red in the editor.

- Assignment to output variables is only allowed at one point in the application.

5.4.4.3 FB calls

Functional principle of FB calls

On calling the function block the formal parameters are supplied with the current values of the input variables or constants, or are assigned to the outputs. The formal parameters and their assigned variables/constants must be of the same data type. The instance name is defined in the declaration section of the editor as a variable with data type = function block name.

FB instances can access only their own POU-wide data and parameters.

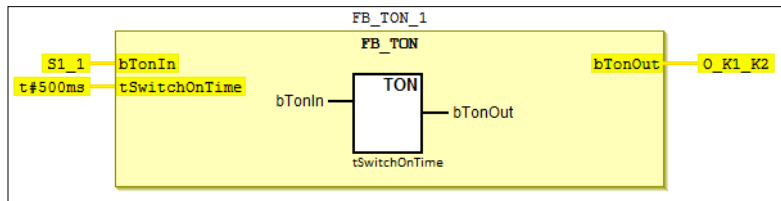


Fig. 52 Example: Instance FB_TON_1 of the FB_TON function block with formal parameters bTonIn, tSwitchOnTime, bTonOut

The function blocks from the "TcEL6900FBs" and "TcEL69xxAnalogFBs" safety libraries are available to the developer in NUM FlexiumTools.

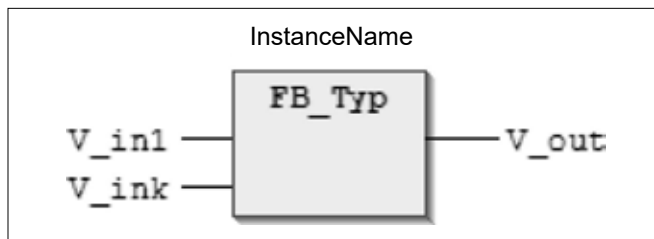


Fig. 53 Example: FB call

Insert Box; Insert Empty Box

The insertion of function blocks takes place in the same way as in standard FBD of CODESYS.

Insertion position

The position in the network at which the selected FB is inserted depends on where the command is executed.

Command is executed:

- In empty network:
The inputs and outputs are additionally labelled according to the signature of the object for function block calls. The instance names of function blocks are displayed via the box.
- At the main output of another box:
The first input of the new box is connected to the main output of the existing box. The insertion of a box at a non-main output of another box is not possible.
- At an input of another box:
The new box is inserted between the ingoing signal flow and the input of the existing box.

Programming guidelines

Deletion of the inputs and outputs: The inputs and outputs of a box that represent a function block call can be deleted with the "Delete" command in the context menu.

In doing so the elements connected to this input or output are also deleted. If the main output of a FB call located within a network is deleted, then the entire tree to the left of this box output will be removed (i.e. including the block itself) in case of a network.

If the box is the (right-hand) end of a network, the last output (main output) can also be removed; the block is then retained and has no outputs.

Negation

"CODESYS Safety for EtherCAT Safety Module" offers the possibility to invert the signal from Boolean inputs. This is done using the "Negation" command, which is available after selecting an input.

Negated inputs are marked by a small circle on the input.

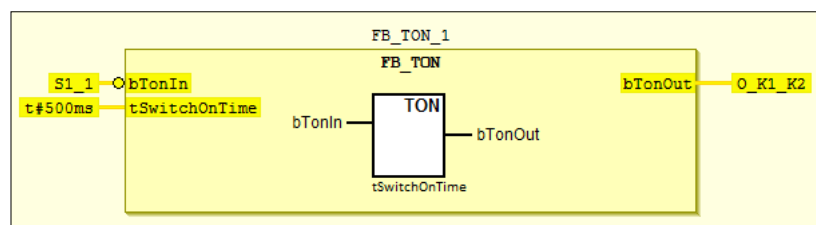


Fig. 54 Example: Negated input

Define output connection

The output that is offered for an output connection to a function block or an assignment can be changed with this command. To this end the output connection must be marked and the "Output connection defined" command must be executed. This command is available only for function blocks with several outputs.

In the example below the standard output connection "bDeactivated" was changed to "bError".

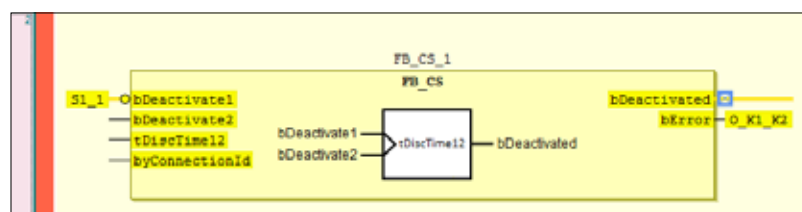


Fig. 55 Example: Define output connection

Update Parameters

The “*Update Parameters*” command updates the inputs and outputs of a function block call.

In doing so, the parameters of the box are compared with those defined in its interface. If a new input or output is added, this is connected to the empty variable. If an input or output is omitted, this is deleted together with all connected elements. Allocation takes place via the names of the inputs or outputs.

Remove unused FB call parameters

The “*Remove unused FB call parameters*” command removes all unoccupied input and output connections of the call.

5.4.4.4 FBD assignments

An assignment is an FBD element that takes up the incoming signal flow in a network and stores it in an operand, i.e. writes it into the variable. A variable is always necessary for an assignment. Assignments can be inserted only at the output of a box.

The insertion of assignments takes place as in the standard FBD of CODESYS.

Examples of assignments

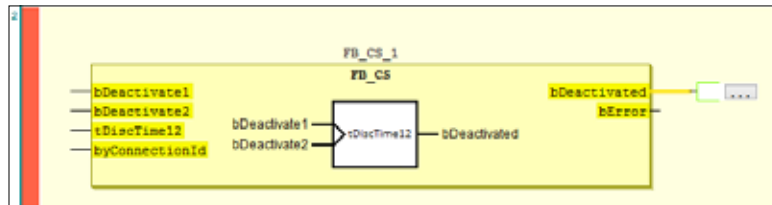


Fig. 56 Example: Assignment at the “bDeactivated” output of the FB_CS

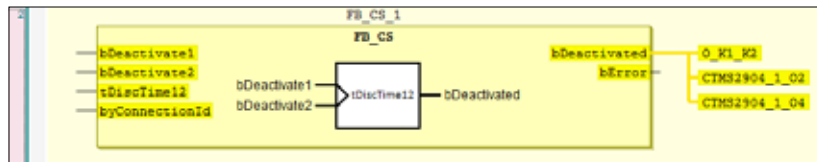


Fig. 57 Example: multiple assignment by inserting a new assignment at the existing assignment

An assignment can be inserted at the beginning of an empty network. This is not permitted, however, and is recognised by the compiler as an error.



Fig. 58 Example: Impermissible assignment



When assigning non-SAFE values to SAFE variables or SAFE inputs, the variables or inputs are marked dark red.

5.4.4.5 Monitoring

Input parameters and output parameters of the Safety FBs can be monitored if the application is logged in (with standard app). As a precondition the application (Safety-App) must be executable and loaded to the controller.

Question marks (?????) instead of the signal states indicates that:

- The library IODrvEL6900 is not up to date. It must be V3.5.3.0 or higher
The EtherCAT stack does not run on the standard controller(after download or reset of the application).
- The application for the CTMP6900 NUMsafe controller or CTMP1960-2600 NUMsafe compact controller in CODESYS does not match to the application on the module.

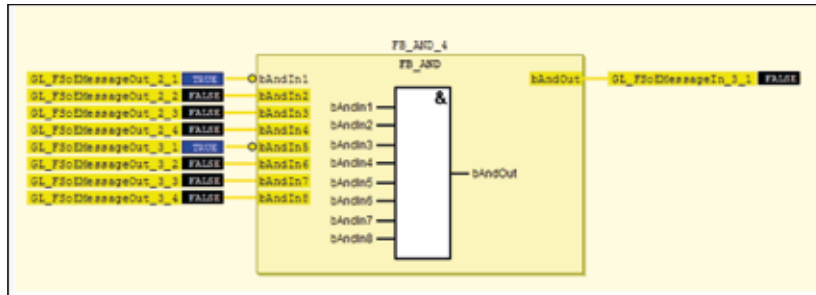


Fig. 59 Monitoring of variables



A change of the signal states during monitoring is not possible.

5.5 I/O access

I/O access takes place exclusively via the logical I/Os.

5.6 Deactivating of Groups

The CTMP1960-2600 NUMsafe compact controller offer the option of deactivating individual groups with an online command. A "group" is defined as a POU running in the NUMsafe controller with its associated logical devices.

As long as a group is deactivated in the NUMsafe controller,

- its POU does not run and cannot be started by RUN.
- its safe field devices are in a fail-safe state.
- other POUs receive replacement values when they read outputs of the FB instances in the deactivated POU

The deactivation of a group takes place in 3 steps

1. Configuration of deactivation methods for a group. The configuration is done in the properties of the POU. Deactivation can be permanent or temporary.
2. Configuration of the substitute values in case of deactivation of the group.
3. Online command to deactivate or reactivate the deactivatable groups.

For details on how to deactivate groups, see the online help.

5.7 Programming guidelines

Independent of the standard on which the certification is based, adherence to the following guidelines is recommended:

- The documentation must be complete, available, legible and understandable.
- All changes over all life cycles must be documented.

The project documentation must contain:

- legal entity (company)
 - functional description of the requirements of the project
 - I/O description
 - version of the function block library employed.
- Each POU must have the following available information:
 - author
 - date of creation
 - date of release
 - version
 - version history
 - sufficient commenting on the networks
 - sufficient commenting on the declaration lines.

Commenting

"CODESYS Safety for EtherCAT Safety Module" offers the following possibilities to comment on the safety application, its objects and their contents:

- Comment field for each POU of the safety application ("Properties" dialog of the POU).
- Comment field for the entire safety application ("Properties" dialog of "Safety App").
The source code information demanded by the standard can be entered here:
company, author, description, inputs and outputs and configuration management history.
- Comment field for each logical I/O ("Properties" dialog of the logical I/Os).
- Comment on each FBD network (optional).
- Network title (optional).

The display of the network title and comments on network and variables can optionally be set ("Options" dialog of the "Tools" menu).



Notice!

It is recommended to display network title and network comment.

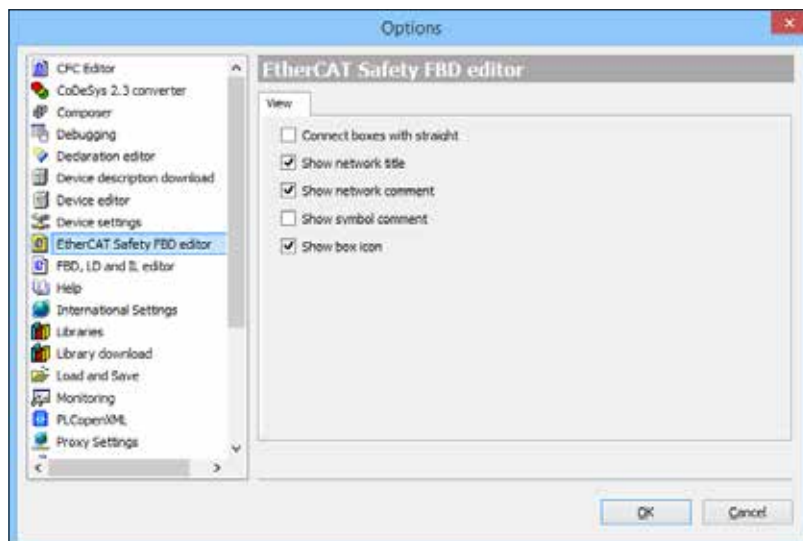


Fig. 60 Settings for the display of comments and titles of the networks

5.8 Example projects

5.8.1 Example project - step by step

Step 1: Creating a new project

At this point an example project will be created with "CODESYS Safety for EtherCAT Safety Module".

Creation of a new project with a safety application:

1. To create a new project: activate the "New Project..." command in the File menu.
2. Select "Empty project" or "Standard project" in the dialog window. Enter a file name and location, then actuate the "OK" button.

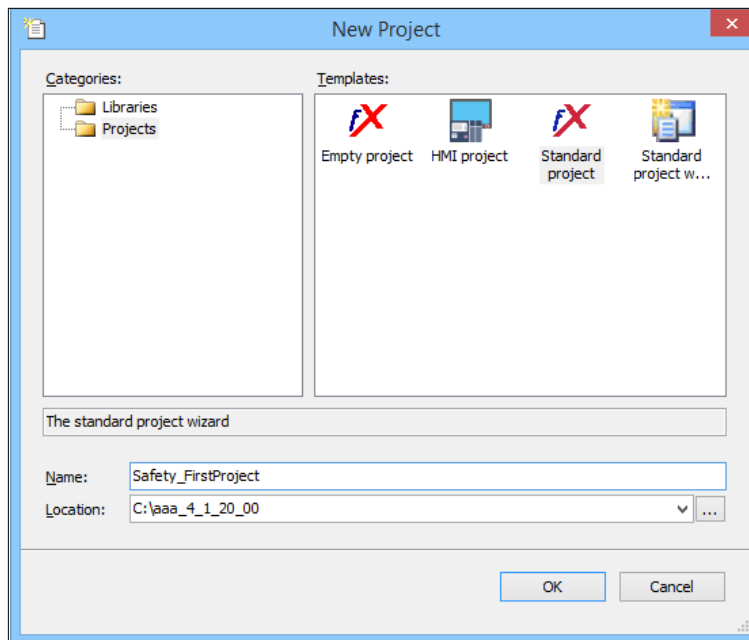


Fig. 61 Creating a new project

3. In the next dialog window push "Next"

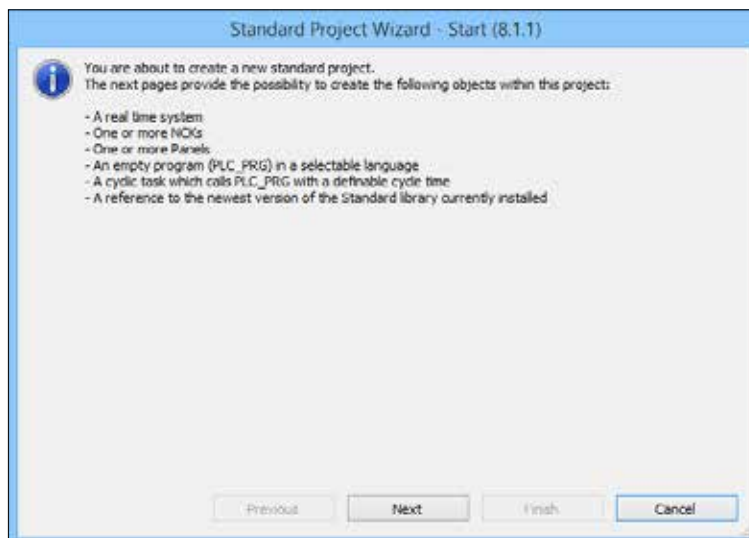


Fig. 62 New project - standard project

4. Choose the right RTS version depending from the Windows operating system where the project will run:
 Flexium RTS (NUM) for 32 bit operating system
 Flexium RTS x64 (NUM) for 64 bit operating system
 Then push "Next"



Fig. 63 New project - Flexium RTS version selection

5. Choose the right NCK type depending from your device:
 Flexium NCK
 or
 Flexium+ NCK
 Then push "Next"

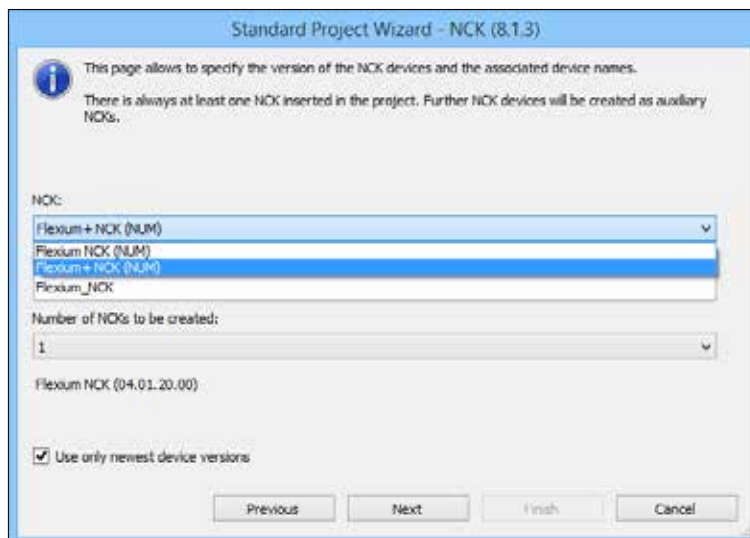


Fig. 64 New project - NCK type selection

6. In the next dialog window push “Next”

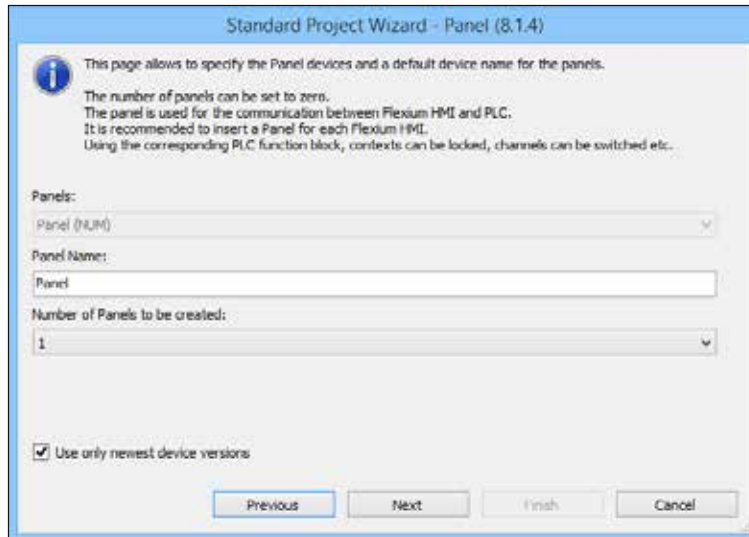


Fig. 65 New project - standard project

7. Choose “PLC Programming Language”: Structured Text (ST)

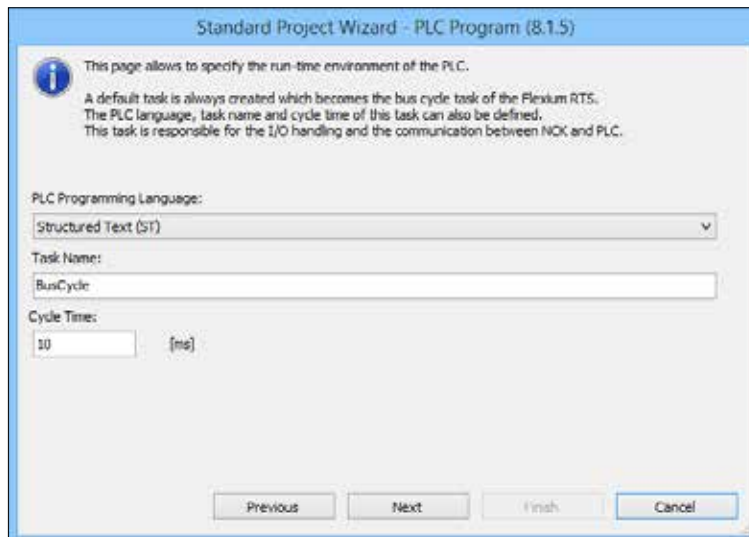


Fig. 66 New project - PLC programming language selection

8. In the next dialog window push “Next”

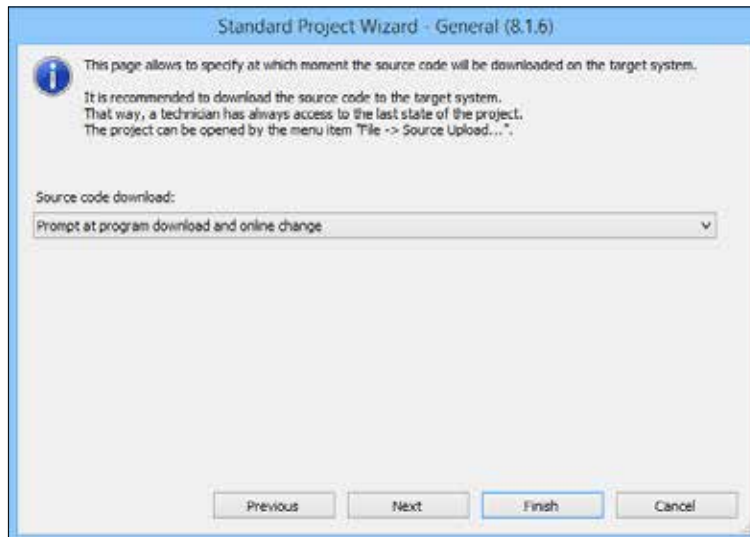


Fig. 67 New project - standard project

9. Check everything and push “Finish”

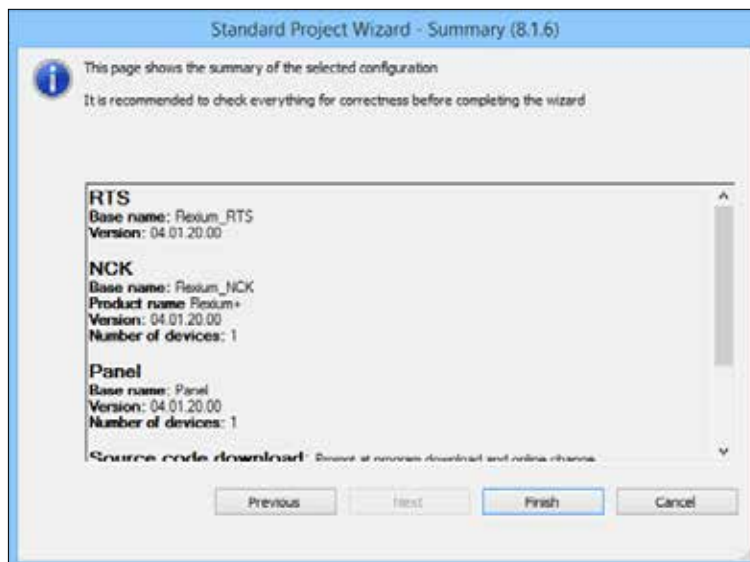


Fig. 68 New project - Configuration end

After quitting the dialog with “Finish” the new project is created; all components are shown in the device tree.

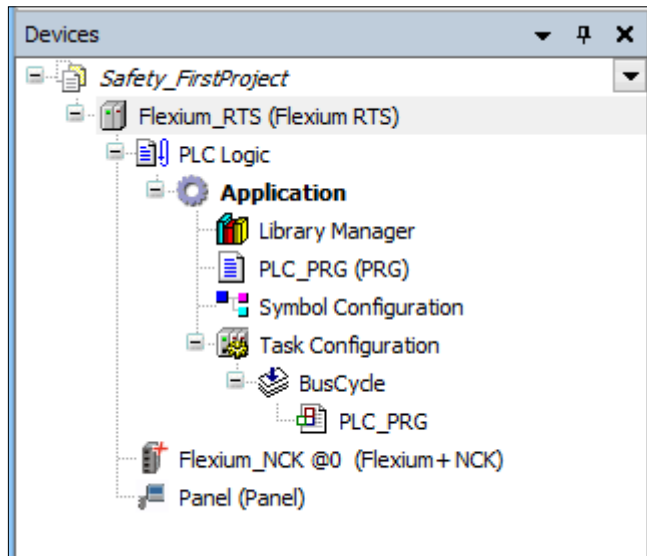


Fig. 69 New project - standard project

Step 2: Adapting the communication settings of the controller

Double-clicking on the device in the device tree opens the associated dialog on the right. The desired controller is now selected in the controller overview.

This is done by selecting the controller and executing the “Set active path” command or with a double-click. The controller is then shown in bold lettering.

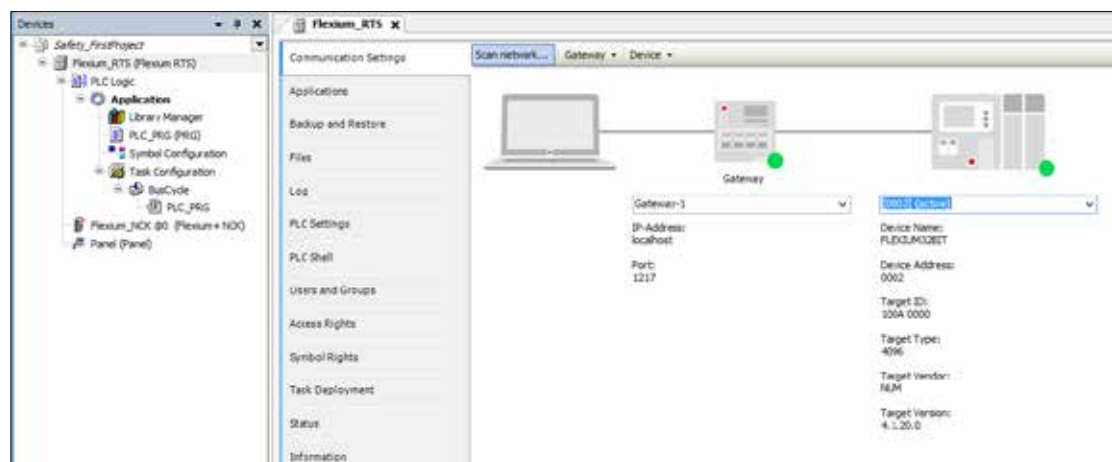


Fig. 70 Communication Settings

Step 3: Adding an EtherCAT Master to the device tree

A right-click on the controller opens the context menu, from which the "Add device..." command is selected. The "EtherCAT Master" device is then selected in the following dialog.

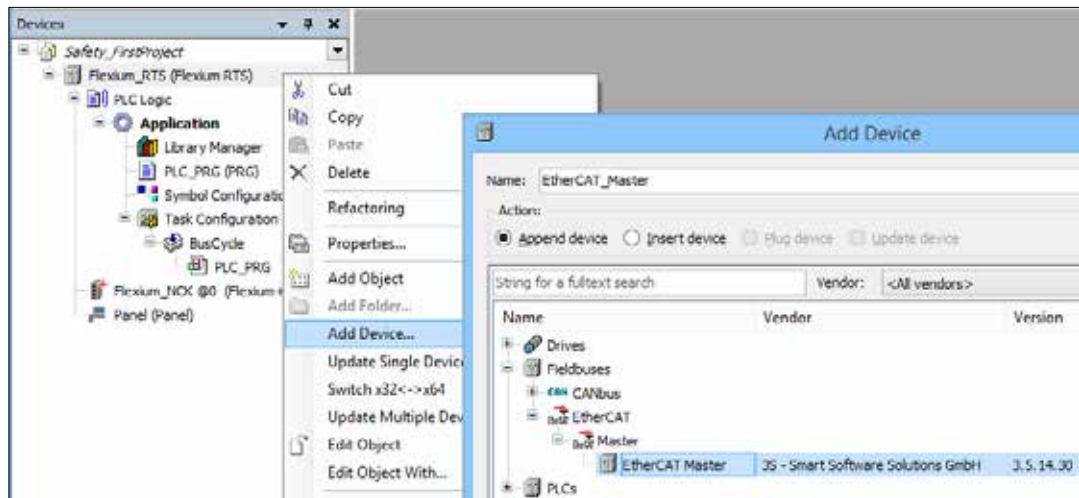


Fig. 71 Inserting an EtherCAT Master

Step 4: Adapting the EtherCAT Master settings

A double-click on the "EtherCAT Master" device in the device tree opens the associated editor for setting the parameters. In this example the desired network adaptor is selected and the dialog is quit with "OK".

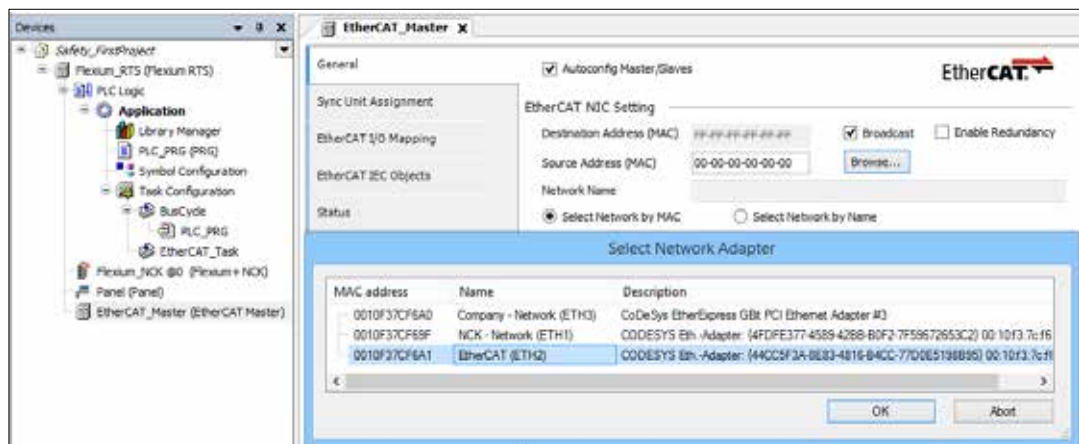


Fig. 72 EtherCAT Master - settings

Step 5: Login and download

After making the settings for communication to the EtherCAT Master, log into this controller ("Online" menu). The application on the controller is thereby overwritten. Subsequently, log out again.

Step 6: Reading in the found hardware

The "Scan For Devices" command opens an overview that lists all devices found.

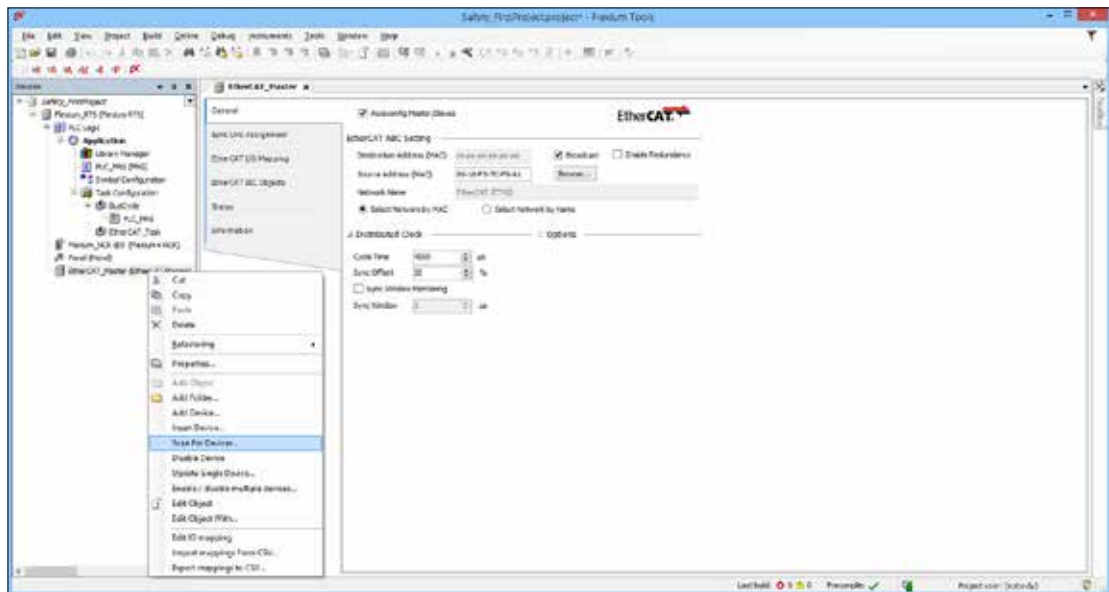


Fig. 73 EtherCAT Master - Scan for devices

Using the command "Copy all devices into project", the found hardware is automatically adopted into the project and shown in the device tree.

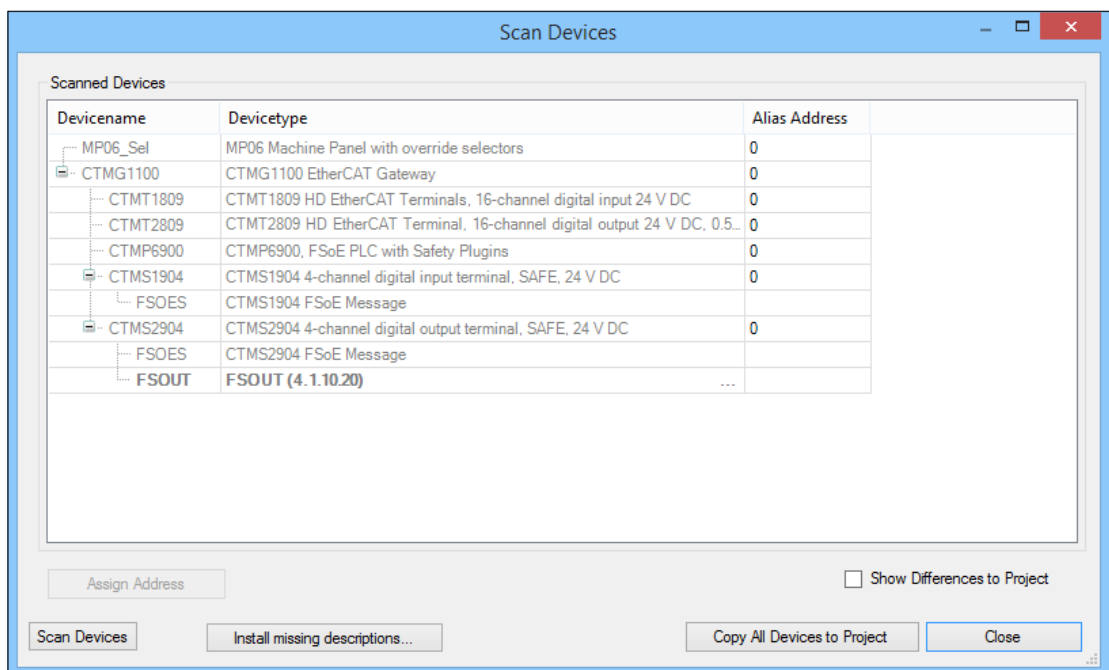


Fig. 74 EtherCAT Master - Found devices

Step 7: Settings of the parameters of the logical I/Os

In the next step the addresses of the logical devices are set. To this end the FSoE address "2" is set for the "CTMS1904" module and the address "3" for "CTMS2904". This address corresponds to the address of the I/O module, which has been set by DIP switches for example.

The connection ID must be parameterized in such a way that every logical I/O has its own unique address. This ensures that the secure IO module can recognize errors arising from wrong forwarding of FSoE telegrams. The connection ID is also tested in the safe IO module. CODESYS does not perform a uniqueness test.

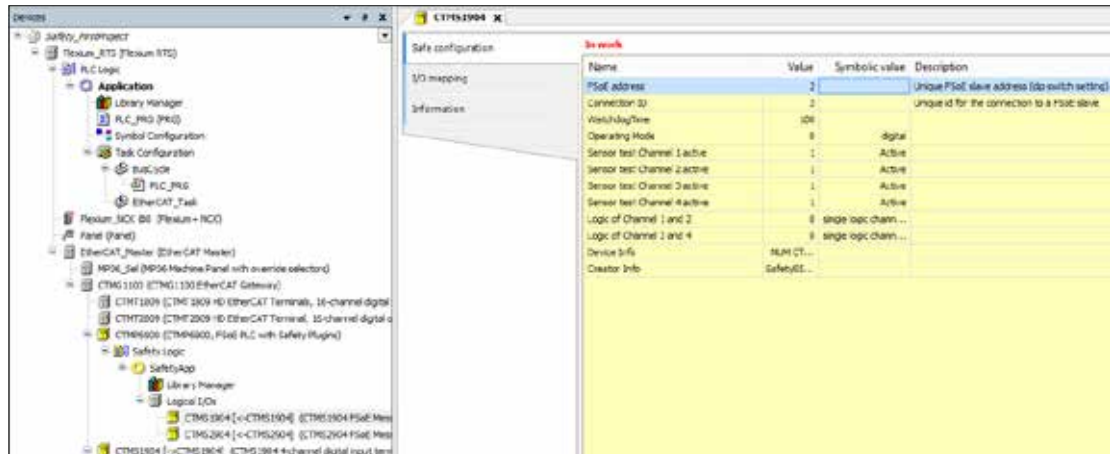


Fig. 75 Logical I/Os - settings

Step 8: Definition of the I/O image of the logical devices

Variables can be assigned to the individual channels of the module on the second tab of the editor. This is defined for the two modules as shown below:

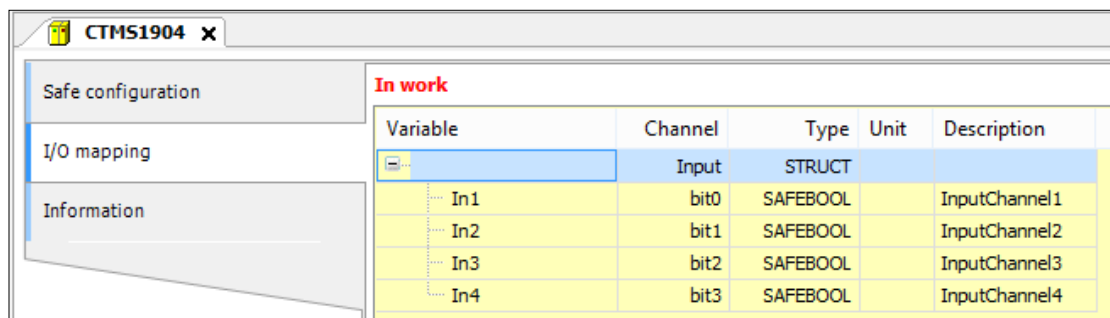


Fig. 76 CTMS1904 logical inputs mapping

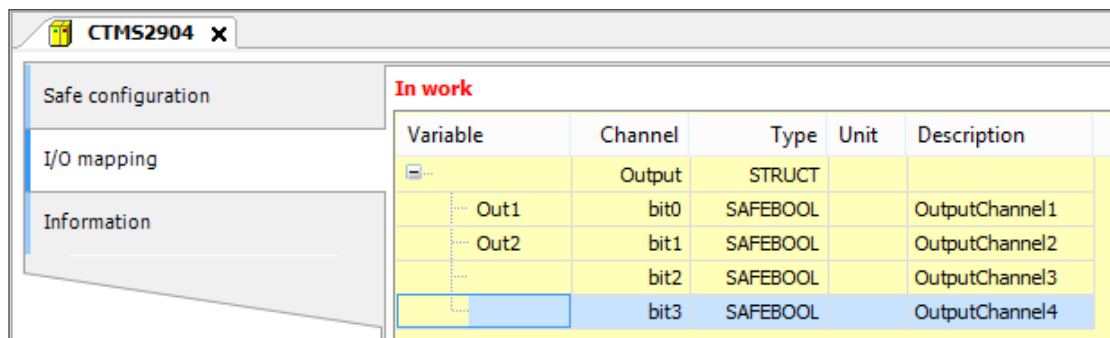


Fig. 77 CTMS2904 logical inputs mapping

Step 9: Creation of a program (POU)

A POU is added via the SafetyApp context menu or via the menu command "Project → Add Object → POU (EtherCAT Safety)"

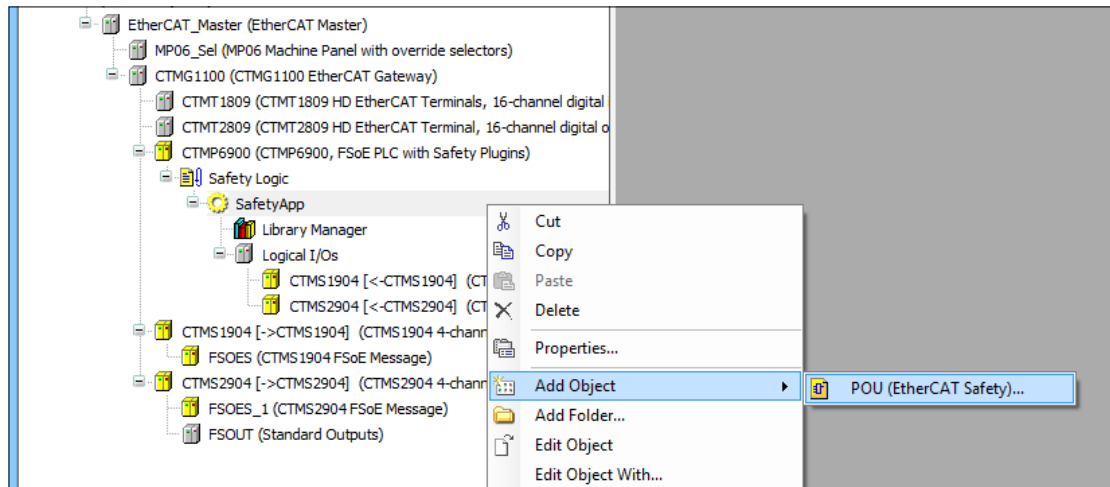


Fig. 78 Addition of a POU

The "FB_TWOHAND" function block is inserted in the editor of the added POU – this is done by selecting the FB in the Tools overview and moving it to the "Start here" field on the desktop by drag&drop.

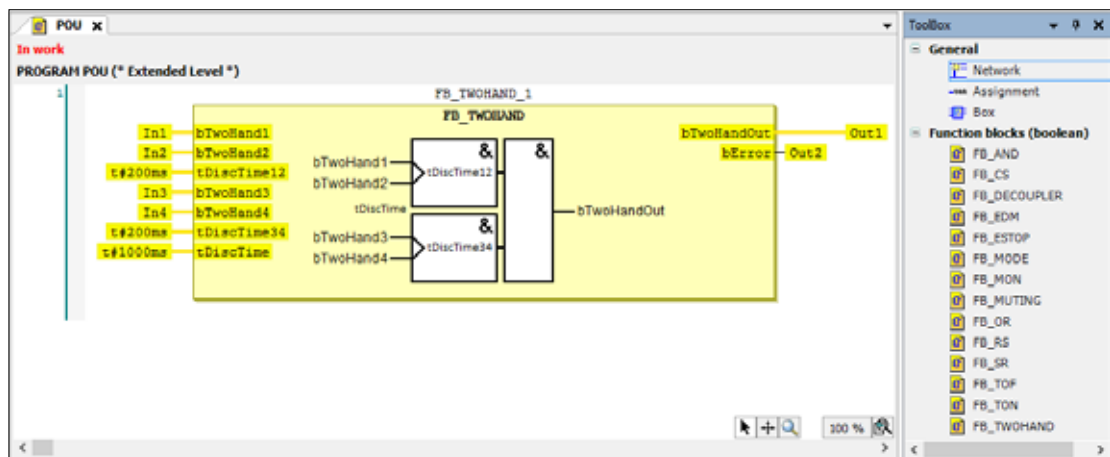


Fig. 79 Addition of the "FB_TWOHAND" FB

The inputs and outputs of the function block are connected with variables according to the diagram above.

Step 10: Setting the group I/Os of the "POU" program

The group I/Os are set in the properties of the newly created POU (right-click on the POU in the device tree - "Properties"). These variables control the state of the POU and can signal errors.

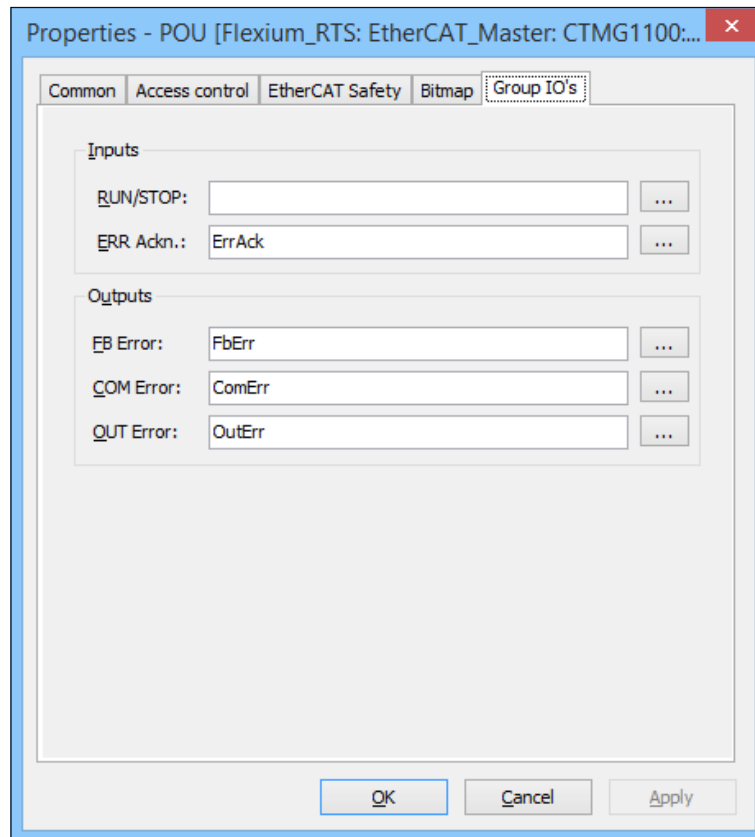


Fig. 80 POU - Properties

Step 11: Addition of logical devices for data exchange

Now further logical devices are added which will handle the exchange of data with the standard controller later on. To do this the "Add logical device" dialog is called ("context menu of logical I/Os → Add Object → Logical Device").



Fig. 81 Addition of a logical device

The device "Exchange 8 bits from CTMP6900 to PLC" is selected from the dialog and is given the name "CTMP6900_to_PLC".

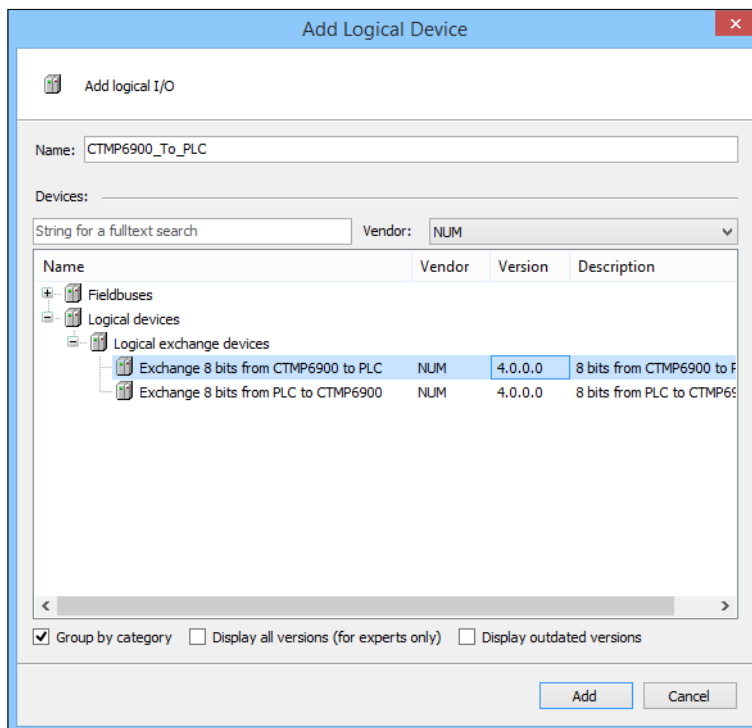
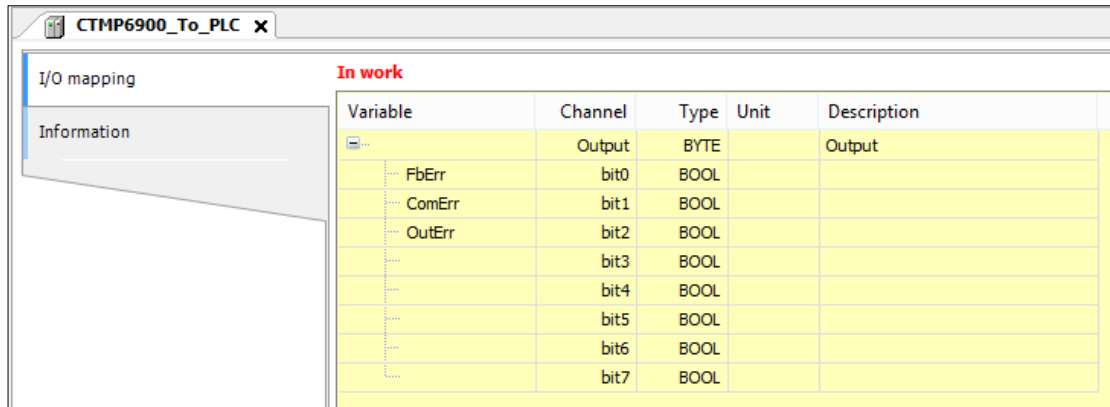


Fig. 82 Addition of a logical device– exchange device

In a similar way a logical device "Exchange 8 bits from PLC to CTMP6900" is created with the name "CTMP6900_from_PLC".

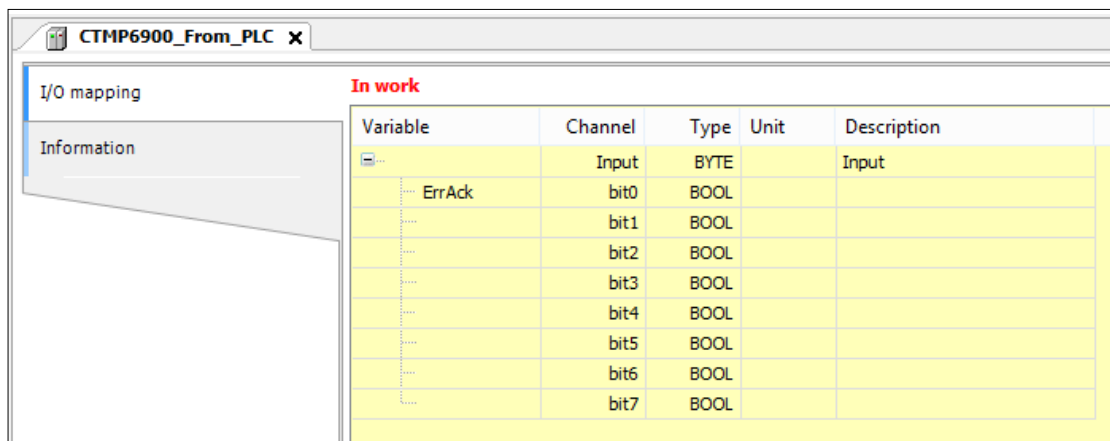
Step 12: Definition of the variables of the exchange objects

Double-clicking on the newly added logical devices opens the associated editor. The variables are entered here as shown below. These variables are subsequently connected with a global variable list of the standard controller.



Variable	Channel	Type	Unit	Description
...	Output	BYTE		Output
... FbErr	bit0	BOOL		
... ComErr	bit1	BOOL		
... OutErr	bit2	BOOL		
...	bit3	BOOL		
...	bit4	BOOL		
...	bit5	BOOL		
...	bit6	BOOL		
...	bit7	BOOL		

Fig. 83 Logical device CTMP6900 --> PLC: variable definition



Variable	Channel	Type	Unit	Description
...	Input	BYTE		Input
... ErrAck	bit0	BOOL		
...	bit1	BOOL		
...	bit2	BOOL		
...	bit3	BOOL		
...	bit4	BOOL		
...	bit5	BOOL		
...	bit6	BOOL		
...	bit7	BOOL		

Fig. 84 Logical device PLC --> CTMP6900: variable definition

Step 13: Addition of global variable lists for the logical exchange

Now GVLs for the logical exchange are added to the device tree ("context menu of the application → Add Object → GVL for logical exchange").

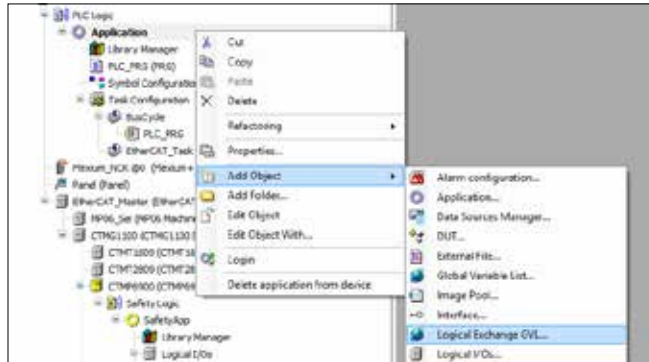


Fig. 85 Addition of a GVL for logical exchange

The new GVL (with the name "fromCTMP") is added to the device tree. A double-click opens its editor. The variables of the logical device "CTMP6900_To_PLC" are now mapped to this GVL in the lower section of the editor.

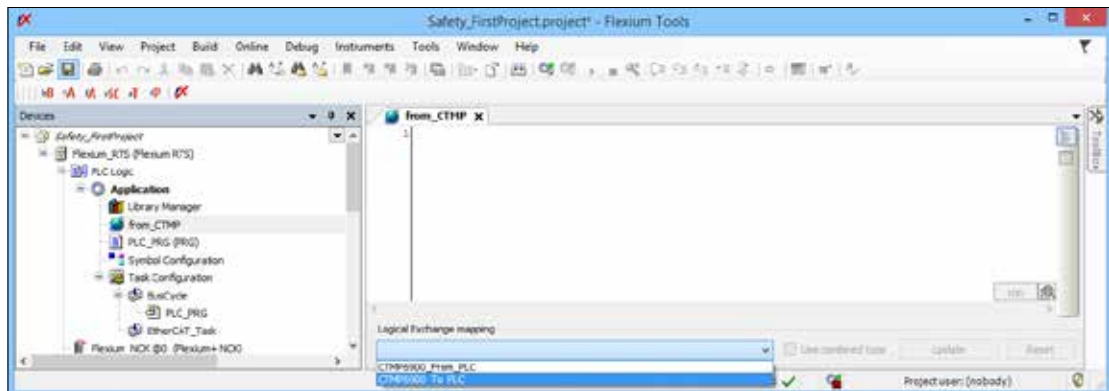


Fig. 86 GVL - Assignment of the logical I/Os

The GVL "toCTMP" is created in a similar way and connected with the logical I/Os of "CTMP6900_To_PLC".

Step 14: "PLC_PRG" program

The variables of the safety application are read and written via the "PLC_PRG" program. The `FbErr`, `ComErr` and `OutErr` variables were inserted so that the states can be monitored. If `bAutostart` is set, the `ErrAck` flag is set for one cycle in the case of an error in the safety block (recognised via `ComErr`) so that the error is acknowledged in the safety controller.

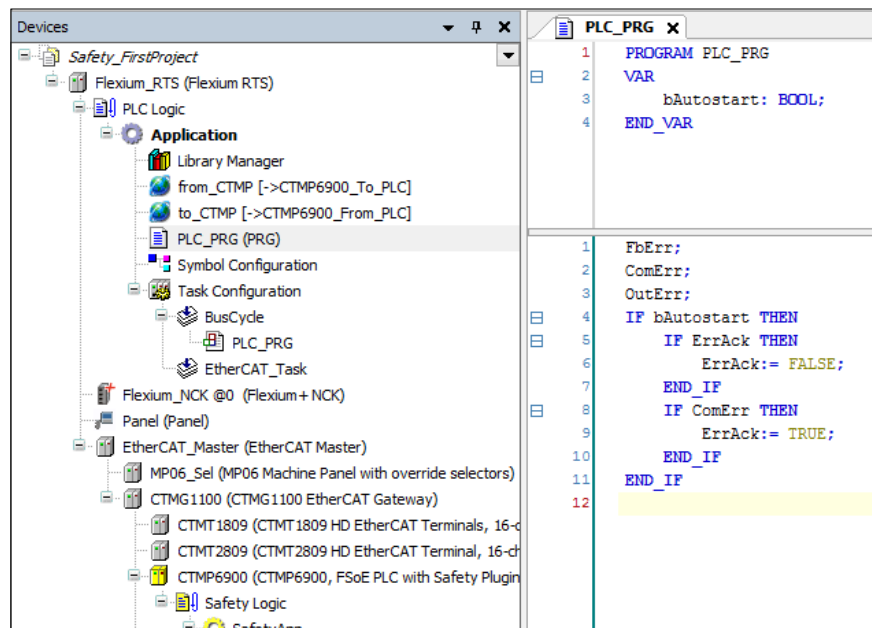


Fig. 87 PLC_PRG program

Step 15: Transferring the standard application to the controller

Finally, log in to the standard controller, transfer the application (download) and start it. The states of the variables can now be monitored in the online view.

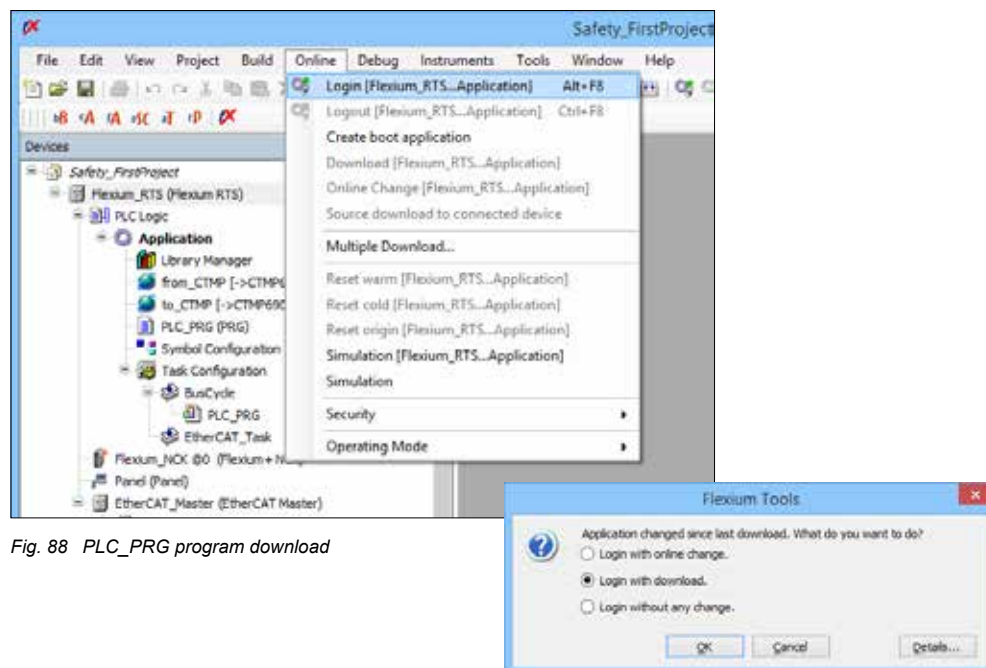


Fig. 88 PLC_PRG program download

Step 16: Downloading the safety program

Now the safety program is loaded to the NUMsafe controller by actuating the “Download project” button in the “Safety parameters” dialog of the NUMsafe controller.

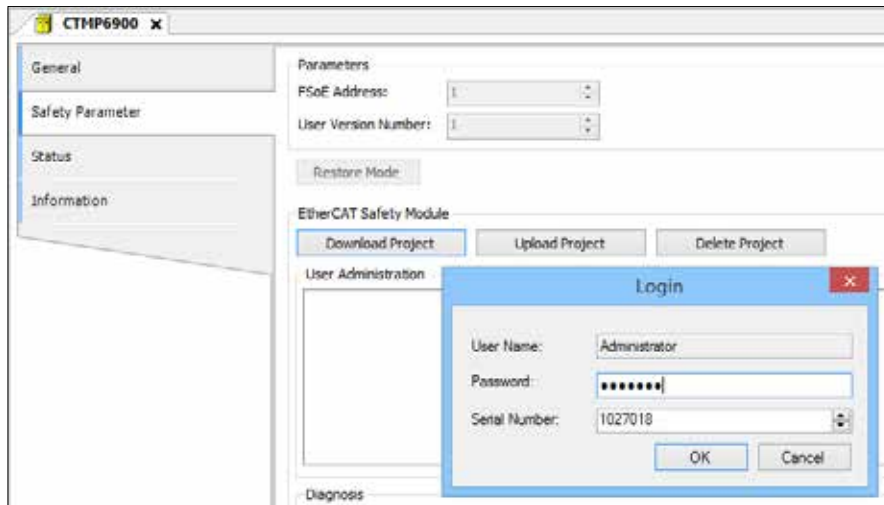


Fig. 89 PLC_PRG program - Download safety program

A login dialog is displayed. In this dialog the user name and password as well as the serial number of the NUMsafe controller must be entered. The user must be registered in the NUMsafe controller user management.

Step 17: Run program on NUMsafe controller

After the project comparison the user name, password and serial number of the NUMsafe controller must be entered a second time to start the project. After confirmation with “OK” the safety program runs on the NUMsafe controller .

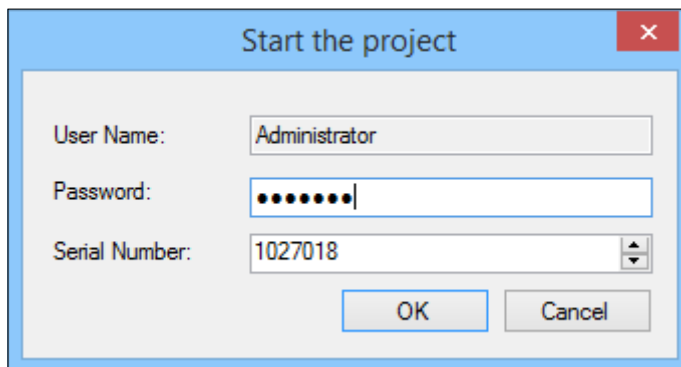


Fig. 90 PLC_PRG program - NUMsafe controller program start

Now the POU can be opened in the editor and the inputs can be monitored

5.8.2 Application example: two-hand control

A two-hand control (category 4, PL-e) is used as a programming example of "CODESYS Safety for EtherCAT Safety Module" and illustrated as an FBD implementation.

The two-hand pushbuttons each consist of a combination of normally closed and normally open contacts on the safe inputs of an input module (e.g. CTMS1904).

Testing of the inputs is active and the signals are checked for discrepancy (200 ms). In addition the synchronous actuation of the two pushbuttons is activated with a monitoring time of 500 ms.

The feedback loop is read in via a safe input. The contactors K1 and K2 are connected in parallel to the safe output. The measurement of current and testing of the output are active for this circuit.

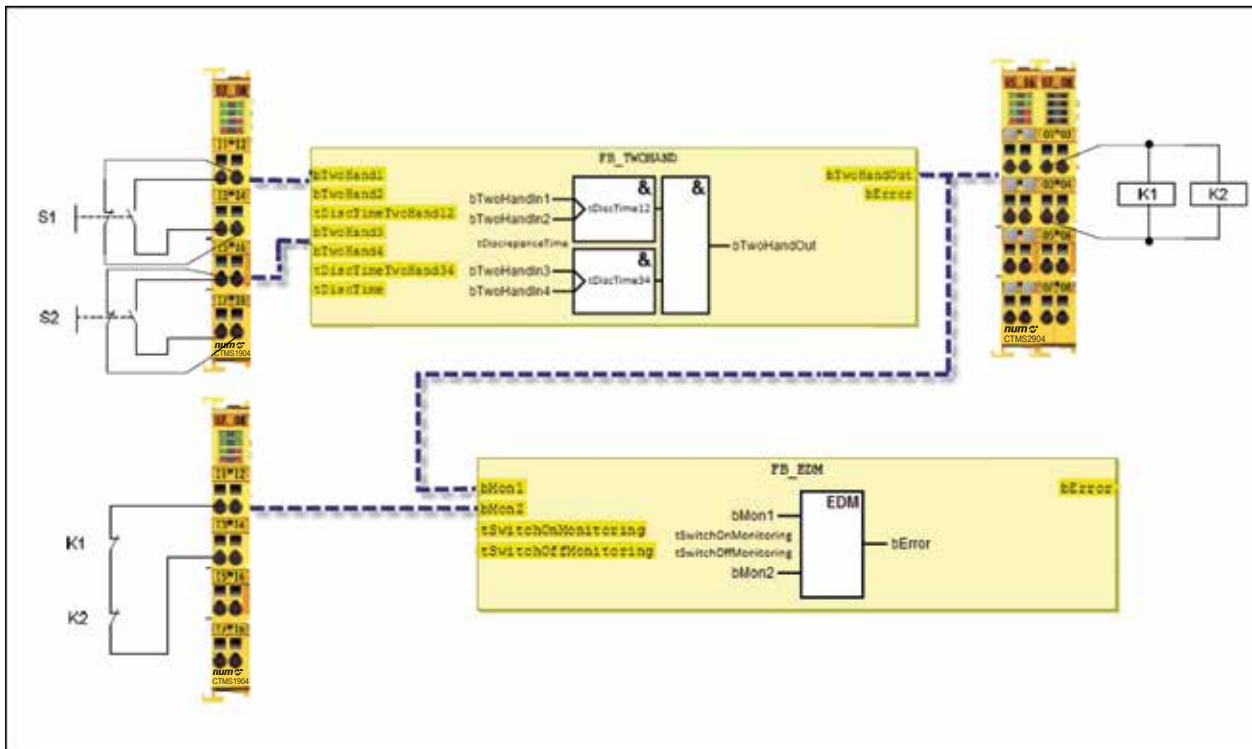


Fig. 91 Structure of the two-hand control with EDM

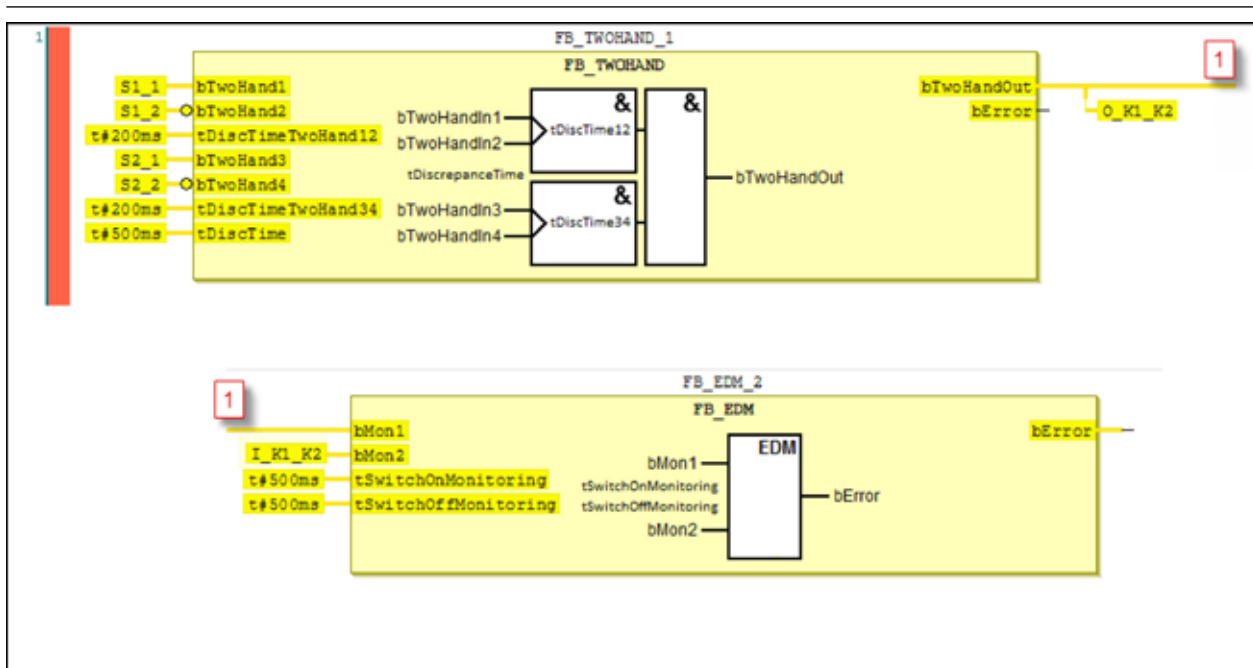


Fig. 92 Implementation for programming example: two-hand control with EDM

Inputs:

Name	Data type	Description
S1_1	SAFEBOOL	Two-hand pushbutton 1 N.O.
S1_2	SAFEBOOL	Two-hand pushbutton 1 N.C.
S2_1	SAFEBOOL	Two-hand pushbutton 2 N.O.
S2_2	SAFEBOOL	Two-hand pushbutton 2 N.C.
I_K1_K2	SAFEBOOL	Contacts of K1 and K2 for reading back

Outputs:

Name	Data type	Description
O_K1_K2	SAFEBOOL	Actuation of the contactors K1 and K2
bError	BOOL	Error flag of FB_TWOHAND_1
bError	BOOL	Error flag of FB_EDM2

Function block parameters

Function Block	Input	Constant value	Description
FB_TWOHAND_1	tDiscTime12	T#200ms	Discrepancy time between normally closed contact and normally open contact of S1
FB_TWOHAND_1	tDiscTime34	T#200ms	Discrepancy time between normally closed contact and normally open contact of S2
FB_TWOHAND_1	tDiscTime	T#500ms	Discrepancy time between S1 and S2
FB_EDM2	tSwitchOnMonitoring	T#500ms	Turn-on delay
FB_EDM2	tSwitchOffMonitoring	T#500ms	Switch-off delay

Page deliberately empty

Page deliberately empty

Summary

6	Online operation	105
6.1	Project download	105
6.2	Project upload	106
6.3	Delete Project.....	107
6.4	Status Display	107

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

6 Online operation

6.1 Project download



To execute the command, you have to be logged in with the standard application. The standard application must be in "Run" state.

Before the command is executed, a user must be selected from the user management. If no user is selected, then the "Administrator" is used. The download procedure is actuated from the "Safety parameters" tab of the editor for the node of the NUMsafe controller. This function is only available in online mode.

After executing the command "Project Download" the user must enter a password and the serial number of the CTMP in order for the data record to be accepted by the device.

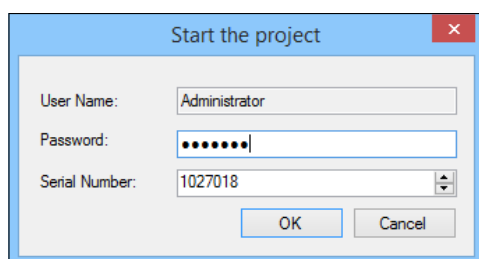


Fig. 93 Login dialog

A re-translation takes place automatically after the download of the application in order to check the generated code. To do this the translated code is decompiled again and compared at source code level. By means of this diverse method the function of the compiler is checked at each download. Both random and systematic errors are thereby determined.

The result of the comparison (same or different) is displayed to the user. For CTMP6900 the number of POU's and networks compared is indicated. For CTMP1960-2600 the number of POU's and function blocks compared is indicated. If equality is determined the download is valid and can be confirmed by the user via the offered dialog box.

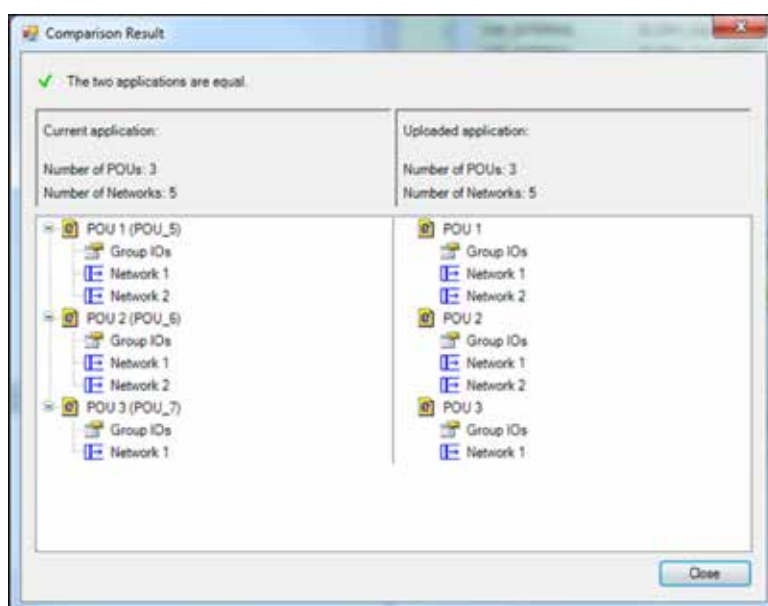


Fig. 94 Dialog: Comparison result = "same projects"

If equality is determined the download is valid and can be confirmed by the user. After entering the user name and password again the enabling data record is loaded to the controller. Afterwards the controller is ready to start. The application must be started from the user program.

This is done by a rising edge of the variables defined in the group I/Os. (see Fig. 38)

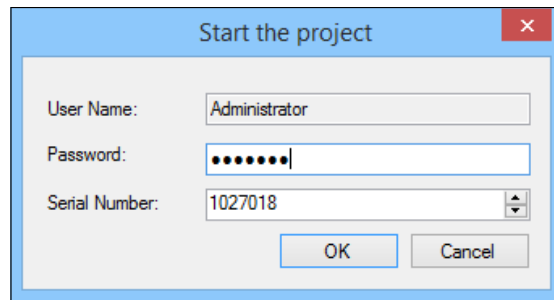


Fig. 95 Dialog: for loading the enabling data record.

If a difference is found the download is not valid and cannot be confirmed. The corresponding confirmation dialog is not offered to the user in this case. Instead, the user is shown where (in which block, network or group IO) the discrepancy occurred.

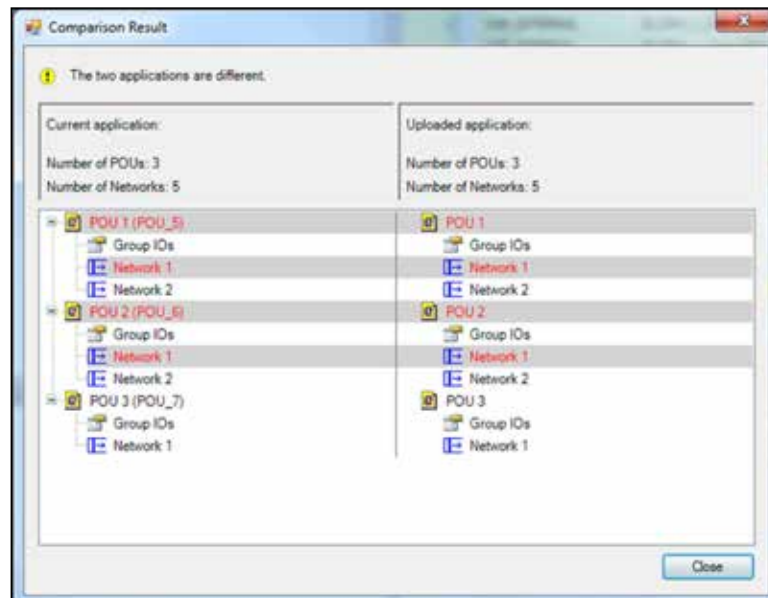


Fig. 96 Dialog: Comparison result = "different projects"

6.2 Project upload



To execute the command, you have to be logged in with the standard application. The standard application must be in "Run" state.

The upload procedure is actuated from the "Safety parameters" tab of the editor for the node of the NUMsafe controller.

The result of the comparison (same or different) is displayed to the user after the upload. For CTMP6900 the number of POUs and networks compared is indicated. For CTMP1960-2600 the number of POUs and function blocks compared is indicated. The comparison view is identical to that of the *Chapter 6.1 "Project download"* function.

Using this function it is simple to check whether the applications on the NUMsafe controller and in the project are the same or different.

6.3 Delete Project



To execute the command, you have to be logged in with the standard application. The standard application must be in "Run" state.

The deletion is actuated from the "Safety parameters" tab of the editor for the node of the NUMsafe controller.

The command is enabled when the application is logged into the NUMsafe controller and the status of the controller is "Run". After the command is executed, the login dialog appears.

After the login data are entered successfully, the project is deleted from the NUMsafe controller. Wrong login data will cause a warning message to appear.

6.4 Status Display

The status of the application and the Safety POU's is displayed in the device tree in squared brackets if the application is logged in (with standard app). As a precondition the application (Safety-App) must be executable and loaded to the controller.

Status of the safety application for CTMP1960-2600

- offline
- run
- stop
- safe
- start
- prepare
- restore
- project crc ok
- global shutdown
- global fault

Status of the POU's for CTMP1960-2600

- run
- stop
- error
- reset
- start
- stoperror
- deactivate
- waitcomerror

The Status "unknown" is displayed if

- the library IODrvEL6900 is not up to date. It must be V3.5.6.0 or higher.
- the application for the NUMsafe controller in CODESYS does not match to the application on the module.
- the EtherCAT stack does not run on the standard controller (after download or reset of the application).

Page deliberately empty

Summary

7	Pinning	111
---	---------------	-----

1

2

3

4

5

6

7

8

9

10

11

12

13

A

Page deliberately empty

7 Pinning

Preparatory measure for the verification

The developer can take preparatory measures for the verification of the safety application.

An important aspect here is to define the version of the safety application intended for the verification and thus to ensure that only precisely this version of the safety application is used for the verification, validation and subsequent acceptance.

"CODESYS Safety for EtherCAT Safety Module" offers the **pinning** function especially for this.



Notice!

Before the programmed or modified safety application is verified, it must be re-pinned. Verification and acceptance may only be carried out with pinned applications. No object of the application may have the status "In Work". Checking for "In Work" takes place during the individual verification steps.

What is pinning?

Pinning means that a reference point to the current version of a safety application is set that identifies the specific version of the safety application and the associated objects.

By means of the pin it is possible to identify a certain version of the application in the project and of an object in the editor on the NUMsafe controller. In addition the verifier, on the basis of the pin, can recognise at any time changes in the application structure, in the contents of its objects and in the library function blocks referred to.



Notice!

A specific version is made identifiable by setting a pin; however, no copy of the specific version is generated when doing this!

Comparison view

The pin functions can be found in the editor of the application object. To this end the safety application object is selected in the project tree and opened using the “*Edit Object*” context menu command. The “*Objects*” dialog shows the comparison view, which displays the version and the checksum of the objects of the current project and of the pinned project.

A precise description of the information and the object list of the comparison view can be found in *Chapter 4.5.4.1 “Safety application object”*.

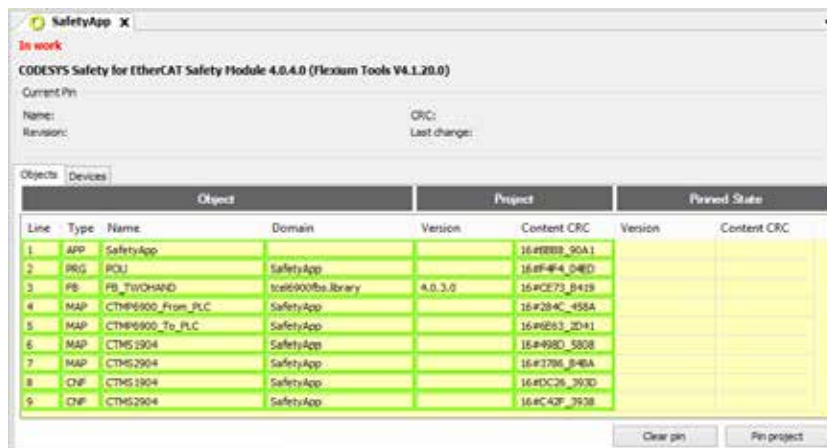


Fig. 97 Comparison view of a safety application that has not yet been pinned

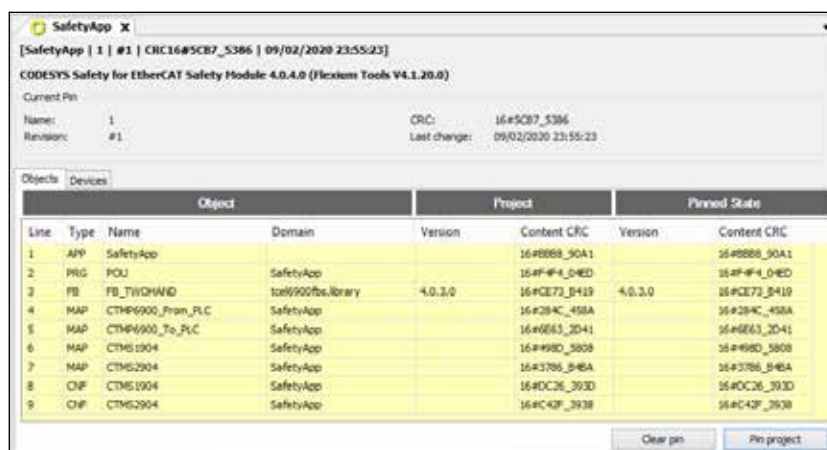


Fig. 98 Comparison view of a pinned safety application

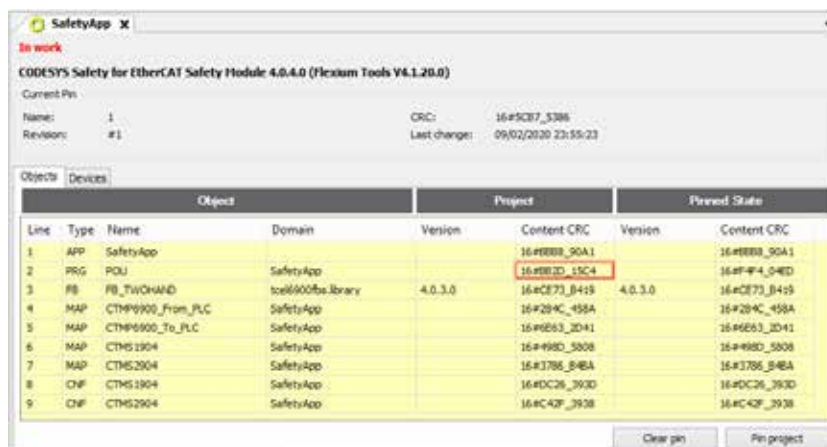


Fig. 99 Comparison view of a pinned safety application with changed POU of the current application

Pinning a safety application

Using the “*Pin project*” command, a pin is set on the current execution version of the listed objects, as a result of which not the contents, but rather the checksum and the version are noted. A pin name can be entered. The revision number of the pin is increased automatically by 1 with each “new” pinning. Using the “*Clear pin*” command, the current pin is deleted and all objects are once again “*In Work*”.

The pinned version of the safety application encompasses:

- Scope of the safety application:
 - which safety objects belong to the application
 - which library function blocks the application requires.
- Execution-relevant version of the objects and library function blocks in the scope of the application:
 - Code of each object of the application
 - Configuration and device parameter of each logical I/O object of the application
 - Interface of the external implementation of each library function block used
 - Version designations of the objects.

The execution-relevant version does not include the object comments. These are not pinned and can thus be updated at the end and during the verification!

The verifier identifies a pinned version by a pin identifier, which is displayed at different points in the programming system. The pin identifier contains:

- Name.
- Revision counter, which is increased automatically by 1 when pinning.
- CRC: a CRC32 of the pinned execution version.

In addition the time of pinning is recorded. However, this is not part of the pin identifier.

Display of the pin information and its deviations

The application pin information for a safety application can be shown to the developer in the comparison view.

The safety application pin information consists of:

- *Name* Name of the pin
- *Revision*
- *CRC* The checksum is created for the entire pinned application.
- *Last change* Time of the pin generation.

In addition, the comparison view of the safety application shows how the current project version differs from the current pinned version of the application. The following differences are shown:

- New objects
- Deleted objects
- Regarding objects with modified code or device parameters
- New function blocks drawn from libraries
- Library function blocks no longer drawn
- Library function blocks differing with regard to interface or implementation version.

Differences are clearly marked in colour so that the verifier can easily recognise them:

- Green: new objects or function library blocks in the project
- Red: change/difference in the contents of the object or device parameter set or library function block
- Blue: objects or library function blocks deleted from, or no longer used in the project.

Pinning in the project and in the object view

If the safety application is pinned, then the comparison view contains the pin information and in the project tree the node point  SafetyApp and its subobjects are marked with the  symbol.

The “SafetyApp” node point is considered to be pinned  if the object and all its subobjects correspond to the object version noted in the pin.

If the application has not yet been pinned or if the pin has been deleted, then only the “In Work” status appears in the top line and in the project tree the node point  SafetyApp and its subobjects are not marked. If a subobject of the safety application is “In Work”, then the safety application is also “In Work”.

The information about the pin or “In Work” is shown in the object view and in the printout of the project.

Verification procedure



The setting of a pin allows execution-relevant changes to be recognised during and after the verification. By setting a pin the developer releases the current version of a safety application in the project for verification or for the continuation of the verification following a change. Pinning makes this version identifiable with regard to the execution-relevant parts for verification tasks and for the acceptance.



Notice!

- The developer must always pin a finished or modified application before releasing it for (continuation of the) verification. No object in the application may be "In Work"!
- The developer must check during reviews, whitebox test development and test execution that the present application has been pinned, that the displayed pin identifier represents the current application to be verified and that no deviation of the project version from the pinned version is reported.
- Before making changes to an accepted application or one that is in verification, he must check that, in the initial state of his change, the present application is pinned, that the correct pin identifier is displayed and that no deviations of the project version from the pinned version are reported.

Page deliberately empty

Page deliberately empty

Summary

8	Verification and acceptance	119
8.1	Verification and acceptance.....	119
8.2	Archiving	119
8.3	Acceptance documentation printout	121

1

2

3

4

5

6

7

8

9

10

11

12

13

Page deliberately empty

8 Verification and acceptance

8.1 Verification and acceptance

**Notice!**

The complete test of the application is carried out by the user.

"CODESYS Safety for EtherCAT Safety Module" supports the user in the verification and acceptance with the following functions:

- Checking of syntax errors by the compiler
- Storing the project in a file on the storage medium
- Archiving the project and all of its referenced files
- Documenting the project by printing
- Pinning of the application for the identification of a version of the project.

8.2 Archiving

**Notice!**

Throughout the entire process, from the development to the acceptance of a safety application, care must be taken on the user side that the correct object versions, the correct libraries and the correct device description files are used (see *Chapter 7 "Pinning"*).

With the aid of archiving, all objects belonging to a safety application are summarised into a single file. This file can be saved as a data backup and unpacked again if needed.

**Notice!**

The archiving of the project for the acceptance of the safety application must be done via the project archiving function. To do this it is necessary to activate the *"Save/send archive"* command in the *"Project archive"* submenu of the *"File"* category. All information relevant to the acceptance must be selected in the Project Archive subdialog.

- Library profile
- Download information files
- Options
- Referenced libraries
- Referenced devices.

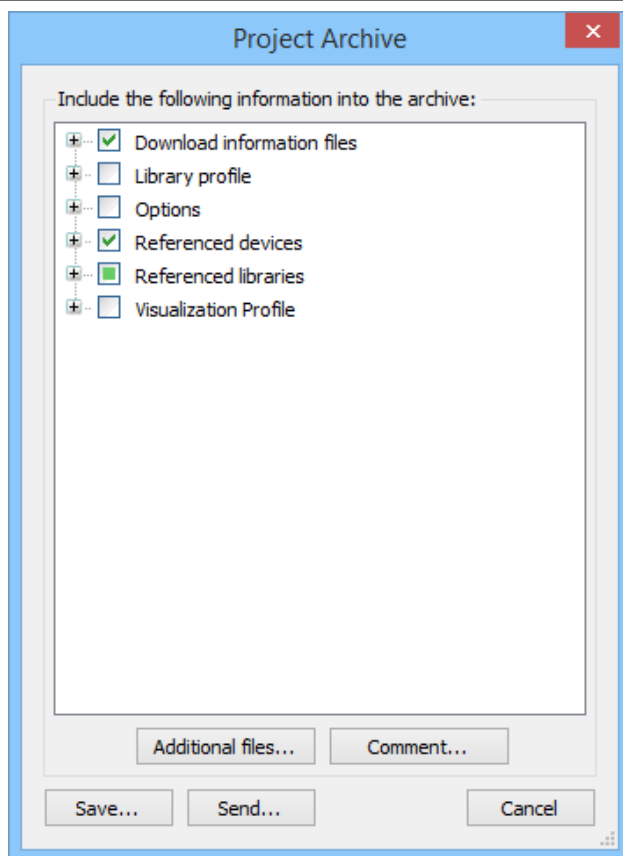


Fig. 100 Selection dialog: Creation of a project archive.

The following items are also recorded in the project archive through the correct selection in the Selection dialog:

- All safety applications with channel mappings and exchange lists.
- All field bus configurations and device parameterizations.
- All libraries used.

The project must be saved before archiving. Automatic saving takes place if necessary.



Notice!

When saving the project, all SDD files are saved with it. This means that, when opening a safety project, the device descriptions currently in the device repository are not used, but rather the SDD files of the project.

8.3 Acceptance documentation printout

The “*Document...*” command in the “*Project*” category is used in “CODESYS Safety for EtherCAT Safety Module” to print out the acceptance documentation. With the object views (object contents), the pin identifier or, if the object is not pinned, the information “*In Work*” is also printed out for each object. The printout optionally also contains a cover page with the following information: file, date and profile as well as a table of contents.



Notice!

For the printout of the acceptance documentation the safety controller and all its sub-node points and objects are marked in the “*Document Project*” dialog.



For detailed information on the documentation of a project, see the CODESYS Standard online help.

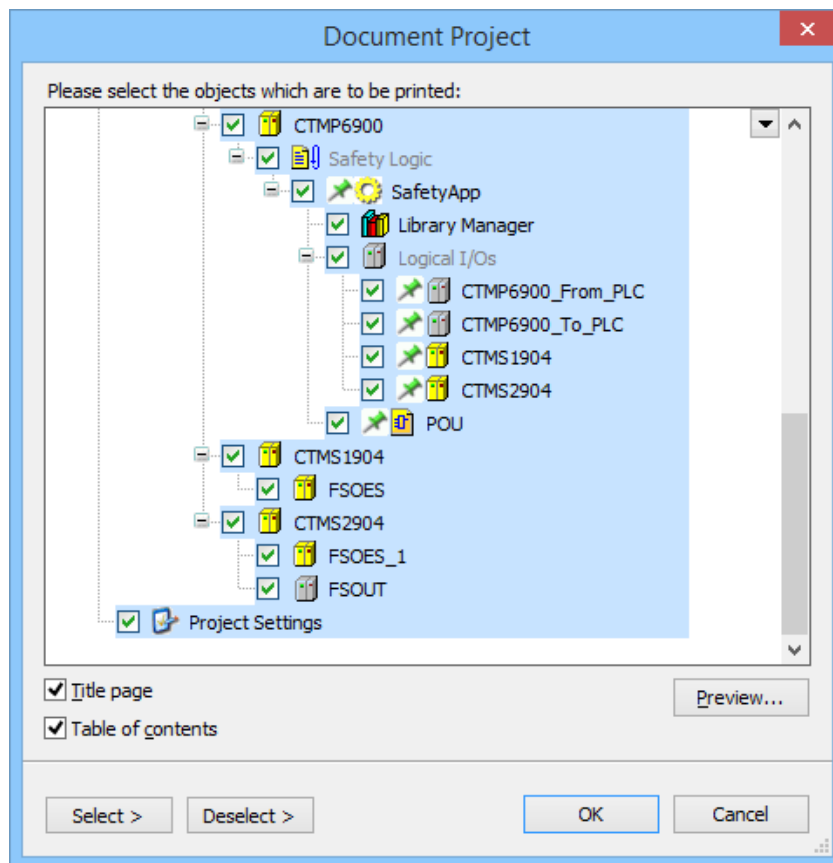


Fig. 101 Dialog: Document Project.

Page deliberately empty

Summary

9	Software updates	125
9.1	Overview of versioning.....	125
9.2	Updating the device version.....	125
9.3	Changing a library version	126
9.4	Updating the CODESYS version	127
9.5	Change request / Bug fix process.....	127

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

9 Software updates

9.1 Overview of versioning

This section describes the possible effects of installing a new device or firmware version or a new CODESYS version on an already verified or accepted project and the consequences of installing a package that is not approved for "CODESYS Safety for EtherCAT Safety Module".

9.2 Updating the device version



Notice!

Changing the contents of a device description can lead to the safety application no longer being pinned, but rather "In Work" again. The safety objects affected by the changed device description and listed with changed checksum in the comparison view of the safety application object must be subjected to an influence analysis and subsequently re-pinned and verified and, if necessary, accepted again.

If a new version of a device description is available, the service employee should inform himself precisely before installing the new device description about which of the possible variants of a new version is concerned and what effects this will have on the project and thus also on the verification and the acceptance:

- The new device is compatible with the previous device:
after installing the new device description in the device repository and updating the device (via the context menu in the device tree), no errors occur when translating the safety application. No further activities need to be carried out.
- If a safety device is being updated:
The update can contain changed device parameters or a changed I/O structure, resulting in changes to safety objects of the safety application.
The safety objects affected by the changed device description and listed with changed checksum in the comparison view of the safety application object must be subjected to an influence analysis and subsequently re-pinned and verified.



New device description files must be installed first in the device repository. In order to update the device, the physical device must be selected in the project tree and the "Update device..." command in the context menu activated.



The detailed information to the installed device descriptions is available in the device repository. To do this the respective device must be selected and the "Details..." button actuated.

9.3 Changing a library version

It is possible that newer library versions may be drawn when updating the device description.

This takes place automatically in such a way that the set libraries always match the device in use. If library function blocks have changed, these have a different checksum and the user must pin, verify and accept his application again.



Notice!

Changing the execution version also changes the pin of the safety application. The application must be re- pinned, verified and accepted again.



Notice!

An update of the device description of the safety application can lead to the loss of the acceptance of the safety application. Contact your device manufacturer with regard to whether an update is possible without problem.



Notice!

The use of another library version may change the pin of a used library function block under certain circumstances. Refer to the comparison view of the safety application to see whether that is the case.
In this case the application must be pinned, verified and accepted again.



Notice!

The correct libraries for a safety controller are automatically drawn in the correct versions with the device description of this NUMsafe controller. The library manager serves only to provide an overview of the available libraries and their function blocks. It is recommended not to remove or add libraries manually.

9.4 Updating the CODESYS version

Two versions are of importance for "CODESYS Safety for EtherCAT Safety Module": the version of CODESYS and the version of the extension "CODESYS Safety for EtherCAT Safety Module" (the information can be found in the *"Help"* menu, *"Information"* command and in the *"Tools"* menu, *"Package Manager"* command).

CODESYS and "CODESYS Safety for EtherCAT Safety Module" are further developed in such a way that a software update is possible without the loss of the acceptance of a safety application.



Notice!

The software packages CODESYS and "CODESYS Safety for EtherCAT Safety Module" should always be updated together.

9.5 Change request / Bug fix process

Change requests or error (bug) reports concerning the "CODESYS Safety for EtherCAT Safety Module" package pass through a sequence defined in the QM process at 3S-Smart Software Solutions:

Change request

1. Customer message requesting a change is received by telephone or email.
2. Entry of the change request in the error data base (Jira) by a support employee of 3S-Smart Software Solutions.

Error

1. Customer message reporting an error is received by telephone or e-mail.
2. Check whether error is comprehensible.
3. Check by the project manager whether this change request can be realised with regard to risk and expense.
4. Entry of the error in the error data base "ESM" (Jira) by a support employee of 3S-Smart Software Solutions.
5. Check by the project manager whether this error is safety-relevant.
6. In the case of safety-relevant errors the registered customers are informed by e-mail.

The error database is accessible to all registered customers over the Internet. They can query the status of all entries there. Registered customers are informed by the QA about product releases.

Registered customers who have subscribed to the RSS feed "CODESYS Safety" are notified about product releases.

Page deliberately empty

Summary

10	Function blocks.....	131
10.1	Introduction	131
10.2	Library " CTMP6900 NUMsafe controller " and "	
	CTMP1960-2600 NUMsafe compact controller"	132
10.2.1	FB_AND	132
10.2.2	FB_CS	134
10.2.3	FB_DECOUPLER	136
10.2.4	FB_EDM	138
10.2.5	FB_ESTOP	140
10.2.6	FB_MON	144
10.2.7	FB_MUTING	149
10.2.7.1	Example with 4 individual muting sensors	153
10.2.7.2	Example with 2 two-channel muting sensors	153
10.2.8	FB_MODE.....	156
10.2.9	FB_OR	159
10.2.10	FB_RS	161
10.2.11	FB_SR	163
10.2.12	FB_TOF	165
10.2.13	FB_TON	167
10.2.14	FB_TWOHAND	169
10.3	Library " CTMP1960-2600 NUMsafe compact controller "	172
10.3.1	FB_ADD	172
10.3.2	FB_COMPARE	174
10.3.3	FB_COUNTER.....	178
10.3.4	FB_DIV	181
10.3.5	FB_ENVELOPE	183
10.3.6	FB_LIMIT	186
10.3.7	FB_LOADSENSING	188
10.3.8	FB_MUL	191
10.3.9	FB_SCALING.....	193
10.3.10	FB_SLI	196
10.3.11	FB_SPEED	198
10.3.12	FB_SUB	200
10.3.13	FB_VIOLATIONCNT	202

Page deliberately empty

10 Function blocks

10.1 Introduction

The "CODESYS Safety for EtherCAT Safety Module" package provides the programmer with function blocks with defined functions.

The functions can be partly configured via additional parameter inputs. The inputs or outputs of a function block can be inputs or outputs of the local process image; however, outputs of function blocks can also be connected with inputs of other function blocks.

Input and output types

Type	Data Type	Description, parameter values
SAFE-In	SAFEBOOL	SAFE input, e.g. at an CTMS1904
Standard-In	BOOL	Logical input variable from standard PLC
FB-Out	SAFEBOOL	SAFE FB output
SAFE-Out	SAFEBOOL	SAFE output, e.g. at an CTMS2904
Standard-Out	BOOL	Logical output variable to standard PLC
FB-In	SAFEBOOL	SAFE FB input
Local-Out	SAFEBOOL	SAFE output at the CTMP1960-2600 (not available for CTMP6900)

10.2 Library " CTMP6900 NUMsafe controller " and " CTMP1960-2600 NUMsafe compact controller"

10.2.1 FB_AND

FB description

Using the FB_AND function block, several input signals can be ANDed to make an output signal. In addition, a setting can be made for each input as to whether it is to be negated. This can be switched over using the "Negation" command from the context menu.

The bAndIn1 input differs from the bAndIn2 - bAndIn8 inputs to the extent that it can also be connected to a standard input. This makes it possible to switch off a safe output via standard signals.

Outputs cannot be switched on via standard signals, but only enabled, since at least two inputs of FB_AND must always be linked (and the second input is a safe input that prevents switching on).

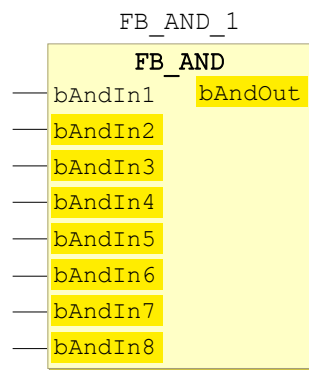


Fig. 102 FB_AND function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bAndIn1	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	1st input channel
bAndIn2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel
bAndIn3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel
bAndIn4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel
bAndIn5	SAFE-In FB-Out	SAFEBOOL	FALSE	5th input channel
bAndIn6	SAFE-In FB-Out	SAFEBOOL	FALSE	6th input channel
bAndIn7	SAFE-In FB-Out	SAFEBOOL	FALSE	7th input channel
bAndIn8	SAFE-In FB-Out	SAFEBOOL	FALSE	8th input channel

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bAndOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	Output channel

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0-15	always 0

STATE INFORMATION

Value	Description, parameter values
0	undefined
1	RUN The RUN state is assumed once all active inputs <code>bAndIn1-bAndIn8</code> are set to 1 (ACTIVE_ANDIN=TRUE). The output assume the following value: <code>bAndOut=1</code>
2	STOP The FB_AND module assumes the STOP state if the input <code>FbRun</code> is FALSE. The output assume the following value: <code>bAndOut= 0</code>
3	SAFE The SAFE state is assumed if one or more of the active inputs <code>bAndIn1-bAndIn8</code> is not 1 (ACTIVE_ANDIN=FALSE). The output assume the following value: <code>bAndOut= 0</code>

10.2.2 FB_CS

FB description

A NUMSafe connection can be deactivated via the FB_CONNECTION SHUTDOWN function block. If the input of the block becomes active the connection is terminated, a shutdown command is sent to the FSoE partner and a feedback signal is output at the output. Furthermore, the connection is terminated and the output is set if a shutdown command is received from the communication partner.

The output is only reset again when the connection to the FSoE partner is once again in the DATA state.

The FSoE Master tries to establish the connection again or, as the case may be, the FSoE Slave replies to the connection again as soon as the input of the block is no longer active.

This block is required for modular safety concepts in which machine parts are sometimes installed and sometimes not. Furthermore, this function is also required for machines in which tool changes take place.

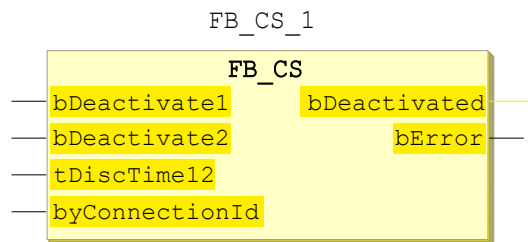


Fig. 103 FB_CS function block

On the opposite side the function block is called without connected inputs. The Deactivated output is set as soon as the connection is terminated due to a shutdown command from the communication partner.

bDeactivate1 is used as a single-channel input if bDeactivate2 and tDiscTime12 are deleted.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bDeactivate1	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
bDeactivate2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel; behaves same as Deactivate1. If the discrepancy time tDiscTime12 is not equal to 0, the 1st and 2nd input channels are considered to be an input group and the discrepancy time between the two channels is monitored if either of the two input channels requests the safe state.
tDiscTime12	-	SAFETIME	TIME#0ms	Discrepancy time
byConnectionId	-	SAFEBYTE	0	ID of the connection (=logical I/O) which is to be deactivated. In CODESYS the connections are numbered in ascending order (starting with the first physical module) This parameter can be set in the "Safe Configuration" dialog of the fieldbus device. (see <i>Safe configuration</i> (4.5.4.2.3.2))

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bDeactivated	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	1st output channel; the safe state corresponds to logic 0. The output is set if the connection is terminated.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: the discrepancy time monitor of a 2-channel input group has detected an error; the error must be acknowledged via the ERR Ackn input of the associated POU (NUMSafe group) FALSE: no error was detected.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	Discrepancy error input group 1

STATE INFORMATION (8-BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN If the CONNECTION module has received a shutdown command on the assigned connection, it switches the connection to SHUTDOWN state and reports this state to the FB_CS module, which then assumes the RUN state. If all activated bDeactivateX inputs are TRUE, the module switches to the RUN state and instructs the CONNECTION module to send a shutdown command via the assigned connection and to set it to SHUTDOWN state. The outputs assume the following values: bError=0 bDeactivated=1
2	STOP The FB_CS module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 bDeactivated=0
3	SAFE If not all activated bDeactivateX inputs are TRUE and the assigned connection is not in SHUTDOWN state, the FB_CS module assumes the SAFE state. The outputs assume the following values: bError=0 bDeactivated=0
4	ERROR If the FB_CS module detects an error, the FB_CS module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bDeactivated=0
5	RESET The FB_CS module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bDeactivated=0

10.2.3 FB_DECOUPLER

FB description

The FB_DECOUPLER function block serves to decouple signals from a NUMSafe connection.

The function block has 8 inputs and 8 outputs, wherein the inputs are looped through to the outputs one-to-one. The corresponding output must be linked as soon as one of the inputs of the block is used. This also applies in the reverse direction.

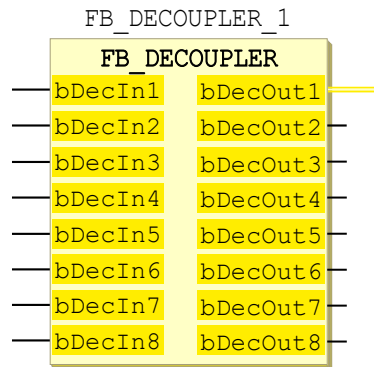


Fig. 104 FB_DECOUPLER function block

Since a NUMSafe connection is always assigned to a POU (NUMSafe group), there is a possibility via the FB_DECOUPLER to distribute the signals of a NUMSafe connection to several POUs (NUMSafe groups) and thus to decouple them.

The module can be used within an existing POU (NUMSafe group) for the distribution of the signals.

The block must be used in a separate POU (NUMSafe group) if the signals are also to be decoupled, because all used outputs of a POU (NUMSafe group) are switched off in the event of a communication error of the connection. The input signals of a NUMSafe group connection can now be linked with the inputs of the FB_DECOUPLER; the outputs can then be distributed to various POUs (NUMSafe groups).

It works exactly the same in the other direction: the outputs of a NUMSafe group connection are linked with the outputs of the FB_DECOUPLER, the inputs of the FB_DECOUPLER can in turn come from different POUs (NUMSafe groups).

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bDecIn1	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel
bDecIn2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel
bDecIn3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel
bDecIn4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel
bDecIn5	SAFE-In FB-Out	SAFEBOOL	FALSE	5th input channel
bDecIn6	SAFE-In FB-Out	SAFEBOOL	FALSE	6th input channel
bDecIn7	SAFE-In FB-Out	SAFEBOOL	FALSE	7th input channel
bDecIn8	SAFE-In FB-Out	SAFEBOOL	FALSE	8th input channel

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bDecOut1	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	1st output channel
bDecOut2	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	2nd output channel
bDecOut3	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	3rd output channel
bDecOut4	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	4th output channel
bDecOut5	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	5th output channel
bDecOut6	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	6th output channel
bDecOut7	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	7th output channel
bDecOut8	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	8th output channel

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	always 0

STATE INFORMATION (8-BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The FB_DECOUPLER module assumes the RUN state if the input FbRun is TRUE. The outputs assume the following values: bDecOutX= bDecInX
2	STOP The FB_DECOUPLER module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bDecOutX= 0

10.2.4 FB_EDM

FB description

The FB_EDM function block (External Device Monitor) monitors the signals `bMon1` and `bMon2` over time. A switch-on and a switch-off monitor can be configured. In the default condition both monitors are inactive.

In the case of the switch-on monitor a check is made as to whether the `bMon2` signal is set to 0 within the specified time (maximally 10000 ms) on a change of the `bMon1` signal from 0 to 1.

In the case of the switch-off monitor a check is made as to whether the `bMon2` signal is set to 1 within the specified time (maximally 10000 ms) on a change of the `bMon1` signal from 1 to 0.

The `bError` output of the block is set if the preset monitoring time is exceeded. This can then be reset via the ERR Ackn signal of the NUMSafe group.

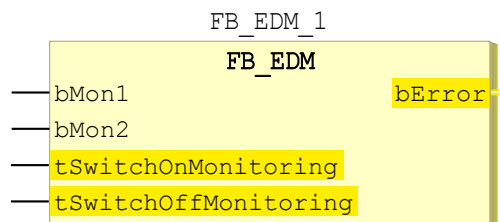


Fig. 105 FB_EDM function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bMon1	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	1st input channel. The "Negation" command can be used to set whether the input is a normally closed or a normally open contact.
bMon2	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	2nd input, which must adopt the opposite value to input 1 within the preset time.
tSwitchOnMonitoring		SAFETIME	TIME#0ms	Switch-on delay
tSwitchOffMonitoring		SAFETIME	TIME#0ms	Switch-off delay

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: SwitchOn or SwitchOff times were exceeded. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	SwitchOff timer elapsed
1	SwitchOn timer elapsed

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
2	STOP The FB_EDM module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0
4	ERROR If the FB_EDM module detects an error, the FB_EDM module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1
5	RESET The FB_EDM module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0
14	MON_OFF If the input bMon1 is FALSE, the FB_EDM module assumes the MON_OFF state in order to perform tSwitchOnMonitoring. The outputs assume the following values: bError=0
15	MON_ON If the input bMon1 is TRUE, the FB_EDM module assumes the MON_ON state in order to perform tSwitchOnMonitoring. The outputs assume the following values: bError=0

10.2.5 FB_ESTOP

FB description

An emergency stop circuit with up to eight emergency stop inputs `bEStopIn1` - `bEStopIn8` can be implemented with the `FB_ESTOP` function block.

Each of the eight inputs can be negated using the "Negation" command from the context menu. As soon as an input requests the safe state, the first output `bEStopOut` immediately enters the safe state ("0") and the second output `bEStopDelOut` enters the safe state after a configurable time delay.

Thanks to the possibility to link the FB output to several outputs, several immediately switching-off outputs `bEStopOut` or delayed switching-off outputs `bEStopDelOut` can be implemented with just one `FB_ESTOP`. In order to quit the safe state of the outputs, a 0->1->0 signal sequence must be recognised at the `bRestart` input.

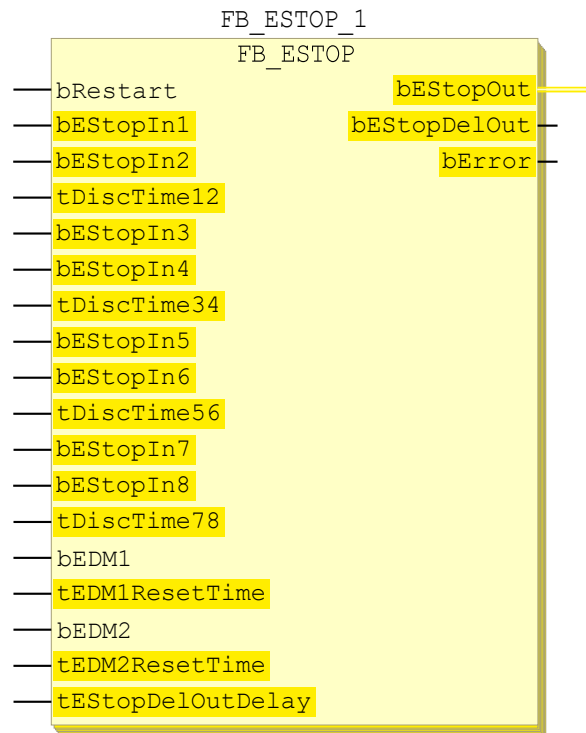


Fig. 106 `FB_ESTOP` function block

Beyond that a feedback loop can be activated for each of the two outputs, with which the `bEStopOut` output is fed back to the `bEDM1` input and the `bEStopDelOut` output to the `bEDM2` input via external wiring.

The `bEDMx` inputs are checked as soon as the safe state is to be quit by means of a 0->1->0 signal sequence at the `bRestart` input. If the `bEDMx` inputs do not have the signal status "1", the `FB_ESTOP` function block enters the error state and sets the `bError` output to 1.

The error state can only be quit again by a 0->1->0 signal sequence at the `ERR Ackn` input of the associated POU (NUMSafe group).

Furthermore 2 inputs (`bEStopIn1` and `bEStopIn2`, `bEStopIn3` and `bEStopIn4`, `bEStopIn5` and `bEStopIn6` as well as `bEStopIn7` and `bEStopIn8`) can be combined in each case to form input pairs, in which the signal states of the two inputs may deviate from each other only within a configurable discrepancy time (`tDiscTimexy`).

If this discrepancy time is exceeded by a pair of inputs, the `FB_ESTOP` block likewise enters the `bError` state. In the `bError` state of the FB the outputs adopt the safe state "0".

`bEStopIn1` is used as a single-channel input if `bEStopIn2` and `tDiscTime12` are deleted. This operating principle applies to all input pairs.

FB_ESTOP CTMP1960-2600 extention

The FB_ESTOP can also be used to monitor the change of the state of the feedback signal `bEDMx` when the outputs are switched on.

The inputs `bEDM1` and `bEDM2` have been assigned a further parameter `tEDMxResetTime`. If its value is not equal to 0, the timer is started when the output `bEStopOut` is activated. If the `bEDMx` input does not switch to FALSE within this time, a function block error is set and the outputs are switched off.

This function can be switched off by entering 0 as `tEDMxResetTime`.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bRestart</code>	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	At the start (when the associated POU (NUMSafe group) is started) or restart (if an input has requested the safe state), the signal sequence 0->1- >0 must be recognised at the Restart input before the safe state of the outputs can be cancelled.
<code>bEStopIn1</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel: the "Negation" command can be used to specify whether the input behaves as a normally closed contact (break contact-safe state is requested with logic 0) or as a normally open contact (make contact-safe condition is requested with logic 1).
<code>bEStopIn2</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel: behaviour same as <code>bEStopIn1</code> . If the discrepancy time is not equal to 0, the 1st and 2nd input channels are considered to be the 1st input pair and the discrepancy time between the two channels is monitored.
<code>tDiscTime12</code>		SAFETIME	TIME#0ms	Discrepancy time for input group 1 (<code>ln1+ln2</code>)
<code>bEStopIn3</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel or 1st input channel of the 2nd input pair, otherwise corresponds to <code>bEStopIn1</code>
<code>bEStopIn4</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel or 2nd input channel of the 2nd input pair, otherwise corresponds to <code>bEStopIn2</code>
<code>tDiscTime34</code>		SAFETIME	TIME#0ms	Discrepancy time for input group 2 (<code>ln3+ln4</code>)
<code>bEStopIn5</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	5th input channel or 1st input channel of the 3rd input pair, otherwise corresponds to <code>bEStopIn1</code>
<code>bEStopIn6</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	6th input channel or 2nd input channel of the 3rd input pair, otherwise corresponds to <code>bEStopIn2</code>
<code>tDiscTime56</code>		SAFETIME	TIME#0ms	Discrepancy time for input group 3 (<code>ln5+ln6</code>)
<code>bEStopIn7</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	7th input channel or 1st input channel of the 4th input pair, otherwise corresponds to <code>bEStopIn1</code>
<code>bEStopIn8</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	8th input channel or 2nd input channel of the 4th input pair, otherwise corresponds to <code>bEStopIn2</code>
<code>tDiscTime78</code>		SAFETIME	TIME#0ms	Discrepancy time for input group 4 (<code>ln7+ln8</code>)
<code>bEDM1</code>	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	<code>bEDM1</code> is the feedback loop for the undelayed output channel (<code>bEStopOut</code>). If this input is parameterised as active, the safe state of the outputs is only quit when restarting if <code>bEDM1</code> supplies the signal "1".

tEDM1ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bEStopOut is activated. If the bEDM1 input does not switch to FALSE within this time, a function block error is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
bEDM2	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	bEDM2 is the feedback loop for the switch-off- delayed output channel (bEStopDelOut). If this input is parameterised as active, the safe state of the outputs is only quit when restarting if bEDM2 supplies the signal "1".
tEDM2ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bEStopDelOut is activated. If the bEDM2 input does not switch to FALSE within this time, a function block error is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
tEStopDelOutDelay		SAFETIME	TIME#0ms	Delay time for the output of the safe state at bEStopDelOut

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bEStopOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	1st output channel; the safe state corresponds to logic 0.
bEStopDelOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	2nd output channel; the safe state corresponds to logic 0. The safe state is output with a delay in accordance with the parameterised delay time tEStopDelOutDelay
bError	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	TRUE: the discrepancy time monitoring of an input pair or one of the feedback loops has detected an error. The error must be reset via the ERR Ackn input of the associated POU (NUMSafe group). FALSE: no error was detected.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	Discrepancy error input group 1
1	Discrepancy error input group 2
2	Discrepancy error input group 3
3	Discrepancy error input group 4
4	EDM monitoring error bEDM1
5	EDM monitoring error bEDM2

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The FB_ESTOP module assumes the RUN state if no error is present and no active bEStopInX input requests a safe state. The outputs assume the following values: bError=0 bEStopOut=1 bEStopDelOut=1
2	STOP The FB_ESTOP module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 bEStopOut=0 bEStopDelOut=0
3	SAFE The FB_ESTOP module assumes the SAFE state as long as at least one of the active bEStopInX inputs has requested the safe state. The outputs assume the following values: bError=0 bEStopOut=0 bEStopDelOut=0
4	ERROR If the FB_ESTOP module detects an error, the FB_ESTOP module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bEStopOut=0 bEStopDelOut=0
5	RESET The FB_ESTOP module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bEStopOut=0 bEStopDelOut=0
6	START The FB_ESTOP module assumes the START state if the bRestart input is TRUE, in order to wait for a rising and falling edge of the bRestart input before entering the RUN state and the outputs leave the safe state. The outputs assume the following values: bError=0 bEStopOut=0 bEStopDelOut=0
8	DELAYOUT The FB_ESTOP module assumes the DELAYOUT state if at least one of the active bEStopInX inputs has requested the safe state and the delay time for the bEStopDelOut output has not yet expired. The outputs assume the following values: bError=0 bEStopOut=0 bEStopDelOut=1

10.2.6 FB_MON

FB description

A safety door circuit with up to four inputs `bMonInx`, for example, can be implemented with the FB_MON function block. Each of the four inputs can be realised as a normally closed contact (break contact - 0 requests the safe state) or as a normally open contact (make contact - 1 requests the safe condition).

As soon as an input requests the safe state, the first output `bMonOut` immediately enters the safe state ("0") and the second output `bMonDelOut` enters the safe state after a configurable time delay. Thanks to the possibility to link the FB output to several outputs, several immediately switching-off outputs `bMonOut` or delayed switching-off outputs `bMonDelOut` can be implemented with just one FB_MON.

In addition there are two `bSecure` inputs with which the request of the safe state by the `bMonIn` inputs can be bypassed. The secure inputs can also be implemented both as normally closed contact (break contact) and as normally open contact (make contact).

The `bRestart` input of the FB can be activated (active or manual restart) by connecting it to a signal or variable. An open input `bRestart` will create a compiler error. If the restart shall be executed automatically (non-active or automatic restart) the input `bRestart` must be deleted..

In order to quit the safe state of the outputs, a 0->1->0 signal sequence must be recognised at the `bRestart` input in the case of active restart. In the case of non-active restart, the safe state is quit as soon as the `bMonInx` or the `bSecurex` inputs no longer request the safe state.

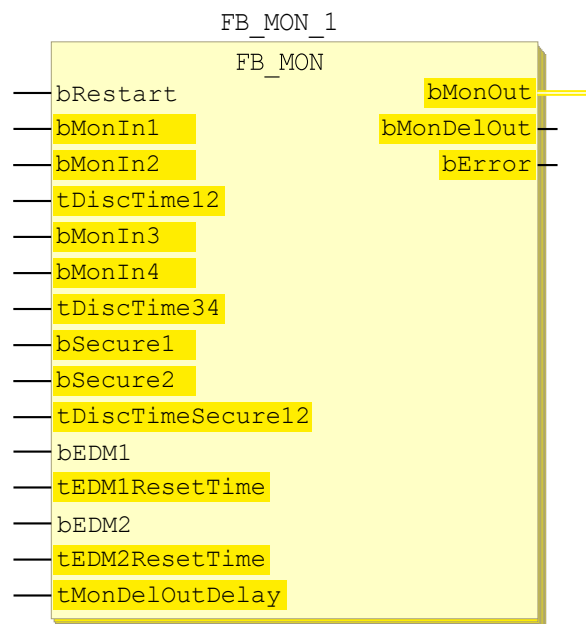


Fig. 107 FB_MON function block

Beyond that a feedback loop can be activated for each of the two outputs, with which the `bMonOut` output is fed back to the `bEDM1` input and the `bMonDelOut` output to the `bEDM2` input via external wiring.

The `bEDMx` inputs are checked as soon as the safe state is to be quit. If the `bEDMx` inputs do not have the signal status "1", the function block enters the error state and sets the `bError` output to 1 if `bRestart` is activated.

If `bRestart` is deactivated, FB_MON remains in the safe state if the `bEDMx` inputs do not have signal status "1" in the moment of restarting. A `bEDMx` error can be thus only recognised if the manual restart is active. The error state can only be quit again by a 0->1->0 signal sequence at the ERR Ackn input of the associated POU (NUMSafe group).

Furthermore 2 inputs (bMonIn1 and bMonIn2, bMonIn3 and bMonIn4 as well as bSecure1 and bSecure2) can be combined in each case into input pairs, in which the signal states of the two inputs may deviate from each other only within a configurable discrepancy time (tDiscTime_{xx}). If this discrepancy time is exceeded by a pair of inputs, the FB_MON block likewise enters the error state.

bMonIn1 is used as a single-channel input if bMonIn2 and tDiscTime12 are deleted.

This operating principle also applies to bMonIn4 and bSecure2.

FB_MON CTMP1960-2600 extention

The FB_MON can also be used to monitor the change of the state of the feedback signal bEDM_x when the outputs are switched on.

The inputs bEDM1 and bEDM2 have been assigned a further parameter tEDMxResetTime. If its value is not equal to 0, the timer is started when the output bESopOut is activated. If the bEDM_x input does not switch to FALSE within this time, a function block error is set and the outputs are switched off.

This function can be switched off by entering 0 as tEDMxResetTime.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bRestart	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	When starting the FB, or if an input has requested the safe state, the signal sequence 0->1->0 must be recognised at the bRestart input before the safe state of the outputs can be quit if manual restart is active. If manual restart is not active, this input is not used. Both starting and quitting the safe state take place automatically as soon as no input requests the safe state any longer.
bMonIn1	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel: the "Negation" command can be used to specify whether a normally closed contact (break contact - safe state is requested with logic 0) or as a normally open contact (make contact - safe condition is requested with logic 1) is linked with this input.
bMonIn2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel: behaviour same as bMonIn1. If the discrepancy time is activated or used, the 1st and 2nd input channels are considered to be the 1st input pair and the discrepancy time between the two channels is monitored.
tDiscTime12		SAFETIME	TIME#0ms	Discrepancy time between bMonIn1 and bMonIn2.
bMonIn3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel or 1st input channel of the 2nd input pair, otherwise corresponds to bMonIn1
bMonIn4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel or 2nd input channel of the 2nd input pair, otherwise corresponds to bMonIn1
tDiscTime34		SAFETIME	TIME#0ms	Discrepancy time between bMonIn3 and bMonIn4.

bSecure1	SAFE-In FB-Out	SAFEBOOL	FALSE	If bSecure1 or bSecure2 are parameterised as active, the evaluation of the bMonInX inputs can to be switched off. If bSecure1 or bSecure2 are parameterised as a normally closed contact (break contact), the bMonInX inputs are ignored, provided bSecure1 and/or bSecure2 are "1". If bSecure1 or bSecure2 are parameterised as a normally open contact (make contact), the bMonInX inputs are ignored, provided bSecure1 and/or bSecure2 are "0". If the discrepancy time is activated or used, bSecure1 and bSecure2 are considered to be an input pair and the discrepancy time between the two channels is monitored.
bSecure2	SAFE-In FB-Out	SAFEBOOL	FALSE	bSecure2 is the 2nd channel of the input pair and otherwise corresponds to bSecure1.
tDiscTimeSecure12		SAFETIME	TIME#0ms	Discrepancy time for bSecure1 and bSecure2.
bEDM1	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	bEDM1 is the feedback loop for the undelayed output channel bMonOut. If this input is parameterised as active, the safe state of the outputs is only quit if bEDM1 supplies the signal "1".
tEDM1ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bMonOut is activated. If the bEDM1 input does not switch to FALSE within this time, a function block error is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
bEDM2	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	bEDM2 is the feedback loop for the switch-off- delayed output channel bMonDelOut. If this input is parameterised as active, the safe state of the outputs is only quit if bEDM2 supplies the signal "1".
tEDM2ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bMonOut is activated. If the bEDM2 input does not switch to FALSE within this time, a function block error is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
tMonDelOutDelay		SAFETIME	TIME#0ms	Delay time for the output of the safe state at bMonDelOut.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bMonOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	1st output channel; the safe state corresponds to logic 0.
bMonDelOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	2nd output channel; the safe state corresponds to logic 0. The safe state is output with a delay in accordance with the parameterised Delay Time ($t_{MonDelOutDelay}$).
bError	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	TRUE: the discrepancy time monitoring of an input pair or one of the feedback loops has detected an error. The error must be reset via the ERR Ackn input of the associated POU (NUMSafe group). FALSE: no error was detected.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	Discrepancy error input group 1
1	Discrepancy error input group 2
2	Discrepancy error in Secure input group
4	EDM monitoring error bEDM1
5	EDM monitoring error bEDM2

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The FB_MON module assumes the RUN state if no error is present and neither the active bMonInx inputs nor the active bSecurex inputs request a safe state. The outputs assume the following values: bError =0 bMonOut =1 bMonDelOut =1
2	STOP The FB_MON module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =0
3	SAFE The FB_MON module assumes the SAFE state as long as at least one of the active bMonInx inputs and at least one of the active bSecurex inputs has requested the safe state. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =0

4	ERROR If the FB_MON module detects an error, the FB MON module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError =1 bMonOut =0 bMonDelOut =0
5	RESET The FB_MON module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =0
6	START The FB_MON module assumes the START state if the bRestart input is active and TRUE, in order to wait for a rising and falling edge of the bRestart input before entering the RUN state and the outputs leave the safe state. This state can only be assumed if manual restart is enabled in the configuration data. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =0
7	ERRORDelay The FB_MON module assumes the ERRORDelay state if a discrepancy error occurs in the RUN state (DiscError=TRUE) and the delay time for the bMonDelOut output has not yet expired. This state can only be assumed if the output delay time in the configuration data is not 0. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =1
8	DELAYOUT The FB_MON module assumes the DELAYOUT state if at least one of the active bMonInx inputs and at least one of the active bSecurex inputs has requested the safe state and the delay time for the bMonDelOut output has not yet expired. This state can only be assumed if the output delay time in the configuration data is not 0. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =1
9	FUNCTEST The FB_MON module assumes the FUNCTEST state if the manual function test is enabled in order to test all active bMonInx inputs once after startup. This state can only be assumed if the manual function test is enabled in the configuration data. The outputs assume the following values: bError =0 bMonOut =0 bMonDelOut =0

10.2.7 FB_MUTING

FB description

The FB_MUTING function block implements an intended suppression of the safety function e.g. for the transport of material into the safety area. The output of the function block remains set even though the connected sensors are interrupted.

Via the muting inputs a check is made that these are actuated in a defined order. The muting can be activated with the `bEnable` input. If the input is logic 0, an interruption of the safety device leads to the immediate switching off of the FB output. In other cases only if the muting sequence is violated.

The `bSequentialInputs` input can be used to define whether 2 inputs are checked in parallel in each case or whether each input is checked sequentially.

In order to prevent bouncing of the muting signals and thus a violation of the muting sequence, a filter time `tFilterTime` (max. 500 ms) can be set for the muting inputs. Furthermore it is possible via `tMaxMutingTime` to monitor the maximum duration of the muting process.

The muting process starts with a logic 1 signal of the first muting input and ends with the logic 0 signal of the last muting input. The value can be configured to maximally 10 minutes or set to 0, which corresponds to a deactivation of the monitoring. The `bMutingActive` output of the function block is set during this period.

The safety device (AOPD - Active Opto-electronics Protection Device), e.g. a light curtain, is connected to the `bOSSDInx` inputs. The feedback signals can be connected to the `bEDMx` inputs. The inputs are deactivated in the default setting.

The direct outputs and those delayed by up to 30 seconds are switched via the `bMuteOut` and `bMuteDelOut` outputs.

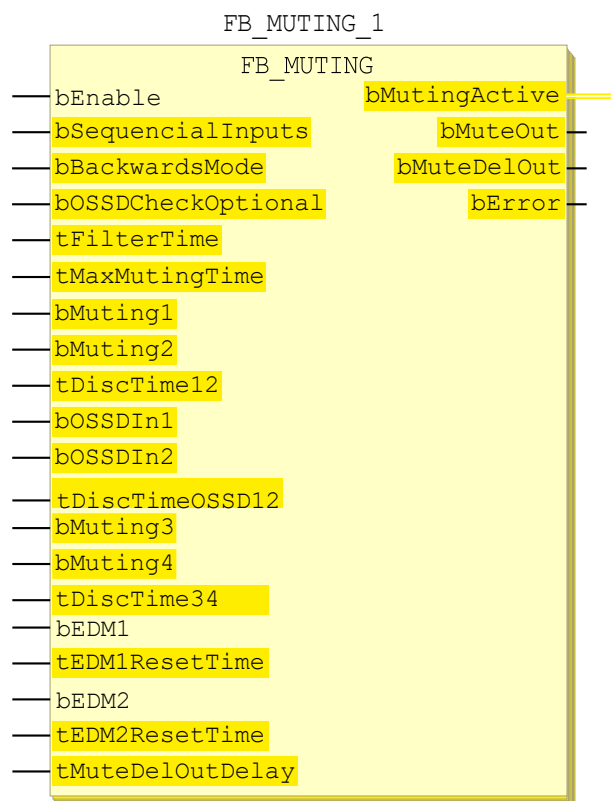


Fig. 108 FB_MUTING function block

bMuting1 is used as a single-channel input if bMuting2 and tDiscTime12 are deleted. This operating principle also applies to bMuting4 and bOSSDIn2.

FB_MUTING CTMP1960-2600 extention

In addition to the existing options, the following additional functions are now available:

Option bBackwardsMode:

If the option is active, a muting sequence is supported in forward and backward direction. If muting sensor bMuting4 is actuated first, backward direction is assumed, forward direction for bMuting1. A muting sequence must be completed before a new sequence (perhaps in the other direction) can be started.

Option bOSSDCheckOptional:

This option is used to remove the light curtain from the sequence check. In this case only the muting sensors bMuting1 to bMuting4 are checked. The light curtain can be operated at any point within the sequence.

Wiring function change:

If the bSequentialInputs option is active, it is possible to leave the bMuting3 and bMuting4 inputs inactive/unconnected. Provided the inputs bMuting1, bMuting2, bOSSDIn1 and bOSSDIn2 are connected.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bEnable	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	The muting can be activated with the Enable input. If the input is logic 0, an interruption of the safety device leads to the immediate switching off of the FB output.
bSequentialInputs	SAFE-In FB-Out	SAFEBOOL	FALSE	This input can be used to define whether 2 inputs are checked in parallel in each case or whether each input is checked sequentially.
bBackwardsMode	SAFE-In FB-Out	SAFEBOOL	FALSE	Available only in CTMP1960-2600 In CTMP6900 if this input will be used a compiler error will arise.
bOSSDCheckOptional	SAFE-In FB-Out	SAFEBOOL	FALSE	Available only in CTMP1960-2600 In CTMP6900 if this input will be used a compiler error will arise.
tFilterTime		SAFETIME	TIME#0ms	Filter time for the avoidance of bouncing of the inputs.
tMaxMutingTime		SAFETIME	TIME#0ms	Maximum duration of the muting process.
bMuting1	SAFE-In FB-Out	SAFEBOOL	FALSE	Via the muting inputs a check is made that these are actuated in a defined order. 1st input channel; the parameterisation specifies whether the input must be negated or is directly used.
bMuting2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel, behaviour same as bMuting1. If the discrepancy time tDiscT12 is not equal to 0, the 1st and 2nd input channels are considered to be the 1st input group and the discrepancy time between the two channels is monitored if either of the two input channels requests the safe state.
tDiscTime12		SAFETIME	TIME#0ms	Discrepancy monitoring time for muting input 1 and 2.

bOSSDIn1	SAFE-In FB-Out	SAFEBOOL	FALSE	The safety device (AOPD - Active Opto-electronics Protection Device), e.g. a light curtain, is connected to the bOSSDIn inputs. bOSSDIn: 1st input channel; the parameterisation specifies whether the input must be negated or is directly used.
bOSSDIn2	SAFE-In FB-Out	SAFEBOOL	FALSE	bOSSDIn2 is the 2nd channel for the safety device and otherwise corresponds to bOSSDIn1
tDiscTime-OSSD12		SAFETIME	TIME#0ms	Discrepancy monitoring time for the inputs bOSSDIn1 and bOSSDIn2
bMuting3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel or 1st input channel of the 2nd input group, otherwise corresponds with bMutingIn1
bMuting4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel or 2nd input channel of the 2nd input group, otherwise corresponds with bMutingIn2
tDiscTime34		SAFETIME	TIME#0ms	Discrepancy monitoring time for muting input 3 and 4.
bEDM1	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	bEDM1 is the feedback loop for the output channel bMuteOut, which is switched off immediately. If this input is parameterised as active, the safe state of the outputs is only quit if bEDM1 is set to 1.
tEDM1ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bMuteOut is activated. If the bEDM1 input does not switch to FALSE within this time, a function block bError is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
bEDM2	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	bEDM2 is the feedback loop for the output channel bMuteDelOut, which is switched off after a delay. If this input is parameterised as active, the safe state of the outputs is only quit if bEDM2 is set to 1.
tEDM2ResetTime		SAFETIME	TIME#0ms	Available only in CTMP1960-2600 If this value is not equal to 0, the timer is started when the output bMuteDelOut is activated. If the bEDM2 input does not switch to FALSE within this time, a function block bError is set and the outputs are switched off. In CTMP6900 if this value is not equal to 0 a compiler error will arise.
tMuteDelOutDelay		SAFETIME	TIME#0ms	Delay time between bMuteOut and bMuteDelOut.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bMutingActive	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	1st output channel; indicates the current muting process by a logic 1.
bMuteOut	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	1st output channel; the safe state corresponds to logic 0.
bMuteDelOut	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	2nd output channel; the safe state corresponds to logic 0; the safe state is output after a delay according to the parameterised output delay time tMutingDelOutDelay.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: the discrepancy time monitor of a 2-channel input group has detected an error; the muting sequence was violated or the maximum muting time was exceeded; the error must be acknowledged via the ERR Ackn input of the associated POU (NUMSafe group) FALSE: no error was detected.

10.2.7.1 Example with 4 individual muting sensors

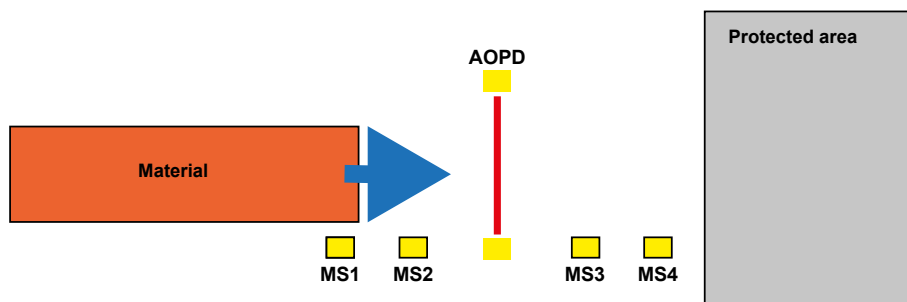


Fig. 109 Configuration example with 4 muting sensor

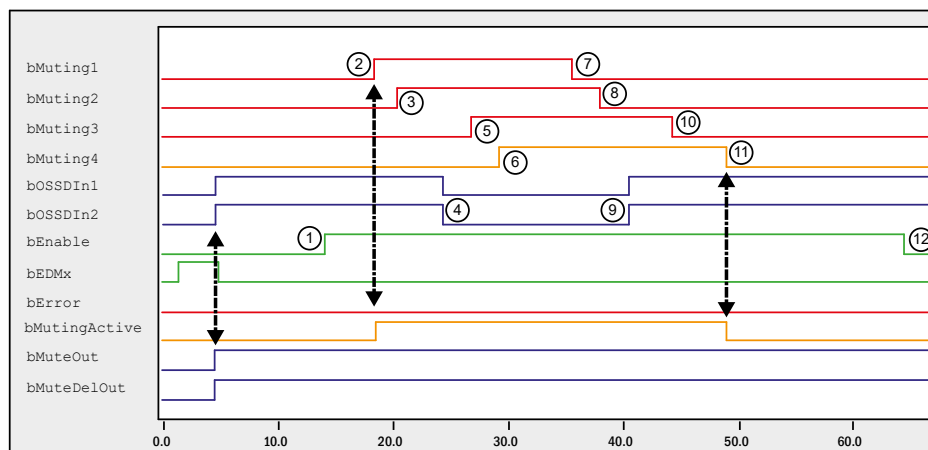


Fig. 110 Muting FB sequence

10.2.7.2 Example with 2 two-channel muting sensors

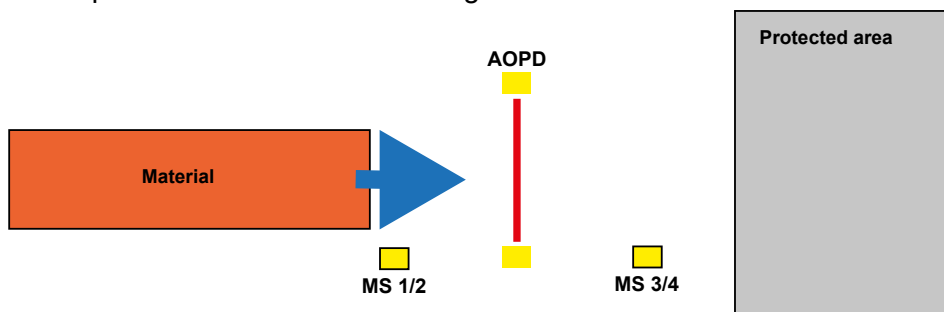


Fig. 111 Configuration example with 2 two-channel muting sensor

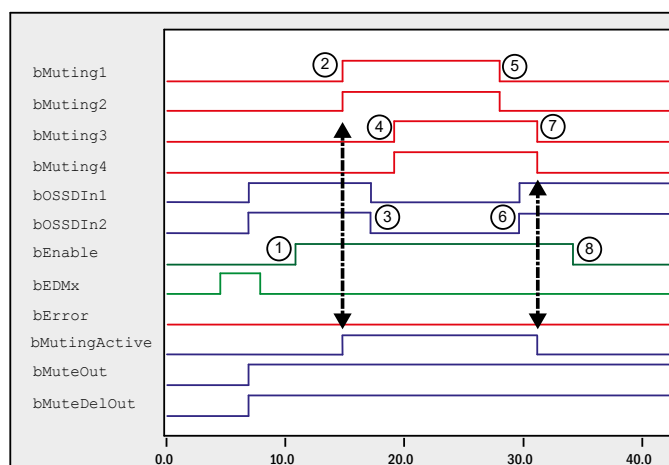


Fig. 112 Muting FB sequence

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	Discrepancy error in muting bMuting group 1
1	Discrepancy error in the bOSSDIn input group
2	Discrepancy error in muting bMuting group 2
4	EDM monitoring error bEDM1
5	EDM monitoring error bEDM2
6	Muting sequence was violated
7	Maximum muting time was exceeded
8	Discrepancy error bMuting1/bMuting2 has not yet been reset (only with CTMP1960-2600)
9	Discrepancy error bOSDDIn1/bOSDDIn2 has not yet been reset (only with CTMP1960-2600)
10	Discrepancy error bMuting3/bMuting4 has not yet been reset (only with CTMP1960-2600)

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN If bOSSDIn is TRUE and bEnable is FALSE or no muting sequence has yet been started, the FB_MUTING module assumes the RUN state. The outputs assume the following values: bError= 0 bMutingActive= 0 bMuteOut= 1 bMuteDelOut= 1
2	STOP The FB_MUTING module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError= 0 bMutingActive= 0 bMuteOut= 0 bMuteDelOut= 0
3	SAFE The FB_MUTING module assumes the SAFE state if bOSSDIn is FALSE and bEnable is FALSE. The outputs assume the following values: bError= 0 bMutingActive= 0 bMuteOut= 0 bMuteDelOut= 0
4	ERROR If the FB_MUTING module detects an error, the FB_MUTING module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError= 1 bMutingActive= 0 bMuteOut= 0 bMuteDelOut= 0
5	RESET The FB_MUTING module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError= 0 bMutingActive= 0 bMuteOut= 0 bMuteDelOut= 0
6	NOT USED

7	NOT USED
8	DELAYOUT The FB_MUTING module assumes the DELAYOUT state if bOSDDIn is FALSE and bEnable is FALSE and the delay time for the bMuteDelOut output has not yet expired. If a discrepancy error occurs in the DELAYOUT state, the FB_MUTING module does not assume the ERROR state until the output delay time has elapsed. The outputs assume the following values: bError= 0 bMutingActive= 0 bMuteOut= 0 bMuteDelOut= 1
9	MUTING1 (Figure 108 number 2-3) The outputs assume the following values: bError= 0 bMutingActive= 1 bMuteOut= 1 bMuteDelOut= 1
10	MUTING2 (Figure 108 number 3-4) Outputs see state 9
11	MUTING3 (Figure 108 number 4-5) Outputs see state 9
12	MUTING4 (Figure 108 number 5-6) Outputs see state 9
13	MUTING5 (Figure 108 number 6-7) Outputs see state 9
14	MUTING6 (Figure 108 number 7-8) Outputs see state 9
15	MUTING7 (Figure 108 number 8-9) Outputs see state 9
16	MUTING8 (Figure 108 number 9-10) Outputs see state 9
17	MUTING9 (Figure 108 number 10-11) Outputs see state 9

10.2.8 FB_MODE

FB description

Operating mode selector switches can be implemented with the FB_MODE function block. The function block has 8 inputs and 8 outputs, which are looped through one-to-one, whereby up to 8 different modes of operation can be selected.

The FB_MODE sets the corresponding output only if precisely one input is set ("1"); the other outputs remain in the safe state ("0"). If no input is set or if more than one is set, all outputs are in the safe state.

The `bRestart` input of the FB can be activated (active or manual restart) by connecting it to a signal or variable. An open input `bRestart` will create a compiler error.

If the restart shall be executed automatically (non-active or automatic restart) the input `bRestart` must be deleted.

If the `bRestart` input is activated, the safe state of the outputs is quit at the start and when changing the operating mode only via a 0->1->0 signal sequence at the `bRestart` input.

Beyond that a discrepancy time can be specified at the `tDiscTime` input with which the change from one mode of operation to the next is monitored.



At least two inputs of FB_MODE must be connected.

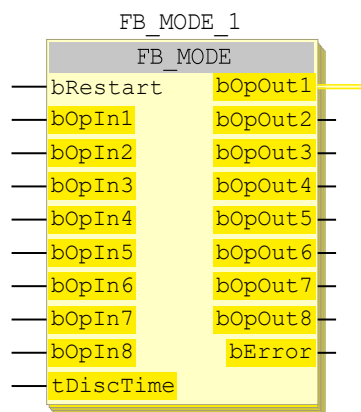


Fig. 113 FB_MODE function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bRestart</code>	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	When starting the FB or if all outputs have been switched to the safe state, the signal sequence 0 >1->0 must be recognised at the <code>bRestart</code> input before the safe state of the outputs is cancelled.
<code>bOpIn1</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel
<code>bOpIn2</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel
<code>bOpIn3</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel
<code>bOpIn4</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel
<code>bOpIn5</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	5th input channel
<code>bOpIn6</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	6th input channel

bOpIn7	SAFE-In FB-Out	SAFEBOOL	FALSE	7th input channel
bOpIn8	SAFE-In FB-Out	SAFEBOOL	FALSE	8th input channel
tDiscTime		SAFETIME	TIME#0ms	Discrepancy monitoring time

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bOpOut1	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	1st output channel
bOpOut2	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	2nd output channel
bOpOut3	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	3rd output channel
bOpOut4	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	4th output channel
bOpOut5	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	5th output channel
bOpOut6	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	6th output channel
bOpOut7	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	7th output channel
bOpOut8	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	8th output channel
bError	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	TRUE: The discrepancy time monitor or the input monitor has detected an error. The error must be acknowledged via the ERR Ackn input of the associated POU (NUMSafe group). FALSE: no error was detected.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	Discrepancy monitoring error

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The FB_MODE module assumes the RUN state if exactly one bOpInX input is TRUE. The outputs assume the following values: bError= 0 bOpOutX = bOpInX (1<=X<=8)
2	STOP The FB_MODE module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError= 0 bOpOutX = 0 (1<=X<=8)
3	SAFE The FB_MODE module assumes the SAFE state if not exactly one bOpInX input is TRUE. The outputs assume the following values: bError=0 bOpOutX = 0 (1<=X<=8)
4	ERROR If the FB_MODE module detects an error, the FB_MODE module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bOpOutX = 0 (1<=X<=8)
5	RESET The FB_MODE module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bOpOutX = 0 (1<=X<=8)
6	START The FB_MODE module assumes the START state if the bRestart input is active and TRUE, in order to wait for a rising and falling edge of the bRestart input before entering the RUN state and the corresponding output bOpOutX becomes TRUE. The outputs assume the following values: bError=0 bOpOutX = 0 (1<=X<=8)

10.2.9 FB_OR

FB description

Using the FB_OR function block, several input signals can be ORed to make an output signal. In addition, a setting can be made for each input as to whether it is to be negated.

This can be set using the "Negation" command from the context menu.

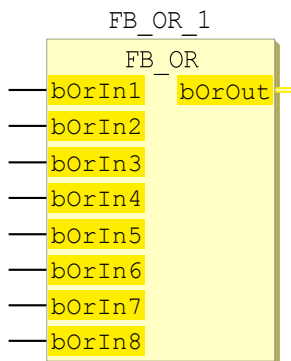


Fig. 114 FB_OR function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bOrIn1	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel
bOrIn2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel
bOrIn3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel
bOrIn4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel
bOrIn5	SAFE-In FB-Out	SAFEBOOL	FALSE	5th input channel
bOrIn6	SAFE-In FB-Out	SAFEBOOL	FALSE	6th input channel
bOrIn7	SAFE-In FB-Out	SAFEBOOL	FALSE	7th input channel
bOrIn8	SAFE-In FB-Out	SAFEBOOL	FALSE	8th input channel

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bOrOut	SAFE-Out FB-In Standard-Out Local-Out	SAFEBOOL BOOL	FALSE	Output channel

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0-15	always 0

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The RUN state is assumed if one or more of the active inputs bOrIn1-bOrIn8 is set to 1 (ACTIVE_ORIN=TRUE). The output assume the following value: bOrOut=1
2	STOP The FB_OR module assumes the STOP state if the input FbRun is FALSE. The output assume the following value: bOrOut=0
3	SAFE The SAFE state is assumed if all active inputs bOrIn1-bOrIn8 are 0 (ACTIVE_ORIN=FALSE). The output assume the following value: bOrOut=0

10.2.10 FB_RS

FB description

The FB_RS function block realises a Reset/Set function.

A logic 1 at the `bSet` input and logic 0 at the `bReset` input leads to a logic 1 at the output.

A logic 0 at the `bSet` input and logic 1 at the `bReset` input leads to a logic 0 at the output.

If both inputs are at logic 1, the Reset signal is dominant and leads to a logic 0 at the output. If both inputs are at logic 0, the output remains in its current state.

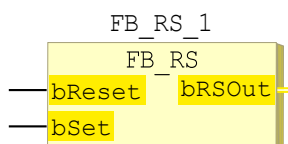


Fig. 115 FB_RS function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bReset</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
<code>bSet</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bRSOut</code>	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Output channel; the safe state corresponds to logic 0.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0-15	always 0

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
2	STOP The FB_RS module assumes the STOP state if the input FbRun is FALSE. The output assume the following value: bRSOut=0
3	SAFE If the bReset input is 1, the FB_RS module assumes the SAFE state. The output assume the following value: bRSOut=0
9	SET If the bReset input is 0 and the bSet input is 1, the FB_RS module assumes the RUN state. The output assume the following value: bRSOut=1

10.2.11 FB_SR

FB description

The FB_SR function block realises a Set/Reset function.

A logic 1 at the `bSet` input and logic 0 at the `bReset` input leads to a logic 1 at the output.

A logic 0 at the `bSet` input and logic 1 at the `bReset` input leads to a logic 0 at the output.

If both inputs are at logic 1, the Set signal is dominant and leads to a logic 1 at the output. If both inputs are at logic 0, the output remains in its current state.

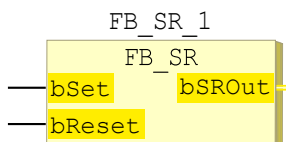


Fig. 116 FB_SR function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bSet</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
<code>bReset</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.

VAR_OUTPUT R

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bSROut</code>	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Output channel; the safe state corresponds to logic 0.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0-15	always 0

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
2	STOP The FB_SR module assumes the STOP state if the input FbRun is FALSE. The output assume the following value: bSROut=0
3	SAFE If the bReset input is 1 and the bSet input is 0, the FB_SR module assumes the SAFE state. The output assume the following value: bSROut=0
9	SET If the bReset input is 0 and the bSet input is 1, the FB_SR module assumes the SET state. The output assume the following value: bSROut=1

10.2.12 FB_TOF

FB description

A switch-off delay is realised with the FB_TOF function block.

A logic 1 at the `bTofIn` input is extended by the set time and forwarded to the output.

If the input is set to 1 again before the switch-off delay time is reached, the output remains switched on. The error output is inactive, since the block does not set errors.

The maximum switch-off delay is 6000 x 100 ms (10 minutes).

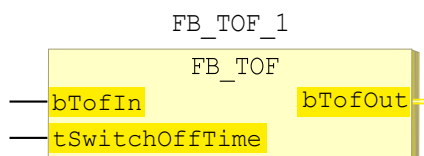


Fig. 117 FB_TOF function block

FB_TOF CTMP1960-2600 extention

In the CTMP1960-2600 the FB_TOF now also supports times between 1 ms and 600 seconds.

The function block has two timebases: 1 ms and 10 ms. With a timebase of 1 ms the maximum time is 60,000 ms in 1 ms steps. With a timebase of 10 ms the maximum time is 600,000 ms in 10 ms steps. The timebase is automatically selected according to the set time.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bTofIn</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
<code>tSwitchOffTime</code>		SAFETIME	TIME#0ms	Switch-off delay time.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bTofOut</code>	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Output channel; the safe state corresponds to logic 0.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0-15	always 0

State information (8- bit value)

Value	Description, parameter values
0	undefined
1	RUN If the <code>bTofIn</code> input is 1, the FB_TOF module assumes the RUN state. If the <code>bTofIn</code> changes to 0 in the RUN state, the FB_TOF module starts the delay timer with the <code>tSwitchOffTime</code> and changes to the DELAYOUT state. The output assume the following value: <code>bTofOut=1</code>
2	STOP The FB_TOF module assumes the STOP state if the input <code>FbRun</code> is FALSE. The output assume the following value: <code>bTofOut=0</code>
3	SAFE If the <code>bTofIn</code> input is 0 and the <code>tSwitchOffTime</code> has expired (<code>DelayTimeExpired=TRUE</code>), the FB_TOF module assumes the SAFE state. The output assume the following value: <code>bTofOut=0</code>
8	DELAY_OUT If the <code>bTofIn</code> input is 0 and the <code>tSwitchOffTime</code> has not yet expired (<code>DelayTimeExpired=FALSE</code>), the FB_TOF module assumes the DELAYOUT state. The output assume the following value: <code>bTofOut=1</code>

10.2.13 FB_TON

FB description

A switch-on delay is realised with the FB_TON function block.

A logic 1 at the `bTonIn` input is extended by the set time and forwarded to the output.

If the input is set to 0 again before the delay time is reached, the output is not switched on.

The maximum switch-on delay is 6000 x 100ms (10 minutes).

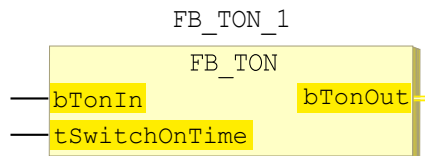


Fig. 118 FB_TON function block

FB_TON CTMP1960-2600 extention

In the CTMP1960-2600 the FB_TON now also supports times between 1 ms and 600 seconds.

The function block has two timebases: 1 ms and 10 ms. With a timebase of 1 ms the maximum time is 60,000 ms in 1 ms steps. With a timebase of 10 ms the maximum time is 600,000 ms in 10 ms steps. The timebase is automatically selected according to the set time.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bTonIn</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
<code>tSwitchOnTime</code>		SAFETIME	TIME#0ms	Switch-on delay time.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bTonOut</code>	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Output channel; the safe state corresponds to logic 0.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0-15	always 0

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN If the <code>bTonIn</code> input is 1 and the delay time has expired (<code>DelayTimeExpired=TRUE</code>), the <code>FB_TON</code> module assumes the RUN state. The output assume the following value: <code>bTonOut=1</code>
2	STOP The <code>FB_TON</code> module assumes the STOP state if the input <code>FbRun</code> is FALSE. The output assume the following value: <code>bTonOut=0</code>
3	SAFE If the <code>bTonIn</code> input is 0, the <code>FB_TON</code> module assumes the SAFE state. If the <code>bTonIn</code> input changes to 1 in the SAFE state, the <code>FB_TON</code> module starts the delay timer with the <code>tSwitchOnTime</code> and changes to the DELAYIN state. The output assume the following value: <code>bTonOut=0</code>
9	DELAY_OUT If the <code>bTonIn</code> input is 1 and the delay time has not yet expired (<code>DelayTimeExpired=FALSE</code>), the <code>FB_TON</code> module assumes the DELAYIN state. The output assume the following value: <code>bTonOut =0</code>

10.2.14 FB_TWOHAND

FB description

The FB_TWOHAND block implements a two-hand circuit in which both input groups must be actuated at the same time in order to switch the output. Renewed setting of the output is only possible if both input groups are at logic 0 at the same time.

An input group can be configured as a single-channel and two-channel input with discrepancy time monitoring. In addition it is possible to define time monitoring of up to 2500 ms between the two input groups. Each input can be configured both as a normally closed contact (N.C.) and as a normally open contact (N.O.) using the "Negation" command.

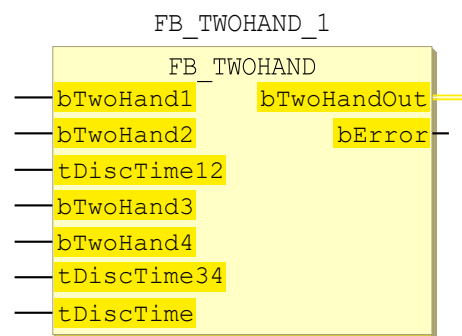


Fig. 119 FB_TWOHAND function block

bTwohandIn1 is used as a single-channel input if bTwohandIn2 and tDiscTime12 are deleted. This operating principle also applies to the second input pair.

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bTwoHand1	SAFE-In FB-Out	SAFEBOOL	FALSE	1st input channel; the "Negation" command can be used to specify whether a normally closed contact (safe state is requested with logic 0) or a normally open contact (safe state is requested with logic 1) is linked with this input.
bTwoHand2	SAFE-In FB-Out	SAFEBOOL	FALSE	2nd input channel: behaviour same as bTwoHand1. If the discrepancy time is not equal to 0, the 1st and 2nd input channels are considered to be the 1st input group and the discrepancy time between the two channels is monitored if either of the two input channels requests the safe state.
tDiscTime12		SAFETIME	TIME#0ms	Discrepancy time for time monitoring of channels 1 and 2.
bTwoHand3	SAFE-In FB-Out	SAFEBOOL	FALSE	3rd input channel or 1st input channel of the 2nd input group, otherwise corresponds to bTwoHand1
bTwoHand4	SAFE-In FB-Out	SAFEBOOL	FALSE	4th input channel or 2nd input channel of the 2nd input group, otherwise corresponds to bTwoHand2
tDiscTime34		SAFETIME	TIME#0ms	Discrepancy time for time monitoring of channels 3 and 4.
tDiscTime		SAFETIME	TIME#0ms	Discrepancy time for time monitoring of the two input groups.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bTwoHandOut	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	1st output channel; the safe state corresponds to logic 0.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: the discrepancy time monitor of a 2-channel input group has detected an error; the error must be acknowledged via the ERR Ackn input of the associated NUMSafe group. FALSE: no error was detected.

DIAGNOSTIC INFORMATION (16- BIT VALUE)

Bit	Description, parameter values
0	Discrepancy error input pair 1
1	Discrepancy error input pair 2
2	Discrepancy error between the two input pairs
6	Two-hand error - one of the two input pairs is actuated and waiting for the second input pair. The error is issued if the second input pair is now actuated and the first input pair is no longer detected as actuated.

STATE INFORMATION (8- BIT VALUE)

Value	Description, parameter values
0	undefined
1	RUN The FB_TWOHAND module assumes the RUN state if all active inputs are TRUE. The FB_TWOHAND module can only assume the RUN state from one of the states 1BUTTON or 2BUTTON. The outputs assume the following values: bError=0 bTwoHandOut=1
2	STOP The FB_TWOHAND module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 bTwoHandOut=0
3	SAFE The FB_TWOHAND module assumes the SAFE state if all active inputs are FALSE. The FB_TWOHAND module exits the SAFE state if all active inputs of an input group are TRUE. The outputs assume the following values: bError=0 bTwoHandOut=0
4	ERROR If the FB_TWOHAND module detects an error, the FB_TWOHAND module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bTwoHandOut=0
5	RESET The FB_TWOHAND module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bTwoHandOut=0

11	1BUTTON If all active inputs of the first input group are TRUE and at least one active input of the second input group is FALSE, the module FB_TWOHAND assumes the 1BUTTON state. The FB_TWOHAND module only assumes the 1BUTTON state from the SAFE state. The outputs assume the following values: bError=0 bTwoHandOut=0
12	2BUTTON If all active inputs of the second input group are TRUE and at least one active input of the first input group is FALSE, the module FB_TWOHAND assumes the 2BUTTON state. The FB_TWOHAND module only assumes the 2BUTTON state from the SAFE state. The outputs assume the following values: bError=0 bTwoHandOut=0
13	RELEASE If at least one active input is FALSE, the FB_TWOHAND module assumes the RELEASE state. If all active inputs are FALSE, the FB_TWOHAND module exits the RELEASE state. The outputs assume the following values: bError=0 bTwoHandOut=0

1

2

3

4

5

6

7

8

9

10

11

12

10.3 Library " CTMP1960-2600 NUMsafe compact controller "

10.3.1 FB_ADD

FB description

FB_ADD adds the two connected analog input values and passes the result to the `AnalogOut` output. If an overflow or underflow occurs during the addition, the ERROR state is assumed. The `AnalogOut` is set to 0 and the output `bError` is set to 1.

If the input `ERR Ackn` of the corresponding group is 1, then the RESET state is activated. When the input `ERR Ackn` of the corresponding group changes to 0 again, the RESET state is exited and the RUN state is entered. In RESET state, the output `AnalogOut` is 0 and the output `bError` is 0.

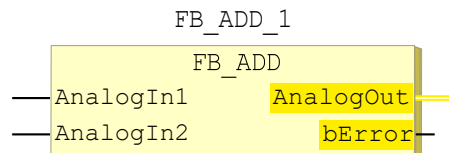


Fig. 120 FB_ADD function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	First analog signal for addition. Permitted data types: UINT, INT, UDINT, DINT
AnalogIn2	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Second analog signal for addition. Permitted data types: UINT, INT, UDINT, DINT

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Analog output that contains the result of the addition. Permitted data types: UINT, INT, UDINT, DINT. The data Type must be selected according to the input data Type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The output <code>AnalogOut</code> has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	The <code>AnalogOut</code> output is correct
1	The <code>AnalogOut</code> output has an underflow (is less than the smallest possible value)
2	The <code>AnalogOut</code> output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4080	An underflow has occurred	FB number	AnalogIn1	AnalogIn2
0x4081	An overflow has occurred	FB number	AnalogIn1	AnalogIn2

STATE INFORMATION

Value	Description, parameter values
0	not used
1	RUN The FB_ADD module cyclically adds the two analog inputs <i>AnalogIn1</i> and <i>AnalogIn2</i> . If no overflow or underflow occurs during the addition, the FB_ADD module is in the RUN state. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> = result of the addition
2	STOP The FB_ADD module assumes the STOP state if the input <i>FbRun</i> is FALSE. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> =0
3	Not used
4	ERROR If the FB_ADD module detects an error when checking the value range of <i>AnalogOut</i> during the addition, the FB_ADD module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: <i>bError</i> =1 <i>AnalogOut</i> =0
5	RESET The FB_ADD module assumes the RESET state if no further error is pending after an error has occurred and the <i>ERR Ackn</i> input of the corresponding group is set to TRUE. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> =0

10.3.2 FB_COMPARE

FB description

FB_COMPARE checks that the 2-5 analog inputs `CompIn1` to `CompIn5` are equal within a window of tolerance based on time and value. The inputs `b1oo2`, `b2oo3`, and `b3oo5` specify the number of inputs to be evaluated. The parameters `udiAllowedDeviation` and `tToleranceTime` (ms) can be used for specifying the deviations and the time interval to be permitted between the input values.

The output `bIsValid` returns a logical 1 if the comparison results in a positive value.

The output `CompOut` contains the first analog value that meets the comparison criteria. The ERROR state is entered if an overflow or underflow occurs. Then the outputs `CompOut` and `bIsValid` are set to 0 and the output `bError` is set to 1.

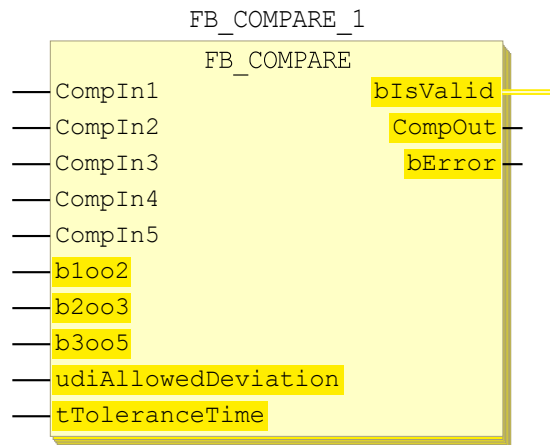


Fig. 121 FB_COMPARE function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
CompIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	First analog signal for the comparison (1oo2, 2oo3, 3oo5). Permitted data types:UINT, INT, UDINT, DINT
CompIn2	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Second analog signal for the comparison (1oo2, 2oo3, 3oo5) Permitted data types:UINT, INT, UDINT, DINT
CompIn3	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Third analog signal for the comparison (2oo3, 3oo5) Permitted data types:UINT, INT, UDINT, DINT
CompIn4	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Fourth analog signal for the comparison (3oo5) Permitted data types:UINT, INT, UDINT, DINT
CompIn5	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Fifth analog signal for the comparison (3oo5) Permitted data types:UINT, INT, UDINT, DINT

b1oo2	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Comparison mode 1oo2: Only the first two inputs CompIn1 and CompIn2 are active. If both inputs are equal (within the permitted deviation), then the first input CompIn1 is written to the output CompOut. If an error occurs, then the output CompOut is deactivated and the output bIsValid is reset. Only a constant value is permitted (no variable). Only one of the inputs b1oo2, b2oo3, or b3oo5 can be TRUE.
b2oo3	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Comparison mode 2oo3: Only the first three inputs CompIn1, CompIn2, and CompIn3 are active. If at least two inputs are equal (within the permitted deviation), then the first input with the valid value is written to the output CompOut. If an error occurs, then the output CompOut is deactivated and the output bIsValid is reset. Only a constant value is permitted (no variable). Only one of the inputs b1oo2, b2oo3, or b3oo5 can be TRUE.
b3oo5	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Comparison mode 3oo5: All five inputs CompIn1, CompIn2, CompIn3, CompIn4, and CompIn5 are active. If at least three inputs are equal (within the permitted deviation), then the first input with the valid value is written to the output CompOut. If an error occurs, then the output CompOut is deactivated and the output bIsValid is reset. Only a constant value is permitted (no variable). Only one of the inputs b1oo2, b2oo3, or b3oo5 can be TRUE.
udiAllowed Deviation	-	SAFEUDINT	0	Permitted deviation when comparing the inputs Only a constant value is permitted (no variable).
tToleranceTime	-	SAFETIME	t#0ms	Time frame (in ms) in which unequal values are tolerated. Only a constant value is permitted (no variable).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bIsValid	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The values are equal (within the permitted deviation).
CompOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Output value: Corresponds to the first CompIn input value that fulfills the comparison. Permitted data types: UINT, INT, UDINT, DINT. The data Type must correspond to the input data Type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	0	TRUE: The output CompOut has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	The <code>CompOut</code> output is correct
1	The <code>CompOut</code> output has an underflow (is less than the smallest possible value)
2	The <code>CompOut</code> output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4098	An underflow has occurred	FB number	<code>CompOut</code>	Smallest permitted value
0x4099	An overflow has occurred	FB number	<code>CompOut</code>	Largest permitted value

STATE INFORMATION

Value	Description, parameter values
0	undefined
1	RUN The FB_COMPARE module assumes the RUN state if a sufficient number of analog inputs deviate from each other by no more than the allowed deviation (ValuesEqual=TRUE). The outputs assume the following values: Error=0 IsValid=1 CompOut=CompInX (X= smallest input that does not deviate)
2	STOP The FB_COMPARE module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: Error=0 IsValid=0 CompOut=0
3	SAFE The FB_COMPARE module assumes the SAFE state if not enough analog inputs deviate from each other by no more than the allowed deviation (ValuesEqual=FALSE) and the DelayOutTimer has expired (DelayOutExpired=TRUE). The outputs assume the following values: Error=0 IsValid=0 CompOut=0
4	ERROR If the FB_COMPARE module detects an error while checking the value range of CompOut, the FB_COMPARE module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: Error=1 IsValid=0 CompOut=0
5	RESET The FB_COMPARE module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: Error=0 IsValid=0 CompOut=0
8	DELAYOUT If not enough analog inputs deviate from each other by no more than the allowed deviation (ValuesEqual=FALSE), the DelayOutTimer is started with the ToleranceTime. The FB_COMPARE module assumes the DELAYOUT state as long as the DelayOutTimer has not expired (DelayOutExpired=FALSE). The outputs assume the following values: Error=0 IsValid=1 CompOut = unchanged

10.3.3 FB_COUNTER

FB description

FB_COUNTER implements an incrementing and decrementing counter. The output `ActValue` specifies the current internal counter value. The counter can be parameterized by means of the inputs `PresetValue` and `CounterLimit`.

For a logical 1 signal at the input `bReset`, the internal counter value is set to the value by means of `PresetValue`. For a rising edge at the input `bCountUp`, the internal counter value is incremented by 1.

For a rising edge at the input `bCountDown`, the internal counter value is decremented by 1. As soon as the counter value specified in `CounterLimit` is reached, the output `bCounterOut` is set. When the counter value is 0, the output `bCounterZero` is set. If the NUMSafe group is started (`Run=1`), then `ActValue` is set to `PresetValue`. When the group is stopped, `ActValue` is set to 0.

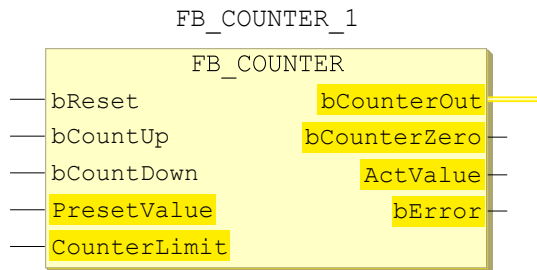


Fig. 122 FB_COUNTER function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bReset	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Rising edge resets the counter to the value <code>PresetValue</code> .
bCountUp	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Rising edge increments the internal counter value by 1.
bCountDown	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Rising edge decrements the internal counter value by 1.
PresetValue	-		0	Initialization value to which the counter is set at Reset. Only a constant value is permitted (no variable).
CounterLimit	-		0	If this value is reached, then the output <code>bCounterOut</code> is set to TRUE. Only a constant value is permitted (no variable).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bCounterOut	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Set to TRUE when the internal counter value has reached the value of Counter-Limit.
bCounterZero	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Set to TRUE when the internal counter value is less than or equal to 0.
ActValue	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Current value of the internal counter value Permitted data types: UINT, INT, UDINT, DINT
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The output <code>ActValue</code> has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	No diagnostic information
1	The <code>ActValue</code> output has an underflow (is less than the smallest possible value)
2	The <code>ActValue</code> output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x40B8	An underflow has occurred	FB number	<code>ActValue</code>	Smallest permitted value
0x40B9	An overflow has occurred	FB number	<code>ActValue</code>	Largest permitted value

STATE INFORMATION

Value	Description, parameter values
1	RUN In the RUN state the FB_COUNTER module sets the <code>ActValue</code> output to <code>PresetValue</code> if the <code>bReset</code> input is TRUE. If the <code>bReset</code> input is FALSE, the FB_COUNTER module increments the <code>ActValue</code> output when a rising edge is detected at the <code>bCountUp</code> input and decrements it when a rising edge is detected at the <code>bCountDown</code> input (this means <code>ActValue</code> remains unchanged when both <code>bCountUp</code> and <code>bCountDown</code> inputs detect a rising edge). The outputs assume the following values: <code>bError=0</code> <code>bCounterOut=(ActValue >= CounterLimit)</code> <code>bCounterZero=(ActValue == 0)</code> <code>bReset=TRUE: ActValue=PresetValue</code> <code>bReset=FALSE: ActValue=ActValue+n (-1 <= n <= 1)</code>
2	STOP The FB_COUNTER module assumes the STOP state if the input <code>FbRun</code> is FALSE. The outputs assume the following values: <code>bError=0</code> <code>bCounterOut=0</code> <code>bCounterZero=0</code> <code>ActValue=0</code>
4	ERROR If the FB_COUNTER module detects an error while checking the value range of <code>CounterOut</code>, the FB_COUNTER module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: <code>bError=1</code> <code>bCounterOut=0</code> <code>bCounterZero=0</code> <code>ActValue=0</code>
5	RESET The FB_COUNTER module assumes the RESET state if no further error is pending after an error has occurred and the <code>ERR Ackn</code> input of the corresponding group is set to TRUE. The outputs assume the following values: <code>bError=0</code> <code>bCounterOut=0</code> <code>bCounterZero=0</code> <code>ActValue=0</code>

10.3.4 FB_DIV

FB description

FB_DIV divides the input `AnalogIn1` by the input `AnalogIn2` and passes the result to the output `AnalogOut`.

If the input `AnalogIn2` = 0, then the output `AnalogOut` is set to 0. In this case, an error is not issued.

If an overflow or underflow occurs during the division, the ERROR state is assumed.

The output `AnalogOut` is set to 0 and the output `bError` is set to 1.

The inputs `bFloor`, `bCeil`, and `bRound` can be used for specifying the rounding method to be applied.

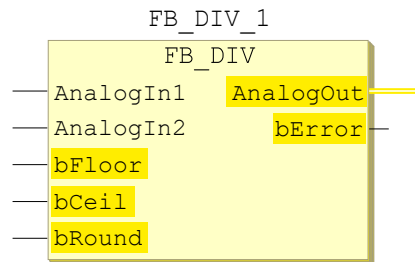


Fig. 123 FB_DIV function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	First analog input signal for division. Permitted data types: UINT, INT, UDINT, DINT
AnalogIn2	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Second analog input signal for division. Permitted data types: UINT, INT, UDINT, DINT
bFloor	-	SAFEBOOL	FALSE	Rounding method: Decimal places are truncated. Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> , and <code>bRound</code> can be TRUE.
bCeil	-	SAFEBOOL	FALSE	Rounding method: The next largest integer is returned as the result. Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> , and <code>bRound</code> can be TRUE.
bRound	-	SAFEBOOL	FALSE	Rounding method: Rounded away from zero (e.g. rounded to 3). Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> , and <code>bRound</code> can be TRUE.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Analog output with the quotient Permitted data types: UINT, INT, UDINT, DINT. The data Type must be selected according to the input data Type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The output <code>AnalogOut</code> has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	The AnalogOut output is correct
1	The AnalogOut output has an underflow (is less than the smallest possible value)
2	The AnalogOut output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4080	An underflow has occurred	FB number	AnalogIn1	AnalogIn2
0x4081	An overflow has occurred	FB number	AnalogIn1	AnalogIn2

STATE INFORMATION

Value	Description, parameter values
0	not used
1	RUN In the RUN state, the FB_DIV module enters the result of the division in the AnalogOut output. The outputs assume the following values: bError= 0 AnalogOut = division result
2	STOP The FB_DIV module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError= 0 AnalogOut=0
3	Not used
4	ERROR If the FB_DIV module detects an error when checking the value range of AnalogOut, the FB_DIV module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError= 1 AnalogOut=0
5	RESET The FB_DIV module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError= 0 AnalogOut=0

10.3.5 FB_ENVELOPE

FB description

FB_ENVELOPE creates an envelope curve from the value of `InValue` plus `uiOffset` and then it checks each time the function block is called whether `InValue` has violated this envelope curve.

If `InValue` reaches the range of $- \text{udiTargetValue}$ and $+ \text{udiTargetValue}$, then the time `tTimeAfterInTarget` begins to run. If `InValue` leaves this range again and the time has not elapsed yet, then this time is reset and it restarts if necessary when `InValue` reaches the range again. If `InValue` remains in the range, then the output `bSafeFunctionOut` is set to FALSE after the time `tTimeAfterInTarget` has elapsed. The output `bSafeFunctionOut` is set to FALSE at the latest after `tMaxTime` has elapsed.

This function block can normally be employed for example for an SS1 or SS2 safety function.

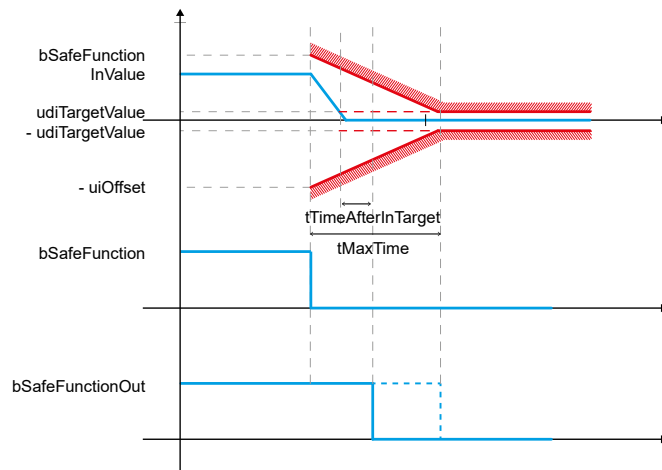


Fig. 124 Diagram showing FB_ENVELOPE

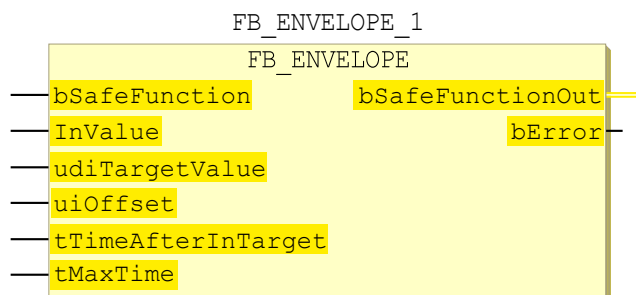


Fig. 125 FB_ENVELOPE function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bSafeFunction</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	For a falling edge, a envelope curve is created from the value of <code>InValue</code> plus <code>uiOffset</code> . At each call cycle of the function block, the system checks whether <code>InValue</code> violates this envelope curve.
<code>InValue</code>	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Analog value that should be monitored when a delay or change occurs in the diction 0 within an envelope curve. Permitted dataTypes: UINT, INT, UDINT, DINT.
<code>udiTargetValue</code>	-	SAFEUDINT	0	Time frame of <code>InValue</code> symmetric by 0 ($- \text{udiTargetValue} \leq \text{InValue} \leq \text{udiTargetValue}$) Only a constant value is permitted (no variable).

uiOffset	-	SAFEUINT	0	Value that is added to the value of InValue and then forms the starting point of the envelope curve as a positive and negative value. Only a constant value is permitted (no variable).
tTimeAfterIn-Target	-	SAFETIME	t#0ms	Time after which the output bSafeFunctionOut is set to FALSE when InValue is located in the range -udiTargetValue and +udiTargetValue. The time begins to elapse when InValue reaches the range between -udiTargetValue and +udiTarget-Value. The time is reset when InValue leaves the range and tTimeAfterInTarget has not elapsed yet. The time is restarted when InValue reaches the range again. Only a constant value is permitted (no variable).
tMaxTime	-	SAFETIME	t#0ms	Maximum time (in ms) after which the output bSafeFunctionOut is set to FALSE. Only a constant value is permitted (no variable).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bSafeFunctionOut	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Output with the safety function (delayed by time and monitored by envelope curve) Set to FALSE when InValue remains within the envelope curve until tTimeAfterInTarget has elapsed or until tMaxTime has elapsed.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: InValue erroneous (e.g. above or below the envelope curve) or tMaxTime elapsed FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	no diagnostic information
1	Underflow (InValue below the envelope curve)
2	Overflow (InValue above the envelope curve)
3	InValue Error
4	MaxTimeExpired

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x40C8	InValueExpired AND InValue < 0	FB instance	InValue	MinAllowedValue
0x40C9	InValueExpired AND InValue > 0	FB instance	InValue	MaxAllowedValue
0x40CA	InValueError=TRUE	FB instance	InValue	-
0x40CB	MaxTimeExpired=TRUE	FB instance	InValue	-

STATE INFORMATION

Value	Description, parameter values
1	RUN The FB_ENVELOPE module assumes the RUN state if the input bSafeFunction is TRUE. The outputs assume the following values: bError=0 bSafeFunctionOut=1
2	STOP The FB_ENVELOPE module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 bSafeFunctionOut=0
3	SAFE The FB_ENVELOPE module assumes the SAFE state if the input bSafeFunction is FALSE, MaxTimeExpired is TRUE or tTimeAfterInTarget Expired is TRUE. The outputs assume the following values: bError=0 bSafeFunctionOut=0
4	ERROR If the FB_ENVELOPE module detects an error, the FB_ENVELOPE module assumes the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bSafeFunctionOut=0
5	RESET The FB_ENVELOPE module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bSafeFunctionOut=0
8	DELAYOUT The FB_ENVELOPE module assumes the DELAYOUT state if the input bSafeFunction is FALSE, InValueInTarget is TRUE, tTimeAfterInTarget Expired is TRUE and MaxTimeExpired is FALSE. The outputs assume the following values: bError=0 bSafeFunctionOut=1
9	MONITOR If the input bSafeFunction is FALSE, InValueInTarget is FALSE and MaxTimeExpired is FALSE, the FB_ENVELOPE module assumes the MONITOR state and monitors whether the InValue is still within the envelope (InValueExpired=FALSE). The FB_ENVELOPE module calculates the maximum permissible difference by first multiplying the InValueDec with the elapsed time since the transition to the MONITOR state. This intermediate result is subtracted from InValueLatch (InValue at the time of the state transition to the MONITOR state); the value must not be smaller than the udiTargetValue. If this difference is less than the current InValue, InValueExpired is set to TRUE. The outputs assume the following values: bError=0 bSafeFunctionOut=1

10.3.6

FB_LIMIT

FB description

FB_LIMIT checks whether the value at the input AnalogIn is within predefined limits.

If the value at AnalogIn is within the limits of MinValue and MaxValue, then the output bInLimit is set.

If the value is greater than MaxValue, then the output bAboveMax is set.

If the value is less than MinValue, then the output bBelowMin is set.

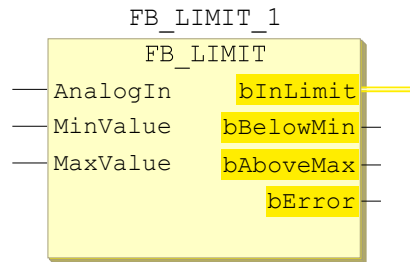


Fig. 126 FB_LIMIT function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Analog input value for the check Permitted data Types: UINT, INT, UDINT, DINT
MinValue	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Minimum value Permitted data Types: UINT, INT, UDINT, DINT Constant value or variable permitted.
MaxValue	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Maximum value Permitted data Types: UINT, INT, UDINT, DINT Constant value or variable permitted.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bInLimit	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: Value AnalogIn is within the minimum and maximum limits.
bBelowMin	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: Value AnalogIn is below the minimum limit.
bAboveMax	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: Value AnalogIn is above the maximum limit.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: MinValue is greater than MaxValue. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	No diagnostic information
1	MinValue is greater than MaxValue

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4090	MinValue is greater than MaxValue	FB number	MinValue	MaxValue

STATE INFORMATION

Value	Description, parameter values
1	RUN If AnalogIn is greater than or equal to MinValue and less than or equal to MaxValue, the FB_LIMIT module assumes the RUN state. The outputs assume the following values: bError=0 bInLimit=1 bBelowMin=0 bAboveMax=0
2	STOP If the input FbRun is FALSE, the FB_LIMIT module assumes the STOP state. The outputs assume the following values: bError=0 bInLimit=0 bBelowMin=0 bAboveMax=0
3	SAFE If AnalogIn is less than MinValue or greater than MaxValue, the FB_LIMIT module assumes the SAFE state. The outputs assume the following values: bError=0 bInLimit=0 bBelowMin=(AnalogIn<MinValue) bAboveMax=(AnalogIn>MaxValue)
4	ERROR If the FB_LIMIT module detects an error, the FB_LIMIT module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 bInLimit=0 bBelowMin=0 bAboveMax=0
5	RESET The FB_LIMIT module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 bInLimit=0 bBelowMin=0 bAboveMax=0

10.3.7 FB_LOADSENSING

FB description

FB_LOADSENSING checks the inputs `AnalogInX` and `AnalogInY` whether the `AnalogInY` values at the position `AnalogInX` are within the range. For the `AnalogInY` values in the table, there is a warning level and a switch-off level.

If `bInactive=FALSE`, then the largest index is determined whose corresponding X-value is still less than the input `AnalogInX`. Then it checks whether the input `AnalogInY` is within the corresponding switch-off level (`Y1`, `Y2`) or within the warning level (`WY1`, `WY2`).

If the value is within the switch-off level, then the output `bValid` is set. If the value is between the values `Y1` and `WY1`, or `Y2` and `WY2`, then the output `bWarning` is also set. There can be up to 25 indices.

The check can be reversed by means of the input `bOutsideY1Y2` so that values outside of `Y1` and `Y2` are valid and the output `bValid` is set. In this case, the warning level has to be specified greater than the switch-off level.

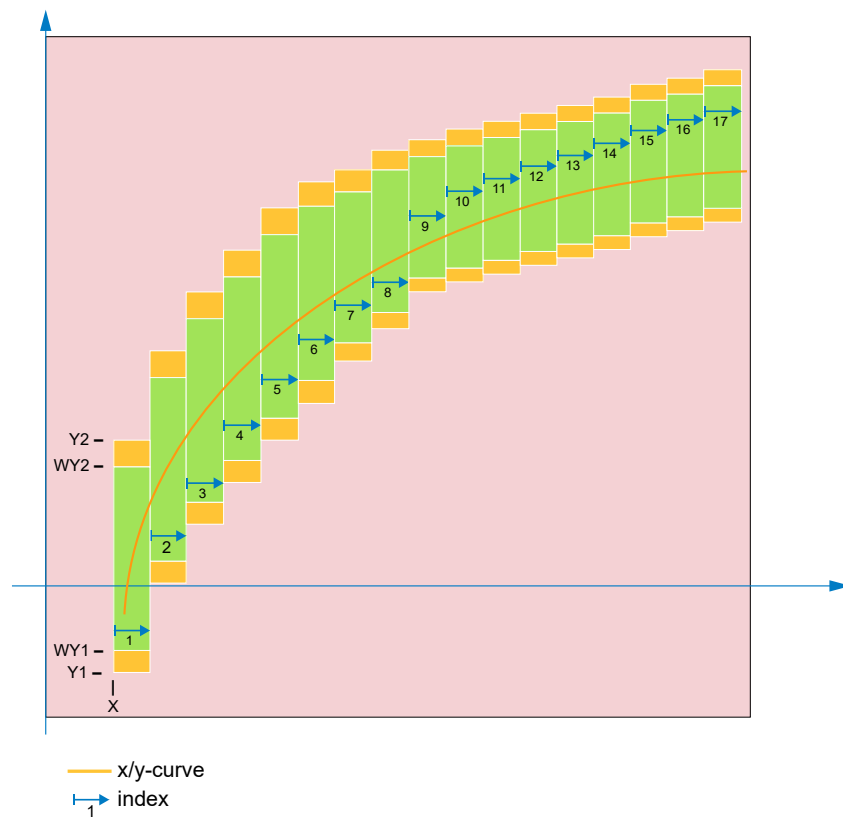


Fig. 127 Diagram showing verification of the characteristic curve

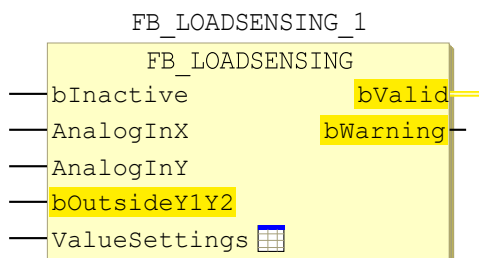


Fig. 128 FB_LOADSENSING function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bInactive	SAFE-In FB-Out Standard-In	SAFEBOOL BOOL	FALSE	Input for deactivating the FB
AnalogInX	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Analog X value Permitted dataTypes: UINT, INT, UDINT, DINT
AnalogInY	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Analog Y value Permitted data Types: UINT, INT, UDINT, DINT
bOutsideY1Y2	-	SAFEBOOL	FALSE	The data of the value table is checked according to the following formula: bOutsideY1Y2 = FALSE: Y1[index] <= WY1[index] < WY2[index] <= Y2[index] bOutsideY1Y2 = TRUE: WY1[index] <= Y1[index] < Y2[index] <= WY2[index]
ValueSettings	-	SAFELINT	-	Table with X values, warning levels (WY1, WY2), and switch-off levels (Y1, Y2). Only constant values are permitted (no variables).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bValid	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: AnalogInY is located within the switch-off level.
bWarning	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: AnalogInY is located between the switch-off and warning level.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
-	No diagnostic information

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
-	-	-	-	-

STATE INFORMATION

Value	Description, parameter values
1	RUN If <code>bInactive</code> is FALSE and <code>AreaValid</code> is TRUE, the FB_LOADSENSING module assumes the RUN state. The outputs assume the following values: <code>bValid=1</code> <code>bWarning=0</code>
2	STOP The FB_LOADSENSING module assumes the STOP state if the input <code>FbRun</code> is FALSE. The outputs assume the following values: <code>bValid=0</code> <code>bWarning=0</code>
3	SAFE If <code>bInactive</code> is FALSE, <code>AreaValid</code> is FALSE and <code>AreaValidButWarning</code> is FALSE, the FB_LOADSENSING module assumes the SAFE state. The outputs assume the following values: <code>bValid=0</code> <code>bWarning=0</code>
16	INACTIVE If <code>bInactive</code> is TRUE, the FB_LOADSENSING module assumes the INACTIVE state. The outputs assume the following values: <code>bValid=0</code> <code>bWarning=0</code>
17	WARNING If <code>bInactive</code> is FALSE and <code>AreaValidButWarning</code> is TRUE, the FB_LOADSENSING module assumes the WARNING state. The outputs assume the following values: <code>bValid=1</code> <code>bWarning=1</code>

10.3.8 FB_MUL

FB description

FB_MUL multiplies the input AnalogIn1 with the input AnalogIn2 and passes the result to the output AnalogOut.

Then the output AnalogOut is set to 0 and the output bError is set to 1.

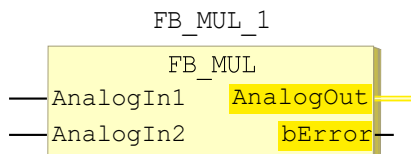


Fig. 129 FB_MUL function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	First analog signal for multiplication. Permitted data Types: UINT, INT, UDINT, DINT
AnalogIn2	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Second analog signal for multiplication. Permitted data Types: UINT, INT, UDINT, DINT

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Analog output that contains the result of multiplication. Permitted data Types: UINT, INT, UDINT, DINT. The data Type must be selected according to the input data type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The output AnalogOut has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	The AnalogOut output is correctW
1	The AnalogOut output has an underflow (is less than the smallest possible value)
2	The AnalogOut output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4080	An underflow has occurred	FB number	AnalogIn1	AnalogIn2
0x4081	An overflow has occurred	FB number	AnalogIn1	AnalogIn2

STATE INFORMATION

Value	Description, parameter values
0	not used
1	RUN In RUN state the FB_MUL module enters the result of the multiplication in the AnalogOut output. The outputs assume the following values: bError=0 AnalogOut = multiplication result
2	STOP The FB_MUL module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 AnalogOut=0
3	Not used
4	ERROR If the FB_MUL module detects an error when checking the value range of AnalogOut, the FB_MUL module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 AnalogOut=0
5	RESET The FB_MUL module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 AnalogOut=0

10.3.9 FB_SCALING

FB description

FB_SCALING multiplies the input `AnalogIn` with the scale factor and then divides and adds to the scaling offset. The input `AnalogIn` can be negated. This corresponds to the Permitted data Types INT16 and INT32 of a multiplication with -1, and for Permitted data Types UINT16 and UINT32 of an XOR function with 0xFFFF and 0xFFFFFFFF. The inputs `bFloor`, `bCeil`, and `bRound` can be used for specifying the rounding method for internal division to be applied. In addition, the input `tWatchdog` can be used for defining that the input `AnalogIn` has to be changed within the specified time. If the input `AnalogIn` remains unchanged within the specified time, then the output `bStuckAtError` is set to TRUE. If input `tWatchdog` is set to `t#0ms` or it is unused, then the check is deactivated. Because the output `bStuckAtError` is not a FB error, the TwinSAFE group does not enter an error state. It has to react to it in the application program.

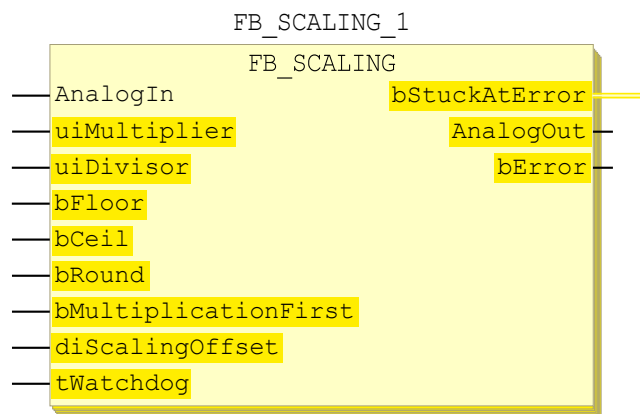


Fig. 130 FB_SCALING function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Input for scaling. The input can be negated. Permitted data Types: UINT, INT, UDINT, DINT
uiMultiplier	-	SAFEUINT	0	Scale factor multiplier Only a constant value is permitted (no variable).
uiDivisor	-	SAFEUINT	0	Scale factor divisor Only a constant value is permitted (no variable).
bFloor	-	SAFEBOOL	FALSE	Rounding method: Decimal places are truncated. Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> and <code>bRound</code> can be TRUE.
bCeil	-	SAFEBOOL	FALSE	Rounding method: The next largest integer is returned as the result. Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> and <code>bRound</code> can be TRUE.
bRound	-	SAFEBOOL	FALSE	Rounding method: Rounded away from zero (e.g. rounded to 3) Only a constant value is permitted (no variable). Only one of the inputs <code>bFloor</code> , <code>bCeil</code> and <code>bRound</code> can be TRUE.

bMultiplicationFirst	SAFE-In FB-Out	SAFEBOOL	FALSE	TRUE: When scaling, multiplication takes place after the optional negation. FALSE: When scaling, division takes place after the optional negation.
diScalingOffset	-	SAFEDINT	0	Offset that is added to the analog value Only a constant value is permitted (no variable).
tWatchdog	-	SAFETIME	t#0ms	Time (in ms) until the input value AnalogIn has to change. If the value is t#0ms, then the check is switched off.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bStuckAtError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE when the input value AnalogIn has not changed for the time tWatchdog.
AnalogOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Output with the scaled input signal Order processing - Negation AnalogIn - Multiplication scaling factor counter - Division scaling factor numerator - Addition scaling offset Permitted data Types: UINT, INT, UDINT, DINT. The data Type must be selected according to the input data Type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: - The output AnalogOut has an overflow or underflow. - A 32-bit overflow occurs for multiplication. - A 32-bit overflow occurs for division when rounding. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	No diagnostic information
1	The AnalogOut output has an underflow (is less than the smallest possible value)
2	The AnalogOut output has an overflow (is greater than the largest possible value)
3	A 32-bit overflow occurred during the multiplication.
4	A 32-bit overflow occurred during the division due to rounding.

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x40B0	The AnalogOut output has an underflow (is less than the smallest possible value)	FB number	AnalogIn	-
0x40B1	The AnalogOut output has an overflow (is greater than the largest possible value)	FB number	AnalogIn	-
0x40B2	A 32-bit overflow occurred during the multiplication.	FB number	AnalogIn	-
0x40B3	A 32-bit overflow occurred during the division due to rounding.	FB number	AnalogIn	-

STATE INFORMATION

Value	Description, parameter values
1	RUN In the RUN state the FB_SCALING module enters the scaling result in the AnalogOut output. The outputs assume the following values: bError=0 AnalogOut = result of the scaling bStuckAtError=StuckAtErrorDetected
2	STOP The FB_SCALING module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 AnalogOut=0 bStuckAtError=0
4	ERROR If the FB_SCALING module detects an error when checking the value range of AnalogOut, the FB_SCALING module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 AnalogOut=0 bStuckAtError=0
5	RESET The FB_SCALING module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 AnalogOut=0 bStuckAtError=0

10.3.10 FB_SLI

FB description

FB_SLI saves the input `Position` with rising edge at the input `bSLI` (`LatchPosition`). As long as `bSLI` is set to TRUE, the system checks that the position is located within the limits `LatchPosition - udiLimitNeg` and `LatchPosition + udiLimitPos`. If this is the case, then the output `bSLIActive` is set to TRUE. If the position leaves the defined range, then `bSLIActive` is set to FALSE. The output `PositionDiff` outputs the current difference between `Position` and `LatchPosition`. With the falling edge at `bSLIActive`, the output `PositionDiff` is also set to 0.

The inputs `udiLimitNeg` and `udiLimitPos` are UINT32 values and therefore they are always specified as positive.

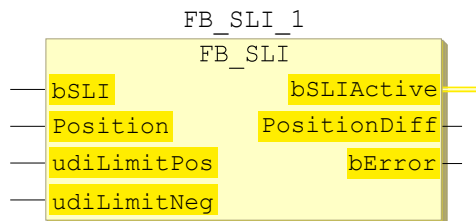


Fig. 131 FB_SLI function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bSLI	SAFE-In FB-Out	SAFEBOOL	FALSE	For a rising edge, the current value of the input <code>Position</code> is saved (<code>LatchPosition</code>) and the deviation is calculated to the limiting values.
Position	SAFE-In FB-Out	SAFEUDINT	0	Current position value Saved with the rising edge at <code>bSLI</code> . As long as <code>bSLI</code> remains set, it is compared with the saved position and the difference is output to <code>PositionDiff</code> . Permitted data Types: UINT, INT, UDINT, DINT
udiLimitPos	-	SAFEUDINT	0	Permitted deviation of the saved position (<code>LatchPosition</code>) in the positive direction Only a constant value is permitted (no variable).
udiLimitNeg	-	SAFEUDINT	0	Permitted deviation of the saved position (<code>LatchPosition</code>) in the negative direction Must be 0 if the data Type of <code>PositionDiff</code> is UINT or UDINT. Only a constant value is permitted (no variable).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bSLIActive	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE when <code>bSLIActive</code> is set and <code>PositionDiff</code> is within the defined limit.
PositionDiff	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Difference position between saved (<code>LatchPosition</code>) and current position If <code>bSLIActive</code> = FALSE, then <code>PositionDiff</code> is set to 0. Permitted data Types: UINT, INT, UDINT, DINT.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: <code>PositionDiff</code> is less than <code>udiLimitNeg</code> or greater than <code>udiLimitPos</code> . FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	no diagnostic information
1	Underflow ($\text{PositionDiff} < -\text{udiLimitNeg}$)
2	Overflow ($\text{PositionDiff} > \text{udiLimitPos}$)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x40C0	An underflow has occurred	FB instance	Position	LatchPosition
0x40C1	An overflow has occurred	FB instance	Position	LatchPosition

STATE INFORMATION

Value	Description, parameter values
1	RUN In RUN state the FB_SLI module determines the <code>PositionDiff</code> and checks whether $-\text{udiLimitNeg} \leq \text{PositionDiff} \leq \text{udiLimitPos}$ applies. The outputs assume the following values: <code>bError=0</code> <code>bSLIActive=1</code> <code>PositionDiff = Position - LatchPosition</code>
2	STOP The FB_SLI module assumes the STOP state if the input <code>FbRun</code> is FALSE. The outputs assume the following values: <code>bError=0</code> <code>bSLIActive=0</code> <code>PositionDiff = 0</code>
3	SAFE If the <code>bSLI</code> input is FALSE, the FB_SLI module assumes the SAFE state. The outputs assume the following values: <code>bError=0</code> <code>bSLIActive=0</code> <code>PositionDiff = 0</code>
4	ERROR If the FB_SLI module detects an error when checking the permitted range for the <code>PositionDiff</code> , the FB_SLI module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: <code>bError=1</code> <code>bSLIActive=0</code> <code>PositionDiff = 0</code>
5	RESET The FB_SLI module assumes the RESET state if no further error is pending after an error has occurred and the <code>ERR Ackn</code> input of the corresponding group is set to TRUE. The FB_SLI module should not exit the RESET state until <code>ERR Ackn</code> is FALSE and <code>bSLI</code> is FALSE. The outputs assume the following values: <code>bError=0</code> <code>bSLIActive=0</code> <code>PositionDiff = 0</code>

10.3.11

FB_SPEED

FB description

FB_SPEED saves the input `Position` and calculates a speed from it with the specified time interval.

The speed at the output is specified in increments for each `tTimeInterval`.

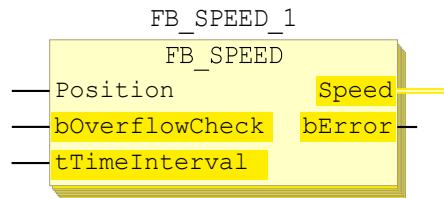


Fig. 132 FB_SPEED function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
Position	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Input channel for the calculation of speed Permitted data Types: UINT, INT, UDINT, DINT
bOverflowCheck	-	SAFEBOOL	FALSE	FALSE: For unsigned types, no overrun is checked. For type UINT, the value 0 follows the position 0xFFFF (and vice versa). For type UDINT, the value 0 follows the position 0xFFFFFFFF (and vice versa). TRUE: For unsigned types, overrun is checked. Only a constant value is permitted (no variable).
tTimeInterval	-	SAFETIME	t #0ms	Time interval for the calculation of speed.

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
Speed	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Output with the calculated speed Permitted data Types: UINT, INT, UDINT, DINT. The data Type must be selected according to the input data type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	TRUE: The output <code>Speed</code> has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	No diagnostic information
1	The <code>Speed</code> output has an underflow (is less than the smallest possible value)
2	The <code>Speed</code> output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4088	An underflow has occurred.	FB number	current position	latched position
0x4089	An overflow has occurred.	FB number	current position	latched position

STATE INFORMATION

Value	Description, parameter values
1	RUN In the RUN state the FB_SPEED module enters the result of the speed calculation in the Speed output. The outputs assume the following values: bError=0 Speed = calculated speed
2	STOP The FB_SPEED module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bError=0 Speed=0
4	ERROR If the FB_SPEED module detects an error when checking the value range of Speed, the module FB_SPEED switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: bError=1 Speed=0
5	RESET The FB_SPEED module assumes the RESET state if no further error is pending after an error has occurred and the ERR Ackn input of the corresponding group is set to TRUE. The outputs assume the following values: bError=0 Speed=0

10.3.12 FB_SUB

FB description

FB_SUB subtracts the input `AnalogIn2` from the input `AnalogIn1` and passes the result to the output `AnalogOut`.

Then the output `AnalogOut` is set to 0 and the output `bError` is set to 1. If the input `ERR Ackn` of the corresponding group is 1, then the RESET state is activated. When the input `ERR Ackn` of the corresponding group changes to 0 again, the RESET state is exited and the RUN state is entered. In RESET state, the output `AnalogOut` is 0 and the output `bError` is 0.

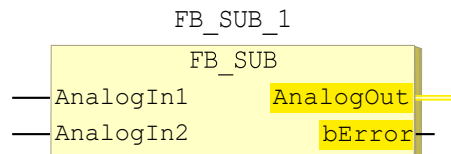


Fig. 133 FB_SUB function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogIn1	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	First analog signal for subtraction. Permitted data Types: UINT, INT, UDINT, DINT
AnalogIn2	SAFE-In FB-Out Standard-In	SAFEUDINT UDINT	0	Second analog signal for subtraction. Permitted data Types: UINT, INT, UDINT, DINT

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
AnalogOut	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Analog output that contains the result of subtraction. Permitted data Types: UINT, INT, UDINT, DINT The data Type must be selected according to the input data type.
bError	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	Replace with: TRUE: The output <code>AnalogOut</code> has an overflow or underflow. FALSE: No error has occurred.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	The <code>AnalogOut</code> output is correct
1	The <code>AnalogOut</code> output has an underflow (is less than the smallest possible value)
2	The <code>AnalogOut</code> output has an overflow (is greater than the largest possible value)

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
0x4080	An underflow has occurred	FB number	AnalogIn1	AnalogIn2
0x4081	An overflow has occurred	FB number	AnalogIn1	AnalogIn2

STATE INFORMATION

Value	Description, parameter values
0	not used
1	RUN The FB_SUB module cyclically subtracts the two analog inputs <i>AnalogIn1</i> and <i>AnalogIn2</i> . If no overflow or underflow occurs during the subtraction, the FB_SUB module is in the RUN state. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> = result of the subtraction
2	STOP The FB_SUB module assumes the STOP state if the input <i>FbRun</i> is FALSE. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> =0
3	Not used
4	ERROR If the FB_SUB module detects an error when checking the value range of <i>AnalogOut</i> during subtraction, the FB_SUB module switches to the ERROR state and transmits the corresponding Diag message to the GROUP module. The outputs assume the following values: <i>bError</i> =1 <i>AnalogOut</i> =0
5	RESET The FB_SUB module assumes the RESET state if no further error is pending after an error has occurred and the <i>ERR Ackn</i> input of the corresponding group is set to TRUE. The outputs assume the following values: <i>bError</i> =0 <i>AnalogOut</i> =0

10.3.13 FB_VIOLATIONCNT

FB description

FB_VIOLATIONCNT is used for implementing a weighted error counter. The counter is activated and deactivated by means of the input `bEnable`. If the counter is deactivated, then the output `bInputOK` is set to FALSE and the input `ActViolationCnt` is set to 0. If `bEnable` is set to TRUE, then the input `bInput` is checked each time the function block is called. If the input is TRUE (when `bInput` is negated: FALSE), then the error counter is decremented by the value `uiDecrementNoError`. If the input is FALSE (when `bInput` is negated: TRUE), then the error counter is incremented by the value `uiIncrementError`. As soon as the error counter reaches `udiCounterLimit`, `bInputOK` is set to FALSE.

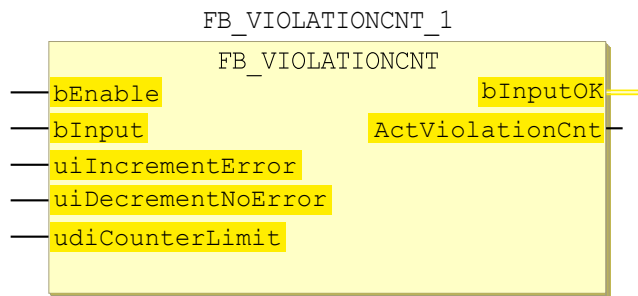


Fig. 134 FB_VIOLATIONCNT function block

VAR_INPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
<code>bEnable</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	Input for activating the function. TRUE: The internal counter is activated. FALSE: The internal counter is deactivated.
<code>bInput</code>	SAFE-In FB-Out	SAFEBOOL	FALSE	If the counter is activated, then the input is checked every time the function block is called. In addition, the error counter is incremented or decremented. Input not negated - TRUE: The counter is decremented by the value <code>uiDecrementNoError</code>. - FALSE: The counter is incremented by the value <code>uiIncrementError</code>. Input negated - TRUE: The counter is incremented by the value <code>uiIncrementError</code>. - FALSE: The counter is decremented by the value <code>uiDecrementNoError</code>.
<code>uiIncrementError</code>	-	SAFEUINT	0	Value by which the internal counter is incremented. If the input <code>bEnable</code> is TRUE and the input <code>bInput</code> is FALSE. Only a constant value is permitted (no variable).
<code>uiDecrementNoError</code>	-	SAFEUINT	0	Value by which the internal counter is decremented. If the input <code>bEnable</code> is TRUE and the input <code>bInput</code> is TRUE. Only a constant value is permitted (no variable).
<code>udiCounterLimit</code>	-	SAFEUDINT	0	Maximum value for the internal counter. Above this value, the output <code>bInputOK</code> is set to FALSE. Only a constant value is permitted (no variable).

VAR_OUTPUT

Name	Permitted Type	Data Type	Initial value	Description, parameter values
bInputOK	SAFE-Out FB-In Standard-Out	SAFEBOOL BOOL	FALSE	This output is set to TRUE when Enable is TRUE and the internal error counter is less than the counter limit.
ActViolationCnt	SAFE-Out FB-In Standard-Out	SAFEUDINT UDINT	0	Indicates the current internal counter state when the input bEnable is TRUE; otherwise the output is set to 0.

DIAGNOSTIC INFORMATION

Bit	Description, parameter values
0	no diagnostic information

DIAG MESSAGE

Text ID	Meaning	Parameter 1	Parameter 2	Parameter 3
-	-	-	-	-

STATE INFORMATION

Value	Description, parameter values
1	RUN If the input bEnable is TRUE and LimitOverrun is FALSE, the FB_VIOLATIONCNT module assumes the RUN state and modifies the ActViolationCnt value. The outputs assume the following values: bInputOK=1 ActViolationCnt = current value
2	STOP The FB_VIOLATIONCNT module assumes the STOP state if the input FbRun is FALSE. The outputs assume the following values: bInputOK=0 ActViolationCnt = 0
3	SAFE If the input bEnable is TRUE and LimitOverrun is TRUE, the FB_VIOLATIONCNT module assumes the SAFE state and modifies the ActViolationCnt value. The outputs assume the following values: bInputOK=0 ActViolationCnt = current value
6	START If the input bEnable is FALSE, the FB_VIOLATIONCNT module assumes the START state. The outputs assume the following values: bInputOK=0 ActViolationCnt = 0

Page deliberately empty

Summary

11	Error messages	221
----	----------------------	-----

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

11 Error messages

The error messages of the compiler for the "CODESYS Safety for EtherCAT Safety Module"-extension of the CODESYS programming system are listed in this section.

In the case of a safety application the user can manually initiate the safety-specific function for the programming system part of the error checks via the standard command *"Build"* (F11).

For this the safety application must be set as active application. Before the safety application is downloaded this process is executed automatically. In case of error messages may appear in the standard message window; it is possible to jump from an error code to its position in the source code.

Error number	Category	Error text	Explanation
[SELC0001]	Create	Assignment not permitted at input.	Depending on the function block type, the inputs can be linked with variables or fixed values can be specified (e.g. times). The direct linking of inputs with assignments is not permitted.
[SELC0002]	Create	Function block expected.	No type name or an invalid type name is specified. A function block must have a valid type name. The following type names are valid: ▪ FB_AND ▪ FB_CS ▪ FB_DECOUPLER ▪ FB_EDM ▪ FB_ESTOP ▪ FB_MODE ▪ FB_MON ▪ FB_MUTING ▪ FB_OR ▪ FB_SR ▪ FB_RS ▪ FB_TOF ▪ FB_TON ▪ FB_TWOHAND ▪ FB_ADD ▪ FB_COMPARE ▪ FB_COUNTER ▪ FB_DIV ▪ FB_ENVELOPE ▪ FB_LIMIT ▪ FB_LOADSENSING ▪ FB_MUL ▪ FB_SCALING ▪ FB_SLI ▪ FB_SPEED ▪ FB_SUB ▪ FB_VIOLATIONCNT
[SELC0003]	Create	Identifier '{0}' already defined.	The identifier has already been assigned in the project. A unique identifier must be used.
[SELC0004]	Create	Invalid COM ERR output.	The output of a group is linked with an invalid variable (not a safe output or a standard output).
[SELC0005]	Create	Invalid ERR Ackn input.	The input of a group is linked with an invalid variable (not a safe input or a standard input). The ERR Ackn input of a NUMSafe group is not linked.
[SELC0006]	Create	Invalid FB ERR output.	The output of a group is linked with an invalid variable (not a safe output or a standard output) or an unknown variable.
[SELC0008]	Create	Invalid identifier '{0}'.	The input of a function block is linked with an invalid variable (not a safe input or a standard input) or an unknown variable.
[SELC0010]	Create	Invalid input '{0}'.	The value at the input has the wrong data type.

Error number	Category	Error text	Explanation
[SELC0011]	Create	Invalid instance name '{0}'.	Wrong or missing instance name.
[SELC0012]	Create	Invalid number of connections.	The maximum number of FSoE connections in the project (128) was exceeded Only for CTMP6900: The maximum number of FSoE connections of a group (50) was exceeded.
[SELC0013]	Create	Invalid number of function blocks.	The number of function blocks in a safety application is incorrect (CTMP6900: 1...255, CTMP1960-2600: 1...512).
[SELC0014]	Create	Invalid number of groups.	The number of groups that can be configured for a safety application is incorrect (CTMP6900: 1...32, CTMP1960-2600: 1...64).
[SELC0015]	Create	Invalid number of output mappings.	There must be at least one output mapping. The maximum permitted number of output mapping data records (65535) that can be configured for a safety application was exceeded.
[SELC0016]	Warning	Invalid number of standard inputs.	
[SELC0017]	Warning	Invalid number of standard outputs.	
[SELC0020]	Create	Invalid OUT ERR output.	The output of a group is linked with an invalid variable (not a safe output or a standard output) or an unknown variable.
[SELC0022]	Create	Invalid RUN/STOP input.	The input of a group is linked with an invalid variable (not a safe input or a standard input) or an unknown variable.
[SELC0023]	Create	Invalid write access '{0}'.	The output of a function block is linked with an invalid variable (not a safe output or a standard output) or an unknown variable.
[SELC0024]	Create	Instance name '{0}' already exists.	The instance name of a function block within a safety POU is not unique.
[SELC0025]	Create	Input '{0}' is missing.	Certain inputs must be present. A required input must be used.
[SELC0026]	Create	Physical device is missing.	A physical device belonging to the logical device is missing.
[SELC0027]	Create	Negation not allowed for input '{0}'.	Certain inputs of some function blocks can be normally closed or normally open contacts. Whether an input should be a normally closed contact or a normally open contact can be defined for this input in the editor via the "Negation" command. A non-negated input is a normally closed contact and a negated input is a normally open contact. The error is output if an input is negated that can only be a normally closed contact, but not a normally open contact.
[SELC0028]	Create	No function block.	There must be at least one function block in a safety application in one of the safety POUs inserted under it.
[SELC0029]	Create	Output already mapped: {0} -> {1}.	There may not be any double output link with an safe output within a safety POU. If a safe output is used doubly, this is displayed as an error.
[SELC0030]	Create	Process image too big.	The total length of the local process image (CTMP6900: max. 2912 Bytes, CTMP1960-2600: max. 16384 Bytes) is exceeded.

Error number	Category	Error text	Explanation
[SELC0047]	Create	The logical device '{0}' is used in different groups ('{1}').	It is not allowed to use a connection in different groups.
[SELC0053]	Create	Logical exchange device '{0}' is not used in safety application and should be removed.	Logical exchange devices are permitted below the safety application only if they are in use. If no variable from the logical exchange device is used in the safety application, then the logical exchange device must be removed.
[SELC0054]	Create	Input '{0}' not supported by the device.	Only for CTMP6900 As of version 4.0.3.0 of the TcEL6900FBs library the function blocks FB_ESTOP, FB_MON and FB_MUTING provide the inputs tEDM1ResetTime and tEDM2ResetTime. These inputs are not supported by the CTMP6900, but only from the CTMP1960-2600. If the inputs are used with the CTMP6900, a compiler error is displayed.
[SELC0055]	Create	Only one of the inputs '{0}' can be set to TRUE.	For FB_COMPARE, there are the inputs b1oo2, b2oo3 and b3oo5. Only one of these inputs can be TRUE. If that is not the case the error is issued during the build. For FB_DIV and FB_SCALING, there are the inputs bFloor, bCeil and bRound. Only one of these inputs can be TRUE. If that is not the case the error is issued during the build. For FB_CAMMONITOR, there are the inputs bExcentricMode and bPendulumMode. Only one of these inputs can be TRUE. If that is not the case the error is issued during the build.
[SELC0056]	Create	At least one of the inputs '{0}' must be set to TRUE.	For FB_COMPARE, there are the inputs b1oo2, b2oo3 and b3oo5. At least one of these inputs must be TRUE. If that is not the case the error is issued during the build. For FB_DIV and FB_SCALING, there are the inputs bFloor, bCeil and bRound. At least one of these inputs must be TRUE. If that is not the case the error is issued during the build. For FB_CAMMONITOR, there are the inputs bExcentricMode and bPendulumMode. At least one of these inputs must be TRUE. If that is not the case the error is issued during the build.
[SELC0057]	Create	At least one of the inputs '{0}' must be set to TRUE.	For FB_EDM, at least one of the inputs tSwitchOnMonitoring and tSwitchOffMonitoring must be active. If that is not the case the error is issued during the build. For FB_TWOHAND, at least one of the inputs bTwoHand1 and bTwoHand2 and at least one of the inputs bTwoHand3 and bTwoHand4 must be active. If that is not the case the error is issued during the build.
[SELC0058]	Create	Invalid output '{0}'.	The function block output is connected to a variable of the wrong data typ (bool <-> analog)
[SELC0059]	Create	Value of input '{0}' is not in valid range ('{1}').	
[SELC0060]	Create	Value of the input '{0}' is not in valid range of the data type specified for '{1}'.	

Error number	Category	Error text	Explanation
[SELC0061]	Create	Value of input '{0}' exceeds the maximum value of the data type specified for '{1}'.	
[SELC0062]	Create	Input '{0}' has a {1}- bit data type, but is connected to a {2}-bit variable.	An analog input configured for a 16-bit data type must not be connected to a variable of 32-bit data type and vice versa. An analog input configured for a 16-bit data type can only be connected to a variable of 16-bit data type. An analog input configured for a 32-bit data type can only be connected to a variable of 32-bit data type.
[SELC0063]	Create	The data type of input '{0}' has to be the same as the data type of the connected function block output.	If an input of a function block is connected to an output of another function block, then both must be of the same data type.
[SELC0064]	Create	Output '{0}' has a {1}-bit data type, but is connected to a {2}-bit variable.	An analog output configured for a 16-bit data type must not be connected to a variable of 32-bit data type and vice versa. An analog output configured for a 16-bit data type can only be connected to a variable of 16-bit data type. An analog output configured for a 32-bit data type can only be connected to a variable of 32-bit data type.
[SELC0065]	Create	The data type of '{0}' has to be the same as the data type of '{1}'.	The inputs <code>MinValue</code> , <code>MaxValue</code> and <code>Analog</code> of the <code>FB_LIMIT</code> function block must be of the same data type. If this is not the case an error will be created when compiling.
[SELC0066]	Create	The {0}-values of the 'Value Settings' table do not match with the selected data type of input '{1}'.	For <code>FB_LOADSENSING</code> : The x values of the value table must be within the range of the data type which is configured for the <code>AnalogInX</code> input. The Y1, Y2, WY1, WY2 values of the value table must be within the range of the data type which is configured for the <code>AnalogInY</code> input. If that is not the case the error is issued during the build.
[SELC0067]	Warning	The X-values of the 'Value Settings' table must be written in ascending order.	For <code>FB_LOADSENSING</code> : The x values of the value table must be in ascending order. If that is not the case the error is issued during the build.
[SELC0068]	Warning	The Y-values of the 'Value Settings' table must meet the condition $Y1[i] \leq WY1[i] < WY2[i] \leq Y2[i]$	For <code>FB_LOADSENSING</code> : If <code>bOutsideY1Y2 == FALSE</code> , then the Y values of the value table must meet the condition „ $Y1[i] \leq WY1[i] < WY2[i] \leq Y2[i]$ “. If that is not the case the error is issued during the build.
[SELC0069]	Create	The Y-values of the 'Value Settings' table must meet the condition $WY1[i] \leq Y1[i] < Y2[i] \leq WY2[i]$	For <code>FB_LOADSENSING</code> : If <code>bOutsideY1Y2 == TRUE</code> , then the Y values of the value table must meet the condition „ $WY1[i] \leq Y1[i] < Y2[i] \leq WY2[i]$ “. If that is not the case the error is issued during the build.
[SELC0070]	Create	Inputs '{0}' have to be connected (operation mode '{1}').	For <code>FB_COMPARE</code> : If <code>b1oo2 == TRUE</code> , then <code>CompIn1</code> and <code>CompIn2</code> must be connected. If <code>b2oo3 == TRUE</code> , then <code>CompIn1</code> , <code>CompIn2</code> and <code>CompIn3</code> must be connected. If <code>b3oo5 == TRUE</code> , then <code>CompIn1</code> , <code>CompIn2</code> , <code>CompIn3</code> , <code>CompIn4</code> and <code>CompIn5</code> must be connected. If that is not the case the error is issued during the build.

Error number	Category	Error text	Explanation
[SELC0071]	Create	Input '{0}' is connected, but it is not activated (operation mode '{1}').	For FB_COMPARE: The error is issued, if • b1oo2 == TRUE and CompIn3, CompIn4 or CompIn5 is connected • b2oo3 == TRUE and CompIn4 or CompIn5 is connected
[SELC0072]	Create	Value of '{0}' does not meet the condition '{1}'.	For FB_CAMMONITOR there are different conditions for the configured values. If one of these conditions is not fulfilled the error is issued during the build.
[SELC0073]	Create	All analog inputs and outputs must have the same data type.	For FB_CAMMONITOR there must be the same data type for all analog inputs and outputs.
[SELC0074]	Create	Input '{0}' must not be used in pendulum mode.	For FB_CAMMONITOR in pendulum mode the inputs OverrunMax, bBackwardsMove, bOverrunCAM and bUpwardCAM must not be connected.
[SELC0075]	Create	Input '{0}' must not be used in excentric mode.	For FB_CAMMONITOR in excentric mode the inputs TDC2LowerLimit and TDC2UpperLimit must not be connected.
[SELC0076]	Create	Output '{0}' must be used in excentric mode.	For FB_CAMMONITOR in excentric mode the output OverrunTDC must be connected
[SELC0077]	Create	Output '{0}' must be used if the input '{1}' is used.	For FB_CAMMONITOR in excentric mode the output OverrunStop must be connected if the input bPress-Started is connected.
[SELC0078]	Create	A constant value must be set at input '{0}' if the input '{1}' is used.	For FB_CAMMONITOR constant values must be defined for the input TDC1LowerLimit, TDC1UpperLimit, TDC2LowerLimit, TDC2UpperLimit, BDCLowerLimit, BDCUpperLimit and OverrunMax, no assignment to variables if the input bSettingMode is connected.
[SELC0080]	Create	Output '{0}' must be used.	Certain outputs must be present. A required output must be used.
[SELC0081]	Create	At least two inputs of the function block must be connected	For FB_AND and FB_OR at least two inputs must be connected
[SELC0082]	Create	This device does not support replacement values	Only for CTMP6900
[SELC0084]	Create	In a passivatable group, only one connection may be defined.	A POU whose properties allow passivation may only access I/O channels of a single field device.
[SELC0085]	Create	This group is not deactivatable. The configured replacement values are ignored.	Replacement values have been defined for signals in a POU, but deactivation is not enabled in the properties of this POU.
[SELC0086]	Create	A replacement value for an output must be defined no more than once.	Several substitute values are defined for one output in a safety POU. However, only one substitute value is permitted.
[SELC0087]	Create	Replacement value does not match the data type of output '{0}'.	The data type of the replacement value must be compatible with the data type of the output.

Page deliberately empty

Summary

12	Glossary and Certificates	229
12.1	Glossary	229

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

12 Glossary and Certificates

12.1 Glossary

API	Protocol-independent application interface. With the aid of API the developer obtains access at the programming level to each safe I/O module at the safety communication level.
Application	Executable unit on a controller made up of program code and associated I/O configuration.
CODESYS	(Controller Development System) programming system from 3S-Smart Software Solutions for the programming of PLCs. Available at present in the second generation and bearing the version designation V3.x.
Component	Software components
Configuration	Description of the system made up of controllers and I/O modules (both standard and fail-safe); definition of the project structure.
Controller	PLC
Gateway	Other name for slave on the fieldbus; connects individual I/O modules with the fieldbus.
Data types	Types
Editor	Component of the PS for editing program sections.
Exchange variables	Variables whose values can be exchanged between a safety application (on a safety controller) and one or more standard applications (on connected standard controllers).
Execution version	Identifier by means of which the checked compatibility between (accepted) boot application and RTS core is controlled. For compatibility with external POU's, see Implementation version . From the user's point of view the execution version corresponds to the compiler version from CODESYS. A changed version number means that something can change in the processing of the application.
F-	Prefix. Short for fail-safe
Fail-safe	Ability of a system to remain in, or to immediately enter a safe state on the occurrence of a failure.
FB	Function block
FBD	Function block diagram
Fieldbus device	Device that implements a fieldbus protocol (e.g. Profibus DP or Profinet) via which the process image is exchanged. Of particular relevance are: - gateway with subordinated I/O modules - drives, i.e. I/O modules that are typically autonomous fieldbus devices.
Field devices	Fieldbus device
F-parameters	Safety-related configuration data in case of PROFIsafe, also called safety protocol parameter .
FSOE	Fail Safe over EtherCAT.
Function block	POU according to IEC 61131-3, can call other function blocks and be called by an application and other function blocks.
Function block diagram	A program/function written in the IEC 61131 function block language (FBD). The language is permissible for safety programming.

I/O module	Hardware module with input channels and/or output channels. Distinction is made between: - I/O modules (also called terminals or "slices") behind a gateway that is a fieldbus device. The input or output channels of each terminal behind the gateway correspond to a section in the process image of the gateway. - I/O modules that are themselves fieldbus devices, e.g. the typical drive. In this case the input or output channels of the I/O module form the entire process image of the fieldbus device.
IEC 61131-3	International standard for programmable logic controllers.
Library manager	Object for the management of the integrated libraries in CODESYS.
Library repository	Data repository of CODESYS, in which libraries are stored with version.
Master	In general, hardware that controls other hardware (slave). In this case, in connection with fieldbus systems, the control of the bus.
Object	A component of the project structure that represents a structuring unit of the project.
Parameterization	Configuration of the parameters of the bus system and the I/ O modules.
PLC	Programmable Logic Controller (PLC).
PLC	Programmable logic controller.
POU	Program Organization Unit (POU)
Program	POU according to IEC 61131-3
Programming system	Development environment on a Windows PC for the creation of programs according to IEC 61131-3 as well as software access to controllers with CODESYS RTS.
Project	A project can contain several controllers with their subordinate objects. These can be safety and/or standard controllers that control a plant or a machine. Stored in CODESYS V3.x in the project file .
PS	Programming system.
RTS	Runtime system.
Safe I/O module	An I/O module that implements a safe I/O protocol and can be used in the safety programming. In a safety application the utilisable data of the I/O channels of safe I/O modules are of one of the SAFExxx data types.
Safety application	Fail-safe application for execution on a fail-safe controller.
Safety controller	A PLC that (usually) offers increased fail-safety through special hardware extensions (e.g. two processors with opposite-acting process observation) and accordingly adapted software.
Safety devices	Fail-safe application for execution on a fail-safe controller.
SAFExxx	Family of data types with the prefix SAFE e.g. SAFEBOOL.
SDD	Safety Devicefacet Description. CRC-secured device description for safety-related device information.
Slave	Hardware that is subordinate to a master by which it is controlled or supplied with data.
Source control	External program for the administration of object data outside of CODESYS. In addition to the actual source data visible to the user, these can also contain data for the administration of these sources within CODESYS. Synonymously used with the term 'version administration'.
Standard CODESYS	CODESYS without additions, as delivered. From a technical point of view it is a CODESYS instance into whose profile the safety extension in particular is NOT loaded.

SW	Software
Terminal	Other name for I/O modules behind a gateway.
Types	Distinction is made between the following sub-types in the safety programming: - Boolean types: BOOL, SAFEBOOL (for truth and bit values) - Bit access types: SAFEBYTE - Arithmetic types: SAFETIME (for operands of arithmetic operations).
User	(from "CODESYS Safety for EtherCAT Safety Module"): developer who works on the safety applications and/or executes them on safety controllers.
User Interface	Interface between humans and software. In particular the graphic user interface between the programming system and the user . To be distinguished from the programming interface .
Validation	Check whether the completely generated and verified product meets the demands placed on it.
Verification	Check whether the created documents and software (functions, modules and components) satisfy their specifications (or their superordinate specifications).

1

2

3

4

5

6

7

8

9

10

11

12

Page deliberately empty

Published by NUM Spa

Via F. Somma 62
20012 Cuggiono (MI) (Italy)
Tel. : +39 02 979 691
website: www.num.com

Printed April 2020