CODESYS Safety for EtherCAT Safety Module

CODESYS® Safety

Read this manual thoroughly before you start working with CODESYS Safety.

3S-Smart Software Solutions GmbH
Memminger Str. 151
87439 Kempten
Deutschland
Telephone: +49-831-54031-0
Fax: +49-831-54031-50
E-mail: info@codesys.com
Internet: www.codesys.com
Dokument Version 13.0
Tool-Version 4.1.0.0

© 2020

Table of contents

1 Introduction **5**

2 Standards **7**

3 Requirements **11**

3.1 System Requirements 11

3.2 Qualified Staff 11

3.3 Installation 11

4 Software Development with "CODESYS Safety for EtherCAT Safety Module" **13**

4.1 Operating Concept 13

4.2 Programming Guidelines 16

4.2.1 Rules for Identifiers of Safety Objects 17

4.2.2 Automatically Checked Programming Guidelines 18

4.3 Setting Up a Safety Project 18

4.3.1 Preparation of Planned Devices 18

4.3.2 Setting Up the Safety Controller and Safety Application 19

4.3.3 Setting Up User Management in the Project 20

4.3.4 Setting Up a User Management on the EtherCAT Safety Module 20

4.3.5 Access Protection when Linking to the Source Code Management 20

4.4 Division of the Application into Groups 20

4.4.1 Configuration of Startup and Acknowledgement 21

4.4.2 Preparing for Deactivation 21

4.5 Programming the Application Logic 21

4.5.1 Difference to Programming in Standard CODESYS 21

4.5.2 POUs 22

4.5.3 Variables 22

4.5.4 Networks 23

4.6 Integration of Field Devices 27

4.6.1 Setting Up the Devices 27

4.6.2 Access to Input and Output Signals 28

4.6.3 EtherCAT Safety SC Modules 28

5 Online Mode **31**

5.1 Connecting to the EtherCAT Safety Module 31

5.2 Project Download 31

5.3 Preparing for Device Replacement 33

5.4 Project Upload 34

5.5 Project Delete 34

5.6 Status Display 35

5.7 Monitoring 35

6 Pinning **37**

Table of contents

7 Verification and Acceptance **43**
7.1 Verification and Acceptance 43
7.2 Archiving 43
7.3 Printing of Acceptance Documentation 45
8 Software Update **47**
8.1 Overview of Version Control 47
8.2 Updating the Device Version 47
8.3 Changing a Library Version 48
8.4 Updating the CODESYS Version 49
8.5 Change Request / Bugfix Process 49
9 Glossary **51**
10 Index **55**

1 Introduction

"CODESYS Safety for EtherCAT Safety Module" is software to program EtherCAT Safety modules from Beckhoff. This manual is part of the "CODESYS Safety for EtherCAT Safety Module" product package and should be read unconditionally before creating a safety application. The manual contains a description of functions in "CODESYS Safety for EtherCAT Safety Module" as well as notices that must be followed unconditionally when creating an application. The following types of safety notices are used:

CAUTION!
Disregard of these safety instructions may lead to personal injury.

NOTICE!
Disregard of these instructions may lead to errors and problems in the development and verification of the safety project.

This symbol indicates information for improved understanding.

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

This manual describes

- How to manage a safety application
- How to program the POUs
- How to configure and use safe I/O modules
- How to protect an application against unauthorized access
- How to version control an applications
- How to configure data exchange between a safety application and a standard application
- How to download a safety application to an EtherCAT Safety module
- How to verify a safety application and prepare it for acceptance

Part of this manual is the exact description of the language subset that is available for the creation of a safety application This contains all data types and all available function blocks. The language subset corresponds to the subset of the EtherCAT Safety modules EL6900, EL6910, and EK1960 from Beckhoff. The available data types are described in chapter ➪ *Chapter 4.5.3.1 “Data Types” on page 23*, and all function blocks provided by the system are listed in the CODESYS Help.

The prerequisite for understanding the manual is good knowledge of CODESYS and knowledge of IEC 61131-3.

We wish you an enjoyable time reading the manual.

2 Standards

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

Valid safety standards

Machine manufacturers and operators of technical plants are responsible for ensuring that machinery or plants meet the requirements for health and safety at work. When the respective relevant standards are respected, this ensures that the machinery 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 machinery directive defines a uniform safety level for the prevention of accidents for machinery brought onto the market within the European Economic Area (EEA) and in Switzerland and Turkey.

The device manufacturer is responsible for providing evidence of compliance with the applicable requirements. The proof can be provided by compliance with the respective European standards (EN). These standards ("harmonized 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 harmonized standards ("product standards"): For the functional part of the safety of individual machinery types (machine tools, woodworking machines, printing machines, packaging machines, etc.)

These standards have the absolute higher priority for the machinery manufacturer. By complying with these standards, it can be assumed that the fundamental safety requirements of the machinery directive are fulfilled.

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

Standards

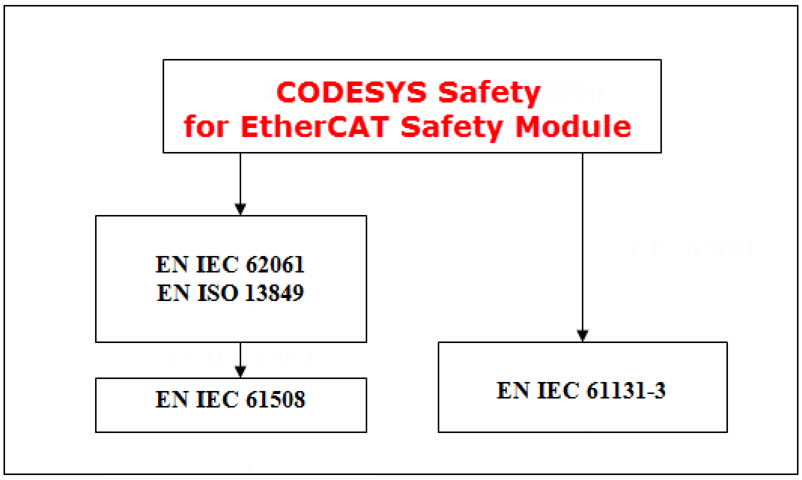


Fig. 1: "CODESYS Safety for EtherCAT Safety Module": Overview of safety standards

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 electrical/electronic/programmable electronic safety-related control systems
EN ISO 13849-1:2015	Safety of machinery – Safety-related parts of control systems
EN IEC 61131-3:2003	Programmable logic 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 machinery.

EN ISO 13849 defines various reliability levels for this, called PL-a to PL-e.

EN IEC 62061 refers to EN IEC 61508, the basic safety standard where the reliability levels SIL1 to SIL4 are defined.

The IEC 61508 standard describes the certification of electrical, electronic, and programmable electronic systems (short form: E/E/PES). 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 level systems correspond to each other approximately as follows:

Table 1: Correlation of PL to SIL

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

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

Demands on software development

In order for the programmed safety function to reach the required reliability level, the development of the control software already has to meet specific requirements.

On the whole, EN IEC 61508 defines the cycle of the overall system with 16 phases, starting with Phase 1: "Concept" and ending with Phase 16: "Decommissioning". "CODESYS Safety for EtherCAT Safety Module" is used as a tool in EN IEC 61508 Phase 10 "Realization, 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.

By using a programming language with a limited language subset (Limited Variability Language; abbreviated: LVL), the requirements of the standard are reduced. The reason is that these languages allow for clearer expression, easier understandability, and easier verification of the program logic. 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 graphical 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 safe aspects from the non-safe aspects

In order to make a clear distinction between safety-relevant signals and standard signals, a new data type was 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 type cannot guarantee that the signal status is safe (for example, if the peripherals are incorrectly wired), it is nevertheless an organizational tool for the minimization 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 case of a violation, an application cannot be downloaded to the EtherCAT Safety module. As a result, compliance with the rules is proven automatically.

IEC 61131-3

IEC 61131-3 describes the programming languages that are used to program controllers. It defines FBD (the language with restricted language subset).

3 Requirements

3.1 System Requirements

The following system requirements apply as a minimum configuration for smaller "CODESYS Safety for EtherCAT Safety Module" projects with max. 100 POUs, max. 10 visualizations, and max. 8 fieldbus devices:

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

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

- Windows 7, 64-Bit
- Windows 10, 64-Bit

3.2 Qualified Staff

A prerequisite for the application of safety-related products is adequate qualification.

For example, the staff who plan, create, and commission safety applications with this product need to meet the following requirements:

- Confidence with respect to the applicable standards and regulations
- Confidence with respect to the relevant safety concepts in automation technology
- Knowledge of accident prevention regulations and any special operating regulations
- Adequate language skills to understand this manual
- Very good knowledge of CODESYS programming

3.3 Installation

The "CODESYS Safety for EtherCAT Safety Module" package is installed with the help of the Package Manager, which is started in CODESYS from the "Tools" menu. The main dialog of the Package Manager provides an overview of the installed packages with their version and installation date.

You need administrative permissions on your computer in order to install and uninstall packages.

Before the installation, check that the "CODESYS Safety for EtherCAT Safety Module" version is released with your CODESYS version. An overview of the released combinations can be found in the Store, on the "All versions" tab, in the "Compatibility" column.

Also check whether there are any known safety-related errors in the version you are using. If there are, then check whether these have a safety-related effects on your CODESYSapplication. For more information about whether and which safety-related errors are known, see the Store, on the "All versions" tab, in the "Release Notes" column.

4 Software Development with "CODESYS Safety for EtherCAT Safety Module"

4.1 Operating Concept

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

- Promoting good programming techniques
- Prohibiting unsafe language features
- Promoting code comprehensibility
- Facilitated testability
- Code documentation procedure

A project is created and the user interface is used according to the operating concept of standard CODESYS. However, some commands of standard CODESYS are not available. Normally, these are visible, but they cannot be activated. The user interface and the handling of the "CODESYS Safety for EtherCAT Safety Module" FBD editor correspond to standard CODESYS.

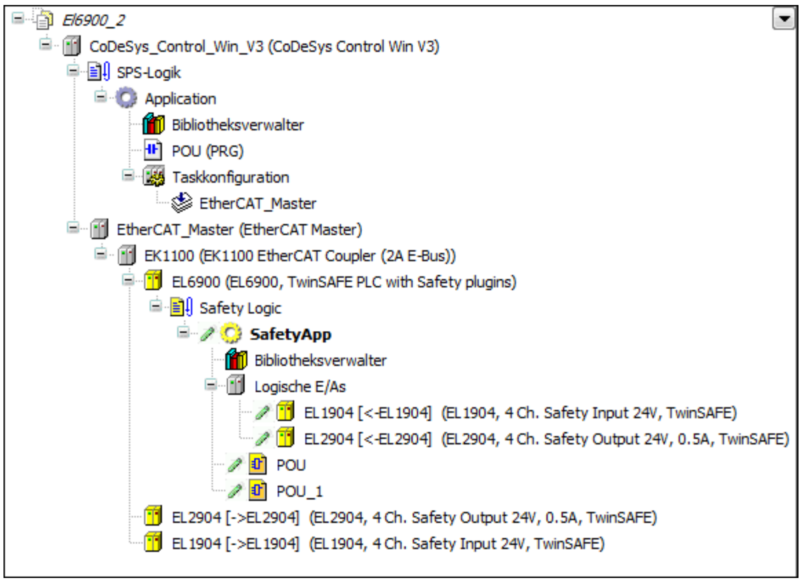
A safety application is programmed in POUs. In the POUs, the program code is implemented in the IEC 61131-3 language FBD (function block diagram). FBD is characterized by its clarity, easy recognition of programming errors, and straightforward data flow.

The operation of "CODESYS Safety for EtherCAT Safety Module" for adding safety applications to an entire project and for creating and editing contents is described in the help for CODESYS Safety.

Ranking of an EtherCAT Safety module in the project tree

The EtherCAT Safety module is displayed in the project tree below an EtherCAT bus coupler (for EL6900 and EL6910) or below an EtherCAT master (for EK1960).

Exactly one application (here: SafetyApp) is located below the EtherCAT Safety module. Below that are all safety objects that belong to a project.



```
El6900_2
  CoDeSys_Control_Win_V3 (CoDeSys Control Win V3)
    SPS-Logik
      Application
        Bibliotheksverwalter
        POU (PRG)
        Taskkonfiguration
          EtherCAT_Master
    EtherCAT_Master (EtherCAT Master)
      EK1100 (EK1100 EtherCAT Coupler (2A E-Bus))
        EL6900 (EL6900, TwinSAFE PLC with Safety plugins)
          Safety Logic
            SafetyApp
              Bibliotheksverwalter
              Logische E/As
                EL1904 [<-EL1904]  (EL1904, 4 Ch. Safety Input 24V, TwinSAFE)
                EL2904 [<-EL2904]  (EL2904, 4 Ch. Safety Output 24V, 0.5A, TwinSAFE)
              POU
              POU_1
        EL2904 [->EL2904]  (EL2904, 4 Ch. Safety Output 24V, 0.5A, TwinSAFE)
        EL1904 [->EL1904]  (EL1904, 4 Ch. Safety Input 24V, TwinSAFE)
```

Fig. 2: Example of a project tree with a standard application and a safety application

For the part of the project tree that contains the objects of the standard application, see the CODESYS Help.

Language subset of safety programming with CODESYS

Safety libraries with pre-certified function blocks are provided to the developer for programming with "CODESYS Safety for EtherCAT Safety Module". The functions can be configured in part by means of additional parameter inputs. The inputs and outputs of a function block can be the inputs and outputs of the local process image, but also the outputs of function blocks can be linked with inputs of other function blocks.

The function blocks were developed by Beckhoff and are included in the firmware of the EtherCAT Safety module.

These POUs are described in two libraries which are part of the "CODESYS Safety for EtherCAT Safety Module" product:

- *“TcEL6900FBs ”*
- *“TcEL69xxAnalogFBs ”*

These libraries are automatically part of the safety applications. In the case of the EL6900 controller, only the library TcEL6900FBs, as this device does not support the other FBs.

These libraries are managed as in standard CODESYS.

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 execution-relevant statuses have to be verified in the comparison view (see *Chapter 6 “Pinning” on page 37*). For acceptance documentation, see *Chapter 7.3 “Printing of Acceptance Documentation” on page 45*.

For a description if the libraries TcEL6900FBs and TcEL69xxAnalogFBs, see the help for CODESYS Safety.

Highlighting of safety structures

The safety objects in the project structure and the safety editors are clearly emphasized 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 in yellow. This supports and facilitates the tasks for the development, verification, and acceptance of the safety application.

Change markings in the implementation part of the editor

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

- Green: Recently 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.

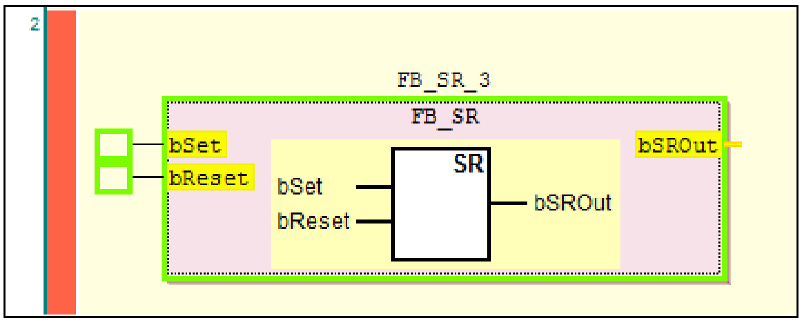


Fig. 3: Example of change marker: Recently added POU call

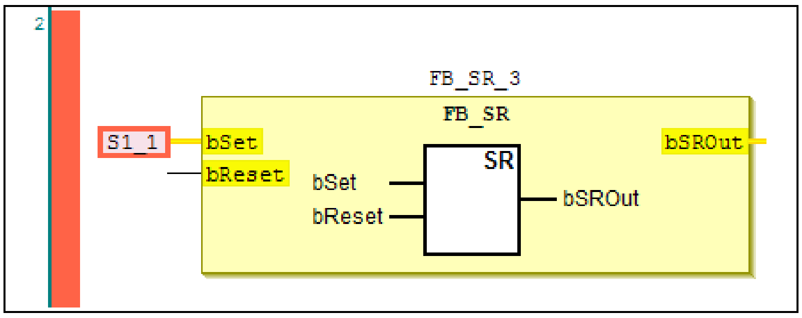


Fig. 4: Example of change marker: Input mapped to S1_1

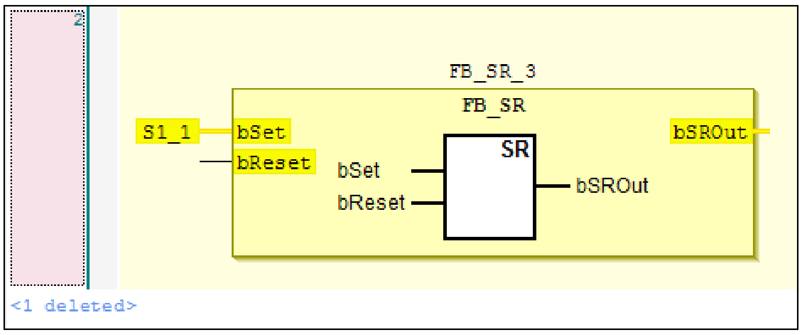


Fig. 5: Example of change marker: Deleted network

4.2 Programming Guidelines

Regardless of the standard on which the certification is based, compliance with the following guidelines is recommended:

- The documentation has to be complete, available, legible, and understandable.
- All changes over all life cycles have to be documented.
- The project documentation has to include the following:
  - Legal entity (company)
  - Functional description of the requirements of the project
  - I/O description
  - Version of the used function block library
- Every POU has to 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

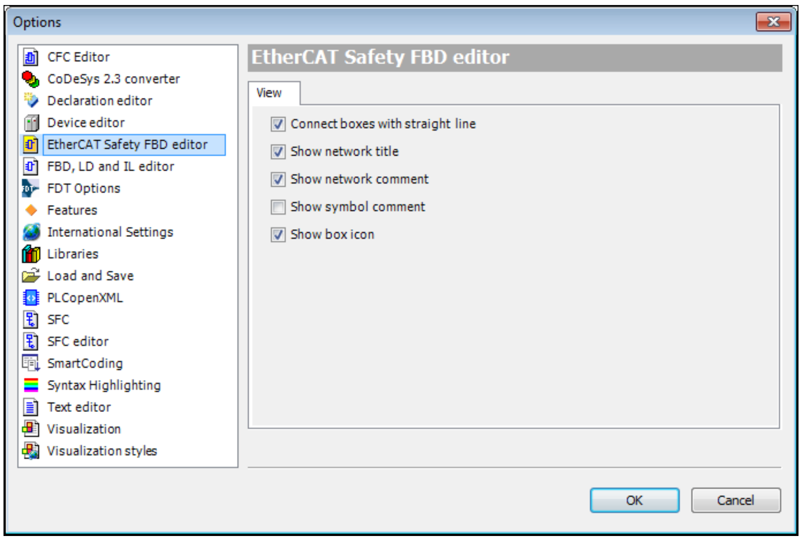
Commenting

"CODESYS Safety for EtherCAT Safety Module" provides 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 required by the standard can be used 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 for each FBD network (optional)
- Network title (optional)

The display of the network title and comment of networks and variables can be set as an option (*“Options”* dialog in the *“Tools”* menu).

NOTICE!
It is recommended to display network titles and network comments.



[✓] Connect boxes with straight line
[✓] Show network title
[✓] Show network comment
[ ] Show symbol comment
[✓] Show box icon

Fig. 6: Settings to display network comments and titles

4.2.1 Rules for Identifiers of Safety Objects

Rules for identifiers

- Names for POUs, I/O modules, variables, networks, etc. are identifiers as defined in IEC 61131-3. Therefore, they have to be in the following form: a sequence of letters, numbers, and underscores, where the first character may not be a number and the last character may not be an underscore, and two underscores may not occur next to each other.
- Identifiers that begin with an underscore are reserved for implicit code and external FBs and must not occur in POUs.

- The IEC keywords listed in IEC 61131-3 (2003-1, Programmable controllers, Part 3: Programming languages - Annex C) must not be used as identifiers.
- The keywords extending beyond the IEC standard that are reserved by standard CODESYS must not be used as identifiers.
- The following keywords are reserved as "CODESYS Safety for EtherCAT Safety Module"-specific and must 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 for external FBs only.

- Among the objects that belong to the safety application POUs, I/O modules, and library function blocks, no two are permitted to have the same name.

4.2.2 Automatically Checked Programming Guidelines

Linkage rules for SAFEBOOL data and analogous linkage rules for the other SAFExxx data types

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

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

- An assignment to output variables is permitted only at one point in the application.

4.3 Setting Up a Safety Project

4.3.1 Preparation of Planned Devices

Only controllers and field devices that are recognized in CODESYS can be programmed and configured. For programming the machinery, corresponding device descriptions have to be installed for all controllers and field devices, and the respective libraries if possible.

NOTICE!
If device descriptions were installed with standard CODESYS without a safety extension, then they cannot be used in "CODESYS Safety for EtherCAT Safety Module". These respective device descriptions have to be reinstalled after the installation of "CODESYS Safety for EtherCAT Safety Module" in order for them to be used.

4.3.2 Setting Up the Safety Controller and Safety Application

Adding the EtherCAT Safety module

The EtherCAT Safety modules EL6900 and EL6910 are inserted below an EtherCAT bus coupler (example: EK1100). The EtherCAT Safety module EK1960 is inserted below the EtherCAT master. This is done by selecting the bus coupler or master and executing the *“Insert Device”* context menu command with selection of the EtherCAT Safety module. The module can also be inserted below the bus coupler or master by means of the "Scan for Devices" context menu command.

After the insertion of the EtherCAT Safety module, 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 EtherCAT Safety module can be updated to a newer version of the device description by means of the *“Update Device”* context menu command. Libraries may be replaced by newer versions.

Structure of the project tree of the safety application

The project tree is displayed in the device window (*“Devices”* view of the *“View”* menu).

The objects relevant for the EtherCAT Safety module and its programming are located below the EtherCAT Safety module. Directly below them is always the symbolic node point *“Safety Logic”*. Below each *“Safety Logic”*, there can only ever be **one** safety application object *“SafetyApp”* (default name), which can contain the following safety objects:

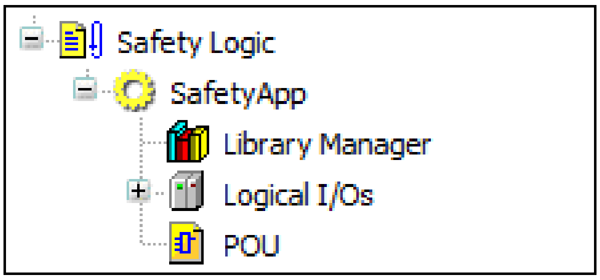


Fig. 7: Example: Safety application object with objects

The following objects have to exist exactly **one time** in a safety application object:

- Library Manager
- Logical I/Os

4.3.3 Setting Up User Management in the Project

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

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

NOTICE!
It is highly recommended to configure a user management with secure access protection for each project.

Without user management in the project, every person who collaborates in the project has all rights to the standard and safety applications in the project. The project manager is responsible for making sure that the safety application is modified by qualified persons only.

4.3.4 Setting Up a User Management on the EtherCAT Safety Module

In addition to the standard user management of CODESYS, the EtherCAT Safety module also provides its own user management. This allows users to be set up with corresponding passwords. This data has to be specified during "Project Download" and "Delete Project".

4.3.5 Access Protection when Linking to the Source Code Management

NOTICE!
When a source code management is used, a user management that corresponds to the user management in the project also has to be configured in the source code management.

4.4 Division of the Application into Groups

For "CODESYS Safety for EtherCAT Safety Module", a group is understood as a POU running in the safety module with its respective logical devices.

4.4.1 Configuration of Startup and Acknowledgement

Each group has its own status and can be started and stopped by the standard application. Errors have to be acknowledged. To do this, the group I/Os have to be named in the object properties of the POU and exchanged with the standard application.

For more information, see the help for CODESYS Safety .

4.4.2 Preparing for Deactivation

The EtherCAT Safety modules EL6910 and EK1960 provides the capability of deactivating individual groups by means of an online command. This means a POU, which is running in a safety module, with its respective logical devices.

As long as a group is deactivated in the safety module:

- Its POU does not run and also cannot be started by RUN.
- Its safe field devices are in fail-safe state.
- Other POUs get substitute values when they read outputs of FB instances of the POU.

Group deactivation in three steps

1. Configuration of the deactivation methods for a group

   The configuration is performed in the properties of the POU. Deactivation can be permanent or temporary.

2. Configuration of the substitute values for the case of deactivation of the group

3. Online command for deactivation or reactivation of deactivatable groups.

For more details about the deactivation of groups, see the help for CODESYS Safety.

4.5 Programming the Application Logic

4.5.1 Difference to Programming in Standard CODESYS

Basic functions of the FBD editor

The following FBD operators which are supported in standard CODESYS are not supported in "CODESYS Safety for EtherCAT Safety Module":

- Standard function blocks: Functionality partially replaced by predefined function blocks. For example, FB_TON from the library TcEL6900FBs
- Operators (standard functions): Functionality replaced by predefined function blocks. For example, FB_AND from the library TcEL6900FBs
- Type conversions, for example INT_TO_WORD
- Edge recognition with assignments (to inputs and variables)
- Set/Reset
- EN/ENO parameters

For detailed and basic information how to use the FBD editor, see the CODESYS Help.

4.5.2 POUs

The program code of a safety application is created in a POU (Program Organization Unit). (For details about adding a new POU see the CODESYS Help.)

The editor of a POU consists of an implementation view where the program code is created.

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

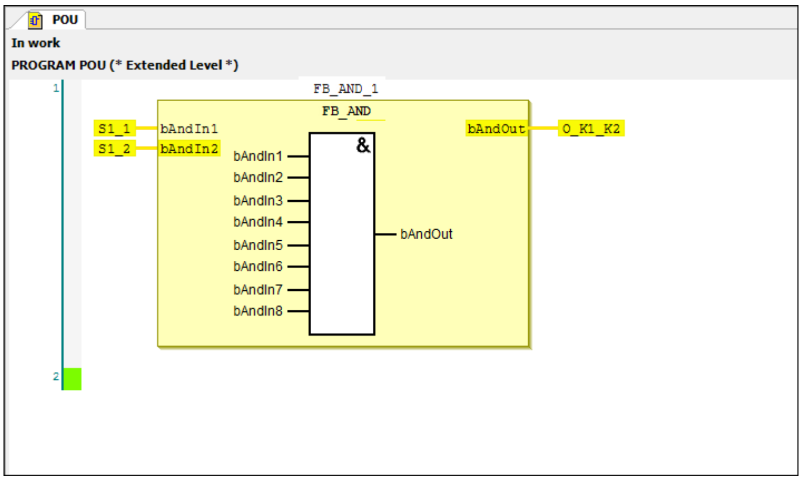


Fig. 8: Example: POU editor: Implementation part with one network

For details about the POU editor, see the CODESYS Help.

Various operating functions are available to the developer for the activation of commands and programming in the FBD. In this manual, the operation from the context menu is usually described for procedures and concrete examples.

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

4.5.3 Variables

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

4.5.3.1 Data Types

The following data types are available to program the EtherCAT Safety modules:

Table 2: IEC standard data types

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} ... $2^{63}-1$

Table 3: Safety-oriented 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} ... $2^{63}-1$
SAFETIME	32	0 ... 2,147,483.647 s

4.5.4 Networks

4.5.4.1 Overview of Networks

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

In a network, the program code has to be in a tree structure, which 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 (short form: tree) with the root to the right; in which the boxes in this tree act as nodes. Exception: Multiple assignments as well as jump and return statements 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.

For detailed and basic information how to use the FBD editor, see the CODESYS Help.

By activating the respective options, a title, a comment, and the comments from the declarations of variables can be displayed in a network. For more information, see the help for CODESYS Safety.

4.5.4.2 Data Flow and Assignment

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

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

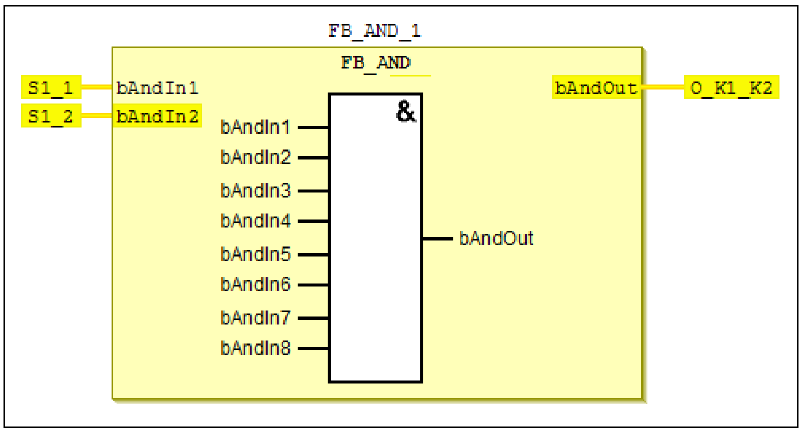


Fig. 9: Example of safe data flow: FB_AND with SAFE variables: `S1_1`, `S1_2`, and `O_K1_K2`

An assignment is an FBD element that takes up the incoming signal flow in a network and stores it in an operand (meaning that is write it to the variable.) A variable is always required for an assignment. Assignments can be inserted only at the output of a box.

Assignments are inserted as in the standard FBD of CODESYS.

Examples of assignments

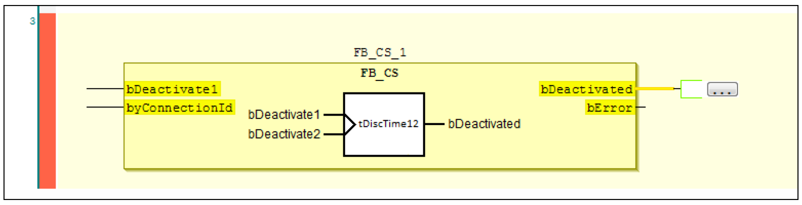


Fig. 10: Example: Assignment at the output "bDeactivated" of FB_CS

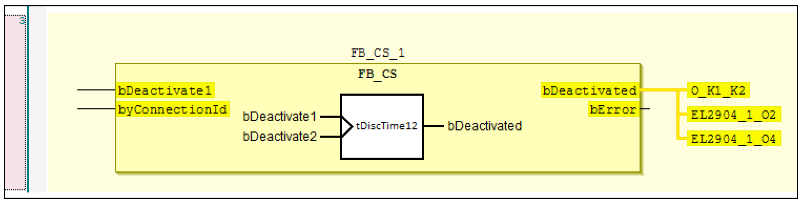


Fig. 11: Example: Multiple assignment by inserting a new assignment at the existing assignment

An assignment can be inserted at the beginning of an empty network. However, this is not permitted and the compiler recognizes this as an error.

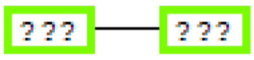


Fig. 12: Example: Impermissible assignment

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

4.5.4.3 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 or constants have to be the same data type. The instance name is defined in the declaration part of the editor as a variable with data type = function block name.

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

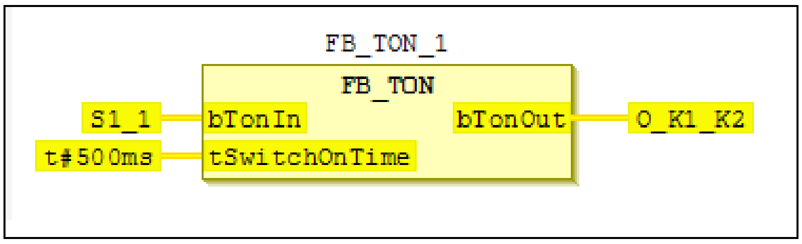


Fig. 13: Example: Instance FB_TON_1 of the POU FB_TON with formal parameters bTonIn, tSwitchOnTime, and bTonOut

In "CODESYS Safety for EtherCAT Safety Module", the developer can use the POUs from the safety library "TcEL6900FBs" and "TcEL69xxAnalogFBs".

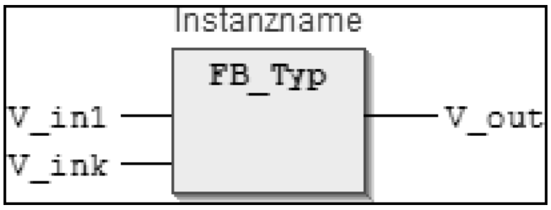


Fig. 14: Example: FB call

Inserting a POU call and inserting a POU

Function blocks are inserted as in the standard FBD of CODESYS.

Negation

"CODESYS Safety for EtherCAT Safety Module" provides the capability of inverting the signal of Boolean inputs. This is done by means of the "Negation" command which is available after an input is selected. Negated inputs are identified by a small circle at the input.

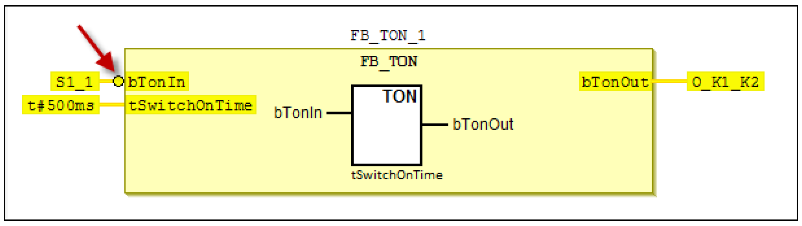


Fig. 15: Example: Negated input

Defining a data type

This command is available for analog function blocks. As a result, the data type of an analog input or output can be changed.

Defining an output connection

The output, which is provided for an output connection to a function block or an assignment, can be changed by means of this command. To do this, the new output connection has to be marked and the "Define output connection" has to be executed. This command is available only for POUs with multiple outputs.

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

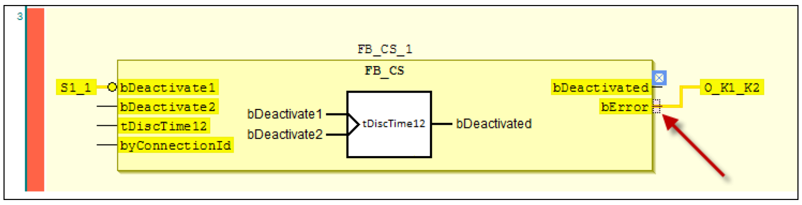


Fig. 16: Example: Define output connection

Removing unused FB call parameters

The *"Remove unused FB call parameters"* command removes all input and output connections of the call that are unassigned. They can be assigned again by means of the "Update Parameters" command.

4.6 Integration of Field Devices

4.6.1 Setting Up the Devices

1. Insert all planned field devices from the planning of the overall system into the project.
2. Provide symbolic names for the inputs/outputs of the devices that map specific input and output signals from the process according to the planning.
3. Set the communication parameters defined during the planning of the overall system and the device parameterization of the safety-oriented field devices.

   For a description of the device editors where you set the parameters and configure the I/O mapping, see the help for CODESYS Safety.

NOTICE!
The device editor is suitable for the display and processing of the device parameters of specific devices. More detailed information is located in the documentation for the respective device, or it can be obtained from the respective device manufacturer.

CAUTION!
Fieldbus-specific requirements for the setting of specific parameters are to be considered.

NOTICE!
The user is responsible for making sure that the devices are correctly parameterized according to the device documentation of the respective device or the respective device manufacturer.

NOTICE!
The device manufacturer has to notify the user about the conditions for the calculation of the system characteristic values.

For a description of the bus-specific safe parameters, see the help for CODESYS Safety.

4.6.2 Access to Input and Output Signals

I/O access takes place exclusively via the logical I/Os. In den mapping settings, symbolic names have to assigned to the I/O channels. Then, like global variables, these can be used in the program.

Type consistency of the I/O channels

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

– if the application revision levels on the safety controller and on the standard 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.

4.6.3 EtherCAT Safety SC Modules

Thanks to the TwinSAFE SC technology from Beckhoff, it is possible to use standard signals for safety-related tasks in any network or fieldbus. In addition, EtherCAT terminals from the analog input, position measurement (angle/distance), or communication (4...20 mA, incremental encoder, IO-Link, etc.) are extended with the TwinSAFE SC function.

The TwinSAFE SC technology typically reaches a safety level corresponding to PL d/Cat. 3 according to EN ISO 13849-1, or SIL 2 according to EN 62061. The input data from an SC module are non-safe analog values. For use in safety functions, the ESM application has to subject them to analysis, a plausibility check and a "vote". This is done by certified function blocks such as `FB_SCALING`, `FB_COMPARE`, `FB_LIMIT`, etc. In doing so, at least one of the data sources has to be a TwinSAFE SC terminal for safety reasons. The other data can originate from other standard bus terminals, drive controls, or measuring transducers. This makes it possible to make all process data in the system accessible for safety engineering.

These terminals provide analog input signals for safety control via a safeguarded TSC connection. Moreover, they have standard I/O channels that you can use in the standard application. In the device tree, the terminal is displayed as an EtherCAT module (EtherCAT Safety SC Module) with the EtherCAT settings and standard I/O channels, and as a submodule for the TSC connection to the safety controller. For a complete description of the terminals with technical data and information on standard I/O channels and configuration, see the corresponding device documentation. For application examples of TwinSAFE SC terminals, see the TwinSAFE Application Manual from Beckhoff.

For a list of TwinSAFE SC modules available for "CODESYS Safety for EtherCAT Safety Module", see the help for CODESYS Safety.

The device descriptions of the supported modules are included in the package.

5 Online Mode

5.1 Connecting to the EtherCAT Safety Module

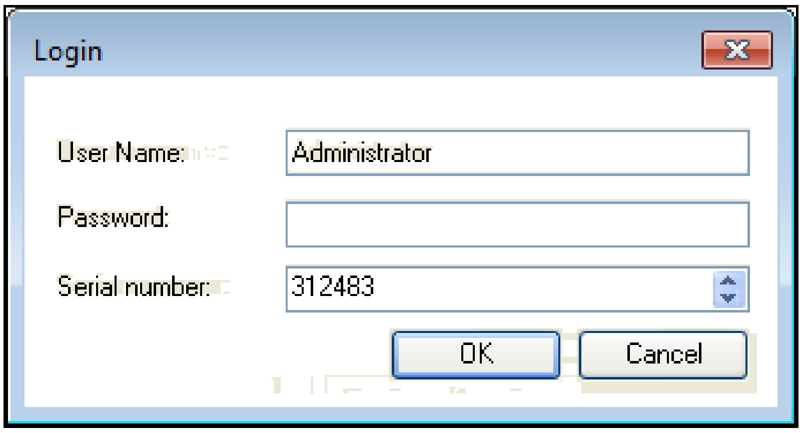
For full access, the requirements include a connection to the standard controller and a confirmation of the identity of the desired EtherCAT Safety module. This is done by specifying the unique FSoE address set on the EtherCAT Safety module on the "Safety Parameters" tab of the device editor.

5.2 Project Download

To execute the command, you have to be logged in with the standard application. The standard application has to be in the state RUN.

Before the command is executed, a user has to be selected from the user management list. If a user is not selected, then the "Administrator" is used. The download operation is executed from the "Safety Parameters" tab of the editor for the node of the EtherCAT Safety module. This function is available in online mode only.

After the *"Project Download"* command is executed, the user has to specify a password, as well as the serial number of the terminal, so that the data record to be accepted by the terminal.


Login

User Name: Administrator

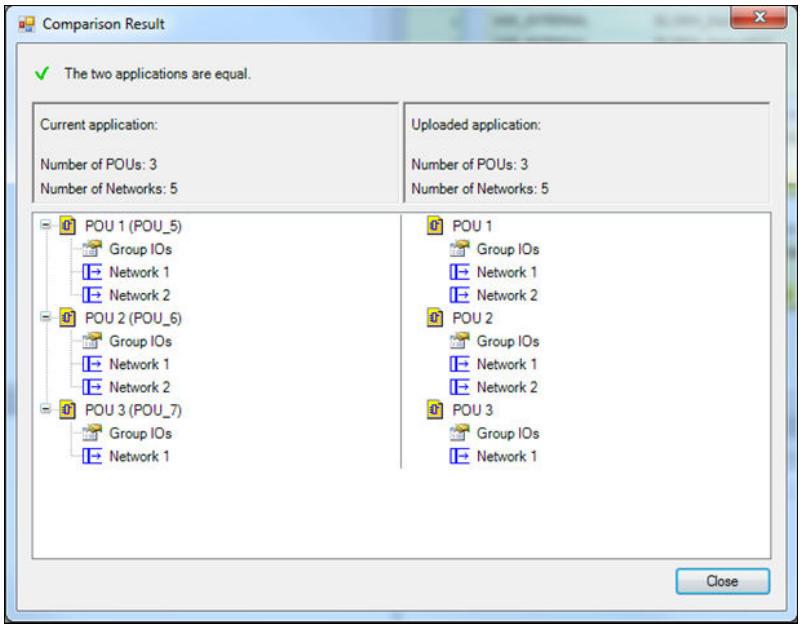
Password:

Serial number: 312483

Fig. 17: Login dialog

In order to check the generated code, the application is reverse compiled after it is downloaded. In doing so, the compiled code is decompiled and compared at source code level. By means of this diverse method, the function of the compiler is checked at each download. In the process, both random and systematic errors are detected.

The result of the comparison (equal or different) is displayed to the user. For EL6900, the number of compared POUs and networks is displayed. For EL6910 and EK1960, the number of compared POUs and function blocks is displayed. When equality is determined, the download is valid and can be acknowledged by the user within the dialog.


Comparison Result

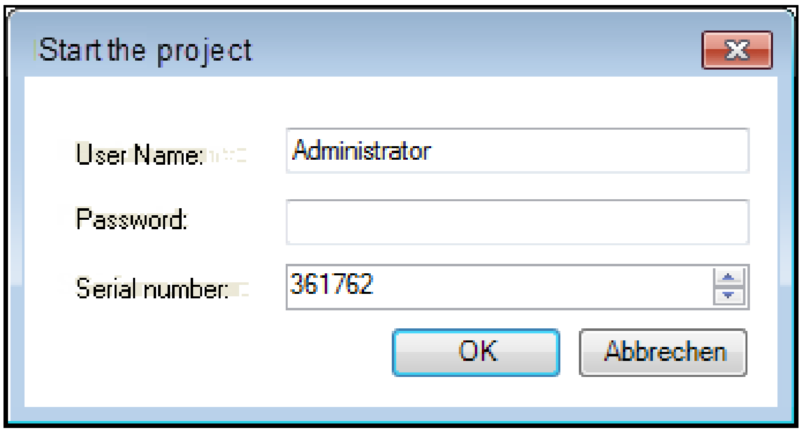
The two applications are equal.

Current application:	Uploaded application:
Number of POUs: 3	Number of POUs: 3
Number of Networks: 5	Number of Networks: 5

```
POU 1 (POU_5)                POU 1
    Group IOs                    Group IOs
    Network 1                    Network 1
    Network 2                    Network 2
POU 2 (POU_6)                POU 2
    Group IOs                    Group IOs
    Network 1                    Network 1
    Network 2                    Network 2
POU 3 (POU_7)                POU 3
    Group IOs                    Group IOs
    Network 1                    Network 1
```

Fig. 18: Dialog 'Comparison Result': Equal projects

When equality is determined, the download is valid and can be acknowledged by the user. After the user name and password are specified again, the activation record is downloaded to the controller. Then the controller is ready to start. The application has to be started from the user program. This is done by a rising edge of the variables defined in the group I/Os. (See the help for CODESYS Safety.)


Start the project

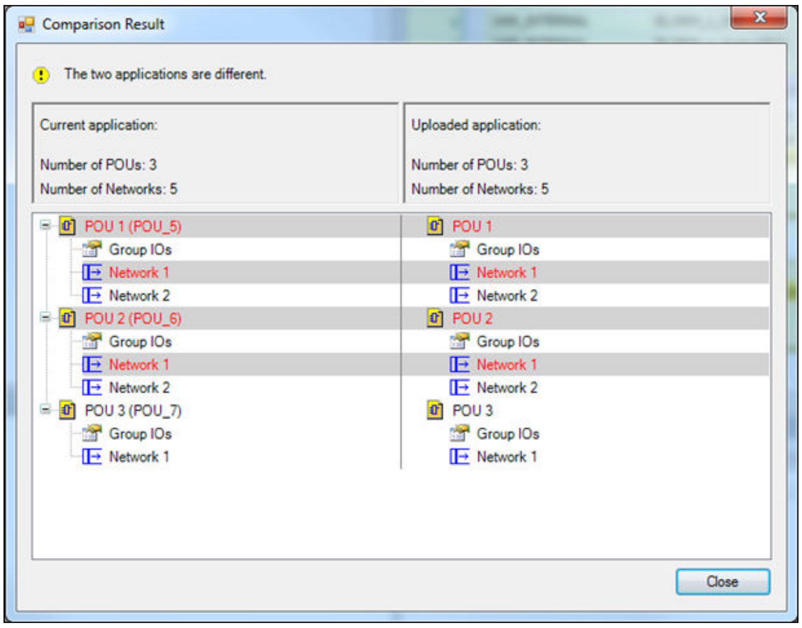
User Name: Administrator

Password:

Serial number: 361762

Fig. 19: Dialog for loading the activation record

If a difference is detected, then the download is invalid and cannot be acknowledged. In this case, the corresponding confirmation dialog is not displayed to the user. Instead, the user is shown where (in which block, network, or group I/O) the discrepancy has occurred.



The two applications are different.

Current application:
Number of POUs: 3
Number of Networks: 5

Uploaded application:
Number of POUs: 3
Number of Networks: 5

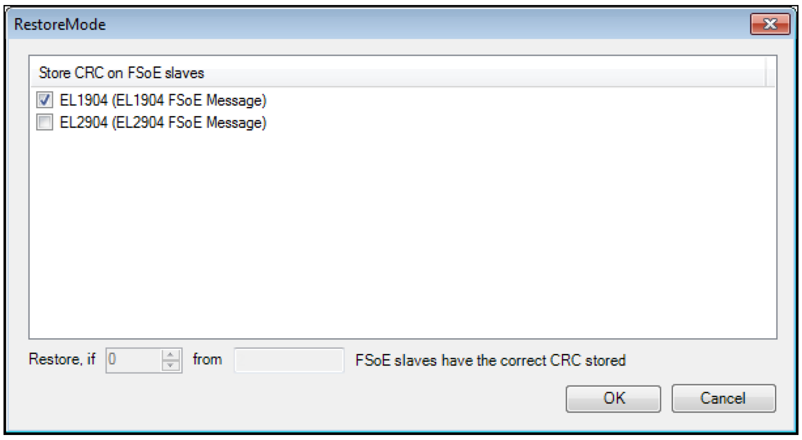
Current application	Uploaded application
POU 1 (POU_5)	POU 1
Group IOs	Group IOs
Network 1	Network 1
Network 2	Network 2
POU 2 (POU_6)	POU 2
Group IOs	Group IOs
Network 1	Network 1
Network 2	Network 2
POU 3 (POU_7)	POU 3
Group IOs	Group IOs
Network 1	Network 1

Fig. 20: Dialog 'Comparison Result': Different projects

5.3 Preparing for Device Replacement

When replacing an EtherCAT Safety module, the previously running project can be restored automatically to the new module and continue running. To do this, you have to select the "Automatic recovery" option on the "Safety Parameters" tab of the editor (after the project download).

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



Store CRC on FSoE slaves
- [✓] EL1904 (EL1904 FSoE Message)
- [ ] EL2904 (EL2904 FSoE Message)

Restore, if 0 from FSoE slaves have the correct CRC stored

Fig. 21: Dialog 'Automatic Restore'

A check box can be used to specify which slaves should store a CRC. Then the lower part defines how many of these slaves have to have the correct CRC so that the project is automatically restored after the EtherCAT Safety module has been replaced.

The same administrator password has to be set on the replaced EtherCAT Safety modules.

5.4 Project Upload

To execute the command, you have to be logged in with the standard application. The standard application has to be in the state RUN.

The upload operation is executed from the "Safety Parameters" tab of the editor for the node of the EtherCAT Safety module. This function is available in online mode only.

The result of the comparison (equal or different) is displayed to the user after the upload. For EL6900, the number of compared POUs and networks is displayed. For EL6910 and EK1960, the number of compared POUs and function blocks is displayed. The comparison view is identical to that of the ➪ *Chapter 5.2 “Project Download” on page 31* function. This function can be used to easily check whether or not the applications on the EtherCAT Safety module and in the project are equal or different.

5.5 Project Delete

To execute the command, you have to be logged in with the standard application. The standard application has to be in the state RUN.

The deletion of a project on the EtherCAT Safety module is executed from the "Safety Parameters" tab of the editor for the node of the EtherCAT Safety module.

The command is enabled when the application is logged on to the EtherCAT Safety module and the module is in the state RUN. After the command is executed, the login dialog opens. After the login credentials have been entered successfully, the project is deleted from the EtherCAT Safety module. An error message is displayed if the login credentials are incorrect.

5.6 Status Display

When the application (safety app) is executable and has been downloaded to the controller, the status of the application and the safety POUs is displayed in square brackets in the device tree when logged in (logged in with standard app).

Statuses of the safety application for EL6910 / EK1960

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

Statuses of the POUs for EL6910 / EK1960

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

The status "unknown" is displayed when

- The library IODrvEL6900 is not the latest version. It has to be V3.5.6.0 or later.
- The application for the EtherCAT Safety module in CODESYS does not match the application on the module.
- The EtherCAT stack is not yet running on the standard controller after a download or reset of the application.

5.7 Monitoring

When the application (safety app) is executable and has been downloaded to the controller, a monitoring of the input and output parameters of the safety FBs can be performed when logged in (logged in with standard app).

Question marks (?????) instead of signal states indicate the following:

- The library IODrvEL6900 is not the latest version. It has to be V3.5.3.0 or later.
- The EtherCAT stack is not yet running on the standard controller after a download or reset of the application.
- The application for the EtherCAT Safety module in CODESYS does not match the application on the module.

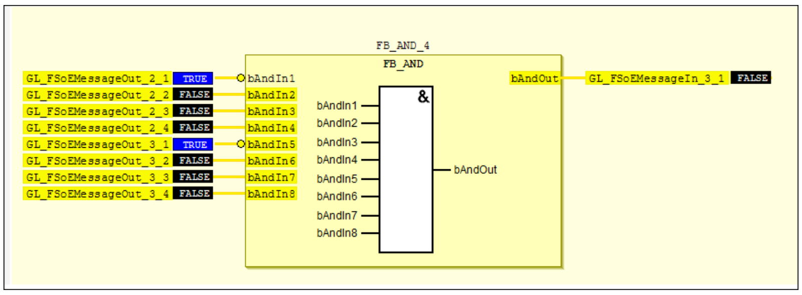

FB_AND_4
FB_AND
GL_FSoEMessageOut_2_1 TRUE
GL_FSoEMessageOut_2_2 FALSE
GL_FSoEMessageOut_2_3 FALSE
GL_FSoEMessageOut_2_4 FALSE
GL_FSoEMessageOut_3_1 TRUE
GL_FSoEMessageOut_3_2 FALSE
GL_FSoEMessageOut_3_3 FALSE
GL_FSoEMessageOut_3_4 FALSE
bAndIn1
bAndIn2
bAndIn3
bAndIn4
bAndIn5
bAndIn6
bAndIn7
bAndIn8
&
bAndOut
bAndOut
GL_FSoEMessageIn_3_1 FALSE

Fig. 22: Monitoring of variables

A change of signal states is not possible while monitoring.

6 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" provides the **pinning** function especially for this purpose.

> **NOTICE!**
> Before the programmed or modified safety application is verified, it has to be re-pinned. Verification and acceptance may only be performed with pinned applications. No object of the application may have the status "In Work". The status "In Work" is checked in the individual verification steps.

What is pinning?

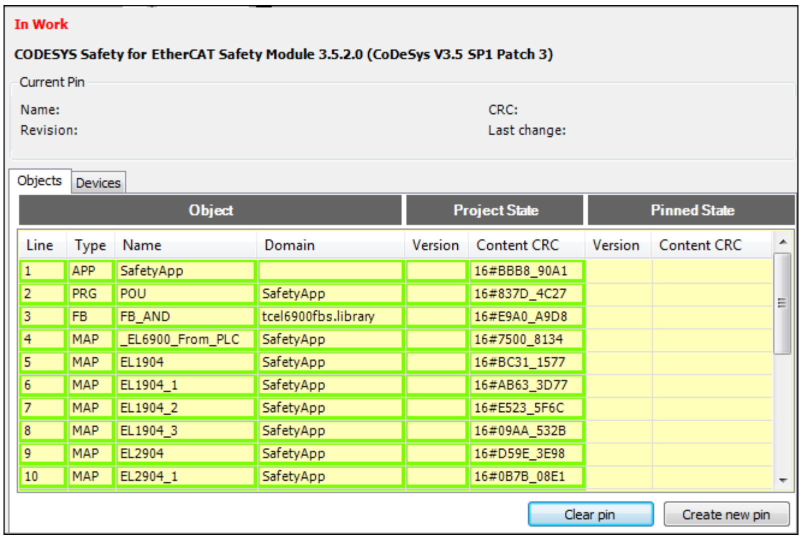
Pinning means that a reference point to the current version of a safety application is set that identifies the specific version of the safety application and the associated objects. By means of the pin it is possible to identify a certain version of the application in the project, of an object in the editor on the EtherCAT Safety module. In addition, the verifier, on the basis of the pin, can recognize 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 do this, the safety application object is selected in the project tree and opened by means of the *“Edit Object”* context menu command. The *“Objects”* dialog shows the comparison view, which displays the version and the CRC of the objects of the current project and of the pinned project.

For a detailed description of the information and the object list of the comparison view, see the help for CODESYS Safety.



In Work

CODESYS Safety for EtherCAT Safety Module 3.5.2.0 (CoDeSys V3.5 SP1 Patch 3)

Current Pin

Name:
Revision:

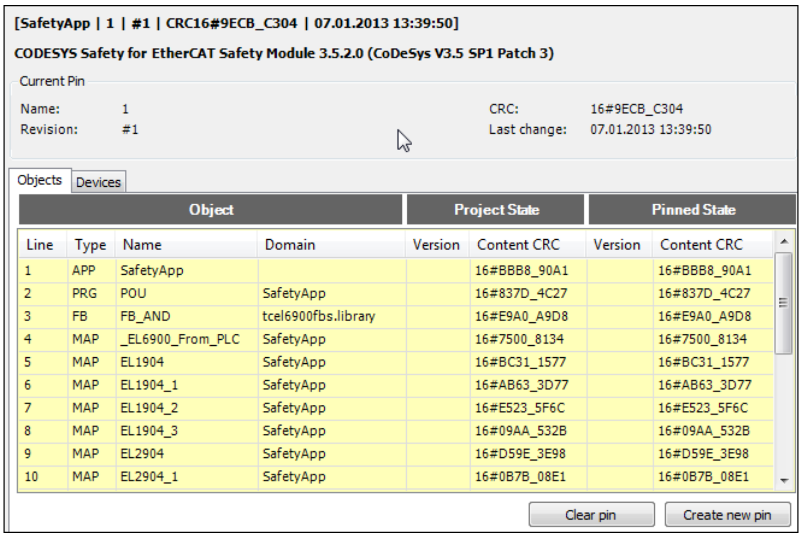
CRC:
Last change:

Objects | Devices

Object				Project State		Pinned State	
Line	Type	Name	Domain	Version	Content CRC	Version	Content CRC
1	APP	SafetyApp			16#BBB8_90A1		
2	PRG	POU	SafetyApp		16#837D_4C27		
3	FB	FB_AND	tcel6900fbs.library		16#E9A0_A9D8		
4	MAP	_EL6900_From_PLC	SafetyApp		16#7500_8134		
5	MAP	EL1904	SafetyApp		16#BC31_1577		
6	MAP	EL1904_1	SafetyApp		16#AB63_3D77		
7	MAP	EL1904_2	SafetyApp		16#E523_5F6C		
8	MAP	EL1904_3	SafetyApp		16#09AA_532B		
9	MAP	EL2904	SafetyApp		16#D59E_3E98		
10	MAP	EL2904_1	SafetyApp		16#0B7B_08E1		

Clear pin | Create new pin

Fig. 23: Comparison view of a safety application that is not pinned yet



[SafetyApp | 1 | #1 | CRC16#9ECB_C304 | 07.01.2013 13:39:50]

CODESYS Safety for EtherCAT Safety Module 3.5.2.0 (CoDeSys V3.5 SP1 Patch 3)

Current Pin

Name: 1
Revision: #1

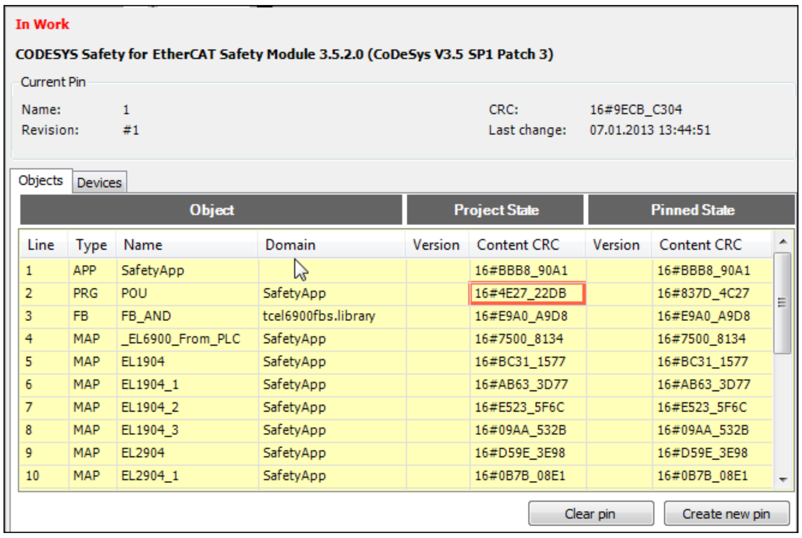
CRC: 16#9ECB_C304
Last change: 07.01.2013 13:39:50

Objects | Devices

Object				Project State		Pinned State	
Line	Type	Name	Domain	Version	Content CRC	Version	Content CRC
1	APP	SafetyApp			16#BBB8_90A1		16#BBB8_90A1
2	PRG	POU	SafetyApp		16#837D_4C27		16#837D_4C27
3	FB	FB_AND	tcel6900fbs.library		16#E9A0_A9D8		16#E9A0_A9D8
4	MAP	_EL6900_From_PLC	SafetyApp		16#7500_8134		16#7500_8134
5	MAP	EL1904	SafetyApp		16#BC31_1577		16#BC31_1577
6	MAP	EL1904_1	SafetyApp		16#AB63_3D77		16#AB63_3D77
7	MAP	EL1904_2	SafetyApp		16#E523_5F6C		16#E523_5F6C
8	MAP	EL1904_3	SafetyApp		16#09AA_532B		16#09AA_532B
9	MAP	EL2904	SafetyApp		16#D59E_3E98		16#D59E_3E98
10	MAP	EL2904_1	SafetyApp		16#0B7B_08E1		16#0B7B_08E1

Clear pin | Create new pin

Fig. 24: Comparison view of a pinned safety application



In Work

CODESYS Safety for EtherCAT Safety Module 3.5.2.0 (CoDeSys V3.5 SP1 Patch 3)

Current Pin

Name:	1	CRC:	16#9ECB_C304
Revision:	#1	Last change:	07.01.2013 13:44:51

Object					Project State		Pinned State
Line	Type	Name	Domain	Version	Content CRC	Version	Content CRC
1	APP	SafetyApp			16#BBB8_90A1		16#BBB8_90A1
2	PRG	POU	SafetyApp		16#4E27_22DB		16#837D_4C27
3	FB	FB_AND	tcel6900fbs.library		16#E9A0_A9D8		16#E9A0_A9D8
4	MAP	_EL6900_From_PLC	SafetyApp		16#7500_8134		16#7500_8134
5	MAP	EL1904	SafetyApp		16#BC31_1577		16#BC31_1577
6	MAP	EL1904_1	SafetyApp		16#AB63_3D77		16#AB63_3D77
7	MAP	EL1904_2	SafetyApp		16#E523_5F6C		16#E523_5F6C
8	MAP	EL1904_3	SafetyApp		16#09AA_532B		16#09AA_532B
9	MAP	EL2904	SafetyApp		16#D59E_3E98		16#D59E_3E98
10	MAP	EL2904_1	SafetyApp		16#0B7B_08E1		16#0B7B_08E1

Fig. 25: 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, whereby the CRC and the version are noted, not the contents. A pin name can be entered. The revision number of the pin is incremented automatically by one with each "new" pinning.

Using the *“Clear Pin”* command, the current pin is deleted and all objects are reset to *“In work”*.

The pinned version of the safety application comprises the following:

- 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 therefore can 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 development system. The pin identifier contains the following information:

- Name
- Revision counter, which is incremented by one 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 developer can display the pin information of the application to a safety application in the comparison view.

The safety application pin information consists of the following:

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

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

- New objects
- Deleted objects
- Objects modified with regard to code or device parameters
- POUs recently reffered from libraries
- Library function blocks no longer referred
- Library function blocks differing with regard to interface or implementation version

Differences are clearly marked in color so that the verifier can easily recognize 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 view and 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 child objects are marked with the symbol. The *“SafetyApp”* node point is considered to be pinned when the object and all its child objects 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 child objects are not marked. If a child object 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 for execution-relevant changes to be recognized during and after the verification. By setting a pin, the developer releases the current status of a safety application in the project for verification or to continue verification after a change. Pinning makes this status identifiable for verification activities and for acceptance with regard to the parts relevant to the acceptance.

NOTICE!

– The developer always has to pin a completed or modified application before releasing it for (continued) verification. It must not be an object of the safety application "In Work".
– The developer has to check during reviews, white box 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 has to 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.

Pinning

7 Verification and Acceptance

7.1 Verification and Acceptance

NOTICE!
The complete test of the application is performed by the user.

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

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

7.2 Archiving

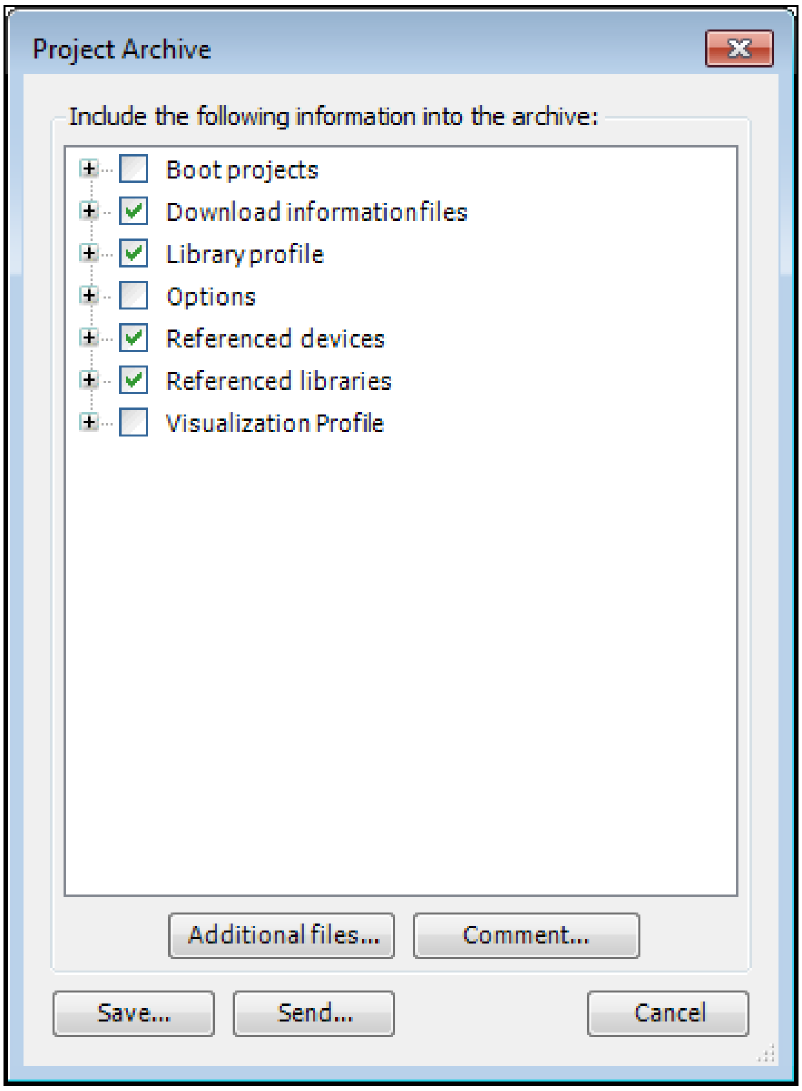
NOTICE!
Throughout the entire process, from the development to the acceptance of a safety application, the user has to take care that the correct object versions, the correct libraries, and the correct device description files are used (see ⇘ *Chapter 6 “Pinning” on page 37*).

Archiving helps to combine all objects belonging to a safety application in a single file.

This file can be saved as a data backup and unpacked again when needed.

NOTICE!
The project for the acceptance of the safety application has to be archived by means of the project archiving function. This has to be done by means of the *“Save/Send Archive”* command in the *“Project Archive”* submenu of the *“File”* category. All information relevant to the acceptance has to be selected in the project archive subdialog:

– Library profile
– Boot projects
– Download information files
– Options
– References libraries
– Referenced devices


Project Archive

Include the following information into the archive:

- [ ] Boot projects
- [x] Download informationfiles
- [x] Library profile
- [ ] Options
- [x] Referenced devices
- [x] Referenced libraries
- [ ] Visualization Profile

Fig. 26: Selection dialog: Creating a project archive

The following items are also included in the project archive by means of correct selection in the selection dialog:

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

The project has to be saved before it is archived. Automatic saving may be done.

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

7.3 Printing of Acceptance Documentation

In "CODESYS Safety for EtherCAT Safety Module", the *“Document”* command in the *“Project”* category is used to print out the project documentation. All marked objects are printed out, including the object information, CRC, and pin information (pin identifier). 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 may also contain a cover sheet with the following information: file, date, profile, and table of contents.

NOTICE!
For the printout of the acceptance documentation, the EtherCAT Safety module and all of its sub-node points and objects have to be marked in the *“Document Project”* dialog.

For detailed information about documenting a project, see the CODESYS Help.

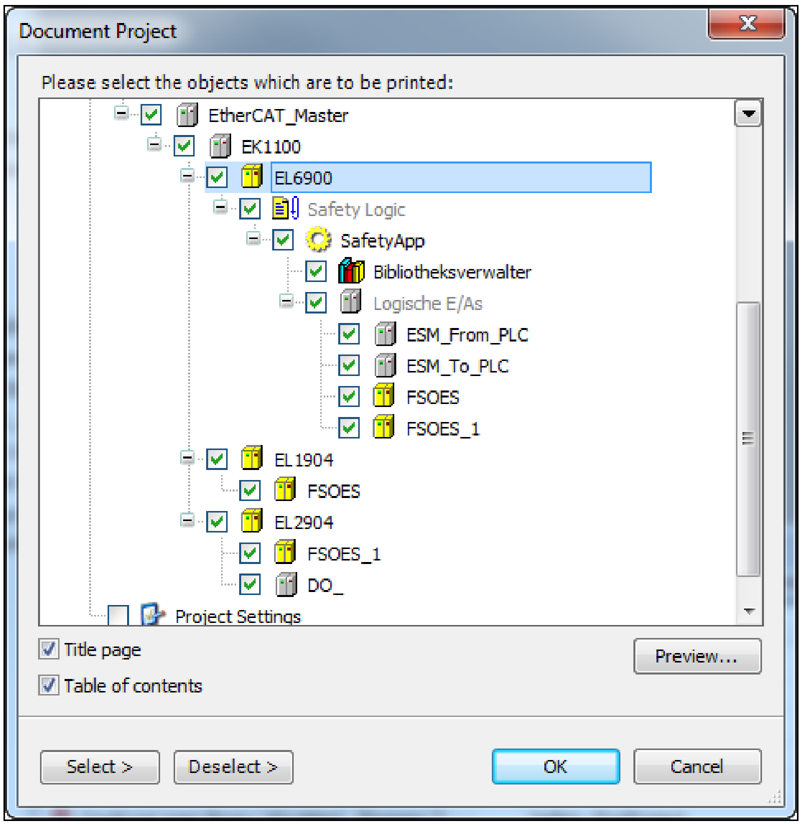


Fig. 27: Dialog 'Document Project'

8 Software Update

8.1 Overview of Version Control

This chapter describes the possible effects of the installation of a new version of a device, a firmware, or CODESYS on an existing verified and accepted project and the consequences of the installation of a package not released for "CODESYS Safety for EtherCAT Safety Module".

8.2 Updating the Device Version

NOTICE!
Changing the contents of a device description can cause the safety application to no longer be pinned, but set back to "In Work". The safety objects affected by the changed device description and listed with a changed CRC in the comparison view of the safety application object have to be subjected to an influence analysis and subsequently re-pinned and verified and, possibly accepted again.

When a new version of a device description is available, the service technician needs to make sure he knows before installing the new device description exactly which of the possible variants of a new version is affected and what effect this will have on the project and also on the verification and the acceptance:

- The new device is compatible with the previous device:
  After the new device description is installed in the device repository and the device is updated (by means of the context menu in the device tree), no errors occur when compiling the safety application. No additional activities have to be performed.
- A safety device is being updated when:
  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 CRC in the comparison view of the safety application object have to be subjected to an influence analysis and subsequently re-pinned and verified.

New device description files have to be installed in the device repository first. In order to update the device, the physical device has to be selected in the project tree and the “Update Device” command in the context menu has to be executed.

The detailed information to the installed device descriptions is available in the device repository. To do this, the respective device has to be selected and the “Details” button has to be clicked.

8.3 Changing a Library Version

An update to the device description may cause newer library versions to be referred. This happens automatically in such a way that the set libraries always match the device in use. When library function blocks have changed, these have a different CRC and the user has to pin, verify, and accept his application again.

NOTICE!
A change to the execution version also changes the pin of the safety application. The application has to be pinned, verified, and accepted again.

NOTICE!
An update to 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 or not an update is possible without any complications.

NOTICE!
In some circumstances, the use of another library version can change the pin of a used library function block. See the comparison view of the safety application to find out whether or not this is the case.

In this case, the application has to be pinned, verified, and accepted again.

NOTICE!
The correct libraries for this EtherCAT Safety module are automatically referred in the correct version with the device description of this EtherCAT Safety module. The Library Manager serves only to provide an overview of the available libraries and their POUs. It is recommended not to remove or add libraries manually.

8.4 Updating the CODESYS Version

Two versions are important for "CODESYS Safety for EtherCAT Safety Module": the version of CODESYS and the version of the "CODESYS Safety for EtherCAT Safety Module" extension. (The information is found in the *"Help"* menu, *"About"* 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 CODESYS and "CODESYS Safety for EtherCAT Safety Module" software packages should always be updated together.

Before you perform the update, check whether or not the combination of CODESYS version and "CODESYS Safety for EtherCAT Safety Module" version are released. For more information, see *Chapter 3.3 "Installation" on page 11*.

After an update, a dialog for updating the project environment may appear when starting CODESYS. We recommend that you use the latest versions of the libraries because they might contain new features.

CAUTION!
Currently, updating the logical I/Os of the EK1960 module breaks the link to the physical device. As a result, it is no longer possible to map the logical I/Os. If you update devices in the "Project Environment" dialog, then always exclude the logical I/Os from the update. Keep using the old device description of the logical I/Os.

8.5 Change Request / Bugfix Process

Change requests and bug reports regarding the "CODESYS Safety for EtherCAT Safety Module" package pass through a workflow as defined in the QM process at 3S-Smart Software Solutions.

Change request

1. The customer message requesting a change is received by telephone or email.
2. The change request is registered in the bug database (Jira) by support staff of 3S-Smart Software Solutions.

Bug

1. The customer message reporting a bug is received by telephone or email
2. A check is performed to determine whether or not the bug is reproducible.
3. A check is performed by the project manager to determine whether or not this change request can be implemented with regard to risk and expense.
4. The bug is registered in the bug database "ESM" (Jira) by support staff of 3S-Smart Software Solutions.
5. A check is performed by the project manager to determine whether or not this bug is safety-relevant.
6. In the case of safety-relevant bugs, the registered customers are notified by email.

The bug database is accessible to all registered customers over the Internet. They can query the status of all entries there.

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

9 Glossary

API	Protocol-independent application interface With the aid of an API, developers get access at the programming level to each safe I/O module at the safety communication level.
Application	Executable unit on a controller consisting of program code and associated I/O configuration
CODESYS	(Controller Development System) Programming system from 3S-Smart Software Solutions for the programming of **PLCs**. Currently available in the second generation and the version designated as V3.x
Configuration	Description of the system consisting of controllers and I/O modules (both standard and fail-safe); definition of the project structure
Controller	PLC
Coupler	Another name for slave on the fieldbus (for example, a DP slave on the Profibus); connects individual I/O modules with the fieldbus.
Data types	**Types**
Development system	Development environment on a Windows PC for creating programs according to **IEC 61131-3** as well as software access to controllers with CODESYS RTS
Editor	Component of the PS for editing program sections
Execution version	Identifier by means of which the checked compatibility between (accepted) boot application and RTS core is controlled. For compatibility with external POUs, see **Implementation version**. From the user perspective, 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.
Fail-safe	Values of variables that ensure the safety of the plant. Fail-safe value for variables of the data type SAFEBOOL have to be FALSE. The user has to make sure that all variables of the data type SAFEBOOL guarantee the safety of the plant. All variables of the data type SAFEBOOL have to be set to the value FALSE both for the initialization and for a safety requirement.
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, **FBD**
FBD	**Function Block Diagram**
Fieldbus device	Device that implements a fieldbus protocol (for example, Profibus DP or Profinet) by means of which the process image is exchanged. Of particular relevance are

	- Couplers with subordinate **I/O modules**, and **-** Drives; including **I/O modules** that are typically autonomous fieldbus devices.
Field devices	**Fieldbus devices**
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 permitted for safety programming.
I/O module	Hardware module with input channels and/or output channels. A distinction is made between **-** I/O modules (also called terminals or "slices") behind a coupler that is a fieldbus device. The input or output channels of each terminal behind the coupler correspond to a section in the process image of the coupler. **-** I/O module that are fieldbus devices (for example, 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
POU	**Program Organization Unit** (POU)
Program	POU according to IEC 61131-3
Programmable logic controller	The name attributed to a device which, by means of special programming software, is used for the automatic control of industrial machine and plants
Program Organization Unit	Object in the project structure that represents software (and not devices/modules) POU
Project	A project can contain multiple 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 utilizable 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) provides increased fail-safety through special hardware extensions (for example, two processors with opposite-acting process observation) and accordingly adapted software.
SAFExxx	Series of data types beginning with "SAFE" (example: SAFEBOOL)
SDD	Safety Devicefacet Description. **CRC**-safeguarded device description for safety-oriented device information
Slave	HW that is subordinate to a master by which it is controlled or supplied with data
Standard CODESYS	CODESYS without additions, as delivered. Technically, it is a CODESYS instance into whose profile the safety extension in particular is NOT loaded
SW	Software
Terminal	Another name for **I/O modules** behind a coupler
Types	A 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** to the **user**. To be distinguished from the **programming interface**.
Validation	Check of whether or not the completely generated and verified product meets the demand placed on it
Verification	Check of whether or not the created documents and software functions, modules, and components satisfy their specifications (or their superordinate specifications)

Glossary

10 Index

A

add user . . . 20
archiving . . . 43

B

bugfix process . . . 49

C

call function blocks . . . 25
change password . . . 20
change request . . . 49
CODESYS version . . . 49
comparison view . . . 19
CRC of an object . . . 19

D

data types . . . 23
deactivation . . . 21
define data type . . . 26
define output connection . . . 26
delete user . . . 20
device update . . . 47

E

EtherCAT safety, user management . . . 20

G

groups . . . 21

I

install CODESYS Safety for EtherCAT Safety Module . . . 11
installation . . . 11

L

library version . . . 48

M

monitor . . . 35
monitoring . . . 35

N

negation . . . 26

P

Package Manager . . . 11
pin CRC . . . 19
preparation for verification . . . 37
print acceptance documentation . . . 45
programming . . . 13
programming guidelines . . . 16
project delete . . . 34
project download . . . 31
project tree . . . 13
project upload . . . 33, 34

S

safety controller . . . 19
safety pinning . . . 37
source code management, access protection . . . 20
standards . . . 7
status, application . . . 35

U

uninstall packages . . . 11
update device . . . 47
upload . . . 20
user management . . . 20

V

version history . . . 20