CODESYS

Anwenderhandbuch

CODESYS® Safety

Vor Beginn aller Arbeiten Anleitung lesen.

TÜVRheinland CERTIFIED
Functional Safety
www.tuv.com
ID 0600000000

PLCopen safety

3S-Smart Software Solutions GmbH
Memminger Str. 151
87439 Kempten
Deutschland
Telefon: +49-831-54031-0
Telefax: +49-831-54031-50
E-Mail: info@codesys.com
Internet: www.codesys.com
Dokument Version: V8.0

Bitte beachten Sie: Nicht alle CODESYS-Funktionen sind in allen Ländern verfügbar. Weitere Informationen zu diesen länderspezifischen Einschränkungen erhalten Sie unter support@codesys.com.

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

Dieses Handbuch ist gültig für CODESYS Safety. Dieses Handbuch ist Teil des CODESYS Safety Produktpakets und vor Erstellen einer Sicherheitsapplikation mit CODESYS Safety unbedingt zu lesen.

Es wendet sich an Planer, Programmierer, Inbetriebnehmer, Betreiber und Wartungspersonal von Automatisierungsanlagen mit Sicherheitsfunktionen.

Voraussetzung für das Verstehen des Handbuchs sind gute CODESYS Kenntnisse und Kenntnisse der IEC 61131-3.

Wir wünschen Ihnen viel Vergnügen beim Lesen des Handbuchs!

1.1 Aufgaben des Handbuchs

Dieses Handbuch enthält Informationen für den bestimmungsgemäßen Gebrauch von CODESYS Safety Steuerungen und deren Programmierung mit CODESYS Safety und zur Sicherheitskonfiguration und Parametrierung angeschlossener Feldgeräte.

Das Handbuch enthält Hinweise, die bei der Erstellung Ihrer Applikation unbedingt zu beachten sind. Es werden folgende Arten von Sicherheitshinweisen verwendet:

GEFAHR!
Bei Nichtbeachtung dieser Sicherheitshinweise können Personen unmittelbar geschädigt werden.

VORSICHT!
Bei Nichtbeachtung dieser Sicherheitshinweise können eventuell später Gefahren entstehen.

HINWEIS!
Die Nichtbeachtung dieser Hinweise kann zu Fehlern und Problemen bei der Entwicklung und der Verfikation des Sicherheitsprojekts führen.

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.2 Gültigkeit des Handbuchs

Dieses Dokument beschreibt alle Versionen von CODESYS Safety, für die die Revision List (Appendix) zum Zertifikat für CODESYS Safety auf der Internetseite *fs-products.tuvasi.com* die auf Seite 2 dieses Dokuments genannte Version des Handbuchs aufführt. (Zum Verifizieren der Version siehe *Kapitel 2.6 „Richtige Version und Konfiguration des Programmiersystems CODESYS Safety“ auf Seite 13*)

Die beschriebene Version von CODESYS Safety kann auf verschiedene Versionen von CODESYS installiert worden sein. Dann kann es zu Abweichungen in der Benutzeroberfläche gegenüber der Darstellung in diesem Handbuch und in der CODESYS Safety Online-Hilfe kommen oder zu anderen Effekten. Siehe *Kapitel 2.6 „Richtige Version und Konfiguration des Programmiersystems CODESYS Safety“ auf Seite 13*

1.3 Einordnung in die Informationslandschaft

Dieses Handbuch konzentriert sich auf

- Security- und Safety-Hinweise
- den Entwicklungsprozess von Sicherheitsapplikationen mit CODESYS Safety
- die genaue Beschreibung des Sprachumfangs, der zur Erstellung einer Sicherheitsapplikation zur Verfügung steht
- den Online-Zugriff und Betrieb von CODESYS Safety Sicherheitssteuerungen
- Fragen der Softwarekonfiguration, der Versionierung und von Änderungen

Dabei bezieht es sich nur auf die Entwicklungsumgebung unter deutscher Lokalisierung.

Für die Arbeit mit CODESYS Safety benötigen Sie je nach Anwendungsfall zusätzliche Dokumentationen. In der vorliegenden Dokumentation wird an geeigneten Stellen auf diese Dokumentation verwiesen.

Die CODESYS Online-Hilfe (erweitert durch Installation der Safety-Erweiterung) hingegen:

- beschreibt die Bedienung von CODESYS
- beschreibt das Handling von Projekten, Bibliotheksversionen und geänderten Bedeutung der Versionsangaben der Safety-Erweiterung
- enthält Informationen zum Programmieren und Konfigurieren der Standardsteuerung und der Feldbusse

2 Voraussetzungen und allgemeine Hinweise

2.1 Bestimmungsgemäße Verwendung

CODESYS Safety ist eine Software zum Programmieren von Sicherheitssteuerungen vom Typ CODESYS Control Safety (CODESYS Safety Steuerungen) und zur Sicherheitskonfiguration und -parametrierung angeschlossener Feldgeräte.

CODESYS Safety ist geeignet zur Entwicklung von Sicherheitsapplikationen im Sinne von Anwendungssoftware bis zu SIL3 nach IEC 61508 / 62061 und Kat4 PL-e nach ISO 13849.

CODESYS Safety Steuerungen realisieren die Logik von Sicherheitsfunktionen in industriellen Umgebungen. Sie können, wenn ihr herstellerspezifisches Handbuch dies zulässt, einsetzbar sein bis zu Sicherheits-Integritätslevel SIL3 gemäß IEC 61508 und PL-e gemäß ISO 13849-1.

Die Anwendungsprogramme dürfen nur mit dem dafür vorgesehenen Programmiertool CODESYS Safety erzeugt werden. Die Umgebungsbedingungen für den bestimmungsgemäßen Einsatz der Steuerung (Temperatur etc.) gemäß herstellerspezifischem Handbuch sind ebenfalls zu berücksichtigen.

2.2 Qualifiziertes Personal

Voraussetzung für die Anwendung sicherheitstechnischer Produkte ist eine ausreichende Qualifikation.

So muss das Personal, welches Sicherheitsapplikationen mit dem vorliegenden Produkt plant, erstellt und in Betrieb nimmt

- eine geeignete technische Ausbildung besitzen
- mit den geltenden Normen und Vorschriften vertraut sein
- mit den einschlägigen Sicherheitskonzepten in der Automatisierungstechnik vertraut sein
- mit grundlegenden Unfallverhütungs- und Arbeitssicherheitsvorschriften und eventuell besonderen Betriebsvorschriften vertraut sein
- die herstellerspezifischen Handbücher der Sicherheitssteuerung und der verbundenen sicheren Feldgeräte und Schutzeinrichtungen kennen
- die Sicherheitshinweise in diesem Handbuch gelesen und verstanden haben
- gute Kenntnisse der CODESYS Programmierung vorweisen

2.3 Gewährleistung und Haftung

Gewährleistungs- und Haftungsansprüche gehen verloren,

- wenn die Schäden auf Nichtbeachtung der gültigen Normen, des Anwenderhandbuchs, des Integrationshandbuchs oder der Online-Hilfe zurückzuführen sind,
- wenn die Anweisungen dieses Anwenderhandbuches nicht befolgt wurden
- das Betreiberpersonal nicht ordnungsgemäß ausgebildet ist

2.4 Allgemeine Sicherheitshinweise

Ein Projekt, das mit CODESYS Safety erstellt wurde, garantiert nicht, dass die gesamte Anwendung sicher ist.

VORSICHT!
Es müssen vom Anwender geeignete Sicherheitskonzepte für ihre Anlagen, Maschinen und Programme erstellt werden.

VORSICHT!
Hersteller und Betreiber von Maschinen mit CODESYS Safety Steuerungen müssen bei Entwicklung, Einbau, Inbetriebnahme, Betrieb und technischen Überprüfungen alle in ihrem Land geltenden nationalen und internationalen Rechtsvorschriften, Sicherheitsregeln und Normen in eigener Verantwortung einhalten.

VORSICHT!
Die Hinweise des Anwenderhandbuchs von CODESYS Safety und der herstellerspezifischen Handbücher der eingesetzten sicherheitstechnischen Geräte müssen beachtet werden.

VORSICHT!
Zum Arbeiten an Sicherheitsapplikationen dürfen nur die Befehle, Werkzeuge, Benutzeroberflächen-Steuerelemente, Editor-Registerkarten und Ansichten verwendet werden, die in ↳ *Kapitel 16 „Liste zulässiger oder geänderter Funktionen“ auf Seite 301* aufgelistet sind.

2.5 Systemvoraussetzungen

Für das Programmiersystem gelten folgende Systemvoraussetzungen als Mindestausstattung für kleinere CODESYS Safety-Projekte mit maximal 100 Bausteinen, maximal 10 Visualisierungen und maximal 8 Feldbusteilnehmern :

- 1 GB RAM
- 1 GHz Pentium
- 1 GB freier Festplattenspeicher
- Bildschirmauflösung 1024 x 768

HINWEIS!

Das Programmiersystem CODESYS Safety ist nur freigegeben für:

– Betriebssysteme: Windows 7, 64 bit oder Windows 10, 64 bit
– Bildschirmauflösung: 96 dpi

2.6 Richtige Version und Konfiguration des Programmiersystems CODESYS Safety

Hinweis Installation1

HINWEIS!

Verifizieren Sie, dass die in diesem Dokument beschriebene Version von CODESYS Safety verwendet wird. Dazu aktivieren Sie in CODESYS Safety den Befehl *„Safety-Versionsinformation anzeigen...“* der Kategorie *„Hilfe“*. Die angezeigte CODESYS Safety Version muss für die auf Seite 2 genannte Dokumentenversion des Anwenderhandbuchs gültig sein. Die Gültigkeit kann durch einen Abgleich mit der Revision List (Appendix) zum Zertifikat für CODESYS Safety auf der Internetseite *fs-products.tuvasi.com* überprüft werden.

Hinweis Installation2

HINWEIS!

Hinweis zur Safety-Version

Verifizieren Sie über den Package Manager, ob zusätzliche Plug-Ins installiert wurden (*„Tools ➔ Package Manager“*). Wenn diese nicht explizit für die Verwendung mit CODESYS Safety zugelassen sind, ist die Eignung der Funktionen von CODESYS Safety für kritische Schritte im Entwicklungsprozess nicht mehr gewährleistet. Das betrifft insbesondere:

- Ansichten der Safety-Objekte, Registerkarte *„Safety Online Information“*
- Onlinebefehle
- "Safety Querverweisliste" und "Gehe zur Definition"
- Projekt dokumentieren
- Projekt archivieren

Die Funktionen dürfen dann nicht mehr für Onlinezugriffe, Verifikation oder Abnahme herangezogen werden. (Es sei denn, Sie führen eine Tool-Validierung Ihrer Installation gemäß IEC 61508, Teil 3 durch).

Erklärung zur Versionsangabe:

Eine Version von CODESYS Safety hat immer die Form "CODESYS Safety 1.2.x ". So wird sie in der Revision List angezeigt und in CODESYS Safety unter *„Hilfe ➔ Safety-Versionsinformation anzeigen...“* (mit weiteren Informationen).

Die dortige Versionszeige hat die Form wie in diesem Beispiel "CODESYS Safety 1.2.0 (CODESYS V3.5 SP8 Patch 4)":

- der erste Teil ist die Version von CODESYS Safety. Ihr entsprechen die ersten 3 Stellen der Package-Version von CODESYS Safety Extension.
- die Angabe in Klammern identifiziert die Version der aktuell verwendeten CODESYS Basis, im Beispiel CODESYS V3.5 SP8 Patch 4.

Es kann bei Beginn des Safety-Engineerings im Projekt eine Meldung "Ungültige Installation" kommen, weil die spezielle, aktuell verwendete Basis nicht von 3S-Smart Software Solutions GmbH für die genutzte Safety-Version freigegeben ist.

Benutzeroberflächen-Varianz

Manche der beim Safety-Engineering verwendeten Ansichten, Befehle und Dialoge stammen aus der CODESYS Basis. Damit kann ihr Aussehen, Verhalten und Funktionsumfang variieren, je nachdem, welche Version von CODESYS als Basis verwendet wird.

Im *Kapitel 16 „Liste zulässiger oder geänderter Funktionen“ auf Seite 301* finden Sie die Auflistungen, welche Ansichten und Befehle für CODESYS Safety verwendet werden dürfen, und welche dabei in Aussehen und Verhalten je nach Version der verwendeten CODESYS Basis variieren können.

- CODESYS-Rahmenfenster
- CODESYS Safety (Ansicht *„Geräte“* und Ansicht *„POUs)“*
- Ansicht *„Meldungen“*
- Ansicht *„Überwachungsliste“*
- Ansicht *„Werkzeuge“*
- Geräteeditor-Rahmenfenster
- Dialog *„Hilfe ➔ Informationen“*
- Geräteeditor, Registerkarte *„Kommunikation“*
- Bibliotheksverwalter
- Dialog *„Tools ➔ Geräte-Repository“*
- Dialog *„Tools ➔ Bibliotheks-Repository“*
  Dialoge *„Bearbeiten ➔ Suchen Ersetzen“*
- und weitere, kleine Dialoge/Ansichten

Wenn ein Safety-Projekt in CODESYS geöffnet ist, wird das Verhalten mancher Standardbefehle und Standardansichten von CODESYS beeinflusst.

CODESYS für die Arbeit mit CODESYS Safety einrichten

Hinweis Installation3

HINWEIS!

CODESYS bietet zwei mögliche Ansichten der Kommunikationseinstellungen. Die neue grafische Ansicht ist nicht für die Verwendung mit CODESYS Safety freigegeben. Sie müssen vor dem Öffnen der Registerkarte *„Kommunikation“* im Dialog *„Tools ➔ Optionen ➔ Geräteeditor“* die Option *„Klassische Darstellung der Kommunikationseinstellungen verwenden “* aktivieren.

Hinweis Installation4

HINWEIS!

Stellen Sie die Sprache für die Benutzeroberfläche und für die Online-Hilfe (einschließlich Anwenderhandbuch) auf die Sprache von CODESYS Safety ein, das auf Ihrem System in Betrieb ist. Die Spracheinstellungen nehmen Sie vor in: *„Tools ➔ Optionen ➔ Internationale Einstellungen“*

2.7 Umgang mit Fehlermeldungen aus CODESYS Safety

Anwenderverhalten bei undefiniertem oder sicherheitsrelevantem Fehler

HINWEIS!

Der Anwender ist dazu verpflichtet, sicherheitsrelevante Fehler unverzüglich an den jeweiligen Gerätehersteller zu melden.

HINWEIS!

Das Auftreten eines undefinierten Fehlers muss unverzüglich an den Gerätehersteller gemeldet werden!

Installationsfehler beim Arbeiten mit CODESYS Safety

Safety Installationsprüfung

Wenn die Meldung erscheint, dass die aktuelle CODESYS Safety Installation ungültig ist und die Applikation beendet wird, kann einer der folgenden Fälle aufgetreten sein:

- *Die Safety-Erweiterung wurde auf eine CODESYS-Version installiert, die nicht für CODESYS Safety freigegeben ist.*

  Lösung des Problems: Verwenden Sie eine CODESYS-Version, die für CODESYS Safety freigegeben ist.
- *Ein zusätzlich installiertes Package hat die gültige Kombination von CODESYS und CODESYS Safety verändert.*

  Lösung des Problems: Deinstallieren Sie das zusätzliche Package.
- *Die Installation der Safety-Erweiterung ist kaputt.*

  Lösung des Problems: Deinstallieren Sie das Package CODESYS Safety Extension und installieren Sie es neu.
- *Die Installation von CODESYS ist kaputt.*

  Lösung des Problems: Deinstallieren Sie CODESYS und installieren Sie es neu und installieren Sie darauf erneut das Package CODESYS Safety Extension.

Interner Softwarefehler beim Öffnen von Projekten

VORSICHT!

Öffnen von Projekten

Erscheint beim Öffnen eines Projekts die Meldung, dass das Projekt nur unvollständig geladen wurde, so darf keine in diesem Handbuch beschriebene Tätigkeit mit einer Sicherheitsapplikation dieses Projekts durchgeführt werden.

Die genannte Meldung (unvollständig geladenes Projekt) ist möglich, wenn das Projekt mit einem anderem Profil gespeichert wurde, wenn Lizenzschlüssel für lizensierte Plug-Ins fehlen, oder wenn die Installation geändert wurde.

Sie müssen in diesem Fall folgendeTätigkeiten ausführen, damit das Projekt sich öffnet, ohne dass die beschriebene Meldung erscheint.

1. Projekt mit richtigem Profil speichern

oder Lizenz besorgen

oder Installation anpassen

2.CODESYS neu starten

3. Projekt öffnen

→ Projekt öffnet ohne Fehlermeldung

Interner Softwarefehler während des Arbeitens mit CODESYS

Trotz aller Sorgfalt in der Produktentwicklung kann das Vorkommen eines Softwarefehlers in CODESYS Safety nicht absolut ausgeschlossen werden.

HINWEIS!

Softwarefehler bitte an Hersteller melden

VORSICHT!

Sollten Sie eine Meldung der Selbstüberwachung der Software sehen (*„Assertionsfehler“*, *„Ausnahme“*, *„Unbehandelte Ausnahme“* oder *„Gefangene Ausnahme “*) darf nicht am Safety-Projekt weitergearbeitet werden. CODESYS Safety muss unbedingt beendet (ggf. mit Speichern des Projekts) und neu gestartet werden.

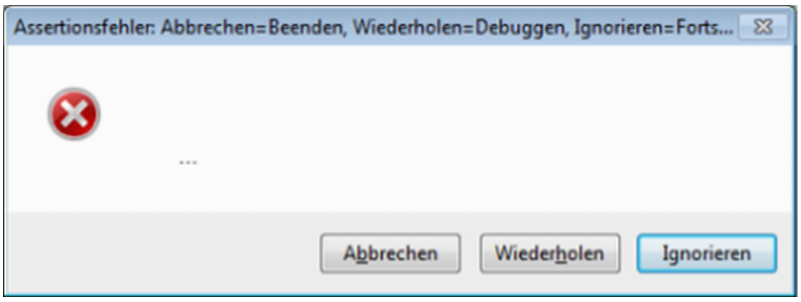


Abb. 1: Beispiel für Fehlermeldung 'Assertionsfehler'

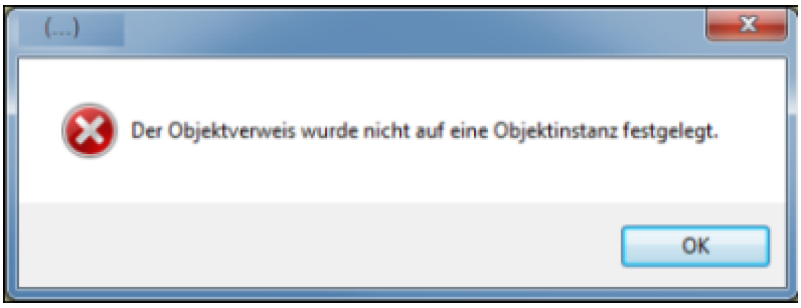


Abb. 2: Beispiel für Fehlermeldung einer Ausnahme

3 Hintergrundwissen

3.1 Sicherheitsnormen

Es wird der Stand der Normen zum 01.09.2017 berücksichtigt.

Die gültigen Sicherheitsnormen

Maschinenhersteller und Betreiber technischer Anlagen sind dafür verantwortlich, dass Maschinen bzw. Anlagen grundlegenden Sicherheits- und Gesundheitsschutzanforderungen genügen. Werden die jeweils maßgeblichen Normen eingehalten, so wird gewährleistet, dass der Maschinen- bzw. Gerätehersteller, oder der Errichter einer Anlage seine Sorgfaltspflicht erfüllt hat und damit der Stand der Technik erreicht ist.

Die Maschinenrichtlinie regelt ein einheitliches Sicherheitsniveau zur Unfallverhütung für Maschinen beim Inverkehrbringen innerhalb des europäischen Wirtschaftsraums (EWR) sowie der Schweiz und der Türkei.

Die Nachweispflicht für die Einhaltung der zu Grunde liegenden Anforderungen liegt beim Gerätehersteller. Der Nachweis kann durch Einhalten der entsprechenden Europäischen Normen (EN) erbracht werden. Diese Normen ("harmonisierte Normen") sind im Amtsblatt der EU aufgelistet.

- EN IEC 62061 und EN ISO 13849: Für den funktionalen Teil der Sicherheit von Steuerungssystemen
- Spezifische harmonisierte Normen ("Produktnormen"): Für den funktionalen Teil der Sicherheit einzelner Maschinentypen (Werkzeugmaschinen, Holzbearbeitungsmaschinen, Druckmaschinen, Verpackungsmaschinen etc.)

Für den Maschinenbauer haben diese Normen die absolut höhere Priorität. Mit Einhaltung dieser Normen kann davon ausgegangen werden, dass die grundlegenden Sicherheitsanforderungen der Maschinenrichtlinie erfüllt werden.

CODESYS Safety ist dafür geeignet, die Sicherheitsfunktionen der Maschine konform zu diesen Normen zu programmieren:

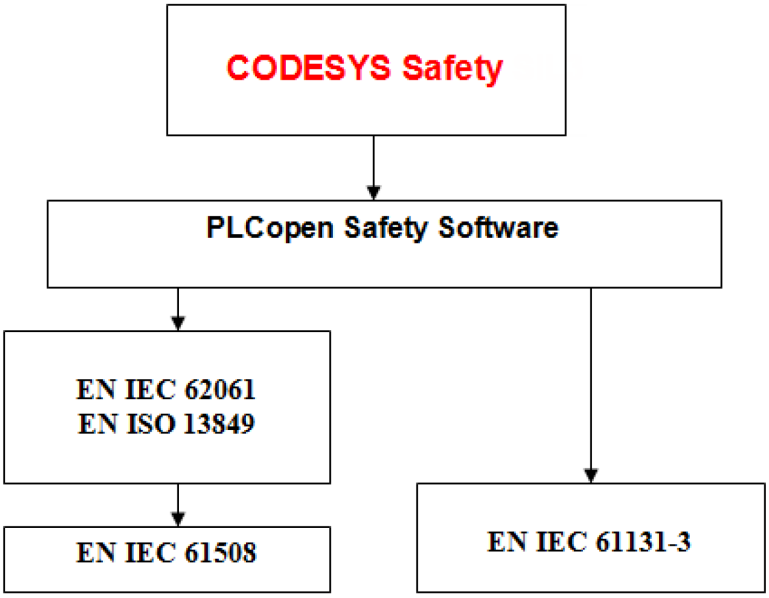


Abb. 3: CODESYS Safety Übersicht Sicherheitsnormen

Norm / Standard	Beschreibung
EN IEC 61508	Sicherheits-Grundnorm, die das Gesamtsystem betrachtet. Definiert SIL, LVL, organisatorische Anforderungen
EN IEC 62061 EN ISO 13849	Bereichs-Sicherheitsnorm für funktionale Sicherheit von Steuerungssystemen. Harmonisierte EU Normen EN IEC 62061: Abstufung nach SIL EN ISO 13849: Abstufung nach PL
EN IEC 61131-3	Norm für Speicherprogrammierbare Steuerungen. Definiert 5 Programmiersprachen, unter anderem auch FUP
PLCopen "Safety Software"	Programmierrichtlinien Trennung Signale (sicher - unsicher) Zertifizierte Bausteine

Sicherheitsstufen

EN IEC 62061 und EN ISO 13849 stellen je nach Höhe des Schadenrisikos unterschiedliche Anforderungen an die Zuverlässigkeit der programmierten Sicherheitsfunktionen der Maschinen.

EN ISO 13849 definiert dazu verschiedene Zuverlässigkeitsstufen genannt PL-a bis PL-e.

EN IEC 62061 verweist dazu auf die Sicherheitsgrundnorm EN IEC 61508, wo die Zuverlässigkeitsstufen SIL1 bis SIL4 definiert sind.

Die Norm EN IEC 61508 beschreibt die allgemeine Zertifizierung elektrischer, elektronischer und programmierbar elektronischer Systeme (kurz: E/E/PES). Nach dieser Norm können auch integrierbare Softwareteile zertifiziert werden. Sie ist die Grundnorm, auf welche EN IEC 62061 und EN ISO 13849 verweisen.

Die beiden Stufensysteme entsprechen sich etwa wie folgt:

Tab. 1: Entsprechung von PL zu SIL

PL	SIL
a	keine Entsprechung
b	1
c	1
d	2
e	3
keine Entsprechung	4

CODESYS Safety ist geeignet für Maschinen der Stufen SIL1 bis SIL3 und PL-a bis PL-e

Anforderungen an die Software-Entwicklung

Damit die programmierte Sicherheitsfunktion die notwendige Zuverlässigkeitsstufe erreicht, muss bereits die Entwicklung der Steuerungssoftware bestimmten Ansprüchen genügen.

Insgesamt definiert EN IEC 61508 den Zyklus des Gesamtsystems mit 16 Phasen, beginnend mit Phase1: "Konzept" und endend mit Phase16: "Außerbetriebnahme". CODESYS Safety wird als Tool in EN IEC 61508 Phase10 "Realisierung, Sicherheits-Software Lebenszyklus" eingesetzt.

Die Norm fordert unter anderem die Definition eines Sicherheitsplans, das Erstellen von Software-Spezifikationen, die Festlegung von Programmierrichtlinien und die Planung der Testaktivitäten.

Durch die Verwendung einer Programmiersprache mit eingeschränktem Sprachumfang (Limited Variability Language, kurz: LVL) reduzieren sich die Anforderungen der Norm. Denn mit diesen Sprachen lässt sich die Programmlogik klarer ausdrücken, leichter verstehen und verifizieren. Diese leichteren Anforderungen sind in den Normen EN IEC 62061 und EN ISO 13849 definiert. (Bei Verwendung von uneingeschränkten Programmiersprachen verweisen beide Normen auf die Grundnorm 61508 (mit allgemeinen, komplexen Anforderungen))

Zu den LVLs zählen insbesondere die grafischen Sprachen der EN IEC 61131-3, wie z.B. FUP oder KOP. Nicht dazu gehören die textuellen Sprachen, wie C, ST, Assembler oder AWL.

3.2 Speicherprogrammierbare Steuerungen

Das PLCopen-Konzept für Sicherheitssoftware

Für Maschinenbauer ist es oft kostenintensiv und kompliziert alle Normen einzuhalten, da sie mit mehreren sicherheitsbezogenen Normen konfrontiert werden. Bei der Konstruktion von Maschinen und Anlagen wird der sicherheitsgerichtete und der funktionale Teil der Applikation häufig voneinander getrennt entwickelt und erst am Schluss zusammengeführt. Auf diese Weise werden die Sicherheitsaspekte zwar berücksichtigt, sie werden jedoch nicht von Anfang an in die gesamte Philosophie des Systems integriert. Moderne, software-basierte Steuerungssysteme und Sicherheitsprotokollschichten für industrielle Feldbusse erlauben inzwischen die verzahnte Entwicklung von sicherheitsgerichtetem und funktionalem Teil der Applikation von Anfang an.

Gemäß aller maßgeblichen Sicherheitsnormen sind die allgemeinen Grundanforderungen an eine Sicherheitsapplikation:

- Unterscheidung zwischen Sicherheits- und Nicht-Sicherheits-Funktionalitäten
- Verwendung von geeigneten Programmiersprachen bzw. Teilmengen von Programmiersprachen
- Verwendung validierter Softwarebausteine
- Verwendung von geeigneten Programmierrichtlinien
- Verwendung von anerkannten Fehler-reduzierenden Maßnahmen für den Lebenszyklus der sicherheitsgerichteten Software

Um den Aufwand für das Erfüllen dieser hohen Anforderungen zu reduzieren, hat das PLCopen Komittee TC eine Lösung für alle Punkte erarbeitet. Diese standardisierte Lösung wird von CODESYS Safety in allen Aspekten unterstützt.

PLCopen-Konzept 1: Integrierte Systeme und integrierte Entwicklung

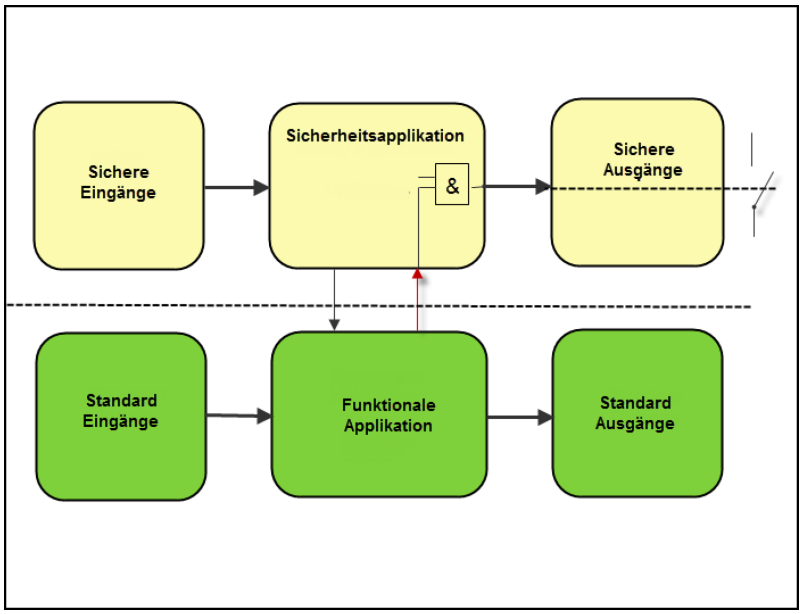


Abb. 4: PLCopen Software-Architekturmodell

Der sicherheitsgerichtete und der funktionale Teil der Applikation wird im Software-Architekturmodell (Abb. 4) als funktionale Applikation bzw. Sicherheitsapplikation dargestellt. Bei CODESYS Safety läuft die funktionale Applikation auf der Standardsteuerung und die Sicherheitsapplikation auf der Sicherheitssteuerung. Ziel der PLCopen ist, statt verschiedener Entwicklungsumgebungen für die jeweiligen Teilapplikationen, die Entwicklungsumgebung der funktionalen Applikation um einen integralen Teil für die Sicherheitsapplikation zu erweitern.

- Die Sicherheitsapplikation bearbeitet die sicheren Eingänge und steuert die sicheren Ausgänge, während die funktionale Applikation die nicht-sicheren Eingänge verarbeitet und die nicht sicheren Ausgänge steuert. Die funktionale Applikation besitzt Lesezugriff auf die sicheren Eingänge und auf die globalen Variablen (angezeigt durch den linken Pfeil).
- Die nicht-sicheren Signale können in der Sicherheitsapplikation nur zur Kontrolle des Programmflusses verwendet werden und können nicht direkt mit sicheren Ausgängen verbunden werden (angezeigt durch den rechten Pfeil und den UND-Operator).

CODESYS Safety ist so eine Erweiterung der Standard-Entwicklungsumgebung CODESYS (für funktionale Applikationen) um einen Teil für die Entwicklung von Sicherheitsapplikationen. Die gesamte Applikation der Maschine bestehend aus funktionaler Applikation und Sicherheitsapplikation mit entsprechenden Ein- und Ausgängen und Datenaustausch zwischen ihnen wird in einer gemeinsamen Entwicklungsumgebung in einem gemeinsamen Projekt entwickelt.

Erweiterung des PLCopen-Konzepts mit Querkommunikation

Die Querkommunikation zwischen Steuerungen stellt eine Erweiterung der klassischen 3-stufigen Architektur Eingänge--Steuerung--Ausgänge wie im PLCopen Software Architekturmodell dar. Durch Verwendung von Safety Netzwerkvariablen in der Sicherheitsapplikation ist auch eine sichere Querkommunikation zwischen den Sicherheitssteuerungen möglich (sofern sie das Feature CODESYS Safety Netzwerkvariablen unterstützen).

Im folgenden Architekturmodell wird die Verwendung von Safety Netzwerkvariablen berücksichtigt. Es gibt eine zweite Sicherheitsapplikation und somit auch eine weitere funktionale Applikation. Die Signale der sicheren Eingänge werden von den PLCopen FBs ausgewertet, in einer der beiden Sicherheitsapplikationen verarbeitet und anschließend in den PLCopen FBs für die Ansteuerung der sicheren Ausgänge ausgewertet.

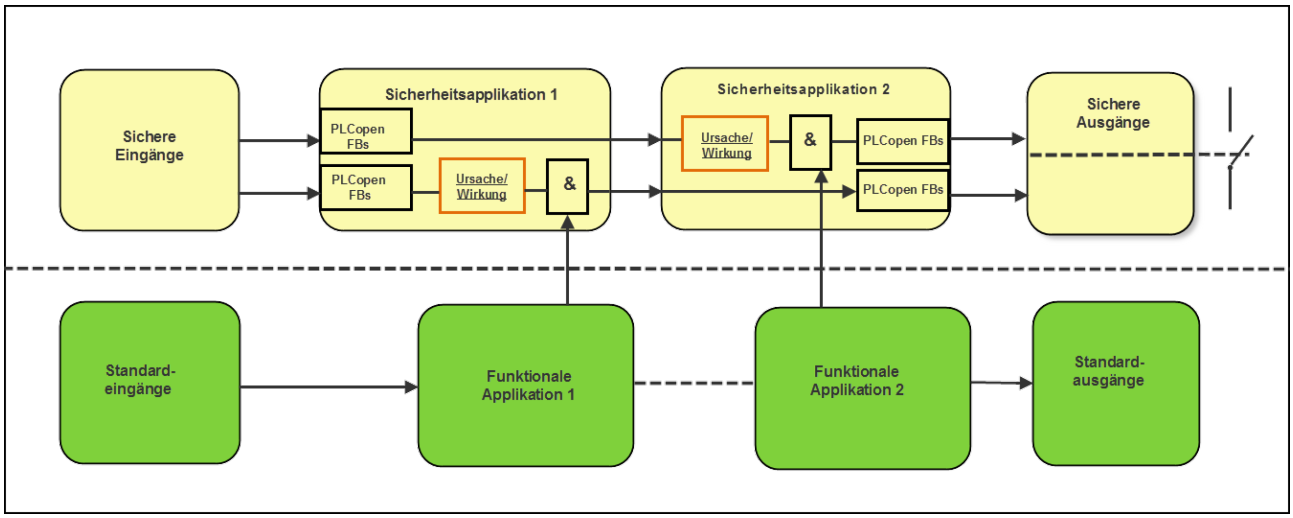


Abb. 5: Software-Architekturmodell mit Querkommunikation

Bei Verwendung von Netzwerkvariablen können die Daten auf 2 verschiedene Arten zwischen Sicherheitsapplikationen kommuniziert werden:

- Ausgänge der Sensorüberwachung, die durch PLCopen Bausteine realisiert werden
- Anforderung einer Sicherheitsfunktion, die im Modell als Ursache/Wirkung darstellt ist.

PLCopen-Konzept 2: Trennung der sicheren Aspekte von nicht-sicheren Aspekten

Um deutlich zwischen sicherheitsrelevanten Signalen und Standardsignalen zu unterscheiden, wurden neue Datentypen mit der Kennzeichnung "SAFE" definiert. Dadurch erkennt der Entwickler, dass die Signale sicherheitsrelevant sind und mit besonderer Sorgfalt behandelt werden müssen. Desweiteren kann durch diese Kennzeichnung die Verbindung von Daten automatisch geprüft werden, um alle unerlaubten Verbindungen zwischen Standardsignalen und sicherheitsrelevanten Signalen zu entdecken. Obwohl die SAFE-Datentypen nicht garantieren können, dass der Signal-Status sicher ist (zum Beispiel bei nicht korrekt verdrahteter Peripherie), ist es dennoch ein organisatorisches Tool zur Minimierung von Fehlern im Anwendungsprogramm. Dies vereinfacht und verkürzt die Verifikation des Signalflusses. SAFE-Datentypen können innerhalb einer sicherheitsrelevanten Umgebung verwendet werden. Sie sollten zur Unterscheidung von sicheren und nicht-sicheren Signalen verwendet werden mit dem Ziel, die Validierung und Zertifizierung zu vereinfachen.

CODESYS Safety behandelt nur Signale von und zu sicheren Feldgeräten als Werte eines SAFE-Datentyps und überprüft automatisch die korrekte Verknüpfung von SAFE-Datentypen in der Applikationslogik.

PLCopen-Konzept 3: geeignete Sprache mit Programmierrichtlinien

Die PLCopen hat einige Regeln zur einheitlichen Definition sicherheitsbezogener Funktionsbausteine aufgestellt und sie in ihren Bausteinen angewendet. Diese "General Rufes for safety-related function blocks" gelten für PLCopen-konforme Funktionsbausteine.

Als Sprachen werden von der PLCopen der Funktionsplan (FUP) und die Anweisungsliste (AWL) bervorzugt, da mit diesen Sprachen Programmiereinheiten relativ leicht erstellt, gut gelesen und geprüft werden können. Die Ansicht des Programmcodes ist klar strukturiert und ähnelt der traditionellen diskreten Verdrahtung von Sicherheitsbaugruppen. In CODESYS Safety steht zur Implementierung des Programmcodes der Sicherheitsapplikation die Programmiersprache FUP zur Verfügung.

Aufgrund der Anforderungen nach Programmierrichtlinien hat die PLCopen Arbeitsgruppe "Safety Software" Programmierregeln in zwei Abstufungen definiert: Basic Level und Extended Level.

Basic Level: Der grundlegende Ansatz ist, dass das Sicherheitsprogramm nur aus zertifizierten Funktionsbausteinen besteht, die in grafischer Form leicht miteinander verbunden werden können. Wenn zusätzlich die Struktur der Verbindung zwischen den FBs eingeschränkt wird, kann eine Ansicht im Stil moderner I (i.e. Integrierte Entwicklungsumgebung) entwickelt werden, die ähnlich der traditionellen diskreten Verdrahtung von Sicherheitsbaugruppen funktioniert. Die Programme haben eine klare Struktur und können leicht gelesen werden. Darüber hinaus wird die Zeit für die Freigabe des Programms signifikant verkürzt, da es aus bereits zertifizierten Bausteinen besteht.

Extended Level: Bei Projekten, für welche der gegenwärtige Stand der zertifizierten Funktonsbausteine nicht ausreichend ist, kann der Anwender die erforderlichen Bausteine (oder sogar das Programm) im Extended Level erstellen. Hierfür wird ein erweiterter Sprachumfang zur Verfügung gestellt. Allerdings kann die Validierung der Funktionalitäten dieser Bausteine und Programme wesentlich aufwendiger, und demzufolge zeitaufwendiger sein, da die Programme dem vollständigen Verifikationsprozess unterliegen. Wenn die Bausteine verifiziert/validiert sind, können sie, zusammen mit den oben beschriebenen Vorteilen im Basic Level verwendet werden.

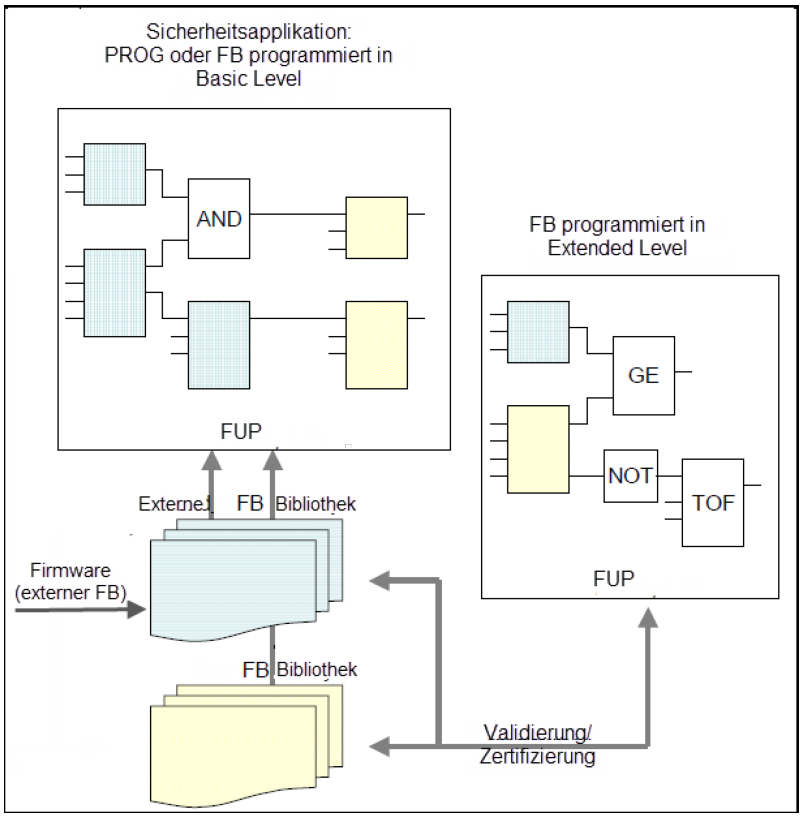


Abb. 6: Empfohlener Anwendungsbereich der beiden Programmierlevel

CODESYS Safety unterstützt diese PLCopen-Regeln durch automatische Prüfung dieser Programmierrichtlinien und der Verknüpfungsregeln. Es prüft Bausteine, die als Basic markiert sind, gegen die Basic-Regeln und Bausteine, die als Extended markiert sind, gegen die Extended-Regeln. Bei Verstoß lässt sich eine Applikation nicht auf die Steuerung laden. So ist das Einhalten der Regeln automatisch nachgewiesen.

Nur noch einige Regeln müssen manuell geprüft werden (siehe *Kapitel 9.3.4 „Manuelle Prüfung der Programmierrichtlinien“ auf Seite 200*).

PLCopen-Konzept 4: standardisierte Funktionsbausteine

Schließlich hat die PLCopen einen Satz von Funktionsbausteinen definiert, welche immer wiederkehrende Funktionalitäten von Sicherheitsapplikationen definieren. Dieser Satz von Funktionsbausteinen unterstützt die typische Safety Ausstattung in der Fertigungsautomation. Die unten aufgeführte Tabelle (*Tab. 2 „Zuordnung der Sicherheitsvorrichtungen zu den PLCopen FBs“ auf Seite 27*) zeigt wie die Funktionalitäten der Safety Ausstattung den einzelnen PLCopen Bausteinen zugeordnet werden.

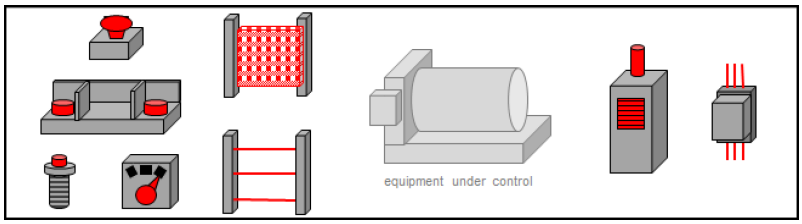


Abb. 7: Gerätetypen der Safety Ausstattung in der Fertigungsautomation

Tab. 2: Zuordnung der Sicherheitsvorrichtungen zu den PLCopen FBs

Bedienelemente	Notaus-Schalter (EN 418)	SF_EmergencyStop
	Zweihandbedienung (EN 574)	SF_TwoHandControlTypeII / III
	Zustimmtaster, Betriebsartenwahlschalter (IEC 60204)	SF_EnableSwitch, SF_ModeSelector
Sensoren	Schutztür (EN 953/ 1088)	SF_SafetyGuardMonitoring, SF_SafetyGuardLocking (Safety Guard Interlocking with Locking)
	Lichtschranke (IEC 61496 / 62046)	SF_ESPE, SF_Testable SafetySensor
	Muting-Sensoren (IEC 61496 / 62046)	SF_MutingSeq, SF_MutingPar, SF_MutingPar_2Sensor
Ausgänge	Antrieb/Ventil mit sicherem Zustand oder sicherer Funktion (IEC 60204)	SF_SafetyRequest
	Relais mit Rückführkontakt (IEC 60204)	SF_EDM
	unsicher geschalteter Ausgang mit Überwachung (IEC 60204)	SF_OutControl

IEC 61131-3

Die IEC 61131-3 beschreibt die Programmiersprachen, die zur Programmierung von Steuerungen eingesetzt werden. Sie definiert die Sprache FUP, von der die PLCopen Teilmengen definiert hat und für die die PLCopen Programmierregeln definiert hat.

3.3 Industrielle Kommunikation

Feldgeräte

Grundsätzlich erfolgt die Kommunikation der Sicherheitssteuerung mit Ein- und Ausgängen, also mit Sensoren und Aktoren von Maschinen und Anlagen, über verschiedene Feldbussysteme mit einer Standardsteuerung als Feldbusmaster. Die Feldgeräte werden über die Feldbusse zyklisch angesteuert, die Prozesswerte (Eingangs- und Ausgangswerte) werden zyklisch als PDOs übertragen.

Modulare Feldgeräte

Ein modulares Feldgerät ist ein Gerät, dessen Prozessabbild mehreren Modulen zugeordnet ist. Das modulare Feldgerät kann eine Kopfstation mit Steckmodulen sein. Dann wird sie auch Koppler genannt.

Das modulare Feldgerät kann auch eine (untergeordnete) Steuerung sein. Diese Steuerung ist dann ein modulares Gerät im (übergeordneten) Feldbus, das die Slaveseite des Feldbusprotokolls implementiert. Die Gliederung des Prozessabbilds in Gerätemodule wird durch die Programmierung der Steuerung definiert.

Sichere Feldgeräte

Die Kommunikation zwischen der Sicherheitssteuerung und den sicheren Feldgeräten erfolgt über Standardfeldbusse oder auch über IP-Netzwerke zwischen Steuerungen. Über diese wird ein Safety-Protokoll zur Absicherung gelegt, so dass das unterlagerte Kommunikationsnetzwerk als schwarzer Kanal betrachtet werden kann, zu welchem auch die Standardsteuerung als Feldbusmaster bzw. modulares Feldgerät gehört.

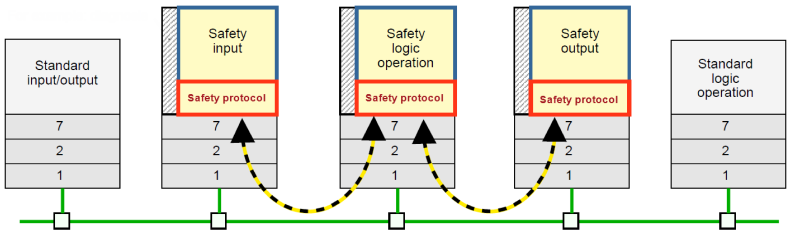


Abb. 8: Kommunikation der Sicherheitssteuerung mit den sicheren Feldgeräten (nach PROFIsafe - Profile for Safety Technology on PROFIBUS DP and PROFINET IO, Version 2.6 - October 2013)

: Schwarzer Kanal

: Nicht sicherheitsbezogene Funktionen, zum Beispiel Diagnose

: Das Safety Protokoll umfasst: Adressierung, Messung der Überwachungszeit, MonitoringNumber

: Die sicheren E/As und Sicherheitslogik sind sicherheitsbezogen aber nicht Teil des Safety Protokolls.

Eigenschaften und Zwecke der Safety-Protokolle:

- Mittels des Safety-Protokolls können Fehler in der Kommunikation festgestellt werden. Die sicheren Feldgeräte beziehungsweise die Sicherheitssteuerung reagieren darauf und stellen Sicherheit her.
- Prozesswerte von sicheren Eingängen zur Sicherheitssteuerung und von der Sicherheitssteuerung zur sicheren Ausgängen werden nur übertragen, wenn eine korrekte, bestätigte, ungestörte Kommunikationsverbindung vorliegt. Ansonsten werden statt der Prozesswerte Nullen als Ersatzwerte übertragen, beziehungsweise die Applikation rechnet statt mit Prozesswerten mit FALSE bzw. 0 als Ersatzwert, was nach dem Prinzip der Failsafe-Logik zu einer sicheren Reaktion führt.
- Der Safety-Master (SPS) steuert seine Safety-Slaves (Feldgerät beziehungsweise untergeordnete SPS) "modulweise" per Steuersignale. Welche Steuersignale es gibt, ist protokollabhängig. Typischerweise gibt es zumindest ein Steuerbit, um einem sicheren Ausgangsmodul den sicheren Zustand zu befehlen, bzw. um die sicheren Ausgänge freizugeben.
- Die Safety-Slaves informieren den Safety-Master "modulweise" per Statussignale. Welche Status es gibt, ist protokollabhängig. Typischerweise gibt es zumindest ein Statusbit für den Fall, dass aktuell keine Prozesseingangssignale an den Master geliefert werden (Slave hat einen Fehler erkannt in der Kommunikation, intern in sich selbst, im angeschlossenen Sensor oder im Fall einer Steuerung als Feldgerät im untergeordneten Feldbus).

Eine wesentliche Eigenschaft der Safety-Protokolle ist die eindeutige Adressierung im gesamten Kommunikationsnetzwerk. Die Adressen eines Feldbusprotokolls müssen nicht nur innerhalb eines Steuerungsprojekts eindeutig sein, sondern über das gesamte Netzwerk mit allen über die Feldbusse kommunizierenden Steuerungen und übergeordneten Steuerungen.

Topologien

Für den sicherheitsbezogenen Teil des Steuerungssystems einer Maschine können, abhängig von den Kommunikationsmöglichkeiten der eingesetzten Sicherheitssteuerungen und ihrer Standardsteuerungen, verschiedene Topologien gewählt werden:

- **T1**: Die Sicherheitsfunktionen der Maschine werden innerhalb eines oder mehrerer Feldbusse jeweils lokal mit einer einzigen Sicherheitssteuerung am Feldbusmaster realisiert. Dabei wird eine einzelne Sicherheitsfunktion jeweils innerhalb eines Feldbusses realisiert und die Gesamtheit der Sicherheitsfunktionen wird somit auf mehreren Feldbussen realisiert.

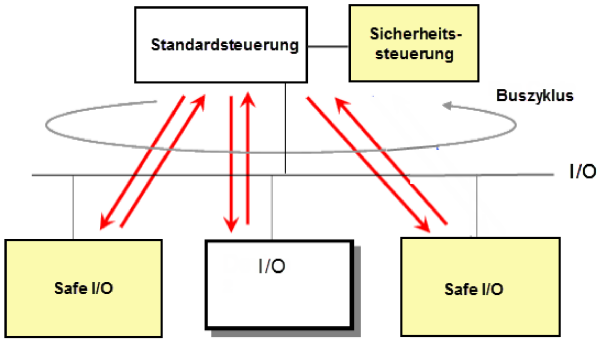


Abb. 9: Topologie T1 (nach PROFIsafe - Profile for Safety Technology on PROFIBUS DP and PROFINET IO, Version 2.6 - October 2013)

- **T2**: Die Sicherheitsfunktionen der Maschine werden innerhalb eines oder mehrere Feldbusse jeweils lokal mit mehreren dezentralen, nicht untereinander kommunizierenden Sicherheitssteuerungen realisiert. Jede Sicherheitsfunktion wird jeweils in einem Feldbus und auf einer Sicherheitssteuerung realisiert.
- **T3**: Die Sicherheitsfunktionen der Maschine werden in mehreren Feldbussen mit mehreren per Safety Netzwerkvariablen untereinander kommunizierenden Sicherheitssteuerungen realisiert.
- **T4**: Die CODESYS Standardsteuerung + Sicherheitssteuerung stellen ein PROFIsafe F-Device dar, das zusammen mit anderen E/A-Modulen über den übergeordneten Feldbus mit der übergeordneten Steuerung kommuniziert. Am übergeordneten Feldbus können mehrere PROFIsafe F-Devices als E/A-Module parallel eingefügt sein, die Sicherheitsfunktionen implementieren. Diese Standardsteuerungen verfügen jeweils über einen PROFINET-Device-Anschluss, über den sichere Signale der jeweiligen Sicherheitssteuerung mit der übergeordneten Sicherheitssteuerung ausgetauscht werden. In jeder Sicherheitssteuerung einer Standardsteuerung werden innerhalb eines oder mehrerer Feldbusse Sicherheitsfunktionen realisiert.

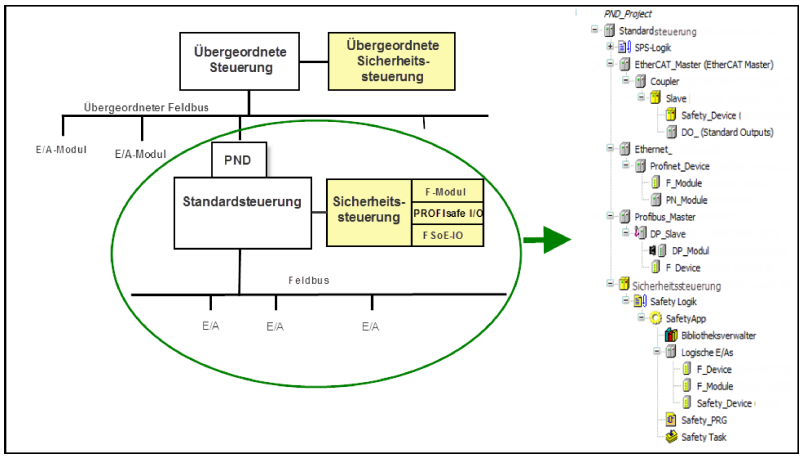


Abb. 10: Topologie T4 in CODESYS Safety

4 Planung des Gesamtsystems

4.1 Übersicht

Aus der Risikoanalyse für die Maschine werden die nötigen Sicherheitsfunktionen ermittelt. Für Realisierung der Sicherheitsfunktionen werden bestimmte Sensoren und Aktoren benötigt, die an einen Feldbus angeschlossen werden, und eine Sicherheitssteuerung, die die Sensoren und Aktoren verknüpfen.

Dabei sind mehrere Punkte gleichzeitig zu berücksichtigen:

1. **Systemstruktur**. Planung der Geräte (Sicherheitssteuerungen, Feldbusse, Sensoren, Aktoren) und Allokation der Sicherheitsfunktionen auf diese. Siehe *Kapitel 4.2 „Strukturierung des Steuerungssystems“ auf Seite 34*. Daraus ergeben sich die Ein-/Ausgangssignale und funktionalen Anforderungen für die zu entwickelnden Sicherheitsapplikationen der einzelnen Sicherheitssteuerungen.

2. **Reaktionszeiten**. Damit die für eine Sicherheitsfunktion nötigen Reaktionszeiten in der Maschine erreicht werden, müssen entsprechend performante Geräte und entsprechende Zeiteinstellungen in den Geräten eingeplant werden. Siehe *Kapitel 4.3.3 „Reaktionszeiten bei F-Device-Steuerungen“ auf Seite 43*. Daraus ergeben sich Anforderungen an Konfigurationseinstellungen in den Sicherheitsapplikationen.

3. **Eindeutigkeit**. Für die Kommunikation zwischen Sicherheitssteuerungen und sicheren Feldgeräten und zwischen Sicherheitssteuerungen untereinander müssen je nach Feldbus verschiedene eindeutige Adressen und IDs vergeben werden. Bei den Topologien T2 und T3 und T4 muss dieses über die einzelne Sicherheitssteuerungen hinweg geplant werden. Siehe *Kapitel 4.4 „Planung der Adressen“ auf Seite 45*. Daraus ergeben sich Anforderungen an Konfigurationseinstellungen in den Sicherheitsapplikationen.

4. **Systemanforderungen**. Für Ethernet-basierte Feldbusse (zum Beispiel EtherCAT, PROFINET) und für Netzwerkvariablen, können zusätzliche Systemanforderungen an Abgrenzung des Netzwerks, Teilnehmer im Netzwerk und die Kommunikationsfehlerrate entstehen. Für die spezifischen Anforderungen siehe die Abschnitte "Systemanforderungen" in *Kapitel 14 „Feldbusse und Netzwerkvariablen“ auf Seite 271*.

5. **Zugriffschutz**: Im Betrieb erfordert der Schutz der Sicherheitsfunktionen Schutzmaßnahmen auf verschiedenen Ebenen. Insbesondere **muss** bei Querkommunikation (T3) das dazu verwendete Netzwerk auf Maschinenebene geschützt werden (siehe *Kapitel 12.1 „IT-Sicherheit im Betrieb“ auf Seite 237*). Die Planung sollte berücksichtigen, dass im Betrieb bestimmte Schutzmaßnahmen ergriffen werden müssen.

4.2 Strukturierung des Steuerungssystems

Für die Realisierung der Sicherheitsfunktionen der Maschine müssen entsprechende Sensoren und Aktoren, entsprechende sichere Feldgeräte in entsprechenden Feldbussen und entsprechende Sicherheitssteuerungen eingeplant werden. Die Sicherheitsfunktionen müssen diesen zugeordnet werden (Allokation).

HINWEIS!
Die maximale Anzahl sicherer Feldgeräte für eine Sicherheitsfunktion kann durch den Feldbus begrenzt sein, z. B. auf 100 bei PROFIsafe (siehe ↳ *Kapitel 14.2 „PROFIsafe“ auf Seite 276*).

Typischerweise wird eine Sicherheitsfunktion durch eine Sicherheitssteuerung und die ihr exklusiv zugeordneten sicheren Feldgeräte im gleichen Feldbus realisiert (immer so bei T1, T2). Die für die Sicherheitsfunktion einer Sicherheitssteuerung benötigten Sensoren und Aktoren sollten der Einfachheit halber genau an dieser Sicherheitssteuerung (und damit der Standardsteuerung) angeschlossen werden.

HINWEIS!
Die maximale Anzahl sicherer Feldgeräte im gleichen Feldbus oder von Sicherheitssteuerungen im gleichen Feldbus (nur T2) kann feldbusspezifisch begrenzt sein. Siehe ↳ *Kapitel 14 „Feldbusse und Netzwerkvariablen“ auf Seite 271*.

Jede geplante Sicherheitssteuerung benötigt eine Sicherheitsapplikation. Für diese muss ein Software-Entwicklungsprozess gestartet werden. Für die Sicherheitsapplikationen muss eine Spezifikation erstellt werden (nicht notwendigerweise als separates Dokument). Zur Verifikation muss für jede Sicherheitsapplikation ein funktionaler bzw. BlackBox Test gegen die Spezifikation eingeplant und in der Verifikationsphase durchgeführt werden. Siehe ↳ *Kapitel 9.4.3 „Vollständiger Funktionstest der Applikation“ auf Seite 212*.

Wenn mehrere Sicherheitssteuerungen erforderlich sind, die für ihre Sicherheitsfunktionen miteinander kommunizieren müssen, kann T3 oder T4 eingesetzt werden.

Aufteilung mit Querkommunikation

Eine Sicherheitsfunktion kann auf Feldgeräte und auf Sicherheitssteuerungen in mehreren Feldbussen (auch Feldbusse verschiedenen Typs) aufgeteilt werden, wenn man eine Kommunikation zwischen den Sicherheitssteuerungen dieser Feldbusse herstellt (T3). Es muss dann auch das entsprechende Netzwerk zwischen den Steuerungen in die Planung einbezogen werden.

Bei so einer Aufteilung erfolgt die Überwachung der Sensoren, Aktoren und Feldgeräte immer in der Sicherheitssteuerung im gleichen Feldbus wie die Feldgeräte. Auf welcher Sicherheitssteuerung die Entscheidung zum Auslösen der Sicherheitsfunktion (Cause/Effect) zu realisieren ist, muss festgelegt werden (siehe *„PLCopen-Konzept 1: Integrierte Systeme und integrierte Entwicklung“ auf Seite 22*): auf der Sicherheitssteuerung mit den Sensoren oder auf der Sicherheitssteuerung mit den Aktoren; oder sind beide Sicherheitssteuerungen an der Entscheidung beteiligt?

Safety Netzwerkvariablen erfüllen die Funktion der sicheren Querkommunikation in CODESYS Safety. Sie erlauben die Konfiguration und den Betrieb eines sicheren Datenaustauschs zwischen CODESYS Safety Sicherheitssteuerungen in einem Projekt.

Sie sind gemacht für den Fall, dass mehrere Sicherheitssteuerungen für ihre Sicherheitsfunktionen den gleichen Sensor (z.B. einen Notaus-Schalter) benötigen: Sie wählen eine Sicherheitssteuerung aus, an die Sie den Sensor anschließen. Diese Sicherheitssteuerung muss den Sensor überwachen und das überwachte Sensorsignal über den NetVar-Mechanismus an die anderen Sicherheitssteuerungen weiterverteilen.

Das Prinzip: Ein CODESYS Projekt enthält beide Sicherheitssteuerungen mit ihren Standardsteuerungen, und ggf. noch weitere. Die Sicherheitssteuerung mit den Sensoren veröffentlicht die Daten und die anderen Sicherheitssteuerungen des Projekts können die Daten lesen. Der Kommunikationsweg erfolgt dabei über die Standardsteuerungen der entsprechenden Sicherheitssteuerungen.

Aufteilung mit untergeordneter Steuerung

Eine Sicherheitsfunktion kann auf eine übergeordnete Sicherheitssteuerung und untergeordnete Sicherheitssteuerungen aufgeteilt werden, wenn diese als modulare Feldgeräte in den übergeordneten Feldbus eingefügt werden.

Bei so einer Aufteilung erfolgt die Überwachung der Sensoren, Aktoren, und Feldgeräte immer in der Sicherheitssteuerung, an deren Feldbus sie angeschlossen sind.

Es muss festgelegt werden, ob die untergeordnete Sicherheitssteuerung dem Safety-Master als ein Safety-Gerätemodul oder gegliedert in mehrere Safety-Gerätemodule angeboten werden sollen. Dabei sollte jedes Safety-Gerätemodul eine Funktionalitätseinheit mit zykluskonsistenten Daten und gemeinsamen Steuersignalen und Status darstellen.

HINWEIS!
Bei Gliederung in mehrere Safety-Gerätemodule müssen die Funktionen der Sicherheitsapplikation und die Ein- und Ausgangssignale im untergeordneten Feldbus den verschiedenen Modulen zugeordnet werden: Welche Funktionen der Sicherheitsapplikation und welche Ausgangssignale werden durch die Steuerbits welches Moduls vom Safety-Master freigegeben bzw. in den sicheren Zustand versetzt? Erkannte Fehler in welchen Eingängen oder Sensoren werden als Fehlerstatus welches Moduls an den Safety-Master gemeldet?

4.3 Planung der Reaktionszeiten

Die Reaktionszeit einer Sicherheitsfunktion (Gesamtreaktionszeit) ist die Zeit von dem Gefährdungsereignis, das ein Sensor dieser Sicherheitsfunktion detektiert, bis zur Reaktion eines Aktors dieser Sicherheitsfunktion, die die Sicherheit herstellt.

Allgemein gilt, eine Sicherheitsfunktion kann mehrere Aktoren umfassen. Für jede Verknüpfung Sensor -> Aktor muss die Reaktionszeit einzeln berechnet werden. Bei der Reaktionszeit ist zu unterscheiden:

- Die tatsächliche Reaktionszeit R_t in der Maschine für ein bestimmtes Ereignis zu einem Zeitpunkt t
- Die im störungsfreien Betrieb technisch bedingte Obergrenze (Worst-Case) R_{wc} tatsächlicher Reaktionszeiten. D.h. $R_{wc} \geq R_t$.
- Die durch das Sicherheitssystem garantierte Obergrenze der Reaktionszeit R_g auch bei einem Fehler, zu der spätestens eine Reaktion erfolgt, notfalls auch indem die sicheren Ausgänge den sicheren Zustand herstellen. Für den reibungslosen Betrieb (ohne notfallmäßigen sicheren Zustand durch die sicheren Ausgänge) ist $R_g \geq R_{wc}$ erforderlich.
- Die durch Sicherheitsanforderungen geforderte, maximal zulässige Reaktionszeit R_{req} für die funktionale Sicherheit der Maschine. Für die Sicherheit ist $R_{req} \geq R_g$ erforderlich.

Die tatsächlichen Reaktionszeiten, ihr Worst-Case R_{wc}, und damit die kürzesten sinnvollen Reaktionszeitgarantien R_g hängen von der Topologie und von verschiedenen geräte-, feldbus- und applikationsabhängigen Faktoren ab. Einen Teil dieser Faktoren kann der Anwender via CODESYS durch Konfiguration beeinflussen (Zykluszeiten, Überwachungszeiten).

Damit die für eine Sicherheitsfunktion geforderten Reaktionszeiten in der Maschine erreicht werden können und sicher erreicht werden, müssen entsprechende koordinierte Zeiteinstellungen in den verschiedenen Sicherheitsapplikationen inklusive ihrer sicheren Feldgeräte und in den Feldbus-Mastern eingeplant werden.

4.3.1 Reaktionszeiten bei Feldbussteuerungen

Reaktionszeit bei Topologie T1/T2

Für jede Sicherheitsfunktion, die innerhalb eines Feldbusses lokal mit einer Sicherheitssteuerung realisiert ist (T1/T2), lässt sich die tatsächliche Reaktionszeit durch folgende Einstellungen beeinflussen: Zykluszeit der Sicherheitssteuerung und Zykluszeit der Mapping-Applikation der Standardsteuerung.

Jeder Sensor (zum Beispiel Lichtschranke oder Not-Halt-Schalter) wandelt das physikalische Signal in ein elektrisches Signal um. Das Signal kann mit einem Eingangsmodul (zum Beispiel einem sicheren Eingang) verbunden werden, der das Signal in logische Information umwandelt.

Jedes dieser Elemente hat eine minimale (= Bearbeitung) und eine maximale Verzögerungszeit (= Bearbeitung und Warten). Die tatsächliche Verzögerungszeit bzw. das Verzögerungszeitintervall liegt zwischen der minimalen und maximalen Verzögerungszeit.

Aufgrund der Annahme, dass alle Komponenten asynchron arbeiten, ist der schlimmste Fall für jede Komponente die doppelte Verzögerung der Komponente.

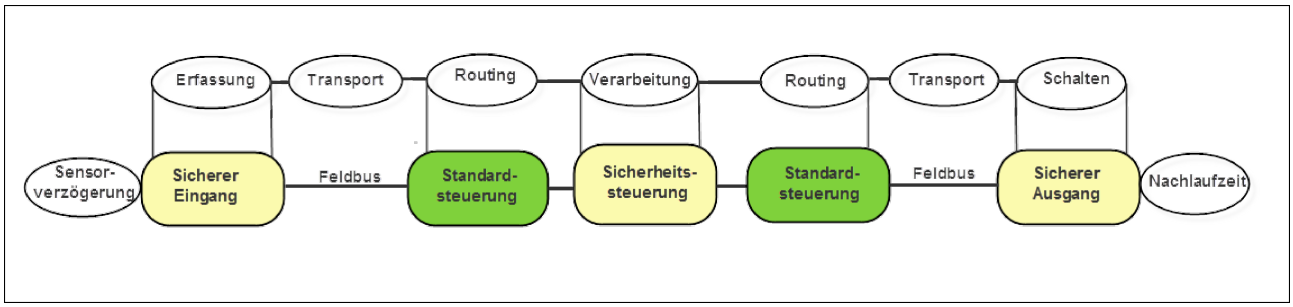


Abb. 11: Zeitfaktoren für Berechnung der tatsächlichen Reaktionszeit R_T

Informationen zu den einzelnen Zeitfaktoren:

- Sensorverzögerung: Der Sensor wandelt einen physikalischen Zustand in ein elektrisches Signal um. Dauer siehe Datenblatt des Sensors
- Erfassung: Das Eingangsmodul wandelt das Sensorsignal in Feldbus-PDOs (logische Information) um. Dauer siehe Datenblatt des Eingangsmoduls
- Feldbustransport: abhängig von der eingestellten Feldbusgeschwindigkeit
- Routing in der Standardsteuerung: abhängig von der Zykluszeit der Mapping-Applikation
- Transport in die Sicherheitssteuerung: systemabhängig
- Verarbeitung: Die Sicherheitssteuerung verarbeitet die logische Information des Eingangsmoduls in logische Ausgangsinformation (Feldbus-PDOs). Dauer abhängig von der Zykluszeit der Sicherheitsapplikation
- Feldbustransport: Dauer abhängig von der eingestellten Feldbusgeschwindigkeit

- Schalten durch den sichereren Ausgang: Umwandlung des Feldbus-PDOs in elektrisches Signal für den Aktor, siehe Datenblatt des Ausgangsmoduls
- Nachlaufzeit: Umsetzen des elektrischen Signals in eine physikalische Aktion durch den Aktor, siehe Datenblatt des Aktors

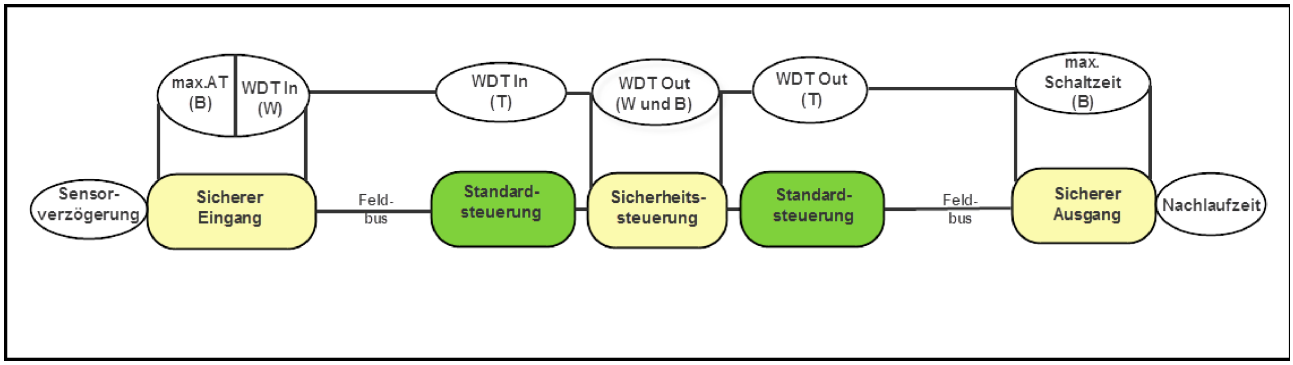


Abb. 12: Zeitfaktoren für die Berechnung der garantierten Obergrenze der Reaktionszeit R_G

WDT Watchdog-Zeit
AT Erfassungsszeit
W Wartzeit
B Bearbeitungszeit
T Transportzeit

Die Faktoren innerhalb jeder Sicherheitskomponente und die Kombination der Faktoren zwischen den Sicherheitskomponenten haben aus Sicherheitsgründen ihre überlagerte Zeitüberwachung gegen ein Zeitlimit, genannt Watchdog-Zeit. Die Zeitüberwachung ergreift die notwendigen Maßnahmen, um den sicheren Zustand zu aktivieren, immer wenn ein Fehler oder eine Störung innerhalb dieser Komponente oder der davorliegenden Komponente auftritt. Diese Watchdog-Zeiten werden in der Sicherheitsapplikation konfiguriert und mit ihnen lässt sich die garantierte Obergrenze R_g der Gesamtreaktionszeit berechnen.

- WDT In: Konfigurierte Watchdog-Zeit der Feldbuskommunikation zwischen sicherem Eingangsmodul und Sicherheitssteuerung
  - Einmal als maximale Wartezeit, die durch das sichere Feldbusprotokoll im Eingangsmodul anfallen kann, bevor es ein geändertes Signal vom Sensor an die Sicherheitssteuerung abschicken kann. So lange muss das Signal des Sensors mindestens anstehen, um zu garantieren, dass es von der Sicherheitssteuerung gesampelt werden kann.
  - Und noch einmal als maximale Transportzeit, in der das abgeschickte Signal in der Sicherheitssteuerung als Eingangssignal ankommt. Diese Zeit überwacht die Performance des Feldbusses und der Standardsteuerung als Vermittler.

    Dauer Feldbustransport + Dauer Routing ≤ WDT In
- WDT Out: Konfigurierte Watchdog-Zeit der Feldbuskommunikation zwischen Sicherheitssteuerung und sicherem Ausgangsmodul
  - Einmal als maximale Wartezeit, inklusive eines Applikationszyklus zur Abbildung der neuen Variablenwerte auf Ausgangssignale, bevor die Sicherheitssteuerung ein geändertes Ausgangssignal an das Ausgangsgerät abschicken kann. So lange muss die Applikation der Sicherheitssteuerung das Signal mindestens anstehen lassen, um zu garantieren, dass es vom Ausgangsmodul gesampelt werden kann.
  - Und noch einmal als maximale Transportzeit, in der das abgeschickte Signal im Ausgangsmodul ankommt. Diese Zeit überwacht die Performance des Feldbusses und der Standardsteuerung als Vermittler.

    Dauer Routing + Dauer Feldbustransport ≤ WDT Out

Die zugesicherte Gesamtreaktionszeit-Obergrenze beträgt:

R_g = Sensorverzögerung

+ max. Erfassungszeit

+ 2 x WDT In

+ 2 x WDT Out

+ max. Schaltzeit

+ Nachlaufzeit

4.3.2 Reaktionszeiten bei Querkommunikation

Reaktionszeit bei Querkommunikation (T3)

Bei einer Sicherheitsfunktion, an der zwei per Safety Netzwerkvariablen kommunizierende Sicherheitssteuerungen beteiligt sind, lässt sich die Gesamtreaktionszeit durch folgende Einstellungen beeinflussen: Zykluszeit der Sender-Sicherheitsapplikation, Zykluszeit der Empfänger-Sicherheitsapplikation, Zykluszeit der Mapping-Applikation der Standardsteuerung.

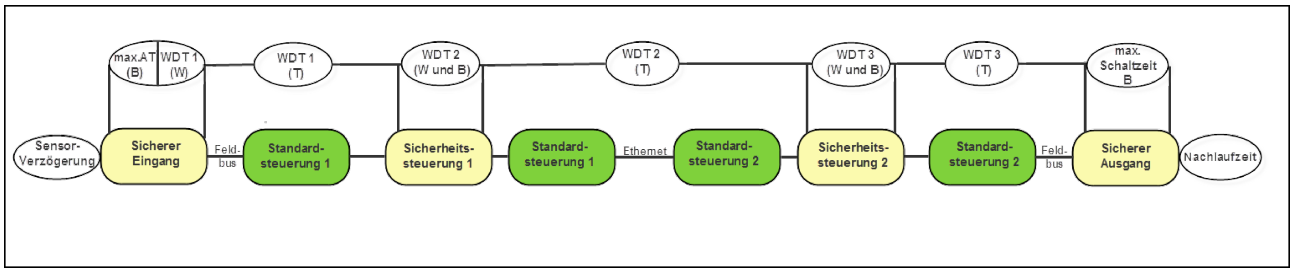


Abb. 13

WDT Watchdog-Zeit
AT Erfassungsszeit
W Wartezeit
B Bearbeitungszeit
T Transportzeit

Reaktionszeitgarantie bei Querkommunikation

Die zugesicherte Gesamtreaktionszeit-Obergrenze beträgt:

R_g = Sensorverzögerung

+ max. Erfassungszeit

+ 2 x WDT 1

+ 2 x WDT 2

+ 2 x WDT 3

+ max. Schaltzeit

+ Nachlaufzeit

In einem Projekt mit Safety Netzwerkvariablen gibt es mehrere Watchdog-Zeiten (siehe *„Reaktionszeit bei Querkommunikation (T3)“ auf Seite 39*), die in Summe die Gesamtreaktionszeit ergeben. Dabei enthält jede Watchdog-Zeit immer mindestens eine Zykluszeit der Applikation, bedingt durch das Zeitmessverfahren in der Sicherheitssteuerung von Zyklusanfang bis Zyklusanfang (siehe *Kapitel 4.3.2 „Reaktionszeiten bei Querkommunikation“ auf Seite 39*).

- WDT 1: Konfigurierte Überwachungszeit der Feldbuskommunikation zwischen sicherem Eingangsmodul und Sicherheitssteuerung 1
  WDT 1 entspricht "WDT In" in *Kapitel 4.3.1 „Reaktionszeiten bei Feldbussteuerungen“ auf Seite 37*.
- WDT 2: Konfigurierte Überwachungszeit der Querkommunikation zwischen Sicherheitssteuerung 1 und Sicherheitssteuerung 2
  - Einmal als maximale Wartezeit, inklusive eines Applikationszyklus zur Abbildung der neuen Eingangssignale auf Netzwerkvariablen, bevor Sicherheitssteuerung 1 neue Variablenwerte an Sicherheitssteuerung 2 abschicken kann. So lange müssen die Werte in Sicherheitssteuerung 1 mindestens anstehen, um zu garantieren, dass sie von Sicherheitssteuerung 2 gesampelt werden können (vgl. *Kapitel 6.6 „Querkommunikation mit Netzwerkvariablen“ auf Seite 150*).
  - Und noch einmal als maximale Transportzeit, in der die abgeschickten Variablenwerte in Sicherheitssteuerung 2 ankommen. Diese Zeit überwacht die Performance der Ethernet-Verbindung und der beiden Standardsteuerungen als Vermittler.
- WDT 3: Konfigurierte Überwachungszeit der Feldbuskommunikation zwischen Sicherheitssteuerung 2 und sicherem Ausgangsmodul
  WDT 3 entspricht "WDT Out" in *Kapitel 4.3.1 „Reaktionszeiten bei Feldbussteuerungen“ auf Seite 37*.

Gefahr des Undersamplings

In jedem Schritt der Sicherheitskette muss ein Signal "lang genug" in der Quelle (sicheres Eingangsmodul, Sicherheitsteuerung 1, Sicherheitsteuerung 2) anliegen (nämlich so lange wie die Watchdog-Zeit der Verbindung ist), damit es auch im Worst-Case zeitlichen Verhaltens eine Chance hat, weitergeschickt zu werden. Das heißt, wenn das Sensorsignal im Eingangsmodul oder eine Netzwerkvariable in Sicherheitsteuerung 1 oder das Ausgangssignal in Sicherheitsteuerung 2 nur kurzzeitig von Wert A auf Wert D wechselt und wieder zurück auf A, kann es sein, dass der Wert D nie übertragen wird (Undersampling). Wenn mit dem verschluckten Wert D eine Sicherheitsfunktion angefordert werden sollte, versagt die Sicherheitsfunktion und die funktionale Sicherheit geht verloren. Wenn ein Glied in der Sicherheitskette das Signal nicht automatisch auf das Mindestmaß für den nächsten Schritt verlängert (WDT k), dann muss das Signal schon im vorherigen Glied entsprechend länger anliegen als bloß WDT k-1.

Zum Beispiel, wenn Sicherheitssteuerung 1 das Signal vom Eingangsmodul (oder einer anderen Sicherheitssteuerung 0) empfängt und unverarbeitet weiter veröffentlicht, ist zu berücksichtigen: Eine Standzeit T_Q des Signals im Eingangsmodul, die gerade sicherstellt, dass das Signal in Sicherheitssteuerung 1 ankommt ($T_Q \geq$ WDT1) reicht noch nicht, um in Sicherheitssteuerung 1 eine danach anschließende Standzeit T_S zu garantieren, die die NVL-

Verbindung zu Sicherheitssteuerung 2 braucht (T_S≥ WDT 2). Die Standzeit muss dazu (T_Q≥ WDT 1 + Zykluszeit Sicherheitssteuerung 1 + WDT 2) sein. Bei einer Sicherheitskette mit mehreren NVL-Verbindungen hintereinander sind die Zeiten aller involvierten Verbindungen und Steuerungen zu kombinieren.

Überwachungszeit bei Querkommunikation

Bei Querkommunikation mit Netzwerkvariablen hängt die Überwachungszeit der Verbindung (WDT 2) vom Empfänger (Sicherheitssteuerung 2) ab.

Eine Überwachungszeit kleiner als der Worst-Case des Kommunikationszyklus der Netzwerkvariablen führt zu keinem stabil laufenden System. Im Kommunikationszyklus kommt es zu Zeitverlusten, da die Applikationszyklen von Sender und Empfänger nicht synchron laufen und auch die systemabhängige Transportzeit berücksichtigt werden muss.

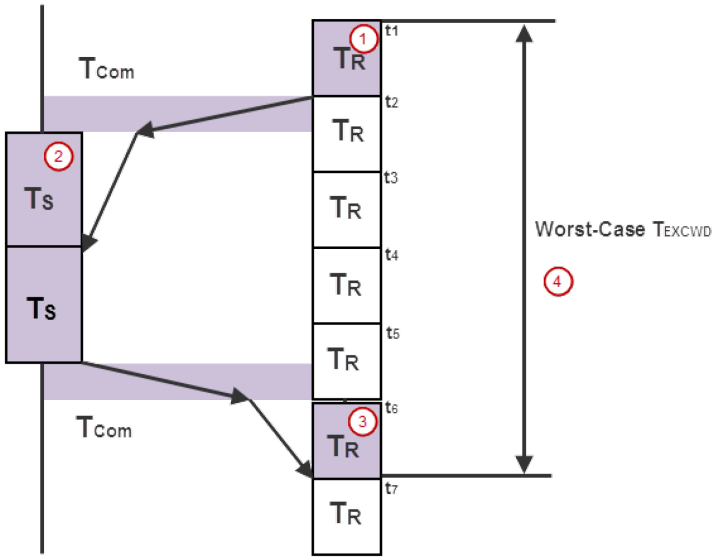


Abb. 14: Sequenzdiagramm eines Netzwerkvariablen-Kommunikationszyklus (links Sender, rechts Empfänger)

T_S Zykluszeit der Applikation von Sicherheitssteuerung 1(NVL-Sender)

T_R Zykluszeit der Applikation von Sicherheitssteuerung 2 (NVL-Empfänger)

T_{Com} Transportzeit von einer Sicherheitssteuerung über deren Standardsteuerung zur anderen Standardsteuerung und deren Sicherheitssteuerung

- (1): Der Empfänger schickt das Protokoll zum Ende des Zyklus ab. Da seine Zeitbasis aber für die Dauer des Zyklus eingefroren ist, beginnt die überwachte Zeit schon mit dem Startzeitpunkt "t_1" des Zyklus.
- (2): Das vom Empfänger gesendete Protokoll kommt zu spät (nach Beginn des Zyklus des Senders). Dies bedeutet, dass sich das Lesen des Protokolls bis zum Start des nächsten Zyklus verzögert.
- (3): Die von Empfänger gesendeten Daten kommen verspätet (nach Beginn des Zyklus des Empfängers). Dies bedeutet, dass sich der Prüfung der Daten und der Überwachungszeit bis zum nächsten Zyklusbeginn (im Beispiel: "t_7") verzögert.
- (4): Worst-Case T_{excWD} des Kommunikationszyklus

Aus diesen Verzögerungsfaktoren ergibt sich die kleinste sinnvolle Überwachungszeit (WDT 2).

Die kleinste sinnvolle Überwachungszeit für eine Safety Netzwerkvariablen-Verbindung ist:

WDT 2 $\geq$ 2 x Transportzeit T_{Com} + 2 x Zykluszeit Sender T_s + 2 x Zykluszeit Empfänger T_R.

4.3.3 Reaktionszeiten bei F-Device-Steuerungen

Reaktionszeit bei T4

Aus Sicht des F-Hosts kann das F-Device zwischen dem PROFINET-Anschluss und den Feldgeräteeingängen/Felgeräteausgängen des F-Devices als Blackbox betrachtet werden. Die F-Device-Blackbox besteht aus folgenden Komponenten: Feldgeräteeingang/Feldgeräteausgang, Feldbus, SPS mit PROFINET-Anschluss und Sicherheitssteuerung. Daraus ergibt sich für den F-Host eine Berechnung der zugesicherten Reaktionszeit wie bei T1:

Die zugesicherte Gesamtreaktionszeit-Obergrenze aus Sicht des F_Hosts:

R_g= Sensorverzögerung

+ max. Erfassungszeit F-In

+ 2 x WDT F-In

+ 2 x WDT F-Out

+ max. Schaltzeit F-Out

+ Nachlaufzeit

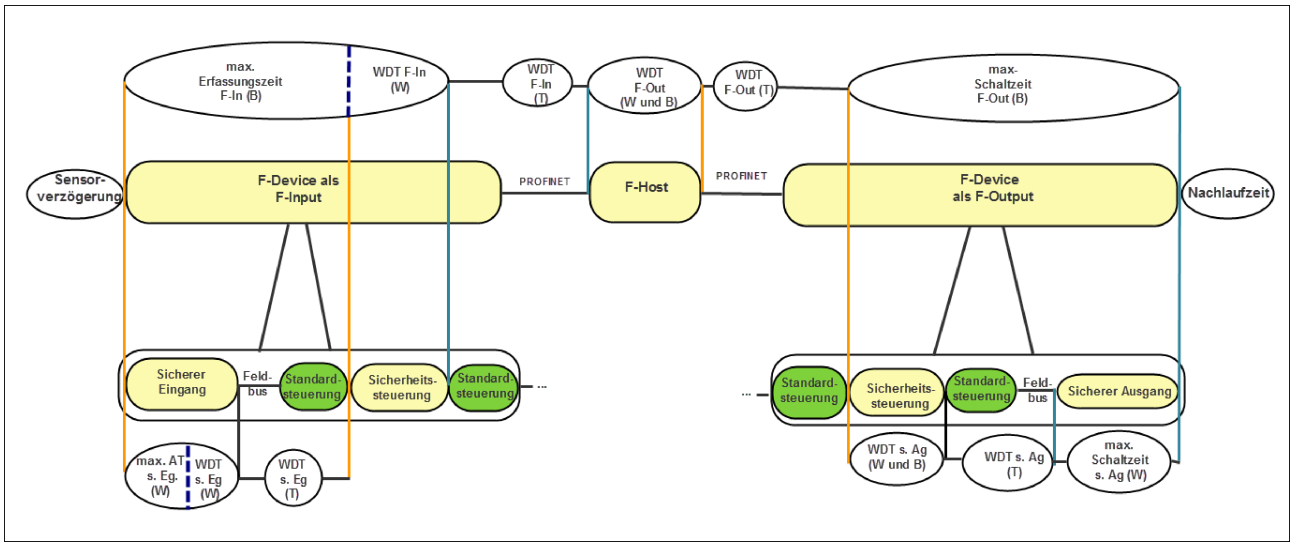


Abb. 15: Reaktionszeit bei T4

W Wartezeit
T Transportzeit
B Bearbeitungszeit

WDT Watchdog-Zeit (Überwachungszeit)
s. Eg sicherer Eingang
s. Ag sicherer Ausgang

- Watchdog-Zeit sicherer Eingang: Konfigurierte Überwachungszeit der Feldbuskommunikation zwischen sicherem Eingangsmodul und Sicherheitssteuerung.

  Maximale Wartezeit und maximale Transportzeit
- Watchdog-Zeit F-In: Konfigurierte Überwachungszeit der Feldbuskommunikation zwischen Sicherheitssteuerung und F-Host

  Maximale Wartezeit maximale Transportzeit
- Watchdog-Zeit F-Out: Konfigurierte Watchdog-Zeit zwischen F-Host und Sicherheitssteuerung

  Maximale Transportzeit, maximale Wartezeit und maximale Bearbeitungszeit
- Watchdog-Zeit sicherer Ausgang: Konfigurierte Überwachungszeit zwischen Standardsteuerung und sicherem Ausgangsmodul

  Maximale Wartezeit, maximale Transportzeit und maximale Wartezeit
- max. Erfassungszeit F-In, max. Schaltzeit F-Out: Die Verzögerung von Sensor- bzw. Aktorsignalen innerhalb der F-Device-Blackbox ist durch die Konstruktion und Programmierung des F-Devices festgelegt. Der Programmierer des F-Hosts benötigt sie für die Berechnung der Gesamtreaktionszeit der Maschine.

Die zugesicherte max. Erfassungszeit F-In des F-Devices als F-Input, das heißt die max. Verzögerung des Sensorsignals durch das F-Device beträgt:

WCDT_In = max. Erfassungszeit sicherer Eingang + 2 x Watchdog-Zeit sicherer Eingang

Die zugesicherte max. Schaltzeit F-Out des F-Devices als F-Output, das heißt die max. Verzögerung des Aktorsignals durch das F-Device beträgt:

WCDT_OUT = 2 x Watchdog-Zeit sicherer Ausgang + max. Schaltzeit sicherer Ausgang

4.4 Planung der Adressen

Adressen für Feldgeräte

Sicherheitsprotokolle wie Profisafe oder FSoE sichern die Verbindung zwischen Sicherheitssteuerung und sicheren Feldgeräten über den Standardfeldbus mittels Safety-Adressen ab (F_Dest_Add und F_Source_Add bei Profisafe und FSoE adress und Connection ID bei FSoE). Deren eindeutige Vergabe innerhalb eines Feldbusses muss geplant werden, insbesondere bei mehreren Sicherheitssteuerungen im gleichen Feldbus (Topologie T2) oder bei der Topologie T4 mit einer Obersteuerung und einem Oberfeldbus. Für genaue Anweisungen zur Eindeutigkeit der entsprechenden feldbus-spezifischen Parameter siehe ↬ *Kapitel 14 „Feldbusse und Netzwerkvariablen“ auf Seite 271*.

Manche Feldgeräte haben mehrere parallele Safety-Verbindungen gleichzeitig. In dem Fall muss für jede Safety-Verbindung eine eigene Adresse und, je nach Feldbus eine eigene Verbindungs-ID eingeplant werden.

Bedeutung der Adresseindeutigkeit:

- bei Topologie T1: eindeutige Slaveadressen in der Sicherheitssteuerungen
- bei Topologie T2: gemeinsamer Slaveadressraum für alle Sicherheitssteuerungen unter gleicher Standardsteuerung
- bei Topologie T3: gemeinsamer Slaveadressraum für alle querkommunizierenden Sicherheitssteuerungen
- bei Topologie T4: gemeinsamer Slaveadressraum für den F-Host und alle F-Device Sicherheitssteuerungen.

  Erklärung: Da die Standardsteuerung ein schwarzer Kanal ist, könnten die Safety-PDOs, die für den untergeordneten Feldbus gedacht sind, fehlerhafterweise in den übergeordneten Feldbus gelangen. Dort könnten sie an ungewollte Adressaten gehen oder sogar durch einen zweiten Fehler via einer zweiten Standardsteuerung als Feldgerät in einen parallel untergeordneten Feldbus gelangen. Deshalb dürfen weder für F-Host- und F-Device-Steuerung noch zwei für F-Device-Steuerungen unter dem gleichen F-Host die gleichen Safety-Adressen für ihre Slaves verwendet werden.

Adressen bei Querkommunikation

Für Netzwerkvariablen werden Safety-Adressen, Connection IDs und Listenidentifikatoren innerhalb eines Projekts automatisch eindeutig vergeben. Listenidentifikatoren dienen der CODESYS-internen Kennung. Für jede Verbindung Sender-Empfänger braucht CODESYS zwei Identifikatoren (einen in jede Richtung). (Beim Empfänger ist ihre Anzahl also 2, beim Sender 2 x Max. Anzahl Empfänger).

Netzwerkvariablen verwenden standardmäßig UDP Broadcasts im Maschinennetzwerk. Um die Netzlast zu reduzieren, können die IP-Adressen der gegenüberliegenden Standardsteuerung fix eingestellt werden. Für diesen Fall ist es sinnvoll, die IP-Adressen aller Steuerungen im Maschinennetzwerk bei der Planung des Gesamtsystems zu vergeben.

Mehrere CODESYS Projekte

Wenn die Software für die Maschine in mehreren getrennten CODESYS Projekten entwickelt wird, müssen für Netzwerkvariablen maschinenweit eindeutige Safety-Adressen, Connection IDs und Listenidentifikatoren bei der Planung des Gesamtsystems vergeben werden und manuell eingestellt werden.

5 Softwareentwicklung mit CODESYS Safety

5.1 Allgemeine Informationen

Im folgenden werden die Struktur des Projektbaums und die Projektbaum-Objekte beschrieben, welche für die Erstellung einer Sicherheitsapplikation mit CODESYS Safety gemäß der Richtlinien IEC 62061 und ISO 13849 zur Verfügung stehen.

Das Anlegen eines Projekts und die Bedienung der Oberfläche erfolgt nach dem Bedienkonzept von Standard CODESYS.

Befehle von Standard CODESYS, die in CODESYS Safety nicht zur Verfügung stehen, sind in der Regel sichtbar, aber nicht aktivierbar.

Gelbe Safety-Strukturen

Durch gelbe Strukturen werden die Safety-Objekte in der Projektstruktur und die Safety-Editoren deutlich hervorgehoben und unterscheiden sich dadurch optisch klar von den entsprechenden Standard-Objekten und Standard-Editoren.

Zusätzlich sind die sicheren Signalflüsse ebenfalls gelb markiert. Dies unterstützt und erleichtert die Tätigkeiten bei der Entwicklung, Verifikation und Abnahme der Sicherheitsapplikation.

Umsetzung des PLCopen Architekturmodells

Das PLCopen Architekturmodell (siehe Abb. 4) wird in CODESYS auf folgende Art und Weise umgesetzt:

- Die Sicherheitssteuerung wird unter der Standardsteuerung eingefügt (siehe *Kapitel 5.5.2 „Sicherheitssteuerung“ auf Seite 59*)
- Alle physikalischen Geräte werden im Gerätebaum unter der Standardsteuerung eingefügt. Sie werden automatisch mit der richtigen Steuerung verknüpft. Dabei werden die E/As der Sicherheitssteuerung zusätzlich als logische E/As unter der Sicherheitsapplikation eingefügt (siehe *Kapitel 5.5.4.2.1 „Überblick logische E/As“ auf Seite 70*)
- Der Datenaustausch zwischen der Sicherheitssteuerung und der Standardsteuerung erfolgt über die Objekte *„GVL für logischen Austausch“* der Standardapplikation und *„Logisches Austauschgerät“* der Sicherheitsapplikation (siehe *Kapitel 5.5.4.2.2.3 „Logisches E/A für Datenaustausch mit der Standardsteuerung“ auf Seite 77*).

5.2 Aufsetzen eines Safety-Projekts

5.2.1 Geplante Geräte vorbereiten

Es können nur Steuerungen und Feldgeräte programmiert und konfiguriert werden, die in CODESYS bekannt sind. Zur Programmierung der Maschine müssen für alle Steuerungen und Feldgeräte aus der Gesamtplanung (siehe *Kapitel 4 „Planung des Gesamtsystems“ auf Seite 33*) entsprechende Gerätebeschreibungen installiert sein, und gegebenenfalls die dazu gehörigen Bibliotheken.

Im Geräte-Repository (siehe *Kapitel 5.3 „Geräteverwaltung “ auf Seite 56*) können Sie die bereits installierten Gerätetypen nachsehen und fehlende nach installieren. Im Bibliotheks-Repository (siehe Onlinehilfe) können Sie die bereits installierten Gerätetypen nachsehen und fehlende nach installieren. Sie können Geräte und Bibliotheken aber auch jederzeit später bei Bedarf nach installieren.

5.2.2 Einrichten der Sicherheitsapplikation

Erstellung eines neuen Projekts mit einer Sicherheitsapplikation

1. Neues Projekt erstellen: Aktivierung des Befehls *„Neues Projekt...“* des Datei-Menüs.
2. Im Dialog *„Neues Projekt“* (siehe Abb. 16) ein *„Leeres Projekt“*, oder *„Leeres Safety-Projekt“* (leeres Projekt mit Template für Safety-Benutzerverwaltung) auswählen und Schaltfläche *„OK“* aktivieren.
3. Dem Projekt die geplanten Standardsteuerungen (Feldbus-Master) hinzufügen: Selektion des Projekts (am Beispiel: Safety_Projekt) im Gerätebaum und Aktivierung des Kontextmenü-Befehls *„Gerät anhängen...“*.
4. Der Reihe nach die Steuerungen auswählen, die mit einer Sicherheitssteuerung ergänzt werden sollen.
5. Den ausgewählten Standardsteuerungen die Sicherheitssteuerung (am Beispiel Abb. 18) hinzufügen: Selektion der Standardsteuerung und Aktivierung des Kontextmenübefehls *„Gerät anhängen...“* und Sicherheitssteuerung auswählen (siehe Abb. 18).
6. Wenn Sie in Schritt 2 *„Leeres Safety-Projekt“* ausgewählt haben, geben Sie nun im Dialog *„Anmelden“* (siehe Abb. 17) den *„Aktueller Benutzer:“* Owner eingeben. Das Passwort ist leer. Schaltfläche *„OK“* aktivieren.
7. Bei Verwendung des Templates mit Safety-Benutzerverwaltung weitere Schritte siehe *„Weiteres Vorgehen bei Safety Projekt mit Benutzerverwaltungs-Template“ auf Seite 52*.

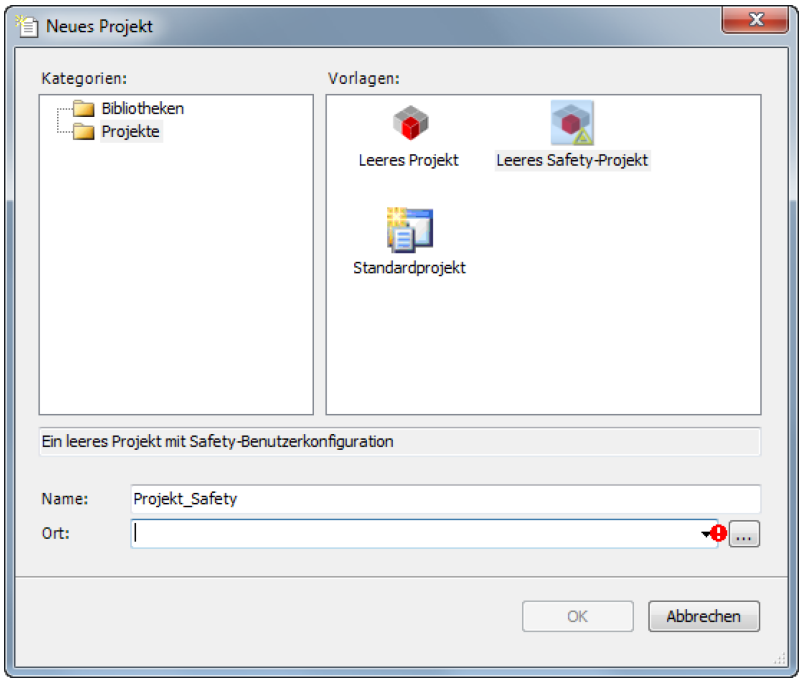


Abb. 16: Dialog 'Neues Projekt'

Es wird empfohlen beim Anlegen eines Safety Projekts im Dialog „Neues Projekt“ die Vorlage „Leeres Safety-Projekt“ auszuwählen. Diese Vorlage ist ein leeres Projekt mit Safety-Benutzerkonfiguration für die CODESYS Safety Benutzerverwaltung.

HINWEIS!
Wird als Vorlage *„Leeres Projekt“* ausgewählt, so muss die Benutzerverwaltung vollständig selbst aufgebaut werden.

Beschreibung der Benutzerverwaltung und der Safety-Benutzerkonfiguration siehe *Kapitel 5.2.3 „Einrichten der Benutzerverwaltung im Projekt“ auf Seite 53*.

Bei Auswahl der Vorlage „Leeres Safety-Projekt“ erscheint beim Einfügen der Sicherheitssteuerung der Dialog „Anmelden“. Die einzigen angelegten Benutzer sind "Owner", "saf" und "ext", das Passwort ist leer.

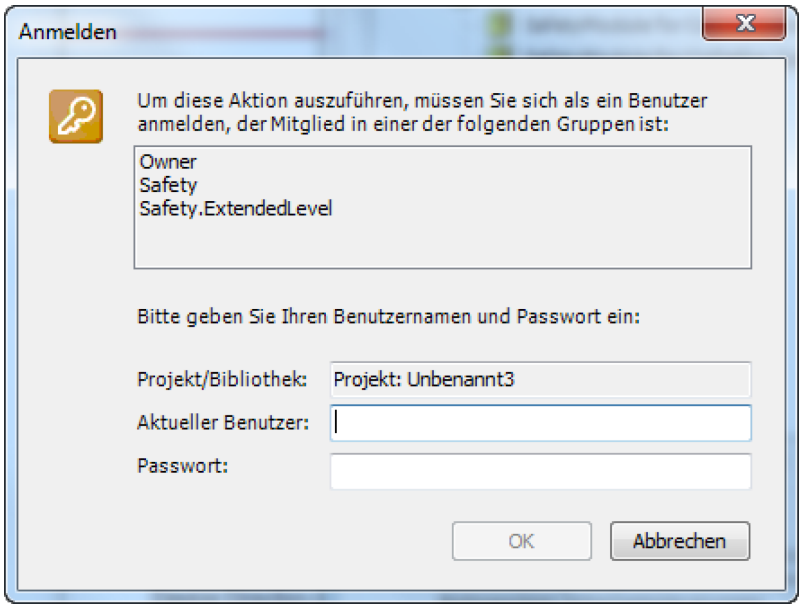


Abb. 17: Dialog: Benutzer- und Passwortabfrage beim Einfügen der Sicherheitssteuerung

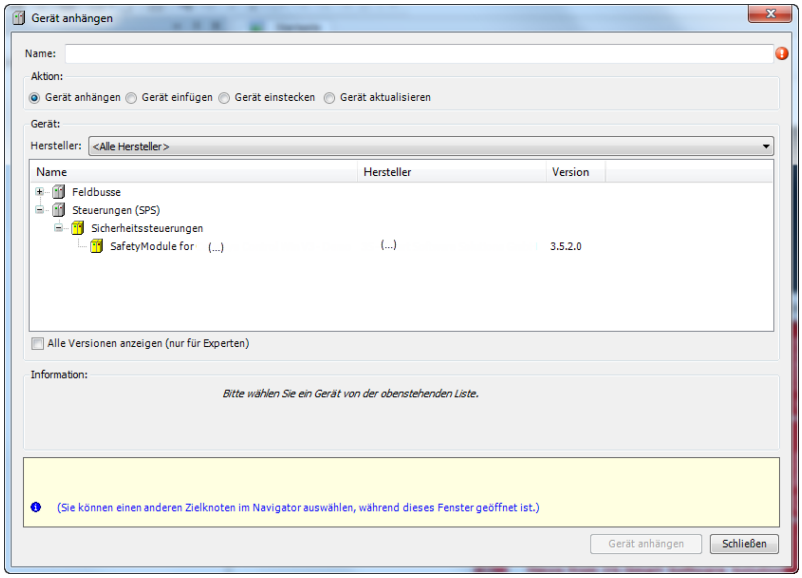


Abb. 18: Dialog beim Anhängen einer Sicherheitssteuerung

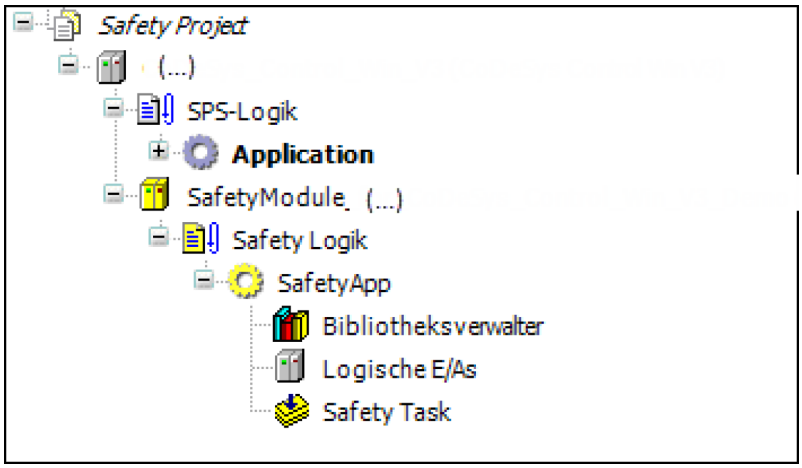


Abb. 19: Projektbaum mit Sicherheitsapplikation

Zusammen mit der Sicherheitssteuerung automatisch eingefügte, und nur einmal vorhandene Objekte bzw. Knotenpunkte:

- **Safety Logik**:
  Logischer Knotenpunkt der Sicherheitssteuerung, unter den genau ein Safety Applikationsobjekt eingehängt werden kann.
- **SafetyApp**:
  Knotenpunkt, unter dem die zum Safety Applikationsobjekt gehörenden Objekte liegen.
  Objekt, das die Ausführungsversion des Codes und den aktuell festgehaltenen Stand (Pin) der Applikation festlegt (siehe ↳ *Kapitel 8 „Pinnen der Software“ auf Seite 189*)
  Der Editor des Objekts verwaltet die Liste der aktuell zum Safety Applikationsobjekt gehörenden Objekte .
- **Bibliotheksverwalter**:
  Enthält die auf der eingefügten Sicherheitssteuerung zur Verfügung stehenden Bibliotheken.
  Diese sind:
  – SafetyPLCopen
  – SafetyStandard
  – ggf. weitere geräteabhängige Bibliotheken
- **Logische E/As**:
  Knotenpunkt, dem logische E/A-Objekte hinzugefügt werden können. Diese hinzugefügten logischen E/As dienen dem Austausch von Daten und E/As mit der Standardsteuerung
- **Safety Task**:
  Dieses Objekt listet alle Programme auf, welche auf die Steuerung geladen und ausgeführt werden.

Objekte, die dem Safety Applikationsobjekt manuell (auch mehrfach) hinzugefügt werden können:

- **Safety Basic-POU**
  POU (Programm oder Funktionsbaustein) mit Programmierlevel Basic
- **Safety Extended-POU**
  POU (Programm oder Funktionsbaustein) mit Programmierlevel Extended

- **Globale Variablenliste (Safety)**
  Deklaration der nur innerhalb des Safety Applikationsobjekts gültigen globalen Variablen
- **Safety Netzwerkvariablenliste (Sender)**
- **Safety Netzwerkvariablenliste (Empfänger)**

Weiteres Vorgehen bei Safety Projekt mit Benutzerverwaltungs-Template

1. Eigenschaften-Dialog der Sicherheitssteuerung öffnen: Sicherheitssteuerung im Projektbaum selektieren und Kontextmenü-Befehl *„Eigenschaften“* aktivieren.
2. Registerkarte *„Zugriffskontrolle“* öffnen.
3. Die Zellen in den Spalten *„Bearbeiten“* und *„Entfernen“* der Benutzergruppe *„Everyone“* durch Doppelklick öffnen, jeweils *„Verbieten“* auswählen und mit *„OK“* bestätigen.
4. Die Zellen in den Spalten *„Bearbeiten“* und *„Entfernen“* der Benutzergruppen *„Safety“* und *„Safety.ExtendedLevel“* durch Doppelklick öffnen, jeweils *„Erlauben“* auswählen und mit *„OK“* bestätigen.

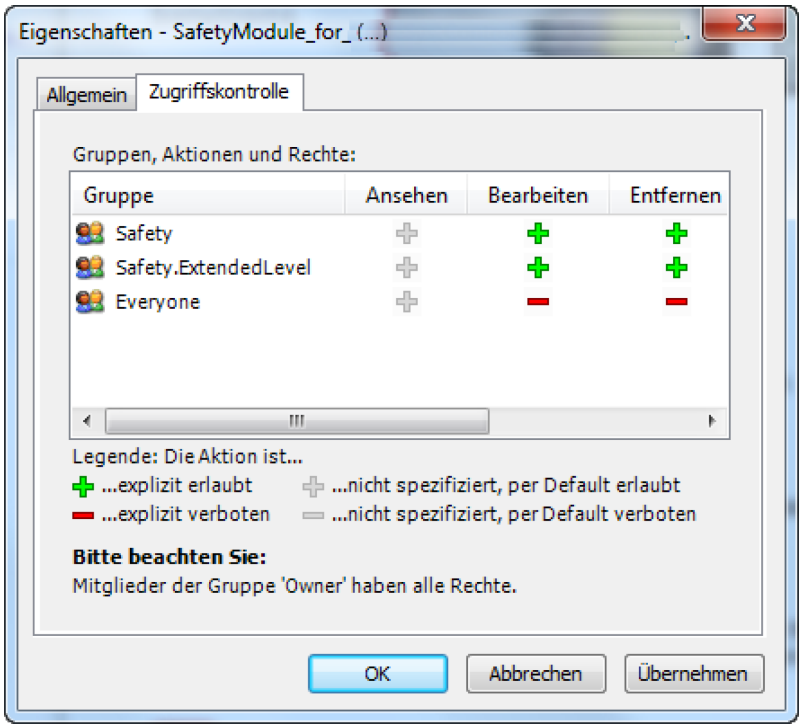


Abb. 20: Dialog: Eigenschaften der S-SPS, Registerkarte 'Zugriffskontrolle'

5.2.3 Einrichten der Benutzerverwaltung im Projekt

In CODESYS Safety ist ein Safety-Benutzerkonfiguration für ein Projekt mit Sicherheitsapplikation integriert. Der Projektleiter kann diese Safety-Benutzerkonfiguration für sein Projekt verwenden, oder aber auch eine eigene Benutzerverwaltung erstellen.

Der Projektleiter muss sich bereits zu Beginn der Erstellung des neuen Projekts im Dialog „Neues Projekt...“ entscheiden, ob er ein „Leeres Projekt“ ohne Benutzerkonfiguration oder ein „Leeres Safety-Projekt“ mit Safety-Benutzerkonfiguration anlegen möchte (siehe Kapitel 5.2.2 „Einrichten der Sicherheitsapplikation“ auf Seite 48). Wird ein „Leeres Projekt“, d.h. eine Projekt ohne Safety-Benutzerkonfiguration erzeugt, so muss der Projektleiter selbst eine Benutzerverwaltung für das Safety Projekt erstellen (siehe Benutzerverwaltung, Rechtevergabe in der CODESYS-Hilfe).

Die Rechtevergabe der Benutzerverwaltung befindet sich im Menü „Projekt“, Untermenü „Benutzerverwaltung“, Auswahl „Rechte...“.

Jeder Benutzer gehört automatisch zu der Gruppe „Everyone“.

Einstellungen der Safety-Benutzerkonfiguration

Diese Benutzerverwaltung enthält bereits folgende Einstellungen:

Benutzergruppen

- *„Owner“*
- *„Safety“*
- *„Safety.ExtendedLevel“*
- *„Everyone“*

Benutzer

- *„Owner“*, der der Gruppe *„Owner“* angehört
- *„saf“*, der der Gruppe *„Safety“*
- *„ext“*, der den Gruppen *„Safety.ExtendedLevel“* und *„Safety“* angehört.

Die voreingestellten Benutzer haben noch kein Passwort. Die Gruppe *„Owner“* enthält bereits den Benutzer *„Owner“*, noch ohne Passwort. Der Benutzer *„Owner“* kann den Benutzergruppen Safety, Safety.ExtendedLevel und Owner entsprechende Benutzer über das Menü *„Projekt* ➔ *Projekteinstellungen* ➔ *Benutzer und Gruppen“* zuordnen und an die Benutzer Passwörter vergeben.

Der Benutzer *„Owner“* kann neue Benutzergruppen definieren. Eine neue Benutzergruppe hat anfangs die Rechte der Benutzergruppe *„Everyone“* der Safety-Benutzerkonfiguration. Durch Zuordnung einer neuen Gruppe zu anderen bereits existierenden Benutzergruppen (z.B. *„Safety“*, *„Safety.ExtendedLevel“*), erhält die neue Gruppe die Zugriffsrechte der zugeordneten Gruppe.

Der Projektleiter muss für den Benutzer "Owner" ein geeignetes, sicheres Passwort vergeben, um den Zugriffschutz des Projekts zu gewährleisten.

Passwort definieren

1. Menü *„Projekt“* öffnen.
2. *„Projekteinstellungen“* selektieren.
3. Im Dialogfenster *„Projekteinstellungen“* die *„Benutzer und Gruppen“* auswählen.
4. Unter dem Tab *„Benutzer“* Owner selektieren.
5. Schaltfläche *„Bearbeiten...“* aktivieren.
6. Im Dialogfenster *„Benutzer bearbeiten“* altes Passwort, neues Passwort und Passwortbestätigung eingeben.
7. Schaltfläche *„OK“* aktivieren.

Weitere Informationen zur Passwortvergabe finden Sie in der CODESYS-Hilfe, siehe
Benutzerverwaltung und Passwort-Manager

In der Benutzerverwaltung sind die Rechte auf Befehle, Benutzer und Gruppen, Objekttypen, Projektobjekte folgendermaßen vorgegeben, können aber von Mitgliedern der Gruppe Owner jederzeit geändert werden.

Rechtevergabe des Safety-Benutzerkonfiguration

- Die Gruppe *„Owner“* besitzt alle Rechte
- Die Gruppe *„Safety“* besitzt **nicht**
  - das Recht, eine *„Safety Extended-POU“* zu erzeugen
  - das Recht, eine *„Safety Externe-POU“* zu erzeugen
  - das Recht zum *„Ausführen“* des Befehls *„Safety FUP: Return einfügen“*
  - das Recht zm *„Ausführen“* des Befehls *„Safety FUP: Sprung einfügen“*
  - die Rechte für das *„Ausführen“* der *„Befehle“*, die die Benutzerverwaltung betreffen.
  - Recht zum *„Bearbeiten“* der *„Benutzer und Gruppen“*.

- Die Gruppe *„Safety.ExtendedLevel“* besitzt **nicht**
  - die Rechte für das *„Ausführen“* der *„Befehle“*, die die Benutzerverwaltung betreffen.
  - Recht zum *„Bearbeiten“* der *„Benutzer und Gruppen“*
- Die Gruppe *„Everyone“* besitzt **nicht**
  - die Rechte zum Ausführen der*„Befehle“* der Kategorie *„Safety“*
  - die *„Rechte“* zum *„Erzeugen“* der Safety-Objekttypen.
  - die Rechte zum *„Ausführen“* der *„Befehle“*, die die Benutzerverwaltung betreffen.
  - das Recht zum *„Bearbeiten“* der *„Benutzer und Gruppen“*.

Hinweise für die Entwicklung einer Sicherheitsapplikation mit der Safety-Benutzerkonfiguration

Zum Einfügen einer *„Safety Extended-POU“* muss der Entwickler Mitglied der Benutzergruppe *„Safety.ExtendedLevel“* sein.

Nach Einfügen einer Sicherheitssteuerung, bzw. des Safety Applikationsobjekts sind die Änderungsrechte für das Safety Applikationsobjekt explizit in dem Dialog *„Eigenschaften“* des Safety Applikationsobjekts unter dem Tab *„Zugriffskontrolle“* wie folgt zu konfigurieren:

Genaue Vorgehensweise siehe *„Weiteres Vorgehen bei Safety Projekt mit Benutzerverwaltungs-Template“ auf Seite 52*

- **+** für Safety
- **+** für Safety.ExtendedLevel
- **-** für Everyone

Nach Einfügen einer Safety Extended-POU, muss der Entwickler im Eigenschaften-Dialog der Safety Extended-POU unter dem Tab *„Zugriffskontrolle“* der Benutzergruppe *„Safety“* das Recht zum *„Bearbeiten“* und *„Entfernen“* der Extended-POU explizit verbieten, falls die Extended-POU nur von Mitgliedern der Benutzergruppe *„Safety.ExtendedLevel“* bearbeitet und entfernt werden soll.

Anlegen eines neuen Projekts ohne Safety-Benutzerkonfiguration

Wird ein neues Projekt als *„Leeres Projekt“* angelegt, so muss der Projektleiter selbst eine geeignete Benutzerverwaltung für die Sicherheitsapplikation des Projekt erstellen. Für die genaue Vorgehensweise und detaillierte Informationen bei der Erstellung dieser Benutzerverwaltung im Projekt wird auf die Hilfe von Standard CODESYS verwiesen, siehe
Benutzerverwaltung, Rechteverwaltung

Um die Auswahl der Safety-spezifischen Befehle und Objekttypen dem Entwickler zu erleichtern, ist bei der Rechtevergabe den Befehlen und den Objekttypen, die für den Safety-Entwickler relevant sind, *„Safety“* vorangestellt.

Wenn dem Entwickler das Recht zu einer bestimmten Operation fehlt, kann er während der Operation das Logon auf einen Benutzernamen mit mehr Rechten wechseln. (Aktivierung des Befehls „Benutzerverwaltung ➔ Benutzer anmelden...“ des Menüs „Projekt“)

5.2.4 Einrichten des Admin-Passworts auf der Steuerung

Die Bootapplikation kann mit einem Admin-Passwort (Administrationspasswort) vor nicht autorisiertem, schreibendem Zugriff geschützt werden. CODESYS Safety besitzt ein Default-Passwort. Nach jedem Öffnen des Projekts muss sich der Anwender vor dem ersten schreibenden Zugriff auf die Bootapplikation mit dem Admin-Passwort autorisieren. Es bleibt solange im Programmiersystem gesetzt gültig, bis das Projekt geschlossen wird.

Detaillierte Informationen siehe *Kapitel 12.1.3 „Schutz der Sicherheitssteuerung vor Schreibzugriff“ auf Seite 242*

5.2.5 Zugriffschutz bei Anbindung an Quellcodeverwaltung

Wird eine Quellcodeverwaltung verwendet, so muss auch in der Quellcodeverwaltung eine der Benutzerverwaltung im Projekt entsprechende Benutzerverwaltung eingerichtet werden.

5.3 Geräteverwaltung

Wurden Gerätebeschreibungen mit Standard CODESYS ohne Safety-Erweiterung installiert, so können sie in CODESYS Safety nicht verwendet werden. Die entsprechenden Gerätebeschreibungen müssen nach Installation von CODESYS Safety neu installiert werden, um sie in CODESYS Safety verwendbar zu machen.

Sichere Geräte werden analog zu Standard-Geräten im *„Geräte-Repository“* (Menü *„Tools“*) verwaltet, installiert und deinstalliert.

Für Detailinformationen wird auf die Online Hilfe von CODESYS Standard verwiesen.

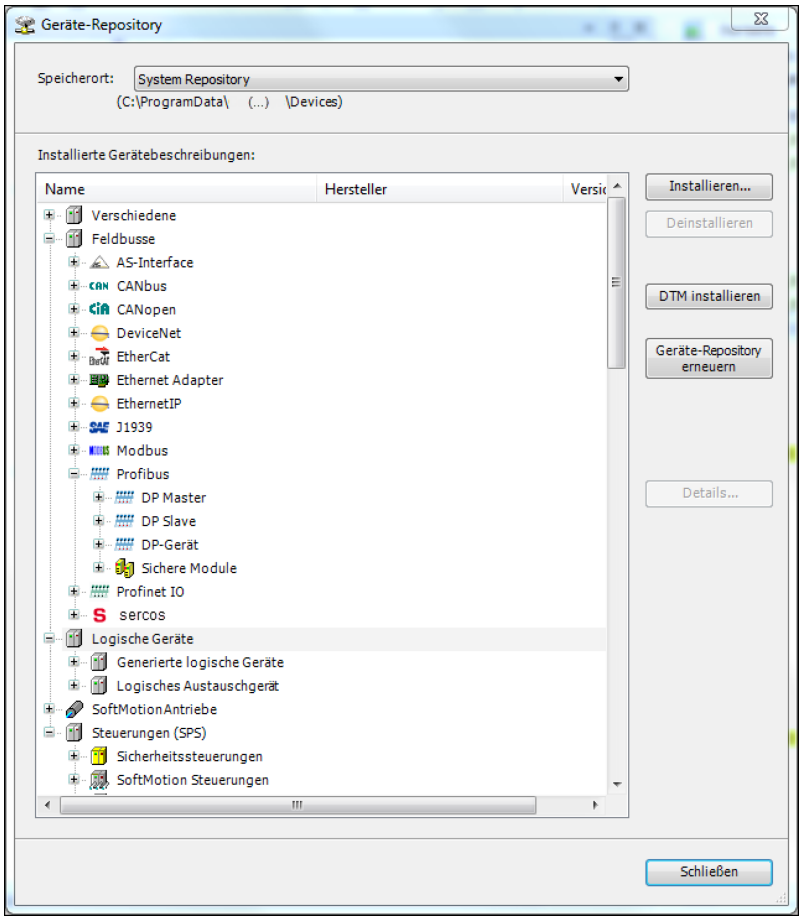


Abb. 21: Geräte-Repository

Das Geräte-Repository kann in dem Menü *„Tools“* selektiert und geöffnet werden.

Hier sind die für CODESYS Safety relevanten Geräte im Geräte-Repository zu finden:

- Sicherheitssteuerungen: in der der Kategorie *„Steuerungen“*, Unterkategorie *„Sicherheitssteuerungen“*
- logische Geräte der sicheren Feldgeräte: abhängig vom jeweiligen Feldbus in einer eigenen Unterkategorie (z.B. *„Sichere Module“*) oder bei den physikalischen Geräten.
- physikalische Geräte für sichere Feldgeräte werden nur explizit aufgeführt, wenn sie nicht als Module oder Submodule als Teil des Feldgeräts installiert werden. Sie werden nicht in eigenen Unterkategorien aufgeführt.
- logische Geräte der Standard Feldgeräte: in der der Kategorie *„Logische Geräte“*, Unterkategorie *„Generierte logische Geräte“*
- logische Geräte zum Datenaustausch mit der Standardsteuerung: in der Kategorie *„Logische Geräte“*, Unterkategorie *„Logische Austauschgeräte“*

Hier sind die für CODESYS Safety relevanten logischen Geräte im Dialog *„Logisches Gerät hinzufügen“* zu finden:

- logische Geräte der sicheren Feldgeräte: abhängig vom jeweiligen Feldbus in einer Unterkategorie der Kategorie *„Feldbusse“*
- logische Geräte der Standard-Feldgeräte: in der Kategorie *„Logische Geräte“*, Unterkategorie *„Generierte logische Geräte“*
- logische Geräte zum Datenaustausch mit der Standardsteuerung: in der Kategorie *„Logische Geräte“*, Unterkategorie *„Logisches Austauschgerät“*

Genaue Informationen zu den logischen E/As siehe ↳ *Kapitel 5.5.4.2.1 „Überblick logische E/As“ auf Seite 70*

Sichere Feldgeräte, die im Geräte-Repository installiert wurden, können im Projektbaum auf Standardseite wie jedes installierte Gerät eingefügt werden.

Die sicherheitsgerichteten Parameter der sicherheitsgerichteten Feldgeräte werden in den jeweiligen logischen E/As (siehe ↳ *Kapitel 5.5.4.2.3.2 „Sichere Parametrierung und Sichere Konfiguration “ auf Seite 84*) editiert. Die Parametrierung der Standard-Feldgeräte und der Standard-Parameter der sicherheitsgerichteten Geräte erfolgt wie in CODESYS Standard.

5.4 Bibliotheken

Für die Programmierung mit CODESYS Safety werden dem Entwickler Sicherheitsbibliotheken mit bereits zertifizierten Funktionsbausteinen zur Verfügung gestellt. Folgende Bibliotheken sind Teil des Produkts CODESYS Safety:

- *„SafetyPLCopen“*
- *„SafetyStandard“*

Diese Bibliotheken sind automatisch Teil der Sicherheitsapplikation. Eine Installation ist nicht notwendig.

Abhängig von der Unterstützung von Safety-Feldbussen und sicherer Querkommunikation durch die Sicherheitssteuerung können weitere Feldbus-spezifische Bibliotheken zur Verfügung stehen (siehe ↳ *Kapitel 14 „Feldbusse und Netzwerkvariablen“ auf Seite 271*).

Sollte der Geräteherstellter weitere Bibliotheken zur Verfügung stellen, werden diese ebenfalls automatisch installiert.

Die Verwaltung dieser Bibliotheken erfolgt wie in CODESYS Standard (siehe Onlinehilfe).

Genauere Beschreibung der Sicherheitsbibliotheken finden Sie in der CODESYS Safety Onlinehilfe.

Die Versionsliste der Bibliotheksbausteine und die Sicherheitshinweise die für die Bibliotheksbausteine beachtet werden müssen finden Sie in ↳ *Kapitel 15 „Vordefinierte Bausteine“ auf Seite 293*.

5.5 Projektstruktur

5.5.1 Einordnung einer Sicherheitssteuerung in den Projektbaum

Die Sicherheitssteuerung wird im Projektbaum als Kind-Knoten der zugeordneten Standardsteuerung dargestellt.

Unter der Sicherheitssteuerung befindet sich genau eine Applikation (hier SafetyApp). Darunter befinden sich sämtliche zu einer Sicherheitsapplikation gehörenden Sicherheitsobjekte.

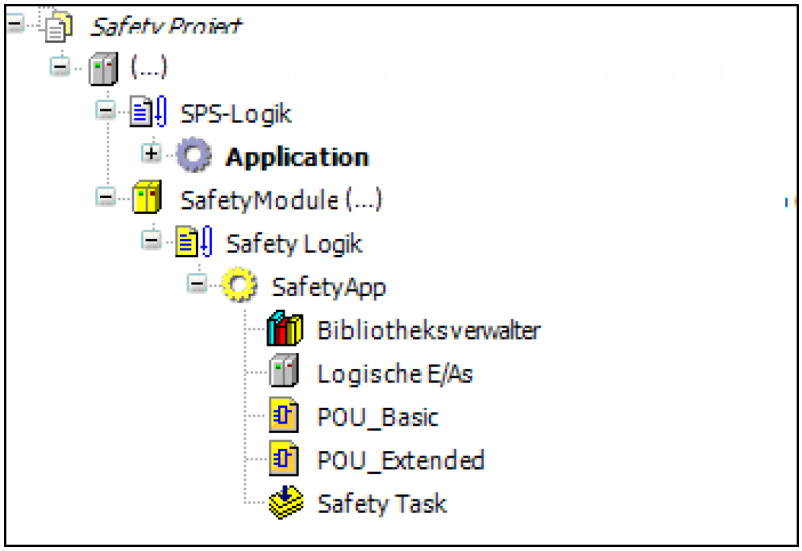


Abb. 22: Beispiel eines Projektbaums mit Standard- und Sicherheitsapplikation

Für den Teil des Projektbaums, der die Objekte der Standardapplikation beinhaltet, wird auf die Online Hilfe von Standard CODESYS verwiesen.

5.5.2 Sicherheitssteuerung

Hinzufügen der Sicherheitssteuerung

Bei der Topologie "Vater-Kind"-Steuerung wird die Sicherheitssteuerung unter der Standardsteuerung eingehängt. Dies erfolgt durch Selektion der Standardsteuerung und Aktivierung des Kontextmenübefehls *„Gerät anhängen...“* mit Auswahl der Sicherheitssteuerung.

Nach dem Einfügen der Sicherheitssteuerung wird immer der logische Knotenpunkt *„Safety Logik“*, das Safety Applikationsobjekt *„SafetyApp“*, das Task-Objekt *„Safety Task“*, der *„Bibliotheksverwalter“* und der Knotenpunkt *„Logische E/As“* automatisch mit eingefügt.

VORSICHT!

Damit nur berechtigte Personen das neue Geräteobjekt bearbeiten können, muss der Entwickler gleich nach dem Einfügen der Sicherheitssteuerung die Änderungsrechte explitzit in dem Dialog *„Eigenschaften“* unter dem Tab *„Zugriffskontrolle“* wie folgt konfigurieren (:

- **+** für Safety
- **+** für Safety.ExtendedLevel
- **-** für Everyone

(siehe *„Weiteres Vorgehen bei Safety Projekt mit Benutzerverwaltungs-Template“ auf Seite 52*)

Die Sicherheitssteuerung kann mit dem Kontextmenü-Befehl *„Gerät aktualisieren“* auf eine neuere Version der Gerätebeschreibung aktualisiert werden. Dabei werden eventuell Bibliotheken durch neuere Versionen ersetzt.

Objekteigenschaften der Sicherheitssteuerung

Eine im Projektbaum eingehängte Sicherheitssteuerung besitzt einen Eigenschaften-Dialog mit den Registerkarten *„Allgemein“* und *„Zugriffskontrolle“*. Der Eigenschaften-Dialog wird durch Selektion der Sicherheitssteuerung im Projektbaum und Aktivierung des Kontextmenü-Befehls *„Eigenschaften...“* geöffnet.

Editor der Sicherheitssteuerung

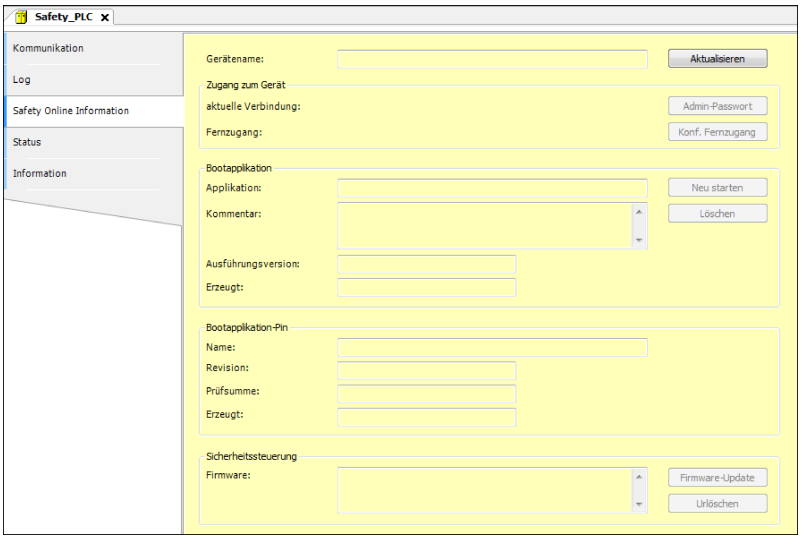


Abb. 23: Editor der Sicherheitssteuerung, Registerkarte 'Safety Online Information'

Der Geräte-Editor einer Sicherheitssteuerung enthält folgende Registerkarten:

- *„Kommunikation“*
  Beschreibung siehe *Kapitel 7.2.2 „Verbindungsaufbau“ auf Seite 164*
- *„Log“*
  Beschreibung siehe *Kapitel 12.3.3 „Logbuch: Diagnose von System- und Laufzeitfehlern“ auf Seite 248*
- *„Safety Online Information“*
  Diese Registerkarte enthält Safety-spezifische Informationen und Befehle der Sicherheitssteuerung:
  siehe *„Safety Online Information“ auf Seite 251*
- *„Status“*
  siehe *Kapitel 12.3.4 „Status: Diagnose der Kommunikation“ auf Seite 250*
- *„Information“*
  Anzeige von allgemeinen Informationen (siehe CODESYS Onlinehilfe)

5.5.3 Safety Logik

Den logischen Knotenpunkt *„Safety Logik“* gibt es aus Gründen der Symmetrie zur Standardsteuerung. Er hat in der Sicherheitssteuerung keine Bedeutung.

Der logische Knotenpunkt *„Safety Logik“* wird automatisch mit der Sicherheitssteuerung in den Projektbaum eingefügt. Unter einer Sicherheitssteuerung kann es nur genau einen Knoten *„Safety Logik“* geben. Ihm kann als Objekt nur eine *„Safety Applikation“* hinzugefügt werden.

Safety Logik besitzt einen Eigenschaften-Dialog mit den Registerkarten *„Allgemein“* und *„Zugriffskontrolle“*. Der Eigenschaften-Dialog wird durch Selektion von *„Safety Logik“* im Projektbaum und Aktivierung des Kontextmenü-Befehls *„Eigenschaften...“* geöffnet.

Safety Logik kann mit keinem Editor geöffnet werden.

5.5.4 Sicherheitsapplikation

5.5.4.1 Safety Applikationsobjekt

Aufbau des Projektbaums der Sicherheitsapplikation

Der Projektbaum wird im Geräte-Fenster (*„Geräte“* des Menüs *„Ansicht“*) dargestellt.

Die für die Sicherheitssteuerung und ihre Programmierung relevanten Objekte befinden sich unterhalb der Sicherheitssteuerung . Direkt darunter liegt immer der symbolische Knotenpunkt *„Safety Logik“* . Unter jedem *„Safety Logik“* kann sich immer nur **ein** Safety Applikationsobjekt *„SafetyApp“* (Defaultname) befinden, welches folgende Safety-Objekte enthalten kann:

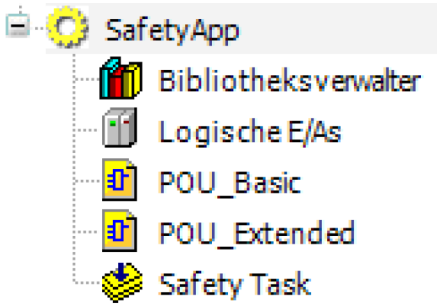


Abb. 24: Safety Applikationsobjekt mit Objekten

Folgende Objekte müssen in einem Safety Applikationsobjekt genau **einmal** vorhanden sein:

- Bibliotheksverwalter
- Safety Task
- Logische E/As

Hinzufügen des Safety Applikationsobjekts

Das Hinzufügen des Safety Applikationsobjekt im Projektbaum erfolgt:

- automatisch mit Hinzufügen der Sicherheitssteuerung
  oder
- manuell durch Selektion des logischen Knotenpunkts *„Safety Logik“* und Kontextmenübefehl *„Objekt hinzufügen“* mit Auswahl *„Safety Applikation“*

Im Eigenschaften-Dialog (Selektion des Objekts *„Safety Applikation“* und Aktivierung des Befehls *„Eigenschaften...“*) kann in der Registerkarte *„Allgemein“* ein Name und in der Registerkarte *„Safety“* ein Kommentar für das Safety Applikationsobjekt editiert werden (siehe Abb. 26).

Wird das Objekt *„Safety Applikation“* manuell hinzugefügt, so können Name und Kommentar für das Safety Applikationobjekt im beim Hinzufügen öffnenden Dialog editiert werden.

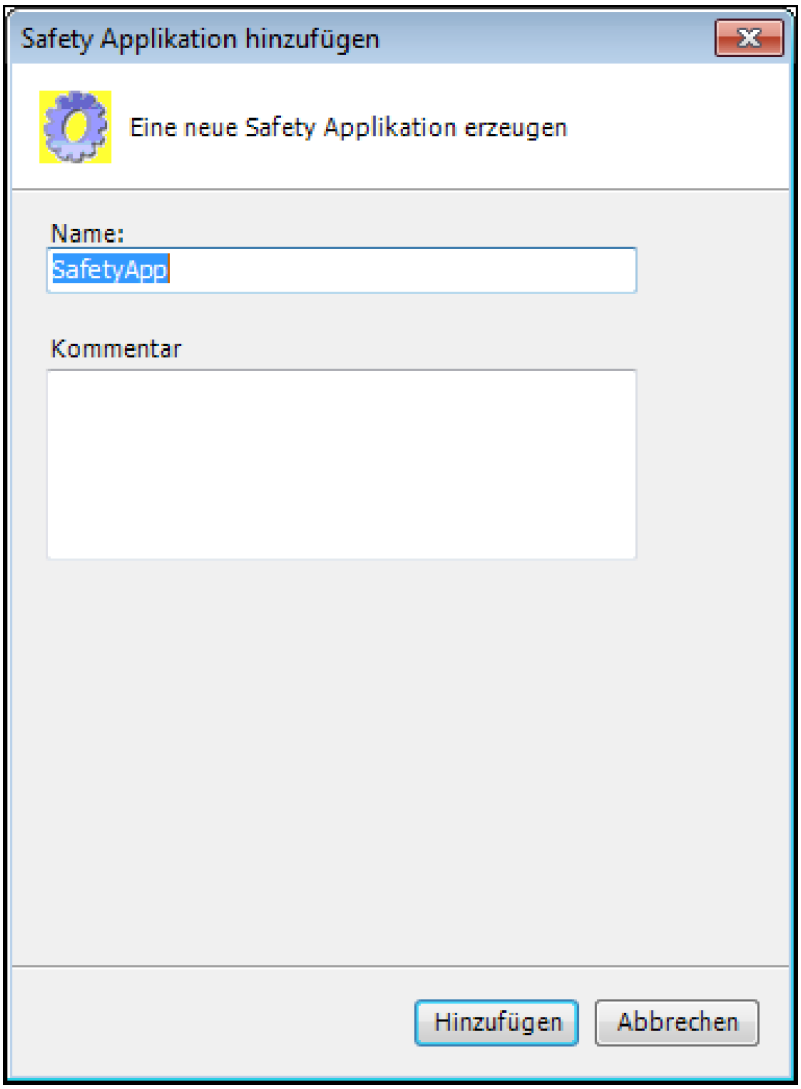


Abb. 25: Dialog beim manuellen Hinzufügen des Safety Applikationsobjekts mit Defaultname "SafetyApp"

Objekteigenschaften des Safety Applikationsobjekts

Der Eigenschaften-Dialog des Safety Applikationsobjekts und aller Objekte des Safety Applikationsobjekts besitzt die Registerkarten „Allgemein“ (mit Namen, Objekttyp und Editor, mit welchem das Objekt geöffnet wird) und „Zugriffskontrolle“ (Zugriffsrechte der Benutzergruppen für das Objekt).

Zusätzliche, objektspezifische Registerkarten werden bei den einzelnen Objekten beschrieben.

Der Eigenschaften-Dialog wird durch Selektion des Objekts im Projektbaum und Aktivierung des Kontextmenübefehls *„Eigenschaften“* geöffnet.

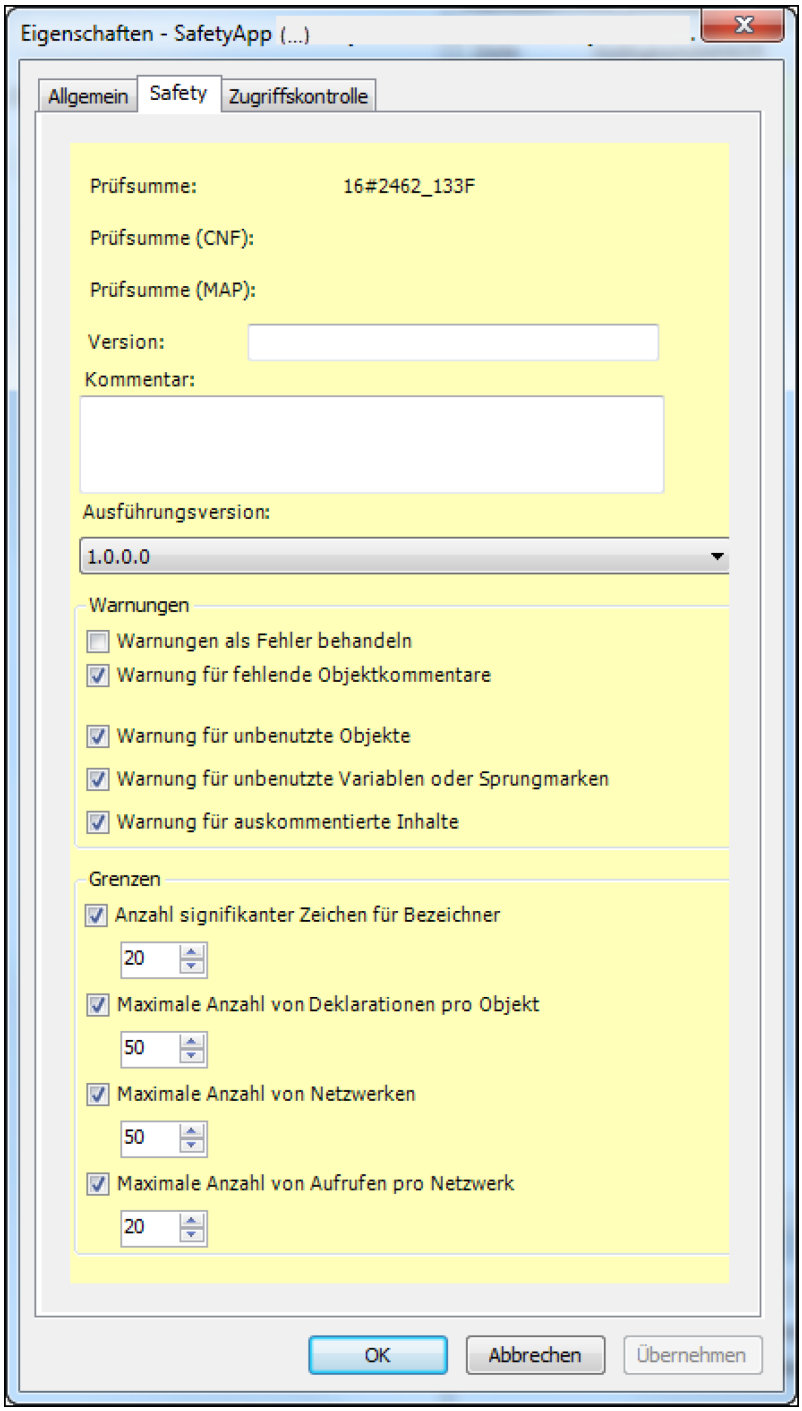


Abb. 26: Dialog Eigenschaften des Safety Applikationsobjekts, Registerkarte 'Safety'

Pin Prüfsumme

Die Pin-Prüfsumme ist eine Prüfsumme über alle Objekte der Sicherheitsapplikation einschließlich der verwendeten Bibliotheksbausteine.

Prüfsumme eines Objekts
Im Gegensatz zur Pin-Prüfsumme identifiziert die Prüfsumme der einzelnen Objekte der Sicherheitsapplikation zusammen mit der Objektversion den Objekt-Inhalt. Die Prüfsumme ist von Bedeutung, wenn einzelne Objekte einer Sicherheitsapplikation in einer anderen Applikation wiederverwendet werden.

Die Registerkarte Safety enthält:

- Informationen
  - *„ Prüfsumme:“*
    Information über die Prüfsumme des Safety Applikationsobjekts
  - *„Prüfsumme (CNF):“* (falls Aspekt *„Sichere Applikationsparameter“* vorhanden, siehe *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)
    Information über die Prüfsumme der Konfiguration
  - *„Prüfsumme (MAP):“* (falls Aspekt *„Sichere Applikationsparameter“* vorhanden, siehe *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)
    Information über die Prüfsumme des Mappings
  - *„Version“* (editierbar)
    Die Version kann vom Entwickler frei vergeben werden und dient dazu, dass die Version des Objekts in der Objektliste leicht erkannt werden kann.
  - *„Kommentar“* (editierbar)
  - *„Ausführungsversion“* (ggf. auswählbar)
    Mithilfe der Ausführungsversion wird die Kompatibilität zwischen (abgenommener) Bootapplikation und LZS überwacht.
- Einstellbare Warnungen und Begrenzungen, die durch den Safety-Checker geprüft werden (automatisch geprüfte Programmierrichtlinien siehe *Kapitel 9.3.3 „Automatische Prüfung der Programmierrichtlinien“ auf Seite 199*)

Die Ausführungsversion ist Teil der Abnahme der Sicherheitsapplikation

Ausführungsversion: In der Regel sollte der Entwickler immer die neuste Ausführungsversion auswählen.

Weitere Informationen zur Ausführungsversion siehe *Kapitel 11.3 „Aktualisieren der Firmware, die Ausführungsversion“ auf Seite 233*

Beim Übersetzen des Sicherheitsapplikation durch den Safety-Checker erzeugte Warnungen und Fehler werden im Meldungsfenster ausgegeben.

Warnungen und Begrenzungen der Registerkate *„Safety“* mit ihren Default-Werten:

- *„Warnungen als Fehler behandeln“* : Nicht aktiv
  Werden Warnungen als Fehler behandelt, kann die Applikation nicht geladen werden.
- *„Warnung für fehlende Objektkommentare für Applikation und POUs“* : Aktiv
  Berücksichtigt Kommentar und Version
- *„Warnung für unbenutzte Objekte“* : Aktiv
- *„Warnung für unbenutzte Variablen oder Sprungmarken“* : Aktiv
- *„Warnung für auskommentierte Inhalte“* : Aktiv
- *„Anzahl signifikanter Zeichen für Bezeichner“* : Aktiv: 20
  Es wird eine Warnung ausgegeben, wenn die ersten 20 Zeichen zweier Bezeichner gleich sind.
- *„Maximale Anzahl von Deklarationen pro Objekt“* : Aktiv: 50
- *„Maximale Anzahl von Netzwerken“* : Aktiv: 50
- *„Maximale Anzahl von Aufrufen pro Netzwerk“* Aktiv: 20

Genauere Informationen zu den Warnungen und Grenzen siehe *„Optional automatisch geprüfte Programmierrichtlinien“ auf Seite 116*

Editor des Safety Applikationsobjekts mit Objektliste

Der Editor wird durch Selektion des Objekts *„Safety Applikation“* und Aktivierung des Kontextmenübefehls *„Objekt bearbeiten“* geöffnet.

Im Projektvergleich kann durch Doppelklick auf das Safety Applikationsobjekt der Vergleichseditor geöffnet werden, der die Unterschiede in den Pin-Kennungen und den Objektlisten der Pins beider Applikationen anzeigt.

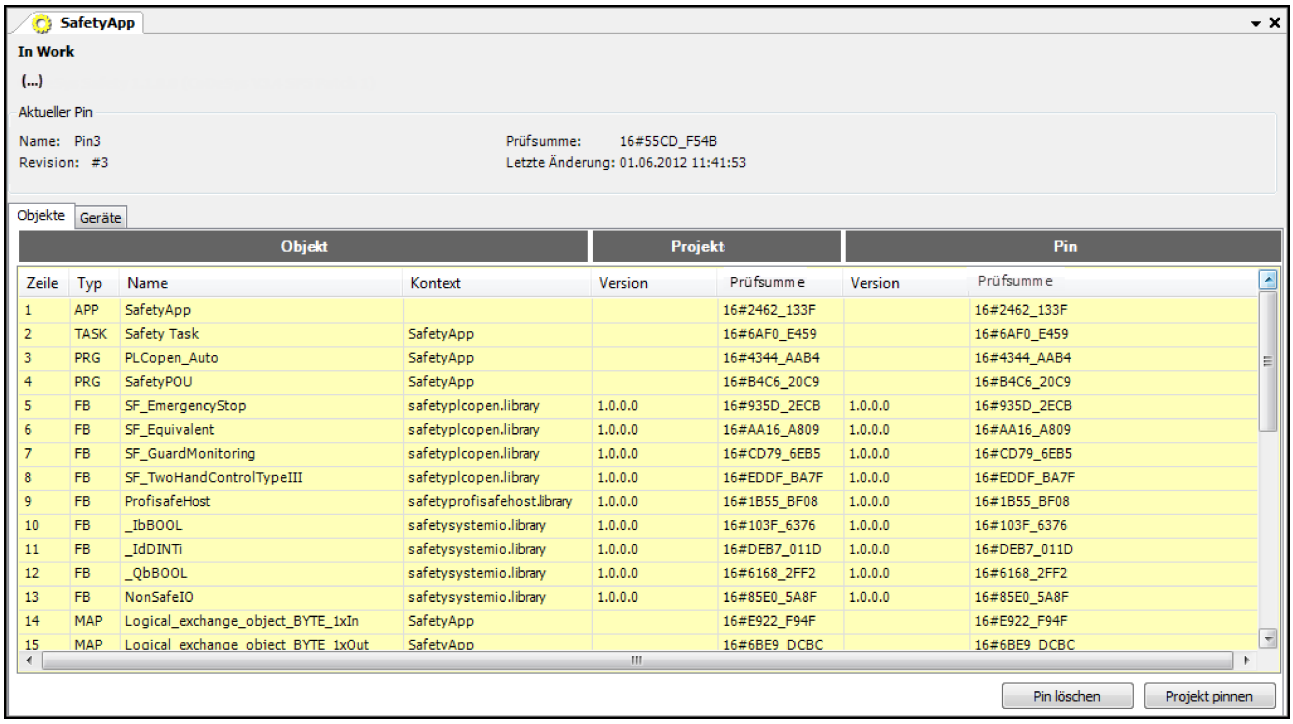


Abb. 27: Editor des Safety Applikationsobjekts mit Objektliste

Registerkarten des Editors des Safety Applikationsobjekts

- Objekte
- Geräte
- optional: Sichere Applikationsparameter
  Die optionale Registerkarte *„Applikationsparamter“* ist geräteabhängig.

Die Applikations-Editor mit den Registerkarten *„Objekte“* und *„Geräte“* enthält folgende Informationen:

- Pin-Informationen oder falls Applikation verändert wurde: *„In Work“*
- Aktuelle CODESYS Safety Version
- *„Aktueller Pin“* der Sicherheitsapplikation bestehend aus :
  - *„Name“*
  - *„Prüfsumme“* (Pin Prüfsumme)
    die Prüfsumme wird über die gesammte gepinnte Applikation erstellt.
  - *„Revision“*
  - *„Letzte Änderung“*

Genauere Informationen zu den Pin-Informationen siehe *Kapitel 8 „Pinnen der Software“ auf Seite 189*.

Objekte

*Die Registerkarte „Objekte“ des Editors des Safety Applikationsobjekts ist die **Objektliste**!*

Er zeigt an, welche Objekte seit dem letzten Pinnen verändert wurden.

Die Vergleichsansicht (Registerkarte 'Objekte') zeigt folgende Informationen an:

Auflistung der zur Sicherheitsapplikation gehörenden Objekte mit

- *„Objekt“*
  - *„Zeile“*
    Laufende Nummer der Einträge
  - *„Typ“*
    Objekttyp (siehe *Tab. 3 „Angezeigte Werte für Typ und Name in der Objektliste (abhängig vom Objekttyp)“ auf Seite 68*)
  - *„Name“*
    Name des Objekts
  - *„Kontext“*
- *„Projekt“*
  - *„Version“* und *„Prüfsumme“*
    Version und Prüfsumme des Objekts im aktuellen Projekt (Projektstatus)
- *„Pin“*
  - *„Version“* und *„Prüfsumme“*
    Version und Prüfsumme des Objekts des zuletzt gepinnten Projekts

Die Objektliste enthält auch Einträge für alle Safety-Objekte aus den Bibliotheken, welche die Applikation einbindet. Dabei werden nur jene aufgenommen, welche aus den Objekten der Applikation referenziert werden. Dies schließt jedoch auch Objekte aus den Bibliotheken mit ein, welche nicht aus Anwender-Code sondern aus System-Code aufgerufen werden, insbesondere aus den E/A-Treibern.

Tab. 3: Angezeigte Werte für Typ und Name in der Objektliste (abhängig vom Objekttyp)

Objekttyp	Typ	Name
Applikation	APP	Name des Applikationsobjekts
Task	TASK	Name des Taskobjekts
Programm POU	PRG	Name des Programmbausteins

Objekttyp	Typ	Name
Funktionsbaustein POU	FB	Name des POU-Objekts des Funktionsbausteins
Globale Variablenliste	GVL	Name des GVL-Objekts
E/A-Mapping	MAP	Name des logischen Geräteobjekts/Applikationsparameter
Sichere Konfiguration	CNF	Name des logischen Geräteobjekts/Applikationsparameter
Sichere Parametrierung	PAR	Name des logischen Geräteobjekts

Zusätzlich enthält die Vergleichsansicht die Schaltflächen *„Pin löschen“* und *„Projekt pinnen“* (siehe *Kapitel 9.3.3 „Automatische Prüfung der Programmierrichtlinien“ auf Seite 199*)

Geräte

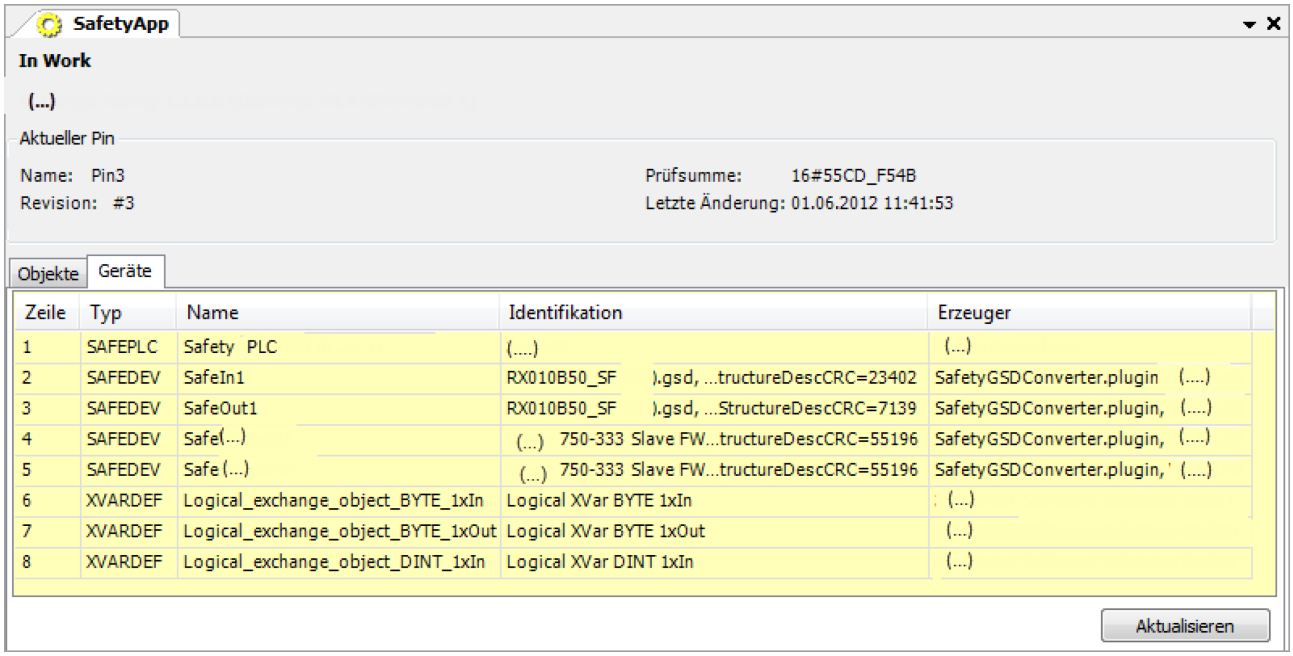


Abb. 28: Editor des Safety Applikationsobjekts mit Geräteliste

Die Registerkarte 'Geräte' zeigt folgende Informationen an:

- Auflistung der zur Sicherheitsapplikation gehörenden Feldgeräte und der Sicherheitssteuerung, zu der die Sicherheitsapplikation gehört, jeweils mit den dazugehörigen Informationen der Beschreibungsdateien.
  - *„Zeile“*
    Laufende Nummer der Einträge
  - *„TYP“*
    Typ des Geräts (siehe *Tab. 4 „Angezeigte Werte für Typ in der Geräteliste“ auf Seite 70*)
  - *„Name“*
    Name des Geräts im Projektbaum
  - *„Identifikation“*
    Gerätespezifische Informationen zur Identifikation
  - *„Erzeuger“*
    Informationen über den Erzeuger der gerätespezifischen Informationen

Tab. 4: Angezeigte Werte für Typ in der Geräteliste

Typ	Beschreibung
SAFEPLC	Sicherheitssteuerung, der die Applikation zugeordnet ist
SAFEDEV	Sicheres logisches Gerät der Applikation
STDDEV	Standard logisches Gerät der Applikation
XVARDEV	Standard logisches Gerät zum Variablenaustausch

Die Schaltfläche *„Aktualisieren“* dient dem Aktualisieren der Geräte-Liste, wenn sich bei geöffnetem Editor die Geräte (logische E/As) oder die Sicherheitssteuerung im Projektbaum geändert haben.

5.5.4.2 Logische E/As

5.5.4.2.1 Überblick logische E/As

Die logischen E/As dienen dem Austausch von Daten zwischen der Standard und der Sicherheitssteuerung. Zwei Arten von Daten können ausgetauscht werden: Globale Variablen und E/A-Daten. Die Sicherheitssteuerung selbst verfügt über keine E/A-Daten. Diese müssen in der Standardsteuerung konfiguriert und dann als logische E/As mit der Sicherheitssteuerung ausgetauscht werden. Wird in CODESYS in der Steuerungskonfiguration ein E/A Modul mit sicheren E/As eingefügt, so wird automatisch ein passendes

logisches E/A in der Sicherheitssteuerung eingefügt (gleicher Name wie das E/A Modul auf Standardseite). Dieses logische E/A enthält alle sicheren E/A-Kanäle sowie die Sicherheitsparameter des Moduls, so dass sämtliche sicherheitsrelevanten Informationen unter der Sicherheitsapplikation zu finden sind.

Prinzip der logischen E/As

Arten der logischen E/As

- logische EAs von sicheren Feldgeräten
  (siehe *Kapitel 5.5.4.2.2.1 „Logisches E/A eines sicheren physikalischen Geräts“ auf Seite 74*)
- logische EAs von Standard-Feldgeräten
  (siehe *Kapitel 5.5.4.2.2.2 „Logisches E/A eines Standard-feldgeräts“ auf Seite 75*)
- logische EAs von globalen Variablen zum Austausch mit der Standardsteuerung des Projekts (*„logische Austauschgeräte“*)
  (siehe *Kapitel 5.5.4.2.2.3 „Logisches E/A für Datenaustausch mit der Standardsteuerung“ auf Seite 77*)

Die logischen EAs der Sicherheitsapplikation sind mit physikalischen Geräten bzw. *„GVLs für logischen Austausch“* (spezielles Objekt auf Standardseite) der Standardapplikation verlinkt. Dies bedeutet, dass es zu jedem physikalischen Gerät, dessen Ein-/Ausgangssignale in der Sicherheitsapplikation verarbeitet werden, genau ein logisches E/A in der Sicherheitsapplikation gibt. Ebenso existiert für jede GVL für logischen Austausch der Standardsteuerung genau ein logisches E/A unter der Sicherheitsapplikation. Die Zuordnung ist variabel.

Die sicherheitsgerichteten Parameter eines Geräts, dessen Ein-/Ausgänge in der Sicherheitsapplikation verarbeitet werden, können nur in dem entsprechenden logischen E/A in der Sicherheitsapplikation editiert werden.

Verwaltet werden die Beschreibungen der logischen E/As in der Kategorie „Logische Geräte“ des Geräte-Repositorys.

Zur besseren Übersicht können im Projektbaum unter „Logische E/A“ Ordner hinzugefügt werden, um die logischen E/As zu gruppieren.

Vorteile der logischen E/As

Durch das Konzept der logischen E/As ergeben sich folgende Vorteile für die Entwicklung und die Verifikation einer CODESYS Safety Sicherheitsapplikation:

- Die Parametrierung der sicheren Parameter (z.B. F-Parameter bei PROFIsafe) von Feldgeräten erfolgt nur in den logischen E/As der Sicherheitsapplikation. Bei eingerichteter Safety-Benutzerverwaltung kann diese Parametrierung nur von Mitgliedern der Benutzergruppe Safety vorgenommen werden.
- Geänderte Zuweisungen von physikalischen Feldgeräten und GVLs für logischen Austausch verändern die Sicherheitsapplikation nicht, da die Änderungen von Zuweisungen unter der Standardapplikation erfolgen und durch eine Download auf die Standardsteuerung wirksam werden.
- Eine bereits verifizierte und abgenommene CODESYS Safety Sicherheitsapplikation kann losgelöst vom ursprünglichen Projekt komplett in eine anderes Projekt integriert werden, ohne dass diese neue Sicherheitsapplikation erneut verifiziert werden muss. Den logischen E/As der Sicherheitsapplikation müssen dabei passende Feldgeräte und GVLs für logischen Austausch neu zugewiesen werden.

Hinweise zu den logischen E/As

Die Zuweisung auf die logischen-E/As erfolgt abhängig vom Namen der Applikation und des logischen E/A-Objekts. Daraus ergeben sich folgende Hinweise:

- Wird der Name eines logischen E/As geändert, so wird der Name des physikalischen Geräts, bzw. der GVL für logischen Austausch automatisch nachgeführt, so dass es nicht neu auf das logische E/A zugeordnet werden muss.
- Das Umbenennen (außer bei "default Zuweisung") oder Verschieben von Feldgeräten im Gerätebaum, bzw. von GVLs für Logischen Austausch in der funktionalen Applikation ändert nichts an der Zuweisung auf das logische E/A. Eine Änderung der Sicherheitsapplikation erfolgt beim Umbenennen automatisch: das zugeordnete logische E/A wird analog zum Feldgerät umbenannt.

  Wird das Feldgerät in eine parallele andere Steuerung verschoben, so ist die Verknüpfung gelöst und das Feldgerät muss neu auf das logische E/A der Sicherheitsapplikation zugeordnet werden.
- Das Löschen des Feldgeräts im Gerätebaum bzw. der GVL für logischen Austausch in der Standardapplikation bedeutet, dass das zugeordnete logische E/A auf nichts mehr gemappt wird (nicht mehr versorgt wird).
- Das Löschen des logischen E/A-Objekts bedeutet, dass keine sicherheitsgerichteten Einstellungen mehr auf das Feldgerät gemappt werden bzw., dass die „von Safety“ Variablen in der GVL für Logischen Austausch nicht mehr versorgt werden.

- Wenn ein anderes logisches E/A-Objekt den alten Namen eines umbenannten oder gelöschten logischen E/A-Objekts erhält, dann werden ab jetzt dessen Einstellungen auf das zugeordneten Feldgerät gemappt bzw. dessen Variablenwerte mit der Austausch-GVL ausgetauscht. Die Zuweisung wird somit indirekt geändert.
  Wurde ein logisches E/A gelöscht und es wird ein neues logisches E/A mit dem Namen des gelöschten logischen E/As eingefügt, dann ist die Zuweisung wieder aktiv.
- Wird ein physikalisches E/A-Objekt kopiert, das mit einem logischen E/A-Objekt verlinkt ist, so wird das logische E/A-Objekt mit seinen Daten ebenfalls kopiert.

Ersatzwerte

Wenn nicht anders festgelegt, wird für Werte, die nicht aktualisiert werden können eine "0" geliefert. Weitere Informationen siehe ↳ *Kapitel 7.8 „Abstimmung mit der Standardsteuerung“ auf Seite 186*

Objekteigenschaften

Der Eigenschaften-Dialog von allen logischen E/As enthält zusätzlich zu den Registerkarten *„Allgemein“* und *„Zugriffskontrolle“* die Registerkarte *„Safety“* (siehe Abb. 29). Informationen der Registerkarte *„Safety“* siehe ↳ *Kapitel 5.5.4.1 „Safety Applikationsobjekt“ auf Seite 61*

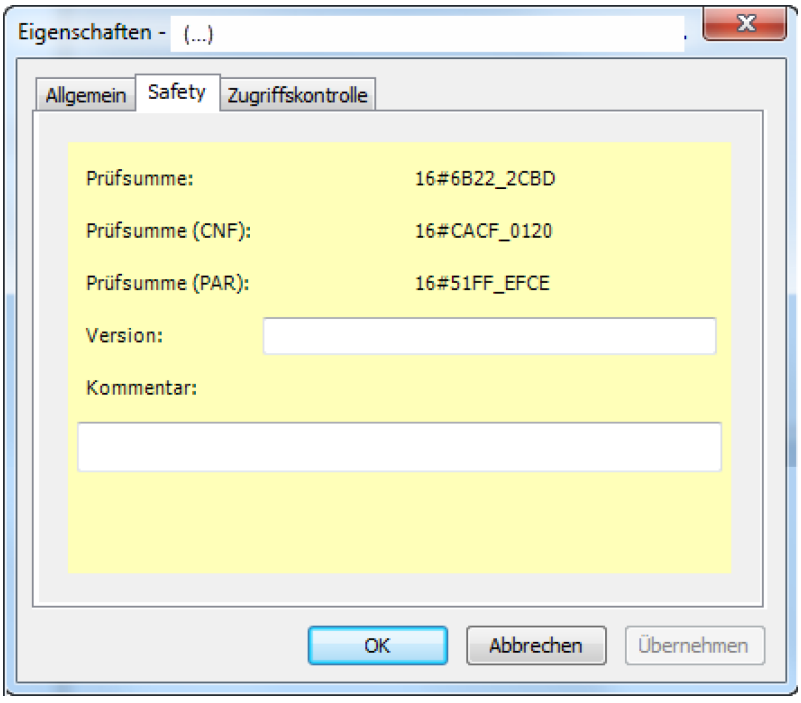


Abb. 29: Dialog: Eigenschaften von logischen E/As, Registerkarte 'Safety'

5.5.4.2.2 Verwendungstypen der logischen E/As

5.5.4.2.2.1 Logisches E/A eines sicheren physikalischen Geräts

Diese logischen E/As dienen dem Austausch von sicherheitsgerichteten E/As zwischen der Sicherheits- und der Standardsteuerung.

Informationen, wie Sie sichere 1oo1-, bzw. 1oo2-Eingabegeräte mit der Sicherheitsapplikation verbinden, und worauf Sie dabei achten müssen, finden Sie im Kapitel ↳ Kapitel 6.5.2 „Anbindung digitaler 1oo1- und 1oo2-Eingangsmodule“ auf Seite 146

Für diesen Austausch muss zuerst das sichere Feldgerät unter der Standardsteuerung wie in CODESYS eingefügt werden.

Hinzufügen eines sicheren Feldgeräts unter der Standardsteuerung

1. Im Projektbaum den entsprechenden Feldbus-Slave unter der Standardsteuerung selektieren.
2. Kontextmenübefehl *„Gerät anhängen...“* aktivieren.
3. Im sich öffnenden Dialog unter der Kategorie *„Feldbusse“* aus der entsprechenden Unterkategorie (z.B. *„Sichere Module“*) das gewünschte sichere Feldgerät auswählen.
4. Schaltfläche *„Schließen“* aktivieren.

Beim Einfügen eines sicheren Feldgeräts unter der Standardsteuerung wird das entsprechende logische E/A automatisch unter den Knotenpunkt „Logische E/As“ der Sicherheitsapplikation eingefügt Voraussetzung: Unter der Standardsteuerung, unter der das sichere Feldgerät eingehängt wird, befindet sich nur eine Sicherheitssteuerung.

Typkonsistenz der E/A-Kanäle

VORSICHT!

Die Typkonsistenz der E/A-Kanäle ist nur gewährleistet,

- wenn der Applikationsstand auf der Sicherheitssteuerung und auf der Standardsteuerung dem Stand des gleichen übersetzbaren Projekts entsprechen

  und
- wenn die Feldgeräte im Projekt den Feldgeräten in der Maschine entsprechen. – Je nach Bussystem wird ein Mismatch hier automatisch erkannt (z.B. bei PROFIBUS).

5.5.4.2.2.2 Logisches E/A eines Standardfeldgeräts

Diese logischen E/As dienen dem E/A-Datenaustausch zwischen Standardfeldgeräten und der Sicherheitssteuerung.

Werden Standard-E/As in der Sicherheitsapplikation verwendet, so sind diese Daten nicht sicher!

Zuerst wird das gewünschte Standard-Feldgerät geräteabhängig unter der Standardsteuerung eingefügt. Die Vorgehensweise entspricht CODESYS Standard.

In der aktuellen Version von CODESYS Safety können nur PROFINET- und PROFIBUS-Geräte als Standard-Geräte in die Sicherheitsapplikation eingebunden werden.

Anschließend muss das entsprechende logische E/A ("Generiertes logisches Gerät") manuell unter dem Safety Applikationsobjekt eingefügt werden:

Hinzufügen des "Generierten logischen Geräts"

1. Im Projektbaum den Knotenpunkt *„Logische E/As“* des Safety Applikationsobjekts selektieren
2. Kontextmenübefehl *„Objekt hinzufügen“* mit Auswahl *„Logisches Gerät“* aktivieren
3. Im Dialog *„Logisches Gerät hinzufügen“* (siehe Abb. 30) in der Kategorie *„Logische Geräte“*, Unterkategorie *„Generierte logische Geräte“* das entsprechende logische E/A auswählen
4. Schaltfläche *„Hinzufügen“* aktivieren.

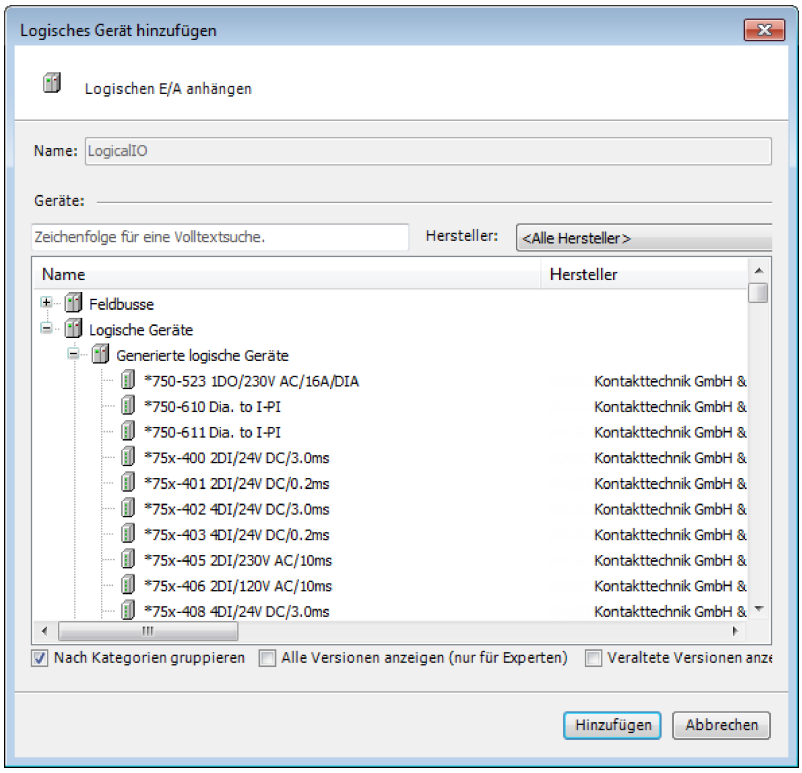


Abb. 30: Dialog 'Logisches Gerät hinzufügen'

Editieren der Variablen siehe *„I/O Abbild“ auf Seite 81*

Verbindung eines Feldgeräts mit dem logischen E/A

Nach erfolgreichem Einfügen des Standard-Feldgeräts unter der Standardsteuerung und des entsprechenden logische E/As unter der Sicherheitssteuerung müssen diese miteinander "verbunden" werden, um E/A-Daten austauschen zu können.

Zuweisung eines Feldgeräts auf das logische E/A

1. Standard-Feldgerät im Projektbaum selektieren
2. Kontextmenübefehl *„Objekt bearbeiten“* aktivieren
3. Tab *„(...) I/O Abbild“*(bei Profibus) öffnen
4. Combobox *„Logisches I/O Abbild“* anklicken
5. Aus der sich öffnenden Liste das entsprechende logische E/A auswählen

   (siehe Abb. 31)

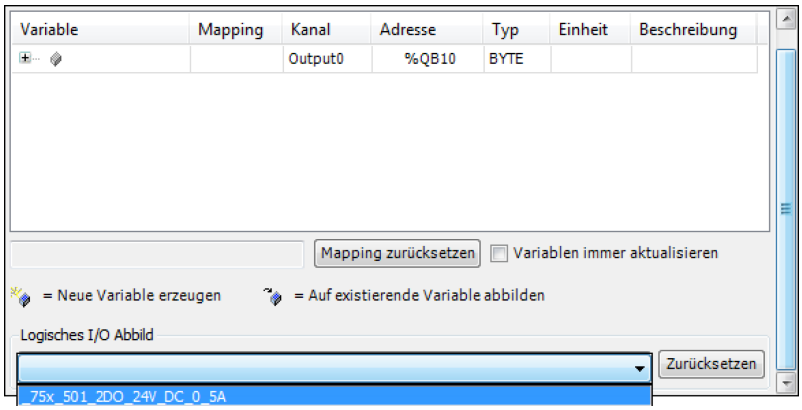


Abb. 31: Bsp. Standardgeräte-Registerkarte '(...) I/O Abbild' mit geöffneter Combobox "Logisches I/O Abbild"

Unter den logischen E/As sind nur solche auswählbar, die noch nicht bereits auf andere Geräte bzw. GVLs für logischen Austausch gemappt sind. Vom System wird nur die Auswahl eines logischen E/As akzeptiert, das die gleiche Gerätebeschreibung wie das physikalische Gerät besitzt.

Als Stackinstanz wird ein Funktionsbaustein vom Typ NonSafeIO erzeugt.

Typkonsistenz der E/A-Kanäle

HINWEIS!

Die Typkonsistenz der E/A-Kanäle ist nur gewährleistet,

- wenn der Applikationsstand auf der Sicherheitssteuerung und auf der Standardsteuerung dem Stand des gleichen übersetzbaren Projekts entsprechen
  und
- wenn die Feldgeräte im Projekt den Feldgeräten in der Maschine entsprechen. – Je nach Bussystem wird ein Mismatch hier automatisch erkannt (z.B. bei PROFIBUS).

5.5.4.2.2.3 Logisches E/A für Datenaustausch mit der Standardsteuerung

Der Datenaustausch zwischen Sicherheitssteuerung und Standardsteuerung erfolgt über Variablen, die in den logischen E/As *„Logisches Austauschgerät“* definiert werden. Auf der Standardseite wird dazu eine *„GVL für logischen Austausch“* angelegt, die mit dem entsprechenden *„Logischen Austauschgerät“* verbunden wird.

Diese Daten werden in der Sicherheitssteuerung wie Ein-/Ausgänge verwendet, in der Standardapplikation sind sie als globale Variablen verfügbar

Der Datenfluss zwischen zwei Variablen ist eindeutig. Dies bedeutet, dass die gleiche Variable der einen Applikation nicht mit zwei Variablen der anderen Applikation ausgetauscht werden kann.

Hinzufügen der "GVL für logischen Austausch"

1. Im Projektbaum das Standardapplikationsobjekt selektieren
2. Kontextmenübefehl *„Objekt hinzufügen“* mit Auswahl *„GVL für logischen Austausch“* aktivieren
3. Im Dialog *„GVL für Logischen Austausch hinzufügen“* kann ein Name für die GVL editiert werden. Der Default-Name lautet *„Logical_GVL“*.
4. Schaltfläche *„Hinzufügen“* aktivieren

Hinzufügen des "Logischen Austauschgeräts"

1. Im Projektbaum den Knotenpunkt *„Logische E/As“* des Safety Applikationsobjekts selektieren
2. Kontextmenübefehl *„Objekt hinzufügen“* mit Auswahl *„Logisches Gerät“* aktivieren
3. Im Dialog *„Logisches Gerät hinzufügen“* (siehe Abb. 32) in der Kategorie *„Logische Geräte“*, Unterkategorie *„Logisches Austauschgerät“* das gewünschte logische E/A auswählen
4. Schaltfläche *„Hinzufügen“* aktivieren.

Editieren der Austauschvariablen siehe „*I/O Abbild*“ *auf Seite 81*

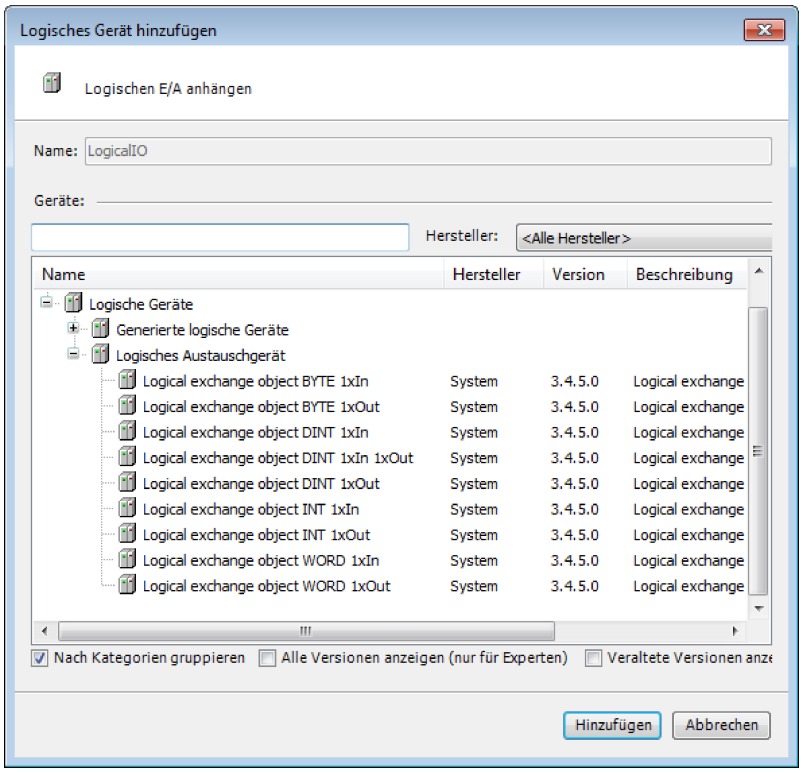


Abb. 32: Dialog 'Logisches Gerät hinzufügen'

Bei der Auswahl des *„Logischen Austauschgeräts“* werden für die einzelnen Variablen festgelegt:

- die Austauschrichtung: IN oder OUT
  IN: von der Standardapplikation in die Sicherheitsapplikation
  OUT: von der Sicherheitsapplikation zur Standardapplikation
- der Datentyp: BYTE, DINT, INT oder WORD

Als Stackinstanz wird ein Funktionsbaustein vom Typ NonSafeIO erzeugt.

Die Austauschvariablen können nur im logischen EA editiert werden. In der GVL für Logischen Austausch können Variablen nicht eingegeben oder verändert werden.

Es können nur Daten ausgetauscht werden, deren Variablen einen Standard-Datentyp besitzen. Variablen eines Datentyps mit Präfix SAFE können nicht zwischen einer Sicherheitssteuerung und einer Standardsteuerung ausgetauscht werden.

Verbindung der "GVL für Logischen Austausch" auf das "Logische Austauschgerät"

1. GVL für Logischen Austausch der Standardapplikation im Projektbaum selektieren
2. Kontextmenübefehl *„Objekt bearbeiten“* aktivieren
3. Combobox *„Abbild für Logischen Austausch“* anklicken
4. Aus der sich öffnenden Liste das entsprechende logische Austauschobjekt auswählen (siehe Abb. 33)

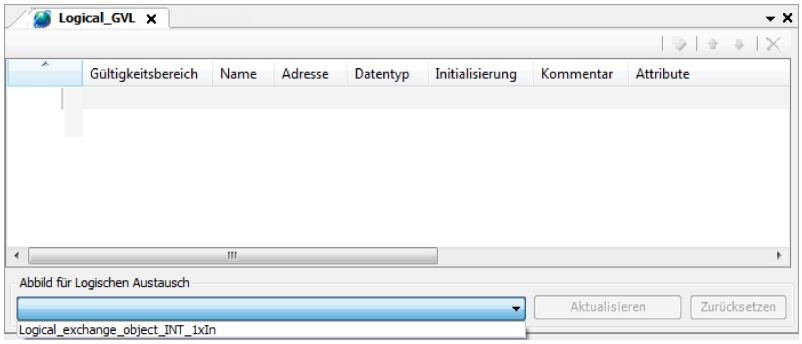


Abb. 33: Editor der GVL für Logischen Austausch mit geöffneter Combobox

In der Combobox stehen alle noch nicht gemappten logischen Austauschobjekte zur Verfügung.

Wird ein logisches Austauschgerät ausgewählt, so werden die Variablen implizit aktualisiert.

Werden im verbundenen logischen Austauschgerät Änderungen vorgenommen, so muss in der GVL für Logischen Austausch die Schaltfläche *„Aktualisieren“* aktiviert werden, um die Variablenliste zu aktualisieren.

Durch Aktivierung der Schaltfläche *„Zurücksetzen“* wird eine bestehende Verbindung aufgelöst.

Hinweise zum Datenaustausch zwischen Standard- und Sicherheitssteuerung

Eine Änderung des Variablenaustauschs kann nur durch einen erneuten Download der Sicherheitsapplikation und der Standardapplikation aktiv werden.

Ersatzwerte

Solange die Applikation nicht beendet ist, aber gerade keine aktuellen Werte ausgetauscht werden können, dann werden für den Variablenaustausch Ersatzwerte verwendet, *„Unterbrechungen durch die Standardsteuerung“ auf Seite 187*

Typkonsistenz der E/A-Kanäle

HINWEIS!
Die Typkonsistenz der E/A-Kanäle ist nur gewährleistet,

– wenn der Applikationsstand auf der Sicherheitssteuerung und auf der Standardsteuerung dem Stand des gleichen übersetzbaren Projekts entsprechen

5.5.4.2.3 Editor der logischen E/As

5.5.4.2.3.1 Information und I/O Abbild

Information

Der Editor aller logischen E/As besitzt die Registerkarte *„Information“* (siehe Abb. 34), welche genaue Informationen und gegebenenfalls ein Bild des jeweiligen logischen E/As enthält.

Als Informationen werden aufgelistet: Name, Hersteller, Kategorien, Typ, ID, Version, Bestellnummer und Beschreibung.

Mögliche Kategorien

- Sichere Module
- Generierte logische Geräte
- Logisches Austauschgerät

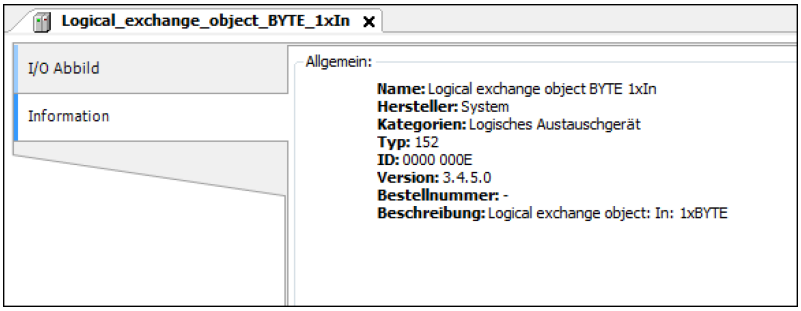


Abb. 34: Registerkarte 'Information' eines logischen E/A

I/O Abbild

Im *„I/O Abbild“* werden die Variablen, die dem Zugriff der Sicherheitsapplikation auf die E/As dienen, definiert.

In CODESYS Safety können E/A-Kanäle nur auf neue Variablen, d.h. nicht auf bereits existierende Variablen, gemappt werden.

Auf Eingangskanäle gemappte Variablen enthalten die entsprechenden Eingangssignale von Feldgeräten und können somit gelesen werden. Variablen, die auf Ausgangskanäle gemappt sind, können geschrieben werden und setzen Ausgangssignale in Feldgeräten.

Für jeden Eingangs- bzw. Ausgangskanal eines E/A-Moduls, welchem eine Variable zugeordnet wurde, wird in der Sicherheitsspplikation eine implizite globale Variable mit dem entsprechenden Namen und dem in der Spalte *„Typ“* angegebenen Datentyp angelegt.

Bereiche der Registerkarte I/O Abbild

- Liste der Variablen des I/O Abbilds mit: Variable (Name), Kanal (Input bzw. Output), Typ, Einheit und Beschreibung
  Die Informationen der Spalten *„Kanal“*, *„Typ“* (IEC Datentyp) , *„Einheit“* und *„Beschreibung“* sind in der Gerätebeschreibungsdatei festgelegt und können nicht verändert werden.
- Physikalisches I/O: Anzeige des mit diesem logischen E/A verbundenen Objekt der Standardapplikation
- Instanzen: Liste der impliziten Instanzen. Diese stehen der Sicherheitsapplikation als globale Variablen zur Verfügung. (siehe *Kapitel 5.5.4.2.4 „Verwendung logischer E/As im Projekt“ auf Seite 86*)

Editieren der Variablen

Die Mapping-Variablen werden in der Registerkarte *„I/O Abbild“* des logischen E/As editiert und angezeigt. Um eine Variable zu editieren, muss die entsprechende Zelle mit einem Doppelklick geöffnet werden.

Durch Aktivierung der Schaltfläche *„Abbild zurücksetzen“* werden alle in der Tabelle eingetragenen Mapping-Variablen gelöscht, d.h. die Abbildung vom physikalischen Gerät auf das logische E/A wird zurückgesetzt.

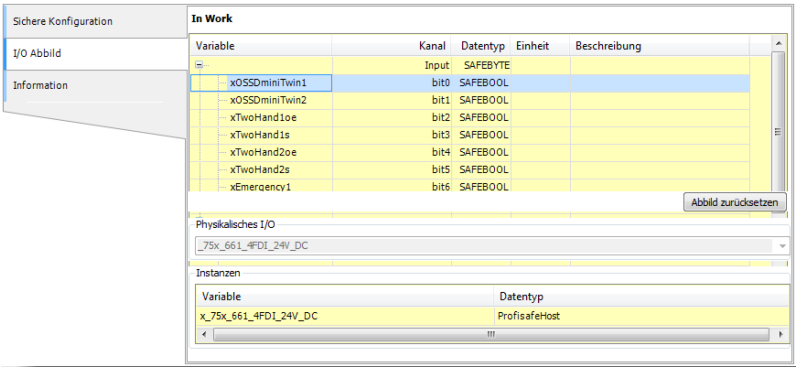


Abb. 35: Registerkarte 'I/O Abbild' des logischen E/s eines sicheren Feldgeräts

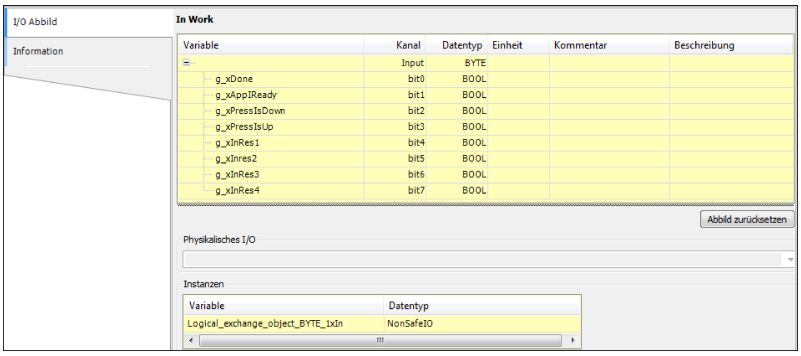


Abb. 36: Registerkarte 'I/O Abbild' eines logischen E/As, Typ Byte mit Bit-Zugriff

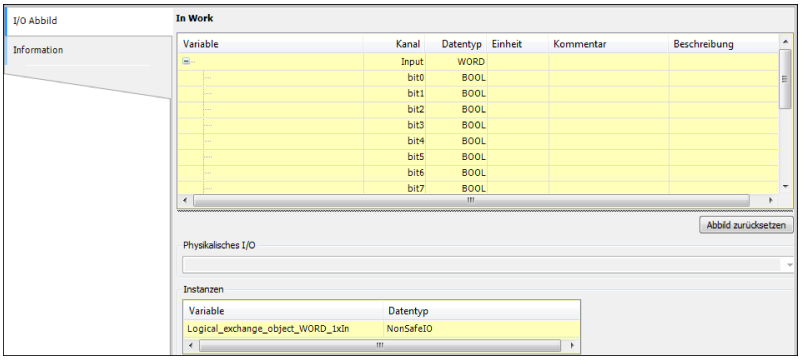


Abb. 37: Registerkarte 'I/O Abbild' eines logischen E/As, Typ WORD

Durchgeführte Änderungen in der Registerkarte „I/O Abbild“ werden rot gekennzeichnet. Es wird dabei immer nur die zuletzt durchgeführte Änderung markiert. Mit Schließen des Editors werden sämtliche Änderungsmarkierungen entfernt.

Verwendung der im *„I/O Abbild“* definierten Variablen im Projekt siehe ↬ *Kapitel 5.5.4.2.4 „Verwendung logischer E/As im Projekt“ auf Seite 86*

Der Editor der logische E/As von Standardgeräten besitzt keine Registerkarte „Sichere Konfiguration“ und „Sichere Geräteparametrierung“. Die Konfiguration und die Geräteparametrierung erfolgt unter der Standardapplikation wie in CODESYS Standard.

Wenn ein Gerät der Sicherheitssteuerung zugewiesen ist, so zeigt die Registerkarte „I/O Abbild“ des Geräteeditors nur diese Zuweisung und nicht mehr das Kanalmapping auf die Variablen der Standardsteuerung.

5.5.4.2.3.2 Sichere Parametrierung und Sichere Konfiguration

Die Registerkarte *„Sichere Parametrierung“* und *„Sichere Konfiguration“* des Editors der logischen E/As gibt es nur bei den sicherheitsgerichteten E/As.

HINWEIS!
Der Geräteeditor ist für die Darstellung und Bearbeitung der Geräteparameter bestimmter Geräte geeignet. Genaue Informationen finden Sie in der Dokumentation des jeweiligen Geräts, bzw. beim jeweiligen Gerätehersteller.

VORSICHT!
Feldbusspezifische Anforderungen an die Einstellung bestimmter Parameter sind zu berücksichtigen.

HINWEIS!
Der Anwender ist dafür verantwortlich, dass die Geräte entsprechend der Gerätedokumentation des jeweiligen Geräts, bzw. des jeweiligen Geräteherstellers korrekt parametriert werden.

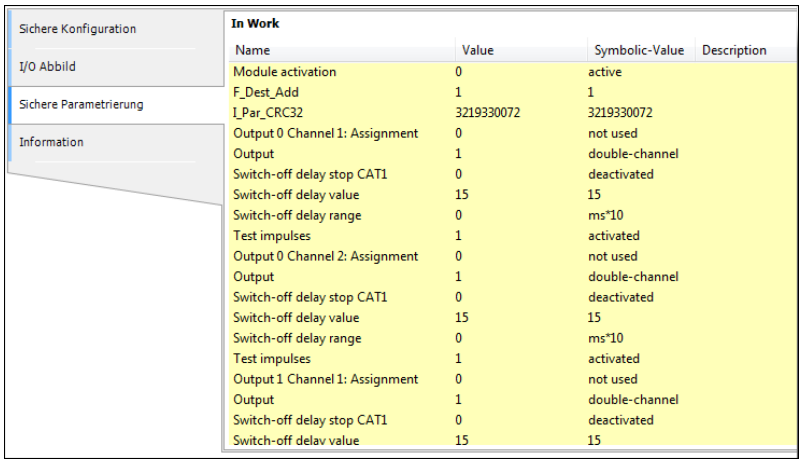
HINWEIS!
Der Gerätehersteller muss den Anwender über die Bedingungen für die Berechnung der System-Kennwerte informieren.

Im oberen Bereich steht die Pin-Information, bzw *„In Work“*. Im anschließenden Bereich sind jeweils die Parameter in tabellarischer Form aufgelistet.

„Sichere Parametrierung“ (siehe Abb. 38) listet die Parameter des jeweiligen sicheren Geräts auf.

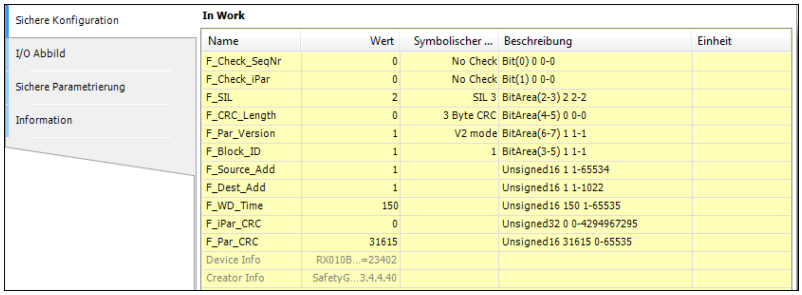
„Sichere Konfiguration“ (siehe Abb. 39) enthält die Konfigurationsparameter für die sichere Kommunikation.

Integer-Parameter können je nach Gerätebeschreibung auch hexadezimal angezeigt werden. Diese optionale Einstellung muss in der Gerätebeschreibungsdatei erfolgen.



Name	Value	Symbolic-Value	Description
Module activation	0	active	
F_Dest_Add	1	1	
I_Par_CRC32	3219330072	3219330072	
Output 0 Channel 1: Assignment	0	not used	
Output	1	double-channel	
Switch-off delay stop CAT1	0	deactivated	
Switch-off delay value	15	15	
Switch-off delay range	0	ms*10	
Test impulses	1	activated	
Output 0 Channel 2: Assignment	0	not used	
Output	1	double-channel	
Switch-off delay stop CAT1	0	deactivated	
Switch-off delay value	15	15	
Switch-off delay range	0	ms*10	
Test impulses	1	activated	
Output 1 Channel 1: Assignment	0	not used	
Output	1	double-channel	
Switch-off delay stop CAT1	0	deactivated	
Switch-off delay value	15	15	

Abb. 38: Registerkarte 'Sichere Parametrierung'



Name	Wert	Symbolischer ...	Beschreibung	Einheit
F_Check_SeqNr	0	No Check	Bit(0) 0 0-0	
F_Check_iPar	0	No Check	Bit(1) 0 0-0	
F_SIL	2	SIL 3	BitArea(2-3) 2 2-2	
F_CRC_Length	0	3 Byte CRC	BitArea(4-5) 0 0-0	
F_Par_Version	1	V2 mode	BitArea(6-7) 1 1-1	
F_Block_ID	1	1	BitArea(3-5) 1 1-1	
F_Source_Add	1		Unsigned16 1 1-65534	
F_Dest_Add	1		Unsigned16 1 1-1022	
F_WD_Time	150		Unsigned16 150 1-65535	
F_iPar_CRC	0		Unsigned32 0 0-4294967295	
F_Par_CRC	31615		Unsigned16 31615 0-65535	
Device Info	RX010B...=23402			
Creator Info	SafetyG...3.4.4.40			

Abb. 39: Beispiel: "Sichere Konfiguration"

Durchgeführte Änderungen in den Registerkarten „Sichere Parametrierung“ und „Sichere Konfiguration“ werden rot gekennzeichnet. Es wird dabei immer nur die zuletzt durchgeführte Änderung markiert. Mit Schließen des Editors werden sämtliche Änderungsmarkierungen entfernt.

Detaillierte Beschreibung der busspezifischen sicheren Parameter

- ProfiSafe-Geräte siehe ↳ *Kapitel 14.2.2 „PROFIsafe Parameter: F-Parameter und i-Parameter“ auf Seite 279*
- FSoE-Geräte siehe ↳ *Kapitel 14.3.2 „FSoE Parameter“ auf Seite 284*

5.5.4.2.4 Verwendung logischer E/As im Projekt

Jede im *„I/O Abbild“* eines logischen E/As deklarierte Mapping-Variable (Kanalvariablen) und alle erzeugten Instanzen der logischen E/As von sicheren Geräten und von Standardgeräten stehen dem Entwickler bei der Programmierung der Sicherheitsapplikation als globale Variablen zur Verfügung. Um sie zur Code-Implementierung in einer POU verwenden zu können, muss sie im Deklarationsteil der POUs als VAR_EXTERNAL deklariert werden. (Variablendeklaration siehe *Kapitel 6.3.3.1 „Allgemeines zu Variablen“ auf Seite 120*)

Alternativ zur expliziten Deklaration können diese Variablen und Instanzen im Implementierungsteil von POUs entweder in der Eingabehilfe oder in der automatisch angezeigten Auswahlliste "Intellisense" ausgewählt werden.

Variablen, die auf einen E/A-Kanal gemappt werden, können beim Debuggen des Programms geschrieben und geforct werden!

5.5.4.3 POUs

POUs (Programm Organisation Units) sind die Programmierobjekte von CODESYS Safety, die entweder als Programm (*„PROGRAM“*) oder als Funktionsbaustein (*„FUNCTION_BLOCK“*) deklariert sind.

Im Projektbaum der Sicherheitsapplikation können beliebig viele POUs hinzugefügt werden.

Merkmale von Programm und Funktionsbaustein:

- Programm

  Ein Programm kann nicht von anderen Programmen aufgerufen werden, kann jedoch Instanzen von Funktionsbausteinen aufrufen.

  Programme werden direkt von der Safety Task aufgerufen. Welche Programme aufgerufen werden, wird im Objekt *„Safety Task“* festgelegt. Nur die aufgerufenen Programme werden auf der Steuerung ausgeführt.
- Funktionsbaustein

  Funktionsbausteine werden immer über eine Instanz aufgerufen, die eine Kopie des Funktionsbausteins ist welche die Daten enthält. Jede Instanz hat einen Bezeichner (Instanzname) und eine Datenstruktur, die ihre Eingabe-, Ausgabe- und internen Variablen beinhaltet.

  In Funktionsbausteinen können Instanzen von Funktionsbausteinen aufgerufen werden.

  Die Deklaration und Verwendung von Funktionsbausteinen erfolgt wie in Standard CODESYS und wird hier nicht weiter erläutert.

Hinzufügen einer POU

1. Im Projektbaum *„SafetyApp“* selektieren
2. Kontextmenü-Befehl *„Objekt hinzufügen“* mit Auswahl *„Safety Basic-POU“* oder *„Safety Extended-POU“* aktivieren
3. Im Dialog *„Safety Basic-POU hinzufügen“* bzw. *„Safety Extended-POU hinzufügen“* (siehe Abb. 40) den Namen und Kommentar) der POU eingeben und Baustein-Typ*„PROGRAM“* oder *„FUNCTION_BLOCK“* auswählen.
4. Im Fall, dass der Bausteintyp *„FUNCTION_BLOCK“* ist, kann die Checkbox *„PLCopen“ „Einmaliger Aufruf“* gesetzt werden. Wird diese Checkbox nicht gesetzt, kann die POU mehrfach aufgerufen werden.

   Bei dem Bausteintyp *„PROGRAM“* ist die Checkbox für den einmaligen Aufruf automatisch gesetzt und kann nicht geändert werden.
5. Schaltfläche *„Hinzufügen“* aktivieren

HINWEIS!

Kommentierung von POUs

Zu jeder POU sollten im Feld *„Kommentar“* gemäß PLCopen folgende Informationen enthalten sein:

- Autor
- Erstelldatum der POU
- Freigabedatum
- Version
- Versionsgeschichte
- funktionale Beschreibung (einschließlich E/A-Parameter)

VORSICHT!

Bei einem Safety Projekt mit der Safety-Benutzerkonfiguration, muss der Entwickler gleich nach dem Einfügen folgende Einstellungen vornehmen, damit nur berechtigte Personen die neue Extended-POU bearbeiten können: Im Dialog *„Eigenschaften“* den Tab *„Zugriffskontrolle“* öffnen und der Benutzergruppe *„Safety“* die Aktionen *„Bearbeiten“* und *„Entfernen“* der Extended-POU verbieten.

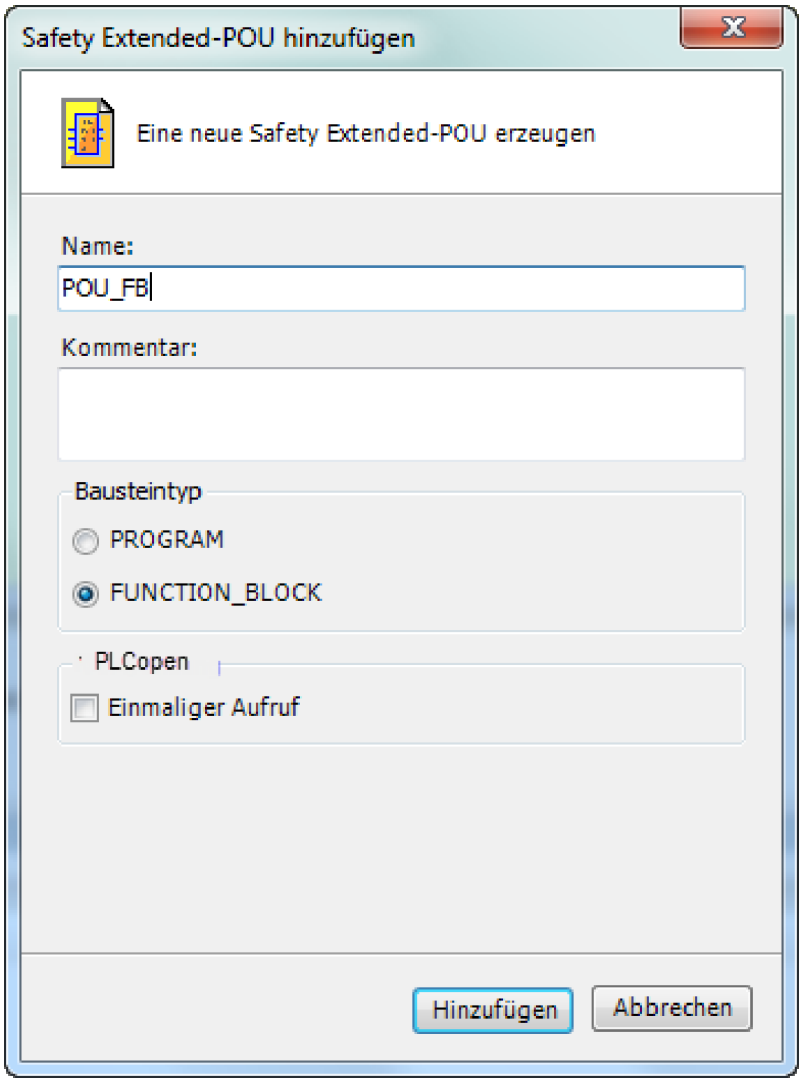


Abb. 40: Beispiel: Dialog beim Erstellen einer Extended-POU, Bausteintyp: "FUNCTION_BLOCK"

Editor einer POU und Erstellung des Programmcodes in POUs siehe ↳ *Kapitel 6.3.2 „POUs“ auf Seite 119*

Objekteigenschaften einer POU

Jede POU der Sicherheitsapplikation besitzt einen Eigenschaften-Dialog mit den Registerkarten *„Allgemein“*, *„Safety“* und *„Zugriffskontrolle“*. Der Eigenschaften-Dialog wird durch Selektion der entsprechenden POU im Projektbaum und Aktivierung des Kontextmenü-Befehls *„Eigenschaften...“* geöffnet.

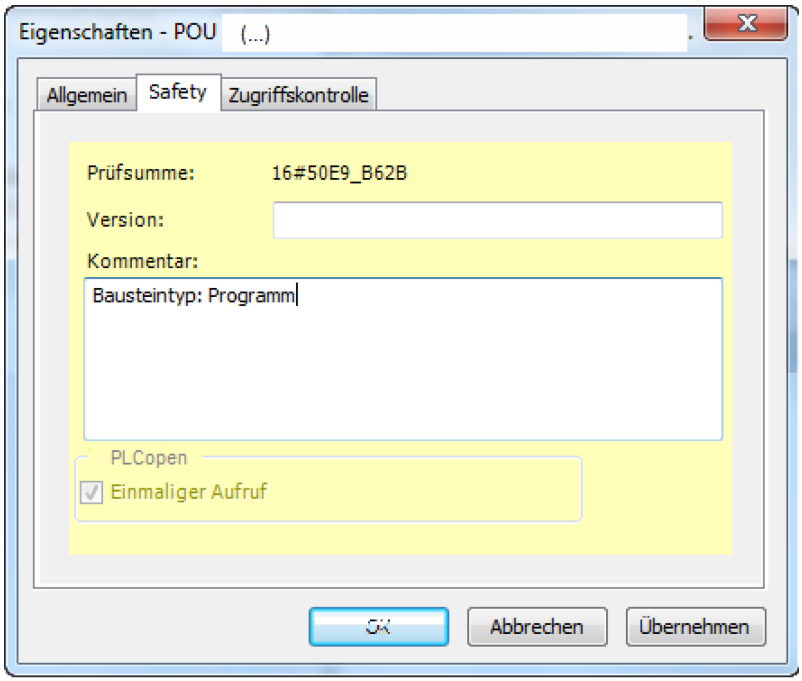


Abb. 41: Dialog Eigenschaften einer POU, Registerkarte 'Safety'

- *„ Prüfsumme“*
  Prüfsumme zu dieser POU
- *„Version“* (editierbar)
  Die Version kann der Entwickler frei vergeben. Mithilfe der Version kann in der Objektliste des Safety Applikationsobjekts schnell erkannt werden, um welche Version des Objekts es sich handelt.
- *„Kommentar “*(editierbar)
- *„PLCopen “*
  Die Checkbox *„Einmaliger Aufruf“* ist
  – beim Bausteintyp *„PROGRAM“* automatisch gesetzt. Die Einstellung kann nicht verändert werden
  – beim Bausteintyp *„FUNCTION_BLOCK“* aktivierbar und kann gesetzt oder zurückgesetzt werden.

Details zum POU-Editor siehe *Kapitel 6.3.2 „POUs“ auf Seite 119*

5.5.4.4 Safety Task

Im Gerätebaum muss genau eine *„Safety Task“* unter dem Safety Applikationsobjekt vorhanden sein. Diese Safety Task ruft die Programme (POUs vom Typ PROGRAM) des Safety Applikationsobjekts auf. Die Aufrufreihenfolge entspricht der Anordnung der Programme in der Liste des Editors der Safety-Task von oben nach unten. Die Aufrufreihenfolge kann im Editor verändert werden. Die

Safety Task definiert also diejenigen Programme, die auf der Steuerung ausgeführt werden. Die Programmauswahl und ihre Aufrufreihenfolge werden durch den Entwickler festgelegt. Die im Editor aufgelisteten Programme sind gleich mit den Programm-POUs des Projektbaums.

Hinzufügen der Safety Task

Das Objekt *„Safety Task“* wird beim Hinzufügen der Sicherheitssteuerung automatisch unter dem Safety Applikationsobjekt hinzugefügt.

Erfolgt das Hinzufügen der Safety Task nicht automatisch oder wurde sie gelöscht, so kann das Objekt über das Kontextmenü-Befehl *„Objekt hinzufügen“* mit der Auswahl *„Safety Task“* dem Safety Applikationsobjekt hinzugefügt werden.

Objekteigenschaften der Safety Task

Die Registerkarte *„Safety“* des Eigenschaften-Dialogs enthält die Prüfsumme des Objekts Safety Task und editierbare Felder für die Version und den Kommentar der Safety Task.

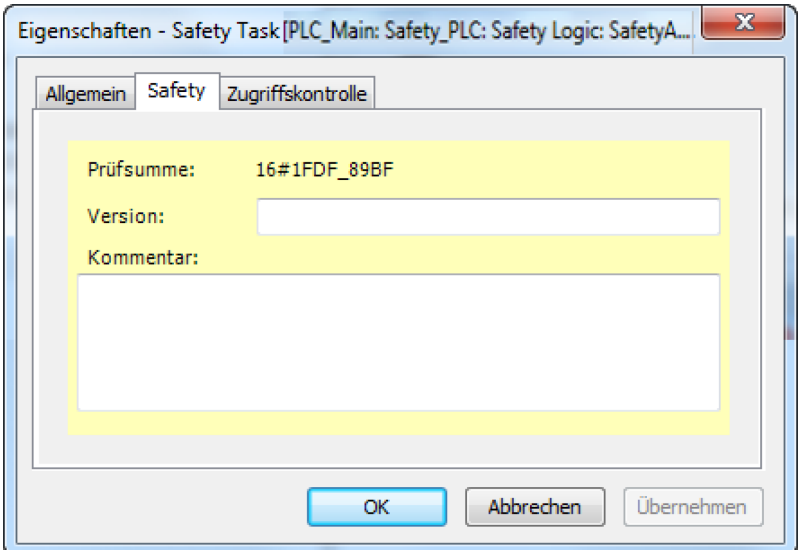


Abb. 42: Dialog: Eigenschaften der Safety Task, Registerkarte 'Safety'

Editor der Safety Task

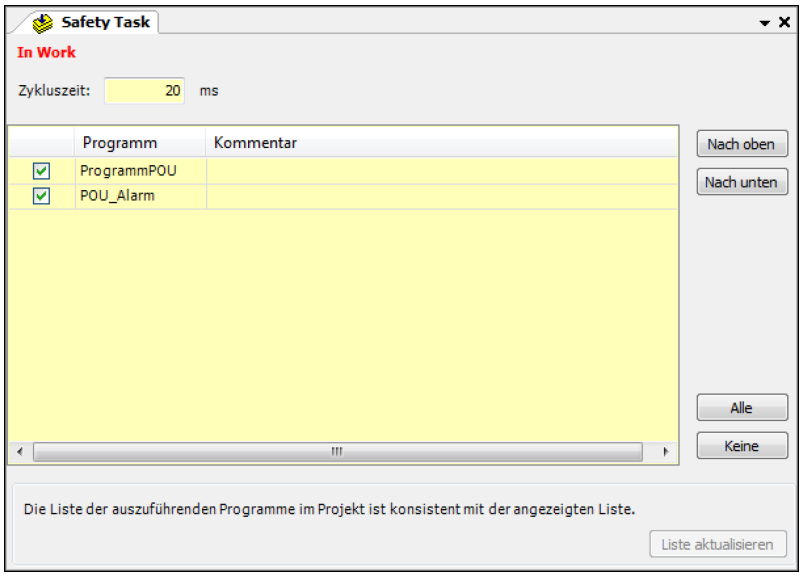


Abb. 43: Editor: Safety Task

Der Editor des Task-Objekts besteht aus 3 Bereichen:

- Pin-Information
- Editierbare Anzeige der Zykluszeit
- Liste der Programme mit Schaltflächen

Zykluszeit

Der Entwickler kann den Wert der Zykluszeit, verändern. Sie wird in ganzen ms (Millisekunden) eingegeben. Als Default wird eine für das angewählte Gerät sinnvolle Zykluszeit voreingestellt (im Beispiel 20 ms).

Wird ein kleinerer Wert als der für diese Sicherheitssteuerung zulässige eingegeben, so wird der Minimalwert verwendet.

Das gleiche gilt für den Maximalwert.

Durch eine Veränderung der Zykluszeit ändern sich auch die Pin-Prüfsumme und die Prüfsumme des Objekts Safety Task!

Programmliste

Die Programmliste enthält alle Programme (POUs vom Typ PROGRAM) der Sicherheitsapplikation.

Die Markierung in der ersten Spalte der Programmliste zeigt an, welche Programme ausgeführt werden. Durch Deaktivierung der Checkbox in der ersten Spalte können die markierten Einträge der Liste auskommentiert werden. Programme von auskommentierten Einträgen werden auf der Steuerung nicht ausgeführt. Auskommentierungen werden durch Aktivieren der Schaltfläche rückgängig gemacht, die entsprechende Programme werden von der Task aufgerufen.

Die Schaltflächen *„Nach oben“* und *„Nach unten“* dienen dazu, die Aufrufreihenfolge der Programme zu ändern:

„Nach oben“: der aktuell selektierte Programmeintrag wird in der Programmliste um eine Position nach oben verschoben.

„Nach unten“: der aktuell selektierte Programmeintrag wird in der Programmliste um eine Position nach unten verschoben.

Durch Aktivieren der Schaltfläche *„Alle“* werden alle Programme der Liste markiert. Aktivierung der Schaltfläche *„Keine“* werden alle Programme abgewählt.

Die zuletzt durchgeführte Änderung in Feldern der Programmliste wird rot markiert. Diese farbliche Markierung wird beim Schließen des Editors entfernt.

Aktualisierung der Programmliste:

Die Programmliste des Safety Task-Objekts wird bei Änderungen der Projektstruktur im Projektbaum automatisch aktualisiert. **Ausnahme:**

Besitzt der Entwickler in der Benutzerverwaltung nicht das Recht zum Bearbeiten des Objekts Safety Task des Projekts, so wird die Programmliste des Task-Objekts nicht automatisch aktualisiert. In diesem Fall ist die Schaltfläche *„Aktualisieren“* aktivierbar, kann aber nur von einem Entwickler aktiviert werden, der das Recht zum Bearbeiten des Objekts Safety Task besitzt.

Zur Dokumentierung kann zu jedem Programmeintrag in der Liste ein Kommentar eingegeben werden.

5.5.4.5 Globale Variablenliste (GVL)

Die Safety Globale Variablenliste (GVL) dient der Darstellung, Deklaration und Bearbeitung global deklarierter Variablen, die innerhalb der ganzen Sicherheitsapplikation gültig sind. Es können einer Sicherheitsapplikation mehrere GVLs hinzugefügt werden.

Im Projektbaum wird eine Safety GVL mit dem Symbol dargestellt.

Die in der GVL der Sicherheitsapplikation deklarierten Variablen sind nicht projektweit, sondern nur innerhalb der Sicherheitapplikation global gültig.

Hinzufügen einer GVL

1. Im Projektbaum das Safety Applikationsobjekt *„SafetyApp“* selektieren
2. Kontextmenü-Befehl *„Objekt hinzufügen“* mit Auswahl *„Safety Globale Variablenliste“* selektieren
3. Im Dialogfenster *„Safety Globale Variablenliste hinzufügen“* den Namen und optional Kommentar für die GVL eingeben.
4. Schaltfläche *„Hinzufügen“* aktivieren

Es öffnet sich der Editor zur Deklaration der globalen Variablen.

Objekteigenschaften GVL

Der Eigenschaften-Dialog einer GVL enthält die Registerkarten *„Allgemein“*, *„Safety“* und *„Zugriffskontrolle“*

Die Registerkarte *„Safety“* des Eigenschaften-Dialogs (siehe Abb. 44) enthält die *„Prüfsumme“* (Prüfsumme der GVL) und editierbare Felder für die Version und den Kommentar der GVL.

Die Version kann der Entwickler frei vergeben. Mithilfe der Version kann in der Objektliste des Safety Applikationsobjekts schnell erkannt werden, um welche Version des Objekts es sich handelt.

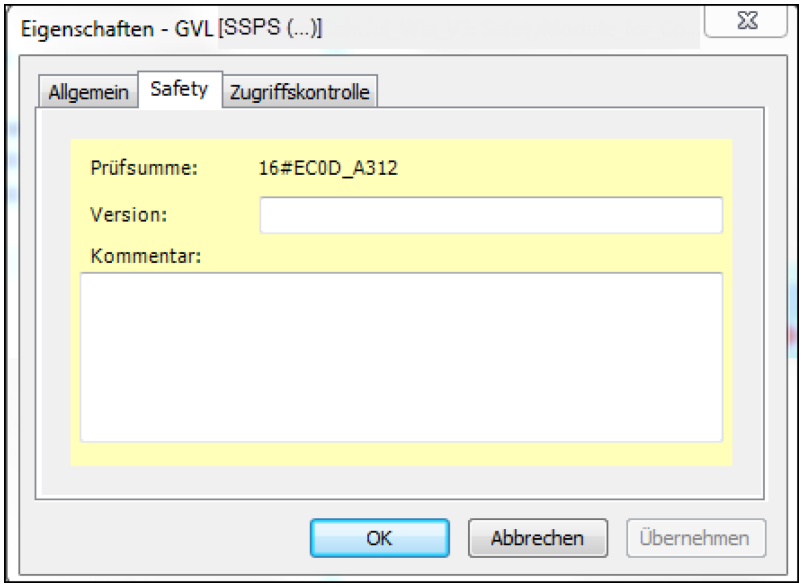


Abb. 44: Dialog Eigenschaften der GVL, Registerkare 'Safety'

Editor einer GVL

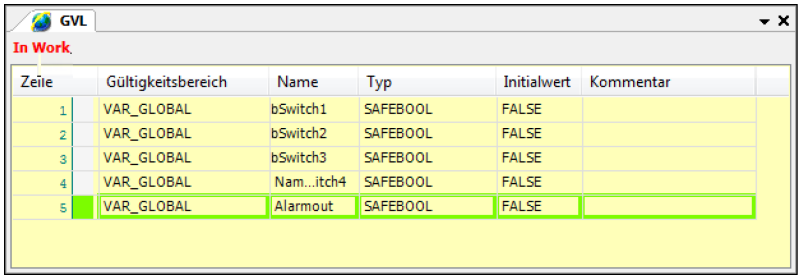


Abb. 45: GVL-Editor

Der Editor besteht aus zwei Bereichen:

- Anzeige der Pin-Information (falls nicht gepinnt: In Work)
- Tabelle zur Variablendeklaration mit den Spalten: Zeile, Bereich, Name, Typ, Initialwert, Kommentar

Deklaration einer globalen Variablen siehe ↳ *Kapitel 5.6 „Variablendeklaration“ auf Seite 97*

Gültigkeitsbereiche für globale Variablen

- VAR_GLOBAL
- VAR_GLOBAL CONSTANT

Datentypen für globale Variablen

- BOOL
- DINT
- INT
- SAFEBOOL
- SAFEDINT
- SAFEINT
- SAFETIME
- SAFEWORD
- TIME
- WORD

In einer GVL deklarierte Variablen stehen in der Eingabehilfe der Sicherheitsapplikation unter der Kategorie "Globale Variablen" zur Verfügung.

Globale Variable ändern

1. GVL des Safety Applikationsobjekts im Projektbaum selektieren
2. Kontextmenübefehl *„Objekt bearbeiten ...“* aktivieren
3. Im Editor (siehe Abb. 45) die zu ändernde Zelle mit Doppelklick öffnen
4. Inhalt der Zelle ändern

5.5.4.6 Netzwerkvariablen - Kommunikation zwischen Sicherheitssteuerungen

Eigenschaften von Safety NetVars

- Die Querkommunikation zwischen Sicherheitssteuerungen dient dem Austausch von sicherheitsbezogenen Signalen.
- Es können Variablen folgender Datentypen ausgetauscht werden: `SAFEBOOL`, `SAFEWORD` und `SAFEINT`.
- Wenn die Querkommunikation mittels der Objekte *„Safety Netzwerkvariablenliste (Sender) “* und *„Safety Netzwerkvariablenliste (Empfänger)“* konfiguriert wurde und ein Download auf Sicherheits- und Standardsteuerungen durchgeführt wurde, wird die Kommunikationsverbindung über die Standardsteuerungen der Sicherheitssteuerung automatisch hergestellt.
- Die Sicherheitssteuerung tauscht immer den Variablenwert aus, den die Variable am Ende des Applikationszyklus hat. Alle ausgetauschten Telegramme (Variablenwerte und Empfangsbestätigungen) werden synchron zum Applikationszyklus in der Ausgangsphase abgeschickt und in der Eingangsphase empfangen werden.

- Ein Sender kann die gleiche Variable an mehrere Empfänger senden und ist dabei programmatisch und funktional unabhängig von seinen Empfängern. Die Empfänger müssen sich zum Aufbau der abgesicherten Kommunikation beim Sender anmelden.
- Ein konfigurierter Variablenaustausch läuft automatisch an, automatisch weiter und nach Wegfall eines Unterbrechungsgrunds automatisch wieder an wenn folgende Bedingungen erfüllt sind:
  – Das Routing der Standardsteuerungen von Sender und Receiver(n) läuft.
  – Die Sicherheitsapplikationen von Sender und Empfänger laufen.
  – Die Kommunikationsstrecke ist schnell genug, die Zykluszeiten sind klein genug und die Watchdog-Zeit ist groß genug.
- Für die Querkommunikation werden die Bausteine *„NetVarReceiver“* und *„NetVarSender“* der Bibliothek *„SafetyNetVar“* verwendet, zusätzlich wird für jede Sender-Empfänger Beziehung eine Instanz des Bausteins *„NetVarSenderStack“* erzeugt.

Vorgehenweise beim Einrichten von sicherer Querkommunikation

1. Fügen Sie das Objekt *„Safety-NVL (Sender) “* in der Sicherheitsapplikation der Sender-Sicherheitssteuerung ein.
2. Geben Sie im Editor des Objekts *„Safety-NVL (Sender) “* in das Eingabefeld *„Safety-Adresse dieser Variablenliste“* die FSoE-Adresse ein.
3. Fügen Sie das Objekt *„Safety-NVL (Empfänger) “* in der Sicherheitsapplikation der Empfänger-Sicherheitssteuerung ein.
4. Wählen im Editor des Objekts *„Safety-NVL (Empfänger) “* in der Auswahlliste *„Zugeordneter Sender“* den Sender aus.
5. Geben Sie in die Eingabefelder *„Connection ID“* und *„Watchdog-Zeit“* die Werte ein.

Für weitere Information wird auf die Online-Hilfe verwiesen.

5.5.4.7 Bibliotheksverwalter

Im Bibliotheksverwalter werden die von der Sicherheitsapplikation verwendbaren Bibliotheken verwaltet. Die Liste der Bibliotheken stammt aus der Gerätebeschreibung der Sicherheitssteuerung, welche die Sicherheitsapplikation mit dem Bibliotheksverwalter enthält. Alle Bibliotheken, die für diese Sicherheitssteuerung zur Verfügung stehen, werden automatisch eingefügt.

Bibliotheksverwalter von CODESYS Safety entspricht dem Bibliotheksverwalter von CODESYS. Details siehe Online-Hilfe von CODESYS

Genauere Beschreibung der Sicherheitsbibliotheken finden Sie in der CODESYS Safety Onlinehilfe.

Die Versionsliste der Bibliotheksbausteine und die Sicherheitshinweise die für die Bibliotheksbausteine beachtet werden müssen finden Sie in *Kapitel 15 „Vordefinierte Bausteine“ auf Seite 293*.

Hinzufügen des Bibliotheksverwalters

Das Objekt *„Bibliotheksverwalter“* wird in der Regel automatisch mit der Sicherheitssteuerung in den Projektbaum eingefügt. Diese Objekt kann unter einem Safety Applikationsobjekt nur einmal vorhanden sein.

Manuell kann er durch Selektion des Safety Applikationsobjekts *„SafetyApp“* im Projektbaum und Ausführen des Kontextmenü-Befehls *„Objekt hinzufügen“* - *„Bibliotheksverwalter“* hinzugefügt werden.

Objekteigenschaften

Der Eigenschaften-Dialog enthält, wie in CODESYS Standard, die Registerkarten *„Allgemein“*, *„Zugriffskontrolle“* und *„Übersetzen“*.

Editor des Bibliotheksverwalters

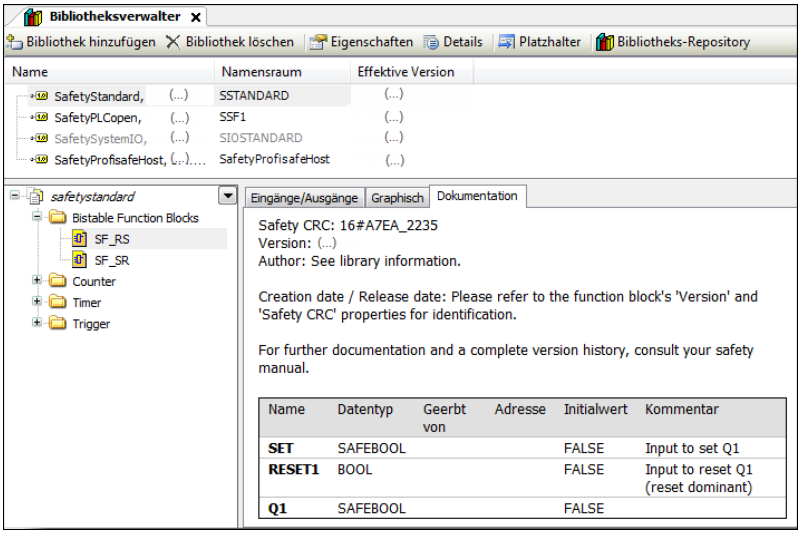


Abb. 46: Beispiel: Editor des Bibliotheksverwalters

Der Editor besteht aus folgenden drei Bereichen:

- Bibliotheksliste
- Bausteinliste (einer selektierten Bibliothek, am Beispiel *„safetystandard“*)
- Bausteinbeschreibung (eines in der Bausteinliste selektierten Bausteins, am Beispiel *„SF_SR“*) mit den Registerkarten: *„Eingänge/Ausgänge“*, *„Graphisch“* und *„Dokumentation“*

Weitere Details zu diesem Editor siehe Online-Hilfe von CODESYS Standard.

Hinweise zum Bibliotheksverwalter

HINWEIS!
Der Bibliotheksverwalter ist für den Nachweis der in der Sicherheitsapplikation verwendeten Bibliotheksbausteine (IEC-Bausteine und externe Bausteine) bei der Verifikation und bei der Abnahme nicht geeignet. Der Nachweis der ausführungsrelevanten Stände muss über die Vergleichsansicht erfolgen, siehe ↳ *Kapitel 8 „Pinnen der Software“ auf Seite 189*. Abnahmedokumentation siehe ↳ *Kapitel 10.1 „Einleitung “ auf Seite 217*.

Die Liste der Bibliotheken kann sich durch ein Aktualisieren der Sicherheitsteuerung ändern.

5.6 Variablendeklaration

Variable deklarieren

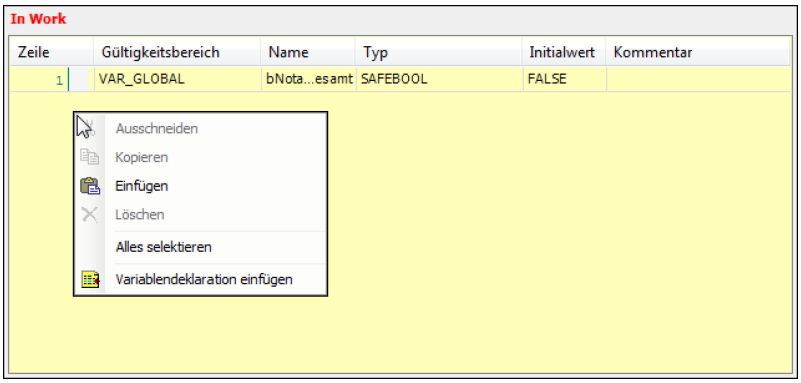


Abb. 47: Editor zur Variablendeklaration mit geöffnetem Kontextmenü

1. Im Variablendeklarationseditor das Kontextmenü aktivieren
2. Im Menüfenster den Befehl *„Variablendeklaration einfügen“* aktivieren
3. Im Dialog *„Safety Variable deklarieren“* die Felder Sichtbarkeit, Name, Typ, Initialwert und optional Kommentar editieren bzw. auswählen
4. Schaltfläche *„OK“* aktivieren.

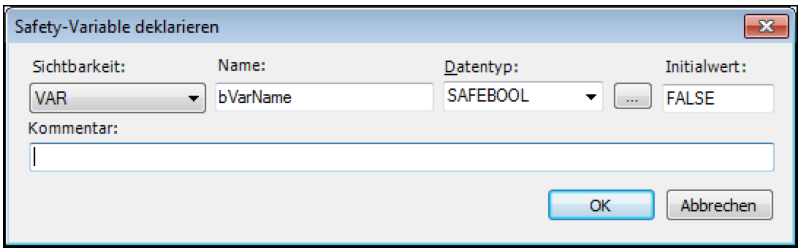


Abb. 48: Dialog 'Safety-Variable deklarieren'

Eingabefelder des Dialog "Safety-Variable deklarieren"

- Sichtbarkeit
- Name
- Typ
- Initialwert
- Kommentar

Bestehende Deklarationen können durch einen Doppelklick auf das entsprechende Feld im Deklarationstabelle geändert werden.

Benutzerdefinierte Datentypen

Die Datentyp-Kategorie *„Benutzerdefinierte Typen“* enthält Funktionsbausteine der Sicherheitsapplikation und die Bibliotheken. Diese Kategorie kann in der Eingabehilfe, die im Dialog Abb. 48 durch Aktivierung der Schaltfläche ... geöffnet wird, ausgewählt werden. Bei bereits deklarierten Variablen wird die Eingabehilfe des Datentyps wie folgt geöffnet:

1. Im Deklarationsfenster die Typ-Zelle der entsprechenden Variable auswählen
2. In der Auswahlliste des Typs das Symbol „...“ anklicken

6 Programmierung

6.1 Überblick Programmierung

CODESYS Safety unterstützt den Entwickler bei der Erstellung einer normenkonformen Sicherheitsapplikation durch

- Förderung guter Programmiertechniken
- Verbot von unsicheren Sprachmerkmalen
- Förderung der Code-Verständlichkeit
- erleichterte Testbarkeit
- Verfahren zur Codedokumentierung

Die Programmierung einer Sicherheitsapplikation erfolgt in POUs, GVLs und dem Task-Objekt.

In den POUs wird der Programmcode in der IEC 61131-3 Sprache FUP (Funktionsplan) implementiert. FUP zeichnet sich durch Eindeutigkeit, leichtes Erkennen von Programmierfehlern und klaren Datenfluss aus.

Die Bedienoberfläche und Handhabung des CODESYS Safety FUP-Editors entspricht CODESYS Standard.

Die für CODESYS Safety FUP spezifischen Befehle werden bei den jeweiligen Sprachelementen beschrieben.

Häufig verwendete Symbolik:

- Eingabehilfe: das Symbol [...], auch Ellipses genannt, kennzeichnet eine Schaltfläche, bei deren Aktivierung sich das Dialogfenster „Eingabehilfe“ öffnet.

Sprachumfang der Sicherheitsprogrammierung mit CODESYS Safety

Der Sprachumfang von FUP wird entsprechend der in PLCopen definierten Sprachteilmengen Basic und Extended eingeschränkt. Die entsprechende Auswahl für den Sprachumfang Basic oder Extended wird beim Neuanlegen einer POU (Programm oder Funktionsbaustein) durch den Entwickler festgelegt (siehe *„Hinzufügen einer POU“ auf Seite 87*).

Im Basic Level kann durch Verknüpfung der bereits zertifizierten Bausteine der PLCopen-Bibliothek (*„SafetyPLCopen“*) und der Standard-Bibliothek (*„SafetyStandard“*) mit relativ geringem Aufwand eine Sicherheitsapplikation implementiert und anschließend verifiziert werden.

Extended Level bietet dem Entwickler zusätzliche Operatoren (boolsche, mathematische, und andere) und bedingte Sprünge/ Returns, um umfangreichere Sicherheitsapplikationen zu erstellen. Sie erfordern im Anschluss an die Entwicklung einen entsprechend aufwendigeren Verifikationsprozess.

Bei der Verwendung von PLCopen Bausteinen läuft die Anlage nach einem Fehlerzustand (Kommunikationsfehler) erst nach Betätigen von Reset wieder an. Werden die PLCopen Bausteine nicht verwendet, so muss dieses Verhalten durch die Applikation implementiert werden.

HINWEIS!

Damit die CODESYS Online-Funktionen und Eingabehilfen für die Sicherheitsapplikation funktionieren, müssen sie neben dem Safety-Sprachumfang auch der Standard-Compilerversion genügen. Wenn im Projekt eine neuere Compilerversion verwendet wird, können sich daraus zusätzliche Einschränkungen der Sicherheitsapplikation ergeben. Zum Beispiel kann es neue Schlüsselwörter geben, die dann nicht mehr als Bezeichner verwendet werden können.

Eine Verletzung solcher zusätzlicher Einschränkungen aus der Compilerversion erkennen Sie nicht beim Befehl *„Erstellen ➔ Übersetzen“*, sondern erst beim Einloggen. Es erscheint dann eine entsprechende Meldung und ein Einloggen ist nicht möglich. Einstellung der Compilerversion siehe *Projektumgebung* in der Standard Online-Hilfe

Abweichungen es Sprachumfangs zu

- PLCopen siehe *Kapitel 6.1.2 „Abweichungen der Sprachelemente von PLCopen Safety“ auf Seite 101*
- Standard-CODESYS siehe *Kapitel 6.1.3 „Unterschiede zur Programmierung in Standard CODESYS“ auf Seite 102*
- IEC 61131-3 siehe *Anhang B.2 „Compliance Liste: Implementierungsspezifische Erweiterungen“ auf Seite 362*

6.1.1 Sprachelemente

Zur Implementierung des Programmcodes einer Sicherheitsapplikation dienen generell folgende Objekte und deren Elemente:

- GVLs
  - Globale Variablen
- NVLs
  - Netzwerkvariablen

- POUs
  - Netzwerke
  - Variablen
  - Operatoren
  - Zuweisungen
  - Sprünge/Returns
  - FB-Aufrufe
- Task
  - Programmliste
- Logische E/As
  - Mapping-Variablen
- Bausteine der Sicherheitsbibliotheken, die im Bibliotheksverwalter angezogen werden,

6.1.2 Abweichungen der Sprachelemente von PLCopen Safety

PLCopen [N2.1.1]	Abweichung in Basic/Extended POUs
Standard-FBs erlaubt	nur als Safety-Varianten vorhanden (SF_TON statt TON, etc)
Datentyp REAL vorgesehen	Datentyp REAL wird nicht unterstützt
(SAFE)TIME in Extended Level: nur als interne Variable oder konstanter Input für FBs	In Extended Level: keine Einschränkung bei Deklarationen, auch möglich als Datentyp für physikalische Ein- und Ausgänge
(SAFE)WORD in Extendend Level: nur als interne Variable oder als Output für Diagnosezwecke	In Extended Level und GVLs: keine Einschränkung bei Deklarationen, auch möglich als Datentyp für physikalische Ein- Und Ausgänge
kein (SAFE)BYTE, (SAFE)DWORD	In Extended Level und GVLs: keine Einschränkung bei Deklarationen, auch möglich als Datentyp für physikalische Ein- und Ausgänge
kein VAR_EXTERNAL in Basic Level	in Programmen: VAR_EXTERNAL erlaubt für Kanalvariablen und Stack-Instanzen VAR_EXTERNAL CONSTANT erlaubt für globale Konstanten
mindestens SAFEBOOL	SAFE-Varianten für alle unterstützten Datentypen
LD	nicht unterstützt
kein mehrfacher Aufruf der selben FB-Instanz	nur im Fall von FBs mit "Einmaliger-Aufruf"-Qualifizierung (alle PLCopen-FBs fallen darunter); Implizite "Einmaliger-Aufruf" "Vererbung": FB mit "Einmaliger-Aufruf"-Instanz ist selbst "Einmaliger-Aufruf" Instanzen anderer FBs können mehrfach oder null mal aufgerufen werden.

PLCopen [N2.1.1]	Abweichung in Basic/Extended POUs
keine FB-Declaration Features nach [N1.1.3-Tab.40] - kein VAR_EXTERNAL von globalen FBs	in Basic-Level Programmen: erlaubt für Stack-Instanzen in Extended Level Programmen: erlaubt für nicht-PLCopen FBs (genauer: für FBs ohne Callonce-Qualifizierung)
keine FB-Declaration Features nach [N1.1.3-Tab.40] - kein VAR_EXTERNAL CONSTANT in einer FB-POU	erlaubt für globale Konstanten (VAR_GLOBAL CONSTANT)

6.1.3 Unterschiede zur Programmierung in Standard CODESYS

Neben der Reduzierung des Sprachumfangs gegenüber der IEC 61131-3 weicht die Programmierung in CODESYS Safety an einigen Stellen von Standard CODESYS ab, weil sie strikter ist:

Einschränkungen

1. IEC-konform erfordert die lokale Verwendung globaler Variablen (explizit und implizit, konstant und nicht) in einer POU immer eine entsprechende External Deklaration
2. IEC-konform sind interne Variable (VAR) von außen nicht sichtbar (statt nur, wie in Standard CODESYS, nicht zuweisbar)
3. IEC-konform können Input-Variablen nur im Aufruf gesetzt werden
4. IEC-konform wird bei Aufrufen der Operatoren SEL und MUX niemals die Auswertung eines Parameters übersprungen
5. IEC-konform ist es ein Fehler, wenn es in einem MUX-Aufruf zum ersten Inputwert keinen entsprechenden Input zum Wählen gibt. CODESYS Safety detektiert diesen Fehler zur Laufzeit und löst eine entsprechende Fehlerreaktion aus
6. IEC-konform haben die Counter-Parameter PV und CV der Safety-Varianten SF_CTU, SF_CTD, SF_CTUD der Standard Funktionsbausteine CTU, CTD und CTUD den Typ INT statt UINT. Ihr höchster Zählerstand ist daher 0x7fff.
7. Zahlen sind (SAFE)INT/DINT-Werte, keine BOOL/BYTE/WORD-Werte. Die Erzeugung eines (SAFE)BYTE/WORD-Wertes erfordert die Qualifzierung der Zahl mit dem Typ, z.B. word#0 oder word#16#8000, die Alternative zu TRUE und FALSE ist nicht 1 und 0 sondern bool#1 und bool#0
8. Ein Nummeral (literale Zahl-Konstante) größer als 215-1, z.B., 16#8000, ist ein (SAFE)DINT-Wert, kein (SAFE)INT-Wert, unabhängig von der Zahlenbasis. Das heißt, 16#FFFF ist nicht gleich dem INT-Wert -1 sondern ein positiver DINT-Wert 216-1.
9. Es gibt keine impliziten Typkonvertierungen, außer INT zu DINT (INT-Polymorphie) und SAFE-Typ zu Typ (SAFE-Polymorphie)

10. Fehler zur Laufzeit bei verschiedenen Konvertierungsfunktionen, wenn Ausgangswert nicht im Wertebereich des Zieltyps ist

6.2 Programmierrichtlinien

6.2.1 Empfehlungen zur Dokumentierung des Codes

Regel P1(Doku)

HINWEIS!

Sie sollten sich im Voraus überlegen, was Sie in Ihrer Applikation durch zusätzliche Kommentare dokumentieren möchten, und das als Richtlinien zur Dokumentation festhalten.

Unabhängig von der zur Zertifizierung zugrunde gelegten Norm und unabhängig von Besonderheiten Ihrer Arbeitsabläufe und Ihres Projekts können folgende allgemeine Richtlinien empfohlen werden.

- Die Dokumentation muss vollständig, verfügbar, lesbar und verständlich sein.
- Alle Änderungen über alle Lebenszyklen müssen dokumentiert werden.
- Die Projektdokumentation muss enthalten
  - juristische Person (Firma)

    Diese Informationen können Sie unter den Objekteigenschaften des Applikationsobjekts, Registerkarte *„Safety“* im Feld *„Kommentar“* eingeben.
  - Funktionsbeschreibung der Anforderungen an das Projekt
  - E/A-Beschreibung
  - Version der verwendeten Funktionsbausteinbibliothek

    Diese Information wird beim Ausdruck des Applikationsobjekts automatisch generiert (siehe *Kapitel 10.3.2 „Ausdruck Projektdokumentation“ auf Seite 225*)
- Jede POU/FB muss folgende verfügbare Informationen haben:
  - Autor
  - Datum der Erstellung
  - Datum der Freigabe
  - Version
  - Versionshistorie

  Diese Informationen können Sie unter den Objekteigenschaften des POU-Objekts, Registerkarte *„Safety“* in den Feldern *„Objektversion“* und *„Kommentar“* eingeben.
- Ausreichende Kommentierung der Netzwerke

  Zur Kommentierung von Netzwerken wird ein Netzwerktitel und ein Netzwerk unterstützt.

Die Anzeige ist optional und kann über „Tools ➔ Optionen ➔ Sichere FUP Optionen“ ein und ausgeschalten werden (siehe Onlinehilfe). Es wird empfohlen, sie immer anzeigen zu lassen.

- Ausreichende Kommentierung der Deklarationszeilen

Kommentierung

CODESYS Safety bietet folgende Möglichkeiten um die Sicherheitsapplikation, ihre Objekte und deren Inhalte zu kommentieren:

- Zu jedem Quellobjekt der Sicherheitsapplikation: Kommentarfeld und Versionsangabe (Applikation, POU, GVL, NVL, logisches E/A, Task, nicht: Bibliotheksverwalter)
  Die Felder für Kommentar und Version befinden sich in den Objekteigenschaften, Registerkarte *„Safety“* (siehe Onlinehilfe)
- Kommentarfeld zur gesamten Sicherheitsapplikation (Dialog *„Eigenschaften“* von *„Safety-App“*)
  Hier kann die Quellcode-Information eingegeben werden: Firma, Autor, Beschreibung, Eingänge und Ausgänge und Konfigurationsmanagement-Geschichte.
- Zu jedem FUP-Netzwerk: Netzwerktitel und Netzwerkkommentar
  Die Anzeige ist optional (siehe Information oben). Es wird empfohlen sie immer anzeigen zu lassen.
- Zu jeder Deklaration: ein Kommentar (POU, GVL, NVL, Variablenmapping logischer E/As)
- Zu jedem Programmeintrag in der Task: ein Kommentar

Der Checker prüft optional, ob bei der Applikation und bei den POUs Kommentare vorhanden sind. Ob diese oder die anderen Kommentare den Programmierrichtlinen genügen, muss manuell verifiziert werden.

Einstellungen zur Anzeige von Warnungen bei fehlenden Kommentar der Sicherheitsapplikation oder einer POU werden im „Eigenschaften“-Dialog der Sicherheitsapplikation vorgenommen.

6.2.2 Regeln für Bezeichner von Safety Objekten und Variablen

Regel P2 (Namen)

HINWEIS!
Die Regeln für die Bezeichner von Safety Objekten und Variablen sollten eingehalten werden.

- Namen für POUs, GVLs, E/A-Module, Variablen, Netzwerke (Labels), etc. sind Bezeichner (Identifier) im Sinne der IEC 61131-3. Damit müssen sie folgende Form haben: Sie sind eine Folge aus Buchstaben, Ziffern und Unterstrich, wobei das erste Zeichen keine Ziffer und das letzte Zeichen kein Unterstrich sein darf, und zwei Unterstriche nicht direkt aufeinander folgen dürfen.
- Bezeichner, die mit Unterstrich beginnen, sind reserviert und dürfen in Basic/Extended POUs und GVLs nicht vorkommen.
- Die in der IEC 61131-3 (2014-06, Programmable controllers, Part 3: Programming languages - Annex C) aufgeführten IEC-Schlüsselwörter dürfen nicht als Bezeichner verwendet werden.
- Die über über die IEC hinausgehenden Schlüsselwörter, die von CODESYS Standard reserviert sind, dürfen nicht als Bezeichner verwendet werden.
- Als CODESYS Safety-spezifische Schlüsselwörter sind reserviert und dürfen nicht als Bezeichner verwendet werden:
  - IOAPI, IOIN, IOOUT, SYSONLY
  - SAFEXXX für alle elementaren Standard Datentypnamen XXX
  - SF_FB für alle Standard Funktionsbausteintypen FB. FBs mit solchen Namen sind nur für externe FBs erlaubt.
- Die Namen von Standard-Funktionen, die im Extended Level der PLCopen verboten sind, stehen nicht als Bezeichner zur Verfügung.
- Variablennamen müssen innerhalb des Gültigkeitsbereichs (Scope) eindeutig sein. Dies bedeutet im Einzelnen:
  - Keine zwei globalen Variablen dürfen den gleichen Namen haben.
  - Keine zwei POU-weiten Variablen in der gleichen POU dürfen den gleichen Namen haben.
  - Es darf keine POU-weite Variable den gleichen Namen haben wie eine globale Variable. Dies ist unabhängig davon, ob die globale Variable mit VAR_EXTERNAL in den Gültigkeitsbereich der POU importiert wird oder nicht.
  - Eine Variable darf nicht gleich heißen wie ein Objekt oder Bibliotheksbaustein der Applikation.
- Namen von Sprungmarken müssen eindeutig sein
  - eine Sprungmarke darf nicht den gleichen Namen haben wie eine POU-weite oder globale Variable
- Das Präfix SF_ ist reserviert für PLCopen konforme FBs und die Safety-Varianten der Standardfunktionsbausteine der IEC

- Unter den Objekten: POUs, GVLs, E/A-Module, Bibliotheksbausteine, die zur Sicherheitsapplikation gehören, dürfen keine zwei den gleichen Namen haben.
- Kein Objekt (POU, GVL, E/A-Modul, Bibliotheksbaustein), das zur Sicherheitsapplikation gehört, darf den gleichen Namen wie eine globale Variable der Applikation haben – mit einer Ausnahme: Implizite globale Variablen eines E/A-Moduls dürfen genau so heißen wie das E/A-Modul selbst (aber nicht wie ein anderes E/A-Modul oder Objekt).

6.2.3 Defensives Programmieren

Verschiedene Prüfungen von Signalen sind gefordert. Prüfungen für Feldgeräte und für Querkommunikation sind in *Kapitel 6.5 „Einbindung von Feldgeräten“ auf Seite 146* und *Kapitel 6.6 „Querkommunikation mit Netzwerkvariablen“ auf Seite 150* aufgeführt.

Regel P3 (Check:Plausibel)

HINWEIS!
Plausibilitätsprüfungen zur defensiven Programmierung sollten programmiert werden (empfohlen in ISO 13649). Zum Beispiel sollte geprüft werden, ob Kombinationen von Eingangssignalen, von Signalen und Zuständen/Zeiten eine physikalisch unmögliche Situation darstellen.

Es muss keine Überwachung von Datenintegrität (gefordert von ISO 13849 und IEC 62061) und Kontrol- und Datenfluss (gefordert von IEC 62061) programmiert werden. Dies erfolgt durch das System.

Basic Level
Im Basic Level muss keine Grenzwertprüfung (Gefordert von IEC 62061) programmiert werden. Diese erfolgt in den vordefinierten Funktionsbausteinen.

Regel P4 (Check:Num)

HINWEIS!
Extended-Level: Für POUs, die numerische Werte verarbeiten gilt: "Vernünftige" Grenzwertprüfungen von Eingangssignalen und Eingangsvariablen (VAR_INPUT) müssen programmiert werden (gefordert von IEC 62061).

6.2.4 Gestaltungsregeln für PLCopen-konforme Funktionsbausteine

Diese Programmierregeln entsprechen den "General Rules for Safety-Related Function Blocks" der PLCopen. Sie gelten für selbst entwickelte PLCopen-konforme Funktionsbausteine und für die vordefinierten Funktionsbausteine der Bibliothek SafetyPLCopen (siehe *Kapitel 15.1.2 „Applikative Bibliotheken“ auf Seite 293*).

Funktionsbausteinspezifische Regeln

PLCopen Funktionsbausteine können nur in Programmen und in Funktionsbausteinen, bei denen „Einmaliger Aufruf“ gesetzt ist, verwendet werden.

Default-Signal	Alle sicherheitsgerichteten Boolschen E/A Signale haben den voreingestellten sicheren Wert "FALSE"
Signallevel	Der Wert SAFEBOOL ist nur wie folgt anwendbar: = 0 entspricht der Sicherheit, wie bei Systemausgängen definiert =1 bedeutet, dass die Sicherheitsaspekte des Systems korrekt arbeiten, z.B., normaler Betrieb ist möglich. Dies reflektiert die Funktionalität der IEC 61131 Umgebungen, wie beispielsweise die Regeln für den Default-Wert und dass alle Ausgänge bei einem Fehlerfall auf "0" gesetzt werden.
Ausgänge	Jeder Ausgang muss in jedem Zyklus zugewiesen werden
Fehlende E/A-Parameter	Fehlende Parameter sind zulässig. Default-Werte gelten. Diese Default-Werte sollen unter keinen Umständen in einen nicht sicheren Zustand führen. Die Default-Werte einschließlich ihrer Attribute (Variable oder Constant) werden in den entsprechenden FBs spezifiziert.
Anlaufverhalten	Anfangs werden die Ausgänge auf die Default-Werte gesetzt. Nach dem ersten Aufruf des FBs, sind die Ausgänge gültig. Es gibt ein konsistentes Anlaufverhalten (Kaltstart).
Zeitdiagramme	Zeitdiagramme, wie sie bei den FBs gezeigt werden, dienen nur zur Erklärung. Sie stellen nicht das exakte Zeitverhalten dar. Das exakte Zeitverhalten hängt von der Implementierung ab.
Fehlerbehandlung und Diagnose	Alle sicherheitsgerichteten Funktionsbausteine haben zwei fehlerbezogene Ausgänge: Error und DiagCode. Sie dienen Diagnosezwecken auf Benutzeranwendungsebene, nicht zur Diagnose auf System- oder Hardewareebene. Die Vorschrift für sicherheitsgerichtete Umgebungen ist, dass die Schaltung der sicherheitsgerichteten Funktion die höchste Priorität hat und dass es in der Folgeschaltung genügend Zeit zur Diagnose gibt, entweder im funktionalen Programm oder in der Bedienschnittstelle.

Allgemeine Eingangsparameter

Name	Datentyp	Beschreibung
Activate	BOOL	Variable oder Konstante. Aktivierung eines Bausteins. Initialwert ist FALSE. Dieser Parameter kann mit der Variablen verbunden werden, die den Zustand (aktiv oder nicht aktiv) des relevanten Sicherheitsgeräts darstellt. Dies gewährleistet, dass keine irrelevanten Diagnosedaten erzeugt werden, wenn das Gerät deaktiviert wird. FALSE: alle Ausgangsvariablen werden auf den Initialwert gesetzt. Wenn kein Gerät verbunden ist, muss ein statisches TRUE-Signal zugewiesen werden.
S_‹sicherheitsgerichteter Eingangs-Name›	SAFExxxx	Jeder Name eines Eingangs eines SAFExxxx-Typs beginnt mit S_. Nur Variable dürfen zugewiesen werden.
S_StartReset	SAFEBOOL	Variable oder Konstante. Aktivierung des automatischen Anlaufs des Bausteins, wenn die S-SPS gestartet wird (warm oder kalt). FALSE (= Initialwert): automatischer Anlauf deaktiviert; manueller Anlauf über den Eingang Reset TRUE: automatischer Anlauf ↳ *„Regel FB1 (S_StartReset)“ auf Seite 115* und ↳ *„Regel FB2 (S_Start_Reset)“ auf Seite 115* sind zu beachten!

Name	Datentyp	Beschreibung
S_AutoReset	SAFEBOOL	Variable oder Konstante. Aktivierung des automatischen Wiederanlaufs des Bausteins FALSE (= Initialwert): automatischer Wiederanlauf deaktiviert; manueller Anlauf über den Eingang Reset TRUE: automatischer Wiederanlauf ↳ *„Regel FB3 (S_AutoReset)“ auf Seite 115* und ↳ *„Regel FB4 (S_AutoReset)“ auf Seite 115* sind zu beachten!
Reset	BOOL	Variable. Initialwert ist FALSE Abhängig von der Funktion, kann dieser Eingang für unterschiedliche Zwecke verwendet werden. ■ Reset der Zustandsmaschine und gekoppelter Fehler- und Zustandsmeldungen wie über DiagCode angezeigt, wenn die Fehlerursache behoben wurde. Dieses Reset-Verhalten ist als Fehler-Reset konzipiert. ■ Manueller Reset einer Wiederanlaufsperre durch den Operator. Dieser Reset ist als funktionaler Reset konzipiert. ■ Zusätzliche bausteinspezifische Reset-Funktionen. Diese Funktion ist nur bei einem Signalwechsel von FALSE nach TRUE aktiv. Ein statisches TRUE-Signal erzeugt keine weiteren Aktionen, kann aber in einigen Bausteinen als Fehler entdeckt werden. ↳ *„Regel FB5 (Reset)“ auf Seite 116* ist zu beachten! Die angemessene Bedeutung ist bei jedem Baustein beschrieben.

Allgemeine Ausgangsparameter

Name	Datentyp	Beschreibung
Ready	BOOL	TRUE: Zeigt an, dass der Baustein aktiviert und die Ausgangs-Ergebnisse gültig sind (gleich wie die "POWER"-LED eines Sicherheits-Relais). FALSE: Der Baustein ist nicht aktiv und das Programm wird nicht ausgeführt. Nützlich im Debug-Modus oder um zusätzliche Bausteine zu aktivieren/deaktivieren. Ebenso für die Weiterverarbeitung im funktionalen Programm.
S_‹sicherheitsgerichteter Ausgangs-Name›	SAFExxxx	Jeder Name eines SAFExxxx-Typ Ausgangs beginnt mit S_.

Name	Datentyp	Beschreibung
Error	BOOL	Fehler-Flag (gleich wie die "K1/K2"-LED eines Sicherheits-Relais.) TRUE: Zeigt an, dass ein Fehler aufgetreten ist und sich der Baustein im Fehlerzustand befindet. Der relevante Fehlerzustand wird am DiagCode-Ausgang angezeigt. FALSE: Es gibt keinen Fehler und der Baustein befindet sich in einem anderen Zustand. Dies wird wiederum durch den DiagCode angezeigt. Nützlich im Debug-Modus ebenso wie für die Weiterverarbeitung im funktionalen Programm.
DiagCode	WORD	Diagnoseregister. Alle Zustände eines Bausteins (Active, Not Active und Error) werden durch dieses Register abgebildet. Es wird immer nur ein konsistenter Code zur gleichen Zeit angezeigt. Im Falle von mehreren Fehlern zeigt der DiagCode-Ausgang den zuerst entdeckten Fehler an. Nützlich im Debug-Modus, ebenso wie für die Weiterverarbeitung im funktionalen Programm.

Diagnose-Codes

Ein transparentes und einheitliches Diagnosekonzept bildet die Basis für alle Bausteine. Dadurch wird gewährleistet, dass ohne Rücksicht auf die Implementierung des Anwenders, dem Endanwender in Form des DiagCode einheitliche Diagnose-Information zur Verfügung steht. Besteht kein Fehler, so wird der interne Zustand des Bausteins (Zustandsmaschine) dargestellt. Ein Fehler wird über einen binären Ausgang (Error) angezeigt. Detaillierte Informationen über interne oder externe Baustein-Fehler kann man über DiagCode erhalten. Der Baustein muss über verschiedene Reset-Eingänge zurückgesetzt werden.

Tab. 5: Allgemeine Diagnose-Code-Bereiche

DiagCode	Beschreibung
$0000\_0000\_0000\_0000_{bin}$	Der Baustein ist nicht aktiviert oder Sicherheits-CPU ist angehalten
$10xx\_xxxx\_xxxx\_xxxx_{bin}$	Zeigt an, dass sich der aktivierte Baustein im Betriebszustand ohne Fehler befindet. X = bausteinspezifischer Code
$11xx\_xxxx\_xxxx\_xxxx_{bin}$	Zeigt an, dass sich der aktivierte Baustein im Fehlerzustand befindet. X = bausteinspezifischer Code

Tab. 6: System- oder gerätespezifische Codes

DiagCode	Beschreibung
$0xxx\_xxxx\_xxxx\_xxxx_{bin}$	X = System- oder gerätespezifische Meldung. Diese Information enthält Diagnose-Information des Systems oder des Geräts. Anmerkung: 0000hex ist reserviert.

Tab. 7: Generische Diagnose-Codes

DiagCode	Beschreibung
$0000\_0000\_0000\_0000_{bin}$ 0000_{hex}	Der Baustein ist nicht aktiviert. Dieser Code stellt den Leerlauf-Zustand dar. Als allgemeines Beispiel könnte die E/A Einstellung wie folgt sein: Activate = FALSE S_In = FALSE oder TRUE Ready = FALSE Error = FALSE S_Out = FALSE
$1000\_0000\_0000\_0000_{bin}$ 8000_{hex}	Der Baustein ist aktiviert, ohne Fehler oder eine andere Bedingung, die den Sicherheits-Ausgang auf FALSE setzt. Dies ist der Standard-Betriebszustand, in dem der Sicherheitsausgang S_Out bei Normalbetrieb TRUE ist. Als allgemeines Beispiel könnten die Ein- und Ausgänge wie folgt gesetzt sein: Activate = TRUE S_In = TRUE Ready = TRUE Error = FALSE S_Out = TRUE
$1000\_0000\_0000\_0001_{bin}$ 8001_{hex}	Eine Aktivierung wurde durch den Baustein entdeckt und der Baustein ist jetzt aktiviert, aber der S_Out-Sicherheitsausgang ist auf FALSE gesetzt. Dieser Code stellt den Init-Zustand des Betriebs-Modus an. Als allgemeines Beispiel könnten die Ein- und Ausgänge wie folgt gesetzt sein: Activate = TRUE S_In = FALSE oder TRUE Ready = TRUE Error = FALSE S_Out = FALSE

DiagCode	Beschreibung
$1000\_0000\_0000\text{-}0010_{bin}$ 8002_{hex}	Der aktivierte Baustein entdeckt eine Sicherheitsanforderung, zum Beispiel S_In = FALSE. Der Sicherheitsausgang ist deaktiviert (S_Out = FALSE). Als allgemeines Beispiel könnten die Ein- und Ausgänge wie folgt gesetzt sein: Activate = TRUE S_IN = FALSE Ready = TRUE Error = FALSE S_Out = FALSE
$1000\_0000\_0000\_0011_{bin}$ 8003_{hex}	Der Sicherheitsausgang des aktivierten Baustein ist durch eine Sicherheitsanforderung deaktiviert worden. Die Sicherheitsanforderung ist jetzt zurückgenommen, aber der Sicherheitsausgang bleib solange FALSE, bis eine Reset-Bedingung entdeckt wird. Dies ist ein Betriebszustand, wo der Sicherheitsausgang S_Out = FALSE. Als allgemeines Beispiel könnten die Ein- und Ausgänge wie folgt gesetzt sein: Activate = TRUE S_In = FALSE =〉 TRUE (mit statischem TRUE fortfahren) Ready = TRUE Error = FALSE S_Out = FALSE

Generisches Zustandsdiagramm

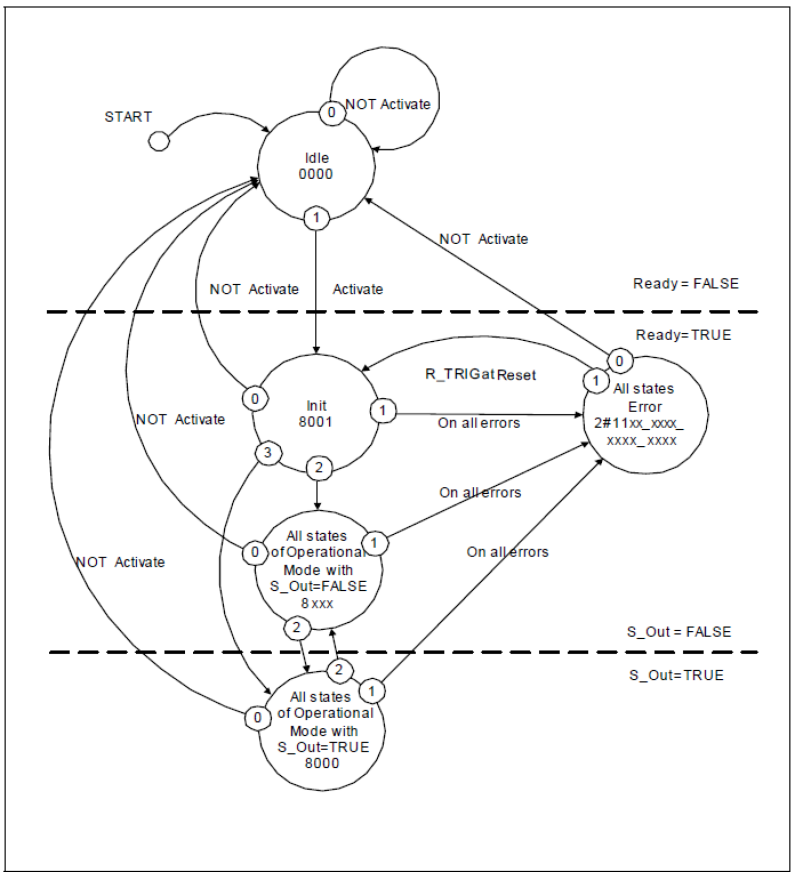


Abb. 49: Generisches Zustandsdiagramm von Sicherheits-FBs

Erklärung des generischen Zustandsdiagramms:

- Es wird ein genereller Überblick über die Zustände und Übergänge gezeigt. Einige Übergänge sind nicht benamt, dies bedeutet, dass sie FB-spezifisch sind und mit dem jeweiligen FB beschrieben werden müssen.
- Das Diagramm zeigt drei Bereiche: Im oberen Bereich ist der FB nicht aktiv und ist im sicheren Zustand (sichere Ausgänge sind FALSE), im mittleren Bereich ist der FB aktiv und im sicheren Zustand (sichere Ausgänge sind FALSE), im unteren Bereich ist der FB in normalem Zustand, d.h. die sicheren Ausgänge sind TRUE.
- Die erste horizontale Linie im Statusdiagramm zeigt den Übergang von einem nicht-aktiven FB zu einem aktiven FB.
- Die zweite horizontale Linie zeigt den Übergang von einem nicht-sicheren zu einem sicheren Zustand
- Die Prioritäten von möglichen parallelen Übergängen werden durch Nummern angegeben (höchste Priorität ist 0)
- Die Zustände enthalten den Zustandsname und den hexadezimalen Diagnosecode.
- Bedingungen "OR, AND, XOR" werden als logischen Operatoren und "NOT" als Negation verwendet.
- Innerhalb der FB-Beschreibung ist der Anfangszustand Idle, mit den Übergängen zu den einzelnen Betriebszuständen über den Zustand Init.

- Activate = FALSE schaltet von jedem Zustand direkt in de Idle-Zustand (0 = höchste Priorität ist für Activate = FALSE reserviert). Zur besseren Übersicht werden diese Übergänge nicht in jedem Statusdiagramm eingezeichnet. Dies wird als Fußnote bei jedem Statusdiagramm erwähnt.
- Aus Gründen der Übersicht, wird das Setzen der Ausgänge nicht im Zustandsdiagramm beschrieben; eine explizite Wahrheitstabelle , die die Information "FB-Zustände zu Ausgang (Ausgängen) " enthält, ist Teil jeder FB-Spezifikation mit der FB-spezifischen Fehler- und Zustands-Codes.

Tab. 8: Bausteinspezifische Fehlercodes

DiagCode	Statusname	Statusbeschreibung und Setzen des Ausgangs
Cxxx	Error	Ready = TRUE S_Out = FALSE Error = TRUE

Tab. 9: Bausteinspezifische Zustands-Codes (kein Fehler)

DiagCode	Statusname	Statusbeschreibung und Setzen des Ausgangs
0000	Idle	Ready = FALSE S_Out = FALSE Error = FALSE
8001	Init-Zustand des Betriebsmodus	Ready = TRUE S_Out = FALSE Error = FALSE
8xxx	Alle Zustände des Betriebsmodus, wo S_Out = FALSE	Ready = TRUE S_Out = FALSE Error = FALSE
8000	Alle Zustände des Betriebsmodus, wo S_Out = TRUE	Ready = TRUE S_Out = TRUE Error = FALSE

6.2.5 Regeln zur Verwendung PLCopen-konformer Funktionsbausteine

Regel FB1 (S_StartReset)

VORSICHT!

S_StartReset

Diese automatische Anlauf soll nur aktiviert werden, wenn gewährleistet ist, dass keine Gefährdung beim Start der Sicherheitssteuerung entstehen kann. Die Verwendung des Features Automatischer-Anlauf in Funktionsbausteinen erfordert deshalb die Implementierung anderer System- oder Applikationsmaßnahmen, um sicherzustellen, dass kein unerwartetes (oder unbeabsichtigtes) Starten auftritt.

Regel FB2 (S_Start_Reset)

VORSICHT!

S_StartReset

Wird der Eingang mit einer Variablen (und nicht mit FALSE) verknüpft, so müssen dafür zusätzliche Validierungsmaßnahmen definiert werden.

Regel FB3 (S_AutoReset)

VORSICHT!

S_AutoReset

Der automatische Wiederanlauf soll nur aktiviert werden, wenn sichergestellt ist, dass es zu keinem Wiederingangsetzen der Maschine nach der Freigabe des NOT-HALT kommen kann. Die Verwendung des Features Automatischer-Wiederanlauf in Funktionsbausteinen erfordert deshalb die Implementierung anderer System- oder Applikationsmaßnahmen, um sicher zu stellen, dass sich die Maschine nicht unerwartet (oder unbeabsichtigt) wieder in Gang setzt.

Regel FB4 (S_AutoReset)

VORSICHT!

S_AutoReset

Wird der Eingang mit einer Variablen (und nicht mit FALSE) verknüpft, so müssen dafür zusätzliche Validierungsmaßnahmen definiert werden.

Regel FB5 (Reset)

VORSICHT!

Reset

Der Eingang ist BOOL. Aber aus applikativen Sicherheitsanforderungen kann sich ergeben, dass er in dieser Applikation trotzdem nur mit SAFEBOOL Signalen beschaltet werden darf.

6.2.6 Automatisch geprüfte Programmierrichtlinien

Die zahlreichen, zusätzlichen, durch den Checker geprüften Programmierrichtlinien der einzelnen Sprachelemente, bei deren Missachtung Fehler, bzw. Warnungen durch den Fehlerchecker erzeugt werden, sind aus Erläuterungen im Kapitel "Safety Fehlermeldungen" in der CODESYS Safety Online-Hilfe ersichtlich.

Folgende auf der PLCopen basierenden Programmierrichtlinien werden von CODESYS Safety automatisch geprüft:

- Sprachteilmenge der Programmierlevel Basic und Extended
- Verknüpfungsregeln für SAFEBOOL Daten und analoge Verknüpfungsregeln für die anderen SAFE-xxx Datentypen

Zusätzliche automatisch geprüfte formale Programmierrichtlinien

- Deklaration von Datenvariablen nur mit explizitem Initialwert. (Ausnahme: FB-Instanzen und External-Deklarationen ohne Initialwert)
- Lesezugriff auf Ausgänge einer FB-Instanz nur in oder nach ihrem Aufruf
- Zuweisung an Ausgangsvariablen nur an einer Stelle in der Applikation
- Keine lokale Variable mit gleichem Namen wie eine globale Variable

Optional automatisch geprüfte Programmierrichtlinien

Zusätzlich können durch den Anwender Einstellungen für die Überprüfung formaler Programmierrichtlinien vorgenommen werden. Diese sind im Dialog *„Eigenschaften“* in der Registerkarte *„Safety“* des Safety Applikationsobjekts einstellbar.

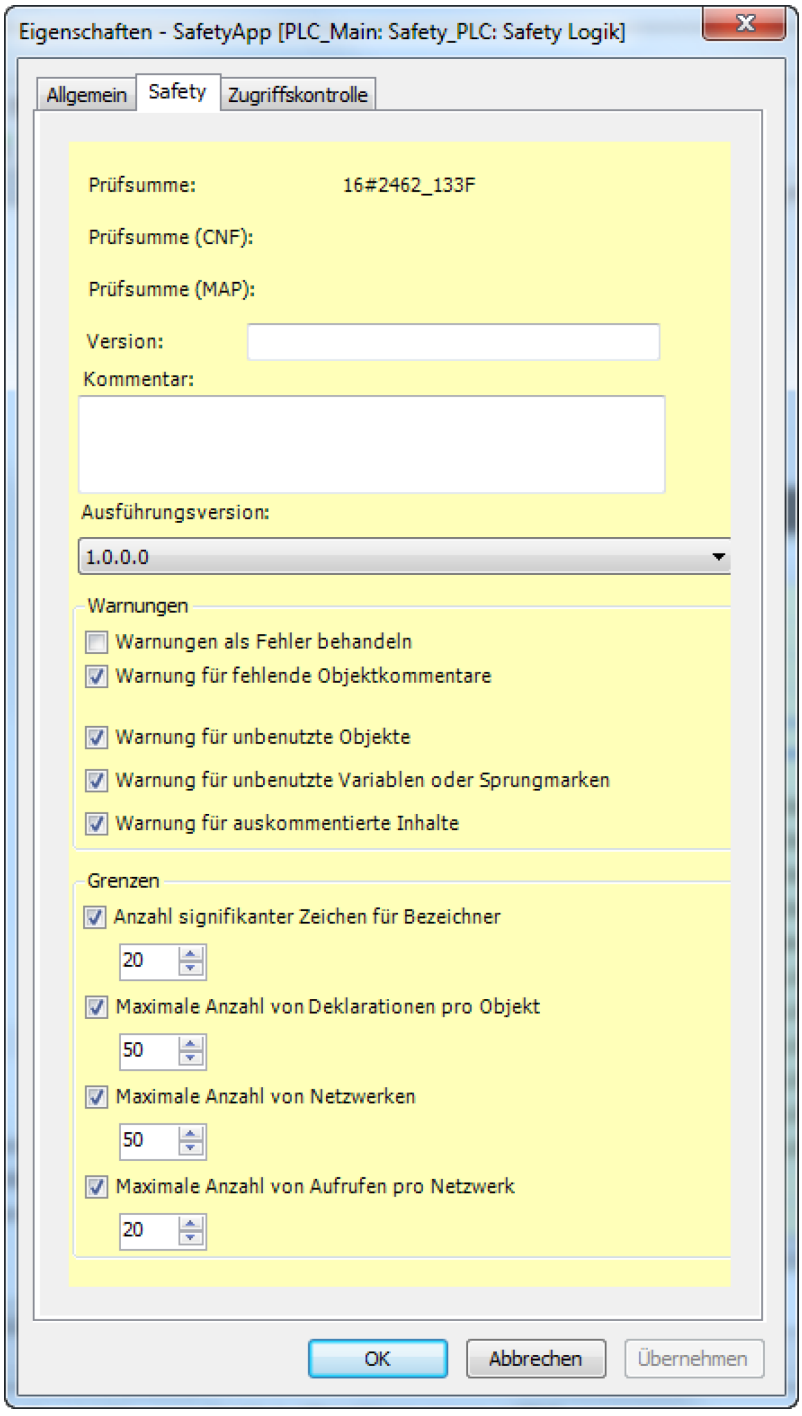


Abb. 50: Eigenschaften-Dialog des Safety Applikationsobjekts

Warnungen:

- *„Warnungen als Fehler behandeln“*
  Treten beim Übersetzen der Sicherheitsapplikation Warnungen auf, werden diese wie Fehler behandelt, was zur Folge hat, dass ein Download auf die Sicherheitssteuerung nicht möglich ist, solange sie Warnungen enthält.
- *„Warnung für fehlende Objektkommentare“*
  Objektkommentare für POUs und die Sicherheitsapplikation
  (weitere Details zur Kommentierung siehe ↳ *Kapitel 6.2.1 „Empfehlungen zur Dokumentierung des Codes“ auf Seite 103*)
- *„Warnung für unbenutzte Objekte“*
  Keine Objekte, die nicht verwendet werden. Dies bedeutet:
  – jedes Programm der Sicherheitsapplikation wird in der Safety Task aufgerufen
  – jeder Funktionsbaustein der Sicherheitsapplikation wird verwendet,
    d.h. er wird in einer GVL oder einem Programm instanziert, oder er wird in einem FB instanziert, der selbst verwendet wird
  – aus jeder GVL wird eine Variable verwendet.
  – aus jedem logischen Gerät wird eine der impliziten Variablen verwendet.
- *„Warnung für unbenutzte Variablen oder Sprungmarken“*
  Keine Variablen, die nicht verwendet werden. Dies bedeutet:
  – jede FB-Instanz hat einen Aufruf
  – jede Konstante, jeder Eingang, jede Eingangsvariable hat einen Lesezugriff
  – jede globale Variable, jede lokale Variable hat einen Schreib- und einen Lesezugriff
  – jeder Ausgang und jede Ausgangsvariable hat einen Schreibzugriff

  Im Extended Level: Keine Sprungmarken, die nicht verwendet werden. Dies bedeutet:
  – jede definierte Sprungmarke ist Ziel eines bedingten Sprungs.
- *„Warnung für auskommentierte Inhalte“*
  Keine Objektinhalte die auskommentiert sind.
- *„Anzahl signifikanter Zeichen für Bezeichner“*
  Begrenzung der Anzahl signifikanter Zeichen von Bezeichnern
- *„Maximale Anzahl von Deklarationen pro Objekt“*
  Begrenzung der Anzahl der Deklarationen in einem Objekt (POU, GVL)
- *„Maximale Anzahl von Netzwerken“*
  Begrenzung der Anzahl der Netzwerke in einer POU (Ausnahme: wiederverwendete validierte POUs)
- *„Maximale Anzahl von Aufrufen pro Netzwerk“*
  Begrenzung der Anzahl der Aufrufe in einem Netzwerk (Ausnahme: wiederverwendete validierte POUs)

6.3 Programmierung der Applikationslogik

6.3.1 GVL

Die GVL enthält Variablendeklarationen, die in Safety Extended-Programmen (POU-Typ PROGRAM) der Sicherheitsapplikation verwendet werden können. Funktionsbausteine (FBs) haben keinen Zugriff auf globale Variablen.

Um eine globale Variable in einem Programmbaustein verwenden zu können, muss sie dort nochmals als VAR_EXTERNAL deklariert werden. Die VAR_EXTERNAL Deklaration wird bei der Verwendung einer globalen Variable in einem Programm automatisch eingefügt.

Globale Variablen dienen nicht der Programmübersicht und sollten sparsam verwendet werden. Gelegentlich sind sie aber zum Austausch von Daten zwischen Programmen unerlässlich.

Informationen zur Erstellung von Variablendeklarationen siehe ↳ *Kapitel 5.6 „Variablendeklaration“ auf Seite 97*

6.3.2 POUs

Der Programmcode einer Sicherheitsapplikation wird in POUs erstellt. (Hinzufügen einer neuen POU siehe ↳ *Kapitel 5.5.4.3 „POUs“ auf Seite 86*)

Eine Sicherheitsapplikation muss mindestens eine, nicht auskommentierte POU enthalten.

Der Editor einer POU besteht aus einem Variablendeklarationsteil, in dem die Variablen deklariert werden und einem Implementierungsfenster, in dem der Programmcode erstellt wird.

Im Implementierungsfenster wird der Programmcode in aufeinanderfolgenden Netzwerken erstellt, die in aufsteigender Reihenfolge abgearbeitet werden.

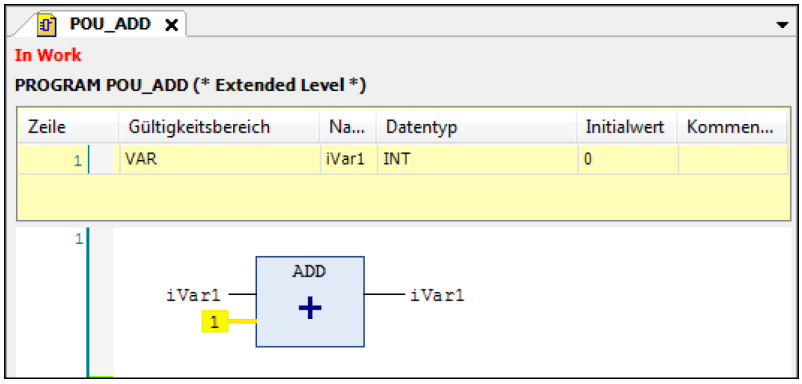


Abb. 51: Bsp. POU-Editor: Deklarationsteil mit einer Variablendeklaration und Implementierungsteil mit einem Netzwerk

Details zum POU-Editor siehe ↳ *Kapitel 6.3.3.1 „Allgemeines zu Variablen“ auf Seite 120*

Zur Aktivierung von Befehlen bei der Programmierung im FUP stehen dem Entwickler verschiedene Bedienfunktionen zur Verfügung. Bei Verfahren und konkreten Beispielen wird in diesem Handbuch meist die Bedienung über das Kontextmenü beschrieben.

Befehle, die im aktuellen Kontextmenü nicht aufgelistet oder nicht aktivierbar sind, können an der momentanen Cursor-Position auch nicht ausgeführt werden.

Standardbefehle

Es können sowohl im Implementierungteil, als auch im Variablendeklarationsteil des FUP-Editors folgende Standard-Befehle von CODESYS ausgeführt werden.

- Kopieren
- Löschen
- Ausschneiden
- Einfügen
- Rückgängig machen
- Wiederherstellen

Des Weiteren werden folgende CODESYS Befehle unterstützt:

- Gehe zur Definition
- Querverweise ausgeben
- Eingabehilfe
- Suchen & Ersetzen
- Alles selektieren
- Lesezeichen

Die Befehle verhalten sich wie in Standard CODESYS.

Für detaillierte und grundlegende Informationen wird auf die Online Hilfe von CODESYS Standard verwiesen.

6.3.3 Variablen

6.3.3.1 Allgemeines zu Variablen

Die Programmierung mit CODESYS Safety erfolgt gemäß IEC 61131-3 mit Hilfe von Variablen. Im Programmcode erfolgt der Zugriff auf Variable über Namen (symbolische Variablen), die entsprechend deklariert werden müssen. Deklarierte Variablen können im Programm-Code gelesen und zugewiesen werden. Details zu den Namen von Variablen siehe ↳ *Kapitel 6.2.2 „Regeln für Bezeichner von Safety Objekten und Variablen“ auf Seite 105*.

Die für die Safety-Programmierung benötigten Variablen werden im Deklarationseditor deklariert und bearbeitet. Dieser Editor ist der Deklarationsteil des POU-Editors. Ausgestattet mit den für die GVL (Globale Variablenliste) notwendigen Besonderheiten, wird dieser Editor auch für die Deklaration und Bearbeitung von globalen Variablen verwendet.

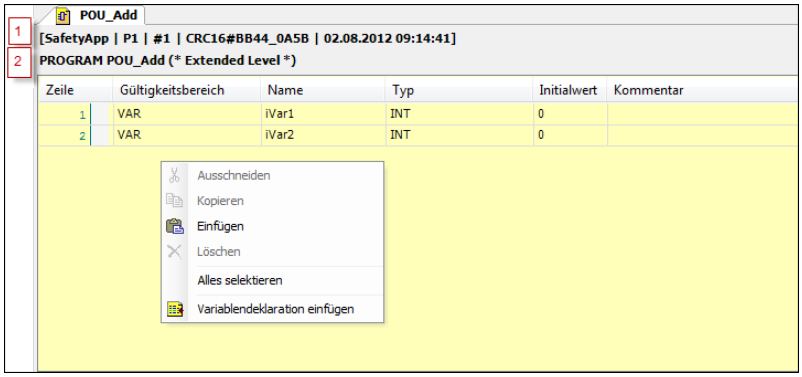


Abb. 52: Deklarationsteil des POU-Editors mit geöffnetem Kontextmenü

Aufbau des Deklarationsteils

- 1: Pin-Informationen des übergeordneten Safety Applikationsobjekts *„SafetyApp“*
- 2: POU-Typ (*„PROGRAM“*), Name der POU mit Programmierlevel (*„POU_Add (*Extended Level*)“*)

Die Verwendung von anderen, als in den folgenden Kapiteln aufgeführten Datentypen der IEC ist nicht erlaubt. Sie stehen auch nicht als freie Bezeichner zur Verfügung und ihre Namen bleiben reservierte Schlüsselwörter und können nicht benutzerdefiniert werden.

Die in den folgenden Abschnitten aufgeführten, in CODESYS Safety automatisch geprüften Programmierrichtlinien werden bei den jeweiligen Sprachelementen konkretisiert.

Änderungsmarkierungen im Deklarationsteil des Editors

Zur besseren Übersicht werden neu editierte, geänderte und gelöschte Variablendeklaration farblich markiert:

Es werden nur die Felder, welche von der zuletzt ausgeführten Editieroperation betroffen sind, hervorgehoben. Es ist immer die Markierung der zuletzt durchgeführten Aktion sichtbar. Mit dem Schließen des Objekts (POU oder GVL) werden alle Markierungen aufgehoben.

Markierungen im Deklarationsteil

- grün: neu hinzugekommene Felder
- rot: geänderte Felder

- `<del 1>` in der Spalte *„Zeile“* der gelöschten Variable
- die Zeile mit der Änderungsmarkierung ist links mit einem roten Balken gekennzeichnet.

Zeile	Gültigkeitsbereich	N...	Typ	Initialwert	Komme...
1	VAR	iVar1	SAFEINT	0	

Abb. 53: Beispiel für Änderungsmarkierung: Datentyp der Variablen `iVar1` wurde geändert

Zeile	Gültigkeitsbereich	Name	Typ	Initialwert	Komme...
1	VAR	iVar1	SAFEINT	0	
2	VAR	iVarCount	SAFEDINT	0	

Abb. 54: Beispiel für Änderungsmarkierung: Neue Variable `iVarCount` wurde eingefügt

Änderungsmarkierungen im Implementierungsteil des Editors

Nach jeder Editieroperation werden die Unterschiede zur vorherigen Version farblich markiert. Es ist immer die Markierung der zuletzt durchgeführten Aktion sichtbar. Mit dem Schließen der POU werden alle Markierung aufgehoben.

- grün: neu hinzugekommene Netzwerke oder Elemente
- rot: Änderung an einem bestehenden Netzwerk/Element
- das Netzwerk mit der Änderung wird rot markiert
- blau: Löschmarkierung für gelöschtes Netzwerk oder Element

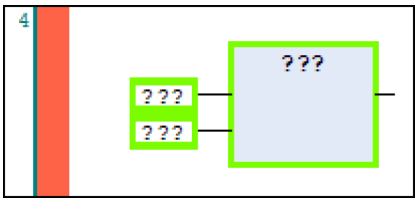


Abb. 55: Beispiel für Änderungsmarkierung: Neu eingefügter Bausteinaufruf

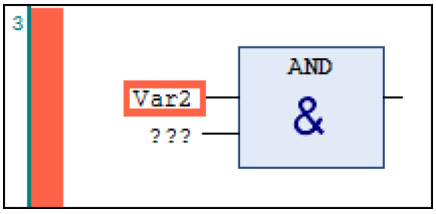


Abb. 56: Beispiel für Änderungsmarkierung: Eingang auf Var2 zugewiesen

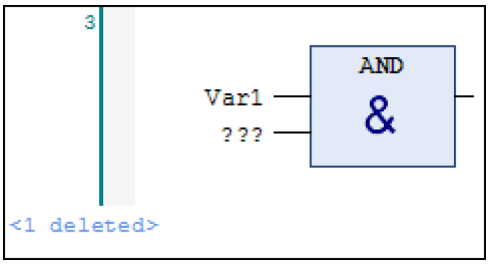


Abb. 57: Beispiel für Änderungsmarkierung: Gelöschtes Netzwerk

Arten von Deklarationen

Mögliche Deklarationen im Deklarationsteil einer POU:

- IEC-Schlüsselwort **VAR** : Kennzeichnung der Deklaration normaler interner Variablen (POU-weite Variable)
- IEC-Schlüsselwort **VAR_INPUT** kennzeichnet Deklarationen von Eingangsvariablen (POU-weite Variable)
- IEC-Schlüsselwort **VAR_OUTPUT** kennzeichnet Deklarationen von Ausgangsvariablen (POU-weite Variable)
- IEC-Schlüsselwort **VAR_EXTERNAL** kennzeichnet Deklarationen von mit VAR_GLOBAL in der Applikation (in GVL oder implizit im logischen E/A) bereits deklarierter globaler Variablen, um sie in der POU benutzbar zu machen.

HINWEIS!
In Programmen mit Programmierlevel Basic sind External-Deklarationen nach PLCopen nicht erlaubt.

Modifizierer

- CONSTANT
  - Zur Deklaration symbolischer Konstanten
  - Bei VAR_EXTERNAL CONSTANT zur Deklaration konstanter globaler Variablen
  - Kann auf VAR und VAR_GLOBAL angewandt werden, wobei die Variable keine FB_Instanz sein darf

Variablennamen

Variablennamen müssen innerhalb des Gültigkeitsbereichs (Scope) eindeutig sein. Dies bedeutet im Einzelnen:

Literalkonstanten

Es stehen folgende Literalkonstanten zur Verfügung:

- Ganzzahl-Literal mit optionaler Typannotation (INT oder DINT) und verschiedenen Zahlenbasen
- Boolesche Literale: TRUE, FALSE

- Bitstring-Literal für BYTE, WORD, DWORD und verschiedenen Zahlenbasen.
  BYTE- und DWORD-Literale stehen nur in implizitem Code und externen FBs zur Verfügung
- TIME-Literale

Das Format der Literalkonstanten muss dem in CODESYS Standard üblichen Format entsprechen.

Bei Zuweisungen von nicht-SAFE-Werten auf SAFE-Variablen oder SAFE-Inputs werden die Variable bzw. der Eingang im Editor dunkelrot markiert

6.3.3.2 Datentypen

In der Sicherheits-Programmierung mit CODESYS Safety wird zwischen sicherheitsbezogenen und nicht-sicherheitsbezogenen Daten unterschieden. Die nicht-sicherheitsbezogenen Daten sind die IEC Standard-Datentypen. Bei den sicherheitsbezogenen Daten sind die IEC Standard-Datentypen mit dem Präfix SAFE erweitert.

Die Mapping-Variablen der Eingangs- und Ausgangskanäle von sicheren Feldgeräten haben immer einen Datentyp SAFExxx, während die Mapping-Variablen von nicht-sicheren Feldgeräten immer einen nicht-SAFE Datentyp haben.

Tab. 10: IEC Standard-Datentypen

Datentyp	Bitlänge	Wertebereich	Beschreibung
BOOL	1	0,1	0 entspricht FALSE 1 entspricht TRUE
DINT	32	- 2.147.483.648 ... 2.147.483.647	
INT	16	-32.768 ... 32.767	
TIME	32	0 ... 2.147.483,647 s	Zeitdauer
WORD	16	0 ... 65.535 (16#00 ... 16#FFFF)	

Die Datentypen BYTE, DWORD, SAFEBYTE und SAFEDWORD können nur in den logischen E/As auftreten und in Extended-Level Programmen als Kanalvariablen (Kategorie: globale Variablen, Deklaration als VAR_EXTERNAL) verwendet werden (siehe Kapitel 5.5.4.2.4 „Verwendung logischer E/As im Projekt“ auf Seite 86).

Tab. 11: SAFE-Datentypen

Datentyp	Bitlänge	Wertebereich	Beschreibung
SAFEBOOL	1	0,1	0 entspricht FALSE 1 entspricht TRUE
SAFEDINT	32	- 2.147.483.648 ... 2.147.483.647	
SAFEINT	16	-32.768 ... 32.767	
SAFETIME	32	0 ... 2.147.483,647 s	
SAFEWORD	16	0 ... 65.535 (16#00 ... 16#FFFF)	

Der Datentyp REAL ist in der Safety-Programmierung nicht zulässig. Wird er dennoch verwendet, führt dies zu einem Übersetzungsfehler.

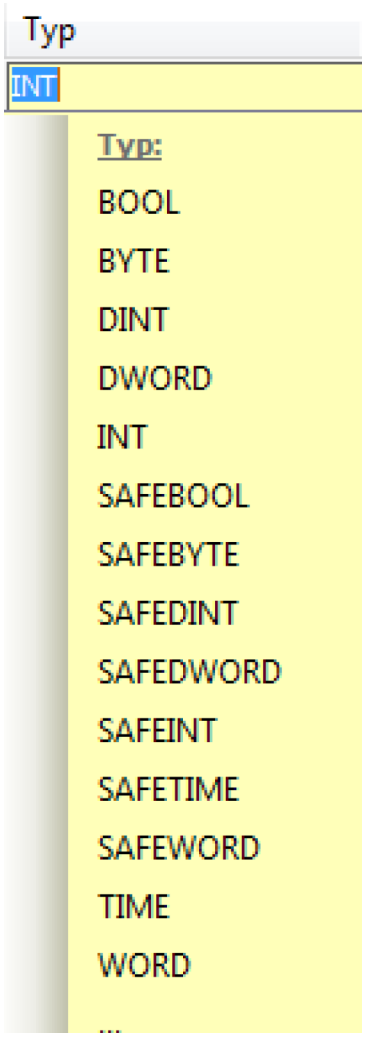


Abb. 58: Deklarationsfenster: Auswahlliste des Typs

6.3.3.3 Variablen für Basic-POUs

Schlüsselwörter der Variablen-deklaration für Basic Level

Variablen für Programm-POU Basic Level

- VAR
- VAR CONSTANT
- VAR_EXTERNAL für Kanalvariablen und Stack-Instanzen
- VAR_EXTERNAL CONSTANT nur erlaubt von VAR_GLOBAL CONSTANT

Variablen für Funktionsbaustein-POU Basic Level

- VAR
- VAR CONSTANT
- VAR_INPUT
- VAR_OUTPUT

Bedeutung der Schlüsselwörter

- VAR: Deklaration normaler interner Variablen, POU-weite Variable
- VAR_INPUT: Deklaration von Input-Variablen
- VAR_OUTPUT: Deklaration von Output-Variablen
- der Modifizierer CONSTANT dient zur Deklaration von symbolischen Konstanten

Datentypen Basic Level

- BOOL
- INT: nur als konstanter Eingangsparameter für einen FB-Aufruf
- DINT: nur als konstanter Eingangsparameter für einen FB-Aufruf
- WORD: nur als Ausgang für Diagnosezwecke
- TIME: nur als konstanter Eingangsparameter in FB-Aufruf
- SAFEBOOL
- SAFEINT nur als konstanter FB-Eingang in Aufruf
- SAFEDINT: nur als konstanter Eingangsparameter in FB-Aufruf
- SAFEWORD: nur als konstanter Eingangsparameter in FB-Aufruf
- SAFETIME: nur als konstanter Eingangsparameter in FB-Aufruf

Der Datentyp REAL steht nicht zur Verfügung.

Funktionsbausteintypen Basic Level

Benutzerdefinierte Typen sind alle Funktionsbausteine der Applikation und die Bausteine der Bibliotheken, die in ↬ *Kapitel 15.1 „Versionsliste der Bausteine“ auf Seite 293* aufgelistet sind.

Im Basic-Level zur Verfügung stehende Bausteine der Bibliothek SafetyStandard:

- SF_CTU
- SF_CTD
- SF_CTUD
- SF_TON
- SF_TOF
- SF_TP

Die bistabilen FBs und Kanten-FBs sind im Basic Level nicht erlaubt.

Einschränkungen für Funktionsbausteintypen

- Es darf zu keiner direkten oder indirekten Rekursion von FBs kommen.
- Die Variable darf nicht als konstant deklariert werden.
- Instanzen normaler FB-Typen können nur als globale Variablen und interne Variablen vorkommen. (Input-Variablen, Output-Variablen und logische E/As können nur von einem Basistyp sein)

6.3.3.4 Variablen für Extended-POUs

Schlüsselwörter der Variablendeklarationen für POUs Extended Level

Variablen für Programm-POU Extended Level

- VAR
- VAR CONSTANT
- VAR_EXTERNAL
- VAR_EXTERNAL CONSTANT

Variablen für Funktionsbaustein-POU Extended Level

- VAR
- VAR CONSTANT
- VAR_INPUT
- VAR_OUTPUT

Bedeutung der Schlüsselwörter:

- VAR: Deklaration normaler interner Variablen, POU-weite Variable
- VAR_INPUT: Deklaration von Eingangsvariablen
- VAR_OUTPUT: Deklaration von Ausgangsvariablen
- VAR_EXTERNAL: Deklaration von mit VAR_GLOBAL in der Applikation bereits deklarierter globaler Variablen, um sie in der POU benutzbar zu machen.

  Globale Variablen, die den Modifizierer CONSTANT haben, müssen als VAR_EXTERNAL CONSTANT deklariert werden.
- der Modifizierer CONSTANT dient zur Deklaration von symbolischen Konstanten

Für die Deklaration als VAR_EXTERNAL und VAR_EXTERNAL CONSTANT stehen bereits existierende Variablen der Kategorie "Globale Variablen" zur Verfügung:

- Globale Variablen des GVL-Objekts der Sicherheitsapplikation.

  Wird eine Variable des GVL-Objekts der Sicherheitsapplikation im Implementierungsteil verwendet, so wird sie automatisch im Deklarationsteil als externe Variable deklariert.

  Nach IEC ist es explizit verboten, in einer POU globale Variablen zu verwenden, ohne sie als "External" zu deklarieren.
- Mapping-Variablen der logischen E/As (Logische Austauschgeräte und sichere Feldgeräte)

In Basic- und Extended-POUs ist die Deklaration von VAR_IN_OUT Variablen nicht möglich.

Datentypen Extended Level

Für die Implementierung einer POU im Programmierlevel Extended stehen dem Entwickler folgende Datentypen zur Verfügung:

Sicherheits-Standardtypen:

- BOOL
- BYTE: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung

- DINT
- DWORD: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung
- INT
- TIME: erlaubt als konstanter Eingangsparameter und für lokale Variablen.

  Nicht zulässig sind External-Deklarationen globaler Variablen vom Typ SAFETIME
- WORD: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung
- Kein Datentyp REAL
- SAFEBOOL
- SAFEBYTE: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung
- SAFEDINT
- SAFEDWORD: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung
- SAFEWORD: für den Austausch kodierter Informationen (Statuscode, Diagnosecode, Steuercode) zwischen vordefinierten Funktionsbausteinen und mit der Umgebung
- SAFEINT
- SAFETIME, erlaubt als konstanter Eingangsparameter und für lokale Variablen.

  Nicht zulässig sind External-Deklarationen globaler Variablen vom Typ SAFETIME, wenn sie weder symbolische Konstanten noch importierte logische E/As sind.

Der Datentyp REAL steht nicht zur Verfügung

Funktionsbausteintypen Extended Level

Bei den Benutzerdefinierte Typen für den Extended Level handelt es sich um alle Funktionsbausteine der Applikation und alle Bausteine der Bibliotheken, die in *Kapitel 15.1 „Versionsliste der Bausteine“ auf Seite 293* aufgelistet sind.

Für Funktionsbausteintypen gelten folgende Einschränkungen:

- Es darf zu keiner direkten oder indirekten Rekusion von FBs kommen.
- Die Variable kann nicht als konstant deklariert werden.
- Instanzen normaler FB-Typen können nur als globale Variablen und interne Variablen vorkommen. (Eingangsvariablen, Ausgangsvariablen und logische E/As können nur von einem Basistyp sein).

6.3.4 Netzwerke

6.3.4.1 Überblick Netzwerke

Programmiereinheiten in FUP sind in aufsteigend durchnummerierte Netzwerke untergliedert, die grafisch dargestellte Elemente, wie Operanden, Bausteinaufrufe, Zuweisungen, Sprünge oder Sprungmarken enthalten können. Die Netzwerke werden in aufsteigender Reihenfolge abgearbeitet.

Innerhalb eines Netzwerks muss der weist der Programmcode eine Baumstruktur auf, dies bedeutet: keine Parallelverschaltung, keine Aufsplittung, keine expliziten Feedbackloops.

Die logischen Elemente eines Netzwerks werden durch Linien zu einer mit der Wurzel rechts liegenden baumartigen Formation (kurz Baum) verbunden, wobei die Boxen in diesem Baum als Knoten fungieren. Ausnahme: Mehrfachzuweisungen sowie Sprung- und Return-Anweisungen fächern diesen Baum in entgegengesetzter Richtung auf. Die Elemente werden vom Editor automatisch, entsprechend ihrer Einfügeposition verbunden, eine freie Platzierung ist nicht möglich.

Grundlegende Funktionalitäten des FUP-Editors

In CODESYS Safety werden folgende in CODESYS Standard unterstützten FUP-Sonderoperatoren nicht unterstützt:

- Flankenerkennung bei Zuweisungen (auf Eingänge und Variablen)
- Set/Reset
- EN/ENO-Parameter

Für detaillierte und grundlegende Informationen zur Bedienung des FUP-Editors wird auf die Online Hilfe von CODESYS Standard verwiesen.

Befehle für FUP-Netzwerke

Befehle für Netzwerke (im Kontextmenü aktivierbar)

- *„Netzwerk einfügen“*
- *„Netzwerk einfügen (unterhalb)“*
- *„Kommentierung ein/aus“*
- *„Bausteinaufruf einfügen“*
  (siehe *Kapitel 6.3.4.3 „Operatoren“ auf Seite 134* und *Kapitel 6.3.4.5 „FB-Aufrufe“ auf Seite 141*)
- *„Leeren Baustein einfügen“*
  (siehe *Kapitel 6.3.4.3 „Operatoren“ auf Seite 134* und *Kapitel 6.3.4.5 „FB-Aufrufe“ auf Seite 141*)
- *„Bausteineingang einfügen“*
  (siehe *Kapitel 6.3.4.3 „Operatoren“ auf Seite 134*)
- *„Zuweisung einfügen“*
  (siehe *Kapitel 6.3.4.2 „Datenfluss und Zuweisungen“ auf Seite 133*)

- *„Sprungmarke einfügen“*
  (siehe ↳ *Kapitel 6.3.4.4 „Sprung/Return und Sprungmarke“ auf Seite 139*)
- *„Sprung einfügen“*
  (siehe ↳ *Kapitel 6.3.4.4 „Sprung/Return und Sprungmarke“ auf Seite 139*)
- *„Return einfügen“*
  (siehe ↳ *Kapitel 6.3.4.4 „Sprung/Return und Sprungmarke“ auf Seite 139*)
- *„Weiterverschaltung festlegen“*
  (siehe ↳ *„Weiterverschaltung Festlegen“ auf Seite 142*)
- *„Nicht verwendete FB-Aufruf-Parameter entfernen“*
  (siehe ↳ *„Nicht verwendete FB-Aufruf-Parameter entfernen“ auf Seite 143*)
- *„Parameter Aktualisieren“*
  (siehe ↳ *Kapitel 6.3.4.5 „FB-Aufrufe“ auf Seite 141*)

Netzwerke einfügen

Durch Aktivierung des Kontextmenü-Befehls *„Netzwerk einfügen“* wird ein Netzwerk im Implementierungsteil des Editors eingefügt. Existieren bereits Netzwerke, wird das Netzwerk vor dem aktuellen Netzwerk eingefügt.

Der Kontextmenü-Befehl *„Netzwerk einfügen unterhalb“* bewirkt, dass ein Netzwerk unterhalb des aktuellen Netzwerks eingefügt wird.

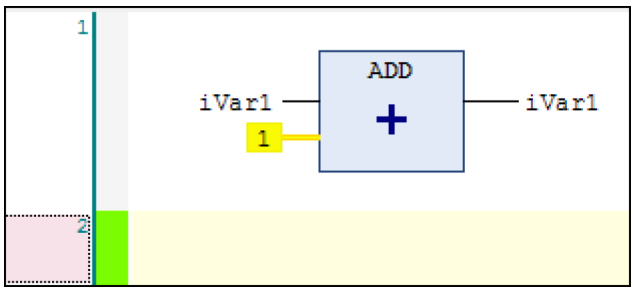


Abb. 59: Bsp. Unterhalb von Netzwerk 1 eingefügtes Netzwerk 2

Kommentierung ein/aus

Durch den Befehl *„Kommentierung ein/aus“* wird ein Netzwerk auskommentiert bzw. in den normalen Zustand gebracht. Bei Auskommentierung werden die im Netzwerk enthaltenen Elemente ignoriert und als inaktiv dargestellt. Somit werden alle auskommentierten Netzwerke bei der Ausführung der Applikation ignoriert und nicht ausgeführt.

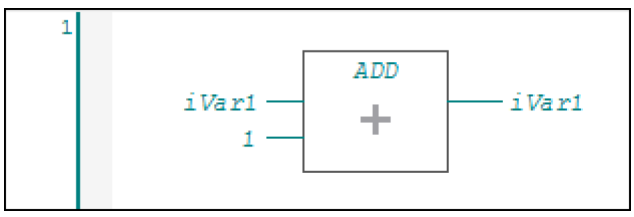


Abb. 60: Bsp.: Auskommentiertes Netzwerk

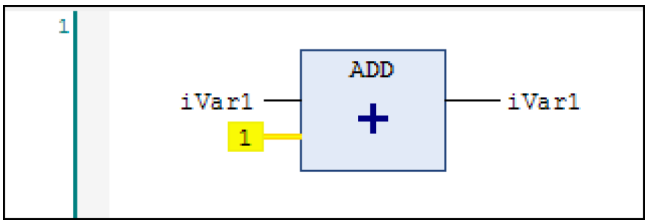


Abb. 61: Bsp. Normaler Zustand des Netzwerks

Netzwerktitel und Netzwerkkommentar

Jedem Netzwerk kann ein Titel und ein Kommentar hinzugefügt werden, wenn die entsprechenden FUP-Optionen aktiviert sind.

Aktivierung der sicheren FUP-Optionen

1. Im Menü *„Tools“* den Dialog *„Optionen...“* öffnen
2. Im Dialog *„Optionen“* den Dialog *„Sichere FUP-Optionen “* aktivieren.
3. Die Optionen *„Netzwerktitel anzeigen“* und *„Netzwerkkommentar anzeigen“* auswählen
4. Schaltfläche *„OK“* aktivieren.

Ein Netzwerktitel kann direkt in der ersten Zeile des Netzwerks, der Netzwerkkommentar kann in der zweiten Zeile des Netzwerks editiert werden. In beiden Fällen muss die entsprechende Zeile zuerst selektiert werden.

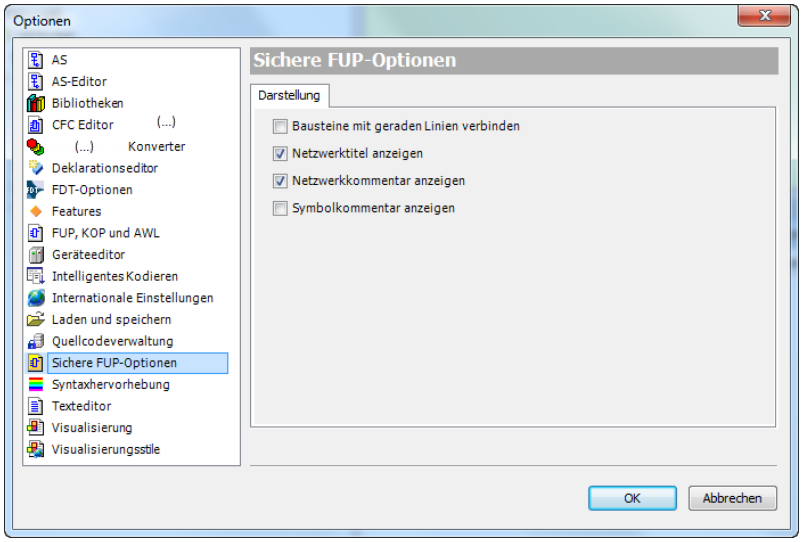


Abb. 62: Dialog: 'Sichere FUP Optionen'

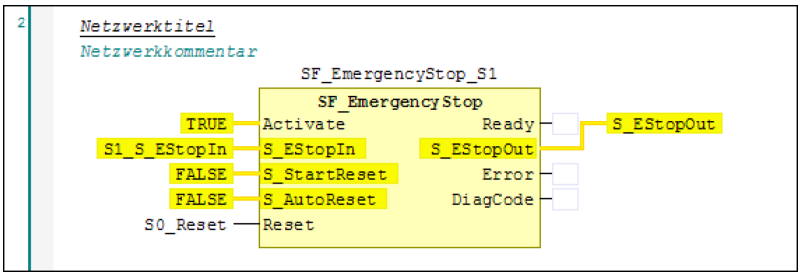


Abb. 63: Netzwerk mit Titel und Kommentar

6.3.4.2 Datenfluss und Zuweisungen

Netzwerke visualisieren den Datenfluss durch die Verbindungen zwischen Variablen, Operatoren und FB-Aufrufen.

Datenfluss sicherheitsgerichteter Signale

Der Datenfluss sicherheitsgerichteter Signale der FUP-Programmierung wird in CODESYS Safety folgendermaßen hervorgehoben:

- Literale und konstant deklarierte Variablen werden gelb hinterlegt.
- SAFExxx-Variablen werden gelb hinterlegt.
- Datenfluss von SAFE-Werten in SAFE-Variablen und in Eingänge von Operatoren und Bausteinen hinein wird durch dicke gelbe Linien dargestellt
- Funktionsbausteine werden gelb dargestellt, wenn sie mindestens einen SAFE-Ausgang haben
- Operator-Aufrufboxen werden gelb ausgefüllt, wenn der Ausgang SAFE ist. Dies ist unter folgenden Bedingungen der Fall:
  - bei Operator AND: der Ausgang ist SAFE, wenn mindestens 1 Eingang SAFE ist.
  - alle anderen Operatoren, inklusiv Konvertierungen: der Ausgang ist SAFE, wenn alle Eingänge SAFE sind.

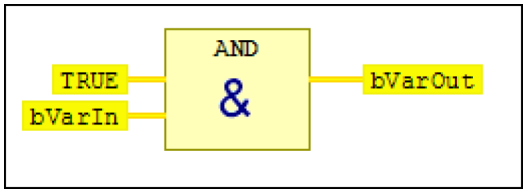


Abb. 64: Beispiel für sicheren Datenfluss: Operator AND mit Literal: TRUE, SAFE-Variablen: VarIn und VarOut, Operator AND

Zuweisungen

Eine Zuweisung ist ein FUP-Element, das in einem Netzwerk den ankommenden Signalfluss aufnimmt und in einem Operanden speichert, d.h. in die Variable schreibt. Für eine Zuweisung ist immer eine Variable notwendig. Zuweisungen können nur am Ausgang einer Box eingefügt werden.

Das Einfügen von Zuweisungen erfolgt wie im Standard-FUP von CODESYS.

Beispiele für Zuweisungen

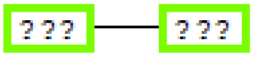


Abb. 65: Beispiel: In leerem Netzwerk eingefügte Zuweisung

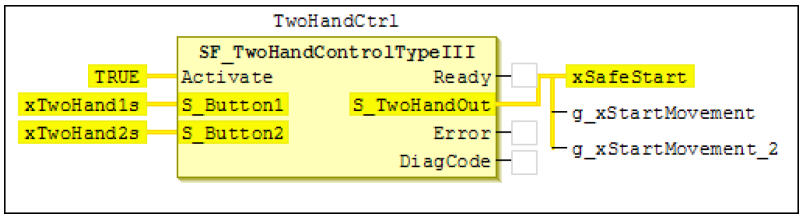


Abb. 66: Beispiel: Mehrfachzuweisung durch Einfügen einer neuen Zuweisung an bestehender Zuweisung

Bei Zuweisungen von nicht-SAFE Werten auf SAFE-Variablen oder SAFE-Eingänge werden die Variablen bzw. Eingänge dunkelrot markiert.

Regel P13 (Bitcodes)

HINWEIS!

Bei der Verschaltung zweier Variablen der Typen (SAFE)BYTE, (SAFE)WORD, (SAFE)DWORD müssen beide die gleiche Information darstellen (zum Beispiel 2 x Antriebssteuerwort, oder 2 x Ventilstatus)

6.3.4.3 Operatoren

Als Operatoren werden die eingebauten Standardfunktionen, die eine Teilmenge der Standardfunktionen der IEC darstellen, bezeichnet. Benutzerdefinierte Funktionen können in der Safety-Programmierung nicht erstellt werden.

Die Operatoren haben die gleiche Semantik wie in Standard-CODESYS. Sie können sowohl mit SAFExxx als auch mit Standard-Datentypen beschaltet werden.

Einige Operatoren lassen sich mit dem Befehl *„Bausteineingang Einfügen“* um zusätzliche Eingänge erweitern. Diese sind im Folgenden markiert

Operatoren im Basic Level

Boolsche Operatoren

- AND (2 Eingänge) - erweiterbar
- AND (3 Eingänge) - erweiterbar
- OR (2 Eingänge) - erweiterbar
- OR (3 Eingänge) - erweiterbar

Alle Operanden des OR müssen SAFEBOOL sein.

Operatoren im Extended Level

Boolsche Operatoren

- AND (2 Eingänge) - erweiterbar
- AND (3 Eingänge) - erweiterbar
- OR (2 Eingänge) - erweiterbar
- OR (3 Eingänge) - erweiterbar
- XOR
- NOT

Regel P5 (NOT/XOR)

VORSICHT!

Der unachtsame Gebrauch der Operatoren XOR und NOT kann zum Verlust der Failsafe-Eigenschaft von SAFExxx-Variablen führen. Der Safety-Checker erzeugt keine Warnung für solche Konstrukte.

Die Operatoren XOR und NOT können die Failsafe-Eigenschaft einer SAFExxx-Variablen negieren, sodass die SAFE-Variable ihre Failsafe Eigenschaft verliert, also "ausfallunsicher" wird. Dies kann zu unbeabsichtigtem Anlaufen der Anlage führen. Es wird dadurch keine Warnung durch den Safety-Checker erzeugt.

Programmierregel: Die SAFExxx-Ausgänge von NOT und XOR müssen ermittelt werden. Anschließend muss sichergestellt werden, dass diese NOT/XOR-Ausgänge nicht auf Ausgänge (E/As) durchgeschaltet werden.

Mathematische Operatoren

- ADD (2 Eingänge) - erweiterbar
- ADD (3 Eingänge) - erweiterbar
- SUB
- MUL - erweiterbar
- DIV - Laufzeitfehler bei Division durch Null (siehe weiter unten)

Das Ergebnis einer Verknüpfung von Werten mit ADD, SUB, MUL, DIV kann eine Zahl ergeben, die außerhalb des Wertebereichs des Datentyps liegt. Kommt es zur Laufzeit zu einer Wertebereichsverletzung stellt das nach IEC 61131-3 [N1.1.3-Kap.6.6.2.5.8] bzw. [N1.1.3-Kap.6.6.2.5.12] einen Fehler im Programm dar. Dieser Fehler wird von CODESYS Safety nicht diagnostiziert! Stattdessen wird die Applikation auf folgende Weise fortgesetzt:

DIV: Wenn die Division der Eingangswerte keine Ganzzahl ergibt, wird CPU-abhängig mit der nächstgrößeren oder nächstkleineren Ganzzahl weiter gerechnet.

ADD, SUB, MUL auf DINT/SAFEDINT-Werten: Wenn die mathematische Operation eine Zahl *N* außerhalb des Wertebereichs von DINT ergibt, wird mit folgender Zahl innerhalb des DINT-Wertebereichs weiter gerechnet: Bei positivem *N*: die erste Zahl im DINT-Wertebereich, die durch wiederholte Subtraktion von 4294967296 (2^{32}) erreicht wird. Bei negativem *N*: die erste Zahl im DINT-Wertebereich, die durch wiederholtes Addieren von 4294967296 (2^{32}) erreicht wird.

ADD, SUB, MUL auf INT/SAFEINT-Werten: Wenn die mathematische Operation eine Zahl *N* außerhalb des INT-Wertebereichs ergibt, dann liegt diese immer noch im DINT- Wertebereich und in anschließenden Operatoren wird mit dieser Zahl *N‘* weiter gerechnet: Dabei werden mathematische Operatoren nach den Regeln für DINT/SAFEDINT-Werte angewendet. Konvertierungen INT_TO_xxx von *N'* führen zu einem Laufzeitfehler. Und bei Zuweisung von *N‘* auf eine INT-Variable wird folgender Wert innerhalb des INT-Wertebereich in die Variable gespeichert: Bei positivem *N'*: die erste Zahl im INT-Wertebereich, die durch wiederholte Subtraktion von 65536 (2^{16}) erreicht wird. Bei negativem *N‘*: die erste Zahl im INT-Wertebereich, die durch wiederholtes Addieren von 65536 (2^{16}) erreicht wird.

Regel P12 (Operatoren)

VORSICHT!

Der unachtsame Gebrauch der Operatoren ADD, SUB, MUL, DIV kann zu Wertebereichsverletzungen führen, die nicht bemerkt werden und ein unerwartetes Verhalten der Applikation nach sich ziehen. Werte dürfen nur mit ADD, SUB, MUL, DIV verknüpft werden, wenn eine Bereichsverletzung entweder ausgeschlossen werden kann (Applikationslogik, Grenzwertprüfungen, o.ä.), oder sie in der Applikation erkannt und der Ergebniswert in diesem Fall nicht verwendet wird (sondern z.B. mittels SEL durch einen passenden Wert ersetzt wird).

Weitere mathematische Operatoren (nicht erweiterbar)

- EQ
- NE
- LT
- LE
- GT
- GE

Andere Operatoren

- SEL
- MUX - erweiterbar, Laufzeitfehler bei ungültigem ersten Input (siehe unten)

Konvertierungen

- BOOL_TO_INT
- BOOL_TO_DINT
- BOOL_TO_TIME
- BOOL_TO_WORD
- BYTE_TO_INT
- BYTE_TO_DINT
- BYTE_TO_TIME
- BYTE_TO_WORD
- DINT _TO_BOOL
- DINT_TO_BYTE - Prüfung des Wertebereichs (siehe unten)
- DINT_TO_INT - Prüfung des Wertebereichs (siehe unten)
- DINT_TO_TIME - Prüfung des Wertebereichs (siehe unten)
- DINT_TO_WORD - Prüfung des Wertebereichs (siehe unten)
- DINT_TO_DWORD
- DWORD_TO_DINT
- DWORD_TO_TIME
- INT_TO_BOOL
- INT_TO_BYTE - Prüfung des Wertebereichs (siehe unten)
- INT_TO_DINT
- INT_TO_DWORD
- INT_TO_TIME - Prüfung des Wertebereichs (siehe unten)
- INT_TO_WORD
- TIME_TO_BOOL
- TIME_TO_BYTE - Prüfung des Wertebereichs (siehe unten)
- TIME_TO_INT - Prüfung des Wertebereichs (siehe unten)
- TIME_TO_DINT - Prüfung des Wertebereichs (siehe unten)
- TIME_TO_WORD - Prüfung des Wertebereichs (siehe unten)
- TIME_TO_DWORD
- WORD_TO_BOOL
- WORD_TO_BYTE - Prüfung des Wertebereichs (siehe unten)
- WORD_TO_DINT
- WORD_TO_INT
- WORD_TO_TIME
- WORD_TO_DWORD

Laufzeitfehler bei Bereichsüberschreitungen der Operatoren im Extended Level

CODESYS Safety reagiert bei den im Folgenden aufgeführten Bereichsüberschreitungen mit einem **Laufzeitfehler**, wodurch die Applikation gestoppt und ein Logbucheintrag erzeugt wird.

Level	Sprachelement	Lauftzeifehler bei
Extended	DIV	Division durch 0
Extended	MUX	Aufruf mit erstem Input mit negativem Wert oder mit Wert N größer als die Anzahl der Eingänge minus 1. Z.B. MUX(2, 16#8000, 16#8001)
Extended	DINT_TO_INT, TIME_TO_DINT, TIME_TO_INT, DINT_TO_TIME, INT_TO_TIME, DINT_TO_WORD, TIME_TO_WORD, DINT_TO_BYTE, INT_TO_BYTE, TIME_TO_BYTE, WORD_TO_BYTE	Ausgangswert ist nicht im Wertebereich des Zieltyps: Bei Konvertierung zwischen zwei ANY_MAGNITUDE Typen (INT, DINT, TIME) muss der numerische Ausgangswert im Wertebereich des Zieltyps liegen (wobei TIME-Werte als Anzahl von Millisekunden gerechnet werden). Bei Konvertierung von/zu Bitstring-Typen (BYTE, WORD, DWORD) muss das Bitmuster des Ausgangswerts ein Bitmuster der Zieltyps sein. Beispiele: DINT_TO_INT(16#0000FFFF), weil 2^{16}-1 kein INT-Wert ist, ebenso DINT_TO_TIME(-1), weil es keine negative TIME-Werte gibt TIME_TO_DINT(t#365d), weil 365 Tage = 3,153,600,000ms = 16#BBF81E00 ist, und damit größer als die größte DINT-Zahl 2^{31}-1 = 16#7FFFFFFF INT_TO_BYTE(-1), da BYTE nur 0 bis 255 umfasst, WORD_TO_BYTE(0xFFFF), da BYTE nur bis 0xFF geht.

Das Standardverhalten von SEL/MUX, dass bei zu großem Eingangswert der maximale Wert und bei negativem Eingangswert der Wert 0 ausgewählt wird, muss in der Sicherheitsapplikation programmiert werden.

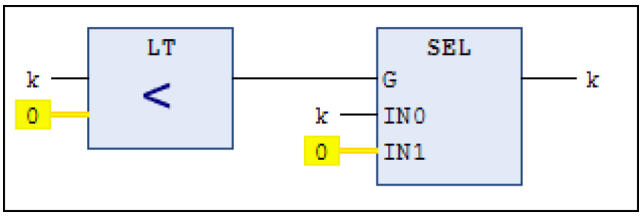


Abb. 67: Programmierung des Standardverhalten von SEL: für k‹0

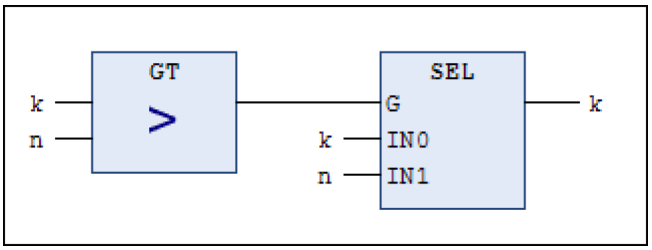


Abb. 68: Programmierung des Standardverhaltens von SEL für k›Max, n...Maximalwert

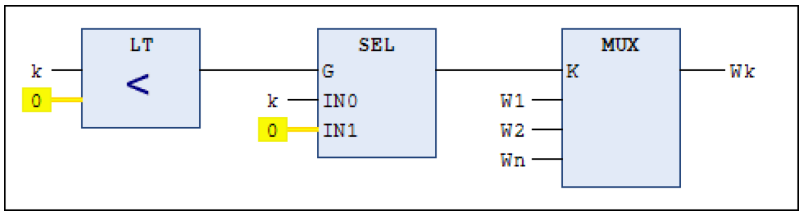


Abb. 69: Programmierung des Standardverhaltens von MUX für k‹0

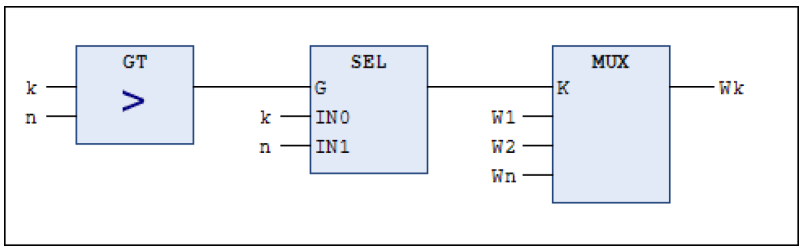


Abb. 70: Programmierung des Standardverhaltens von MUX für k›Max, n...Maximalwert

6.3.4.4 Sprung/Return und Sprungmarke

Durch einen bedingten Sprung wird die sequentielle Abarbeitungsreihenfolge der POU unterbrochen. Wenn die Sprungbedingung TRUE ist, wird auf ein mit der Sprungmarke gekennzeichnetes Netzwerk gesprungen.

Mit einer bedingten Return-Anweisung wird die Abarbeitungsfolge des Bausteins unterbrochen. Wenn die Returnbedingung erfüllt ist, wird der Baustein verlassen.

Sprünge und Returns sind nur als bedingte Vorwärtssprünge und bedingte Returns erlaubt. Sie sind nur im Programmierlevel Extended möglich. Im Programmierlevel Basic sind generell keine Sprünge/Returns erlaubt.

Sprung/Return

- Bedinge Vorwärtsprünge und Returns sind nur am Netzwerkende erlaubt (dabei bei Mehrfachzuweisungen nach der letzten Zuweisung)
- Als Sprungziel muss ein Netzwerk mit Sprungmarke innerhalb der gleichen POU existieren.
- Das Sprungziel-Netzwerk muss hinter dem Netzwerk mit dem Sprung liegen.
- Das Sprungziel darf sich in keinem auskommentierten Netzwerk befinden.
- Die Bedingung eines Sprung/Return muss boolesch sein.

Regel P6 (Sprung)

HINWEIS!
Bedingte Vorwärtssprünge dürfen gemäß den PLCopen-Regeln für "Sicherheitssoftware" nur für Zustandsmaschinen verwendet werden.

Regel P7 (Return)

HINWEIS!
Returns dürfen gemäß den PLCopen-Regeln für "Sicherheitssoftware" nur als Fehleraussprung verwendet werden.

VORSICHT!
Der unachtsame Gebrauch von bedingten Sprüngen und Returns kann zum Verlust der Failsafe-Eigenschaft von SAFExxx-Variablen führen. Der Safety-Checker erzeugt keine Warnung für solche Konstrukte.

Sprünge mit sicherer Bedingung sind in dieser Hinsicht unkritisch.

Ein bedingter Sprung, der von einem unsicheren Wert abhängt und als Sprungziel eine Zuweisung auf eine SAFExxx Variable hat, erlaubt es einem unsicheren Eingang, eine sicheren Ausgang zu beeinflussen. Hierfür gilt folgende Regel:

Alle Zuweisungen auf SAFExxx Variable, die Ziele von bedingten Sprüngen sind, die von unsicheren Variablen abhängen, müssen ermittelt werden. Es muss dafür gesorgt werden, dass die Sicherheit der Maschine in allen Fällen gewährleistet ist.

Sprungmarke

Sprungmarken sind Zieladressen für Sprünge und können nur am Anfang eines Netzwerks mit dem Kontextmenübefehl *„Sprungmarke einfügen“* eingefügt werden.

Eine Sprungmarke zwischen dem Aufruf des FBs und dem Auslesen eines Ausgangs dieses FBs ist nicht erlaubt

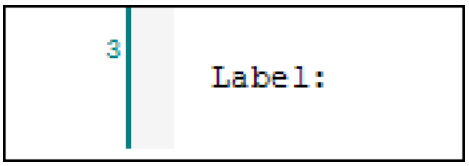


Abb. 71: Beispiel: Eingefügte Sprungmarke

6.3.4.5 FB-Aufrufe

Funktionsweise von FB-Aufrufen

Beim Aufruf des Funktionsbausteins werden die Formalparameter mit den aktuellen Werten der Eingangsvariablen oder Literalen versorgt, bzw. den Ausgängen zugewiesen. Die Formalparameter und ihre zugewiesenen Variablen/Konstanten müssen im Datentyp übereinstimmen. Der Instanzname wird im Deklarationsteil des Editors als Variable mit Datentyp = Bausteinname deklariert.

FB-Instanzen können nur auf ihre eigenen POU-weiten Daten und Parameter zugreifen.

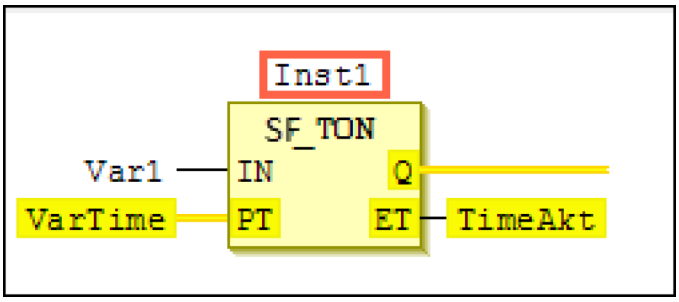


Abb. 72: Beispiel: Instanz Inst1 des Bausteins SF_TON mit Formalparametern IN, PT, Q, ET

In CODESYS Safety stehen dem Entwickler neben den selbst erstellten Funktionsbausteinen, die Bausteine der Sicherheitsbibliotheken "SafetyPLCopen", "SafetyStandard" und gegebenenfalls weiterer Sicherheitsbibliotheken zur Verfügung.

HINWEIS!

Bevor Sie einen Bibliotheksbaustein verwenden, müssen Sie die Dokumentation dieses Bausteins kennen. Die Dokumentation des Bibliotheksbausteins muss mit der aktuell in der Applikation verwendeten Version des Bibliotheksbausteins übereinstimmen: Sie verifizieren dies, indem Sie die Versionsinformation der Bausteindokumentation mit der Objektversion des Funktionsbausteins vergleichen, die in der Registerkarte *„Objekte“* des Editors des Safety Applikationsobjekts angezeigt wird.

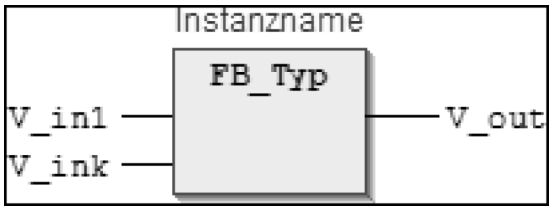


Abb. 73: Beispiel: FB-Aufruf

Bausteinaufruf einfügen, Leeren Baustein einfügen

Das Einfügen von Funktionsbausteinen erfolgt wie in Standard-FUP von CODESYS.

Einfügeposition

An welcher Stelle des Netzwerks der ausgewählte Baustein eingefügt wird, hängt davon ab, wo das Kommando ausgeführt wird:

Kommando wird ausgeführt in

- in einem Netzwerk, so dass eine Verknüpfung zweier Bausteine mit sicheren Ein- bzw. Ausgängen entsteht:
  Es werden automatisch immer die ersten SAFE-Eingänge bzw. SAFE-Ausgänge miteinander verknüpft.
- leerem Netzwerk:
  Für Funktionsbausteinaufrufe werden die Ein- und Ausgänge entsprechend der Signatur des Objekts zusätzlich beschriftet. Die Instanznamen von Funktionsbausteinen werden über der Box angezeigt.
- am Hauptausgang einer anderen Box:
  Der erste Eingang der neuen Box wird mit dem Hauptausgang der bestehenden Box verbunden. Das Einfügen einer Box an einem Nicht-Hauptausgang einer anderen Box ist nicht möglich.
- an einem Eingang einer anderen Box:
  Die neue Box wird zwischen dem eingehenden Signalfluss und dem Eingang der bestehenden Box eingefügt.

Löschen der Ein- und Ausgänge

Die Ein- und Ausgänge einer Box, die einen Funktionsbausteinaufruf darstellen, können mit dem Befehl *„Löschen“* des Kontextmenüs gelöscht werden. Dabei werden die an diesem Ein- oder Ausgang anliegenden Elemente auch gelöscht. Wird der Hauptausgang eines Bausteinaufrufs, der sich innerhalb eines Netzwerks befindet, gelöscht, wird im Falle eines Netzwerkes der gesamte Baum links von diesem Bausteinausgang (also inklusive des Bausteins selbst) entfernt. Ist der Baustein das (rechte) Ende von einem Netzwerk, kann auch der letzte Ausgang (Hauptausgang) entfernt werden, der Baustein bleibt dann erhalten und besitzt keine Ausgänge.

Weiterverschaltung Festlegen

Der Befehl *„Weiterverschaltung festlegen“* legt fest, welcher Ausgang einer Box mit mehreren Ausgängen für die Weiterverschaltung (Hauptausgang) benutzt werden soll. Jede Box mit Ausgängen kann nur einen einzigen Hauptausgang haben. Andere Boxen können mit dieser Box nur an deren Hauptausgang verbunden werden. Das Kommando kann nur am Ausgang einer Box mit mehreren Ausgängen ausgeführt werden. Beim Umdefinieren des Hauptausganges werden mit dem alten Hauptausgang verbundene Elemente automatisch auf den neuen Hauptausgang gesetzt und mit dem alten Nebenausgang verbundene Zuweisungsziele automatisch auf den neuen Nebenausgang gesetzt.

Parameter aktualisieren

Der Befehl *„Parameter Aktualisieren“* aktualisiert die Ein- und Ausgänge eines Funktionsbausteinaufrufs. Dabei werden die Parameter der Box mit den in ihrer Schnittstelle definierten verglichen. Kommt ein neuer Ein- oder Ausgang hinzu, wird dieser mit der leeren Variable beschaltet. Fällt ein Ein- oder Ausgang weg, wird dieser zusammen mit allen beschalteten Elementen gelöscht. Die Zuordnung erfolgt dabei über die Namen der Ein- bzw. Ausgänge.

Nicht verwendete FB-Aufruf-Parameter entfernen

Der Befehl *„Nicht verwendete FB-Aufruf-Parameter entfernen“* entfernt alle Ein- und Ausgängsanschlüsse des Aufrufs, welche leer belegt sind. Die minimale vom Baustein geforderte Anzahl von Eingängen bleibt bestehen.

6.4 Implementierung von F-Modulen

Dieser Abschnitt betrifft nur den Fall, dass die Sicherheitssteuerung als F-Device einer übergeordneten Sicherheitssteuerung (F-Host) verwendet wird - siehe Topologie 4 in *„Topologien“ auf Seite 29*. Das ist der Fall, wenn die Standardsteuerung einen PROFINET-Device-Knoten hat, unter dem F-Module eingefügt wurden. Siehe CODESYS PROFIsafe F-Device-Hilfe.

PROFIsafe-Adresse eine F-Moduls konfigurieren

1. Doppelklicken Sie im Gerätebaum das logische E/A des F-Moduls.

   ⇨ Das logische E/A öffnet sich im Editor.

   Die Registerkarte *„Sichere Konfiguration“* ist geöffnet.

2. Doppelklicken Sie das Feld *„Wert“* der *„Source Address“* und geben Sie die Adresse des F-Hosts ein.

3. Doppelklicken Sie das Feld *„Wert“* der *„Destination Address“* und geben Sie die eindeutige Adresse des F-Moduls (PROFIsafe-Adresse) ein.

   Beachten Sie hierbei die Hinweise in *„Dokumentation implementierter F-Module für die Host-Programmierung“ auf Seite 227*.

Neben dem zyklischen Datenaustausch mit dem F-Host **muss** die Applikation eines F-Devices dem F-Host bestimmte Statussignale zurückmelden und **muss** die Applikation bestimmte Steuersignale des F-Hosts befolgen. Diese Signale werden über die FB-Schnittstelle des Treiberbausteins des entsprechenden F-Moduls ausgetauscht. Diese ist in *Kapitel 14.5 „PROFIsafe F-Device“ auf Seite 291* beschrieben.

Hinweis FDev_1 (Einsatzbereich)

WARNUNG!

Für den Fall, dass die für ein F-Modul in der Applikation und im untergeordneten Feldbus mit angeschlossenen Sensoren und Aktoren implementierten Funktionen nur für einen eingeschränkten SIL oder nicht für Prozessanwendungen ausgelegt sind, gilt:

Die Applikation muss prüfen, für welchen Einsatzbereich der F-Host das F-Modul konfiguriert: Für SIL1, SIL2, SIL3 oder für die Prozessanwendung. Dazu muss die Applikation den FB-Ausgang ‹Modulname›.S_F_SIL abfragen, sobald der FB-Ausgang ‹Modulname›.S_ConfigOK TRUE ist.

Im Fehlerfall darf das F-Modul den sicheren Zustand nicht verlassen und muss den FB-Eingang ‹Modulname›.S_ModuleOK_DS auf FALSE setzen, um dem F-Host einen Gerätefehler (device fault) zu melden.

Hinweis FDev_2 (device fault)

WARNUNG!

Die Sicherheitsapplikation muss für jedes implementierte F-Modul bestimmen, ob die Sicherheitssteuerung und die angeschlossenen Feldgeräte betriebsbereit sind oder ob ein Fehler in ihnen vorliegt (device fault).

Im Fehlerfall muss die Applikation für das F-Modul den sicheren Zustand einnehmen und muss den FB-Eingang ‹Modulname›.S_ModuleOK_DS auf FALSE setzen, um dem F-Host einen Gerätefehler (device fault) zu melden.

Eine Sicherheitsapplikation kann den Gerätefehler bezüglich eines F-Moduls auf zwei Weisen weiter behandeln:

a. Die Sicherheitsapplikation verlässt den Zustand S_ModuleOK_DS = FALSE nicht mehr. Nach der Behebung der Fehlerursache muss der Betreiber die Steuerung neu starten.

b. Die Applikation erkennt, wenn die Fehlerursache behoben wurde und setzt S_ModuleOK_DS = TRUE, wodurch die Prozesskommunikation unvermittelt wieder anläuft.

Hinweis FDev_4 (safe state)

WARNUNG!
Die Sicherheitsapplikation muss für ein implementiertes F-Modul an seinen angeschlossen Aktoren den sicheren Zustand einnehmen und Failsafe-Werte in den Ausgangsmodulen im untergeordneten Feldbus aktivieren, solange der F-Host das F-Modul nicht per FB-Ausgang ‹Modulname›.S_ActivateModule_DC freigibt.

Hinweis FDev_3 (process values)

WARNUNG!
Die Sicherheitsapplikation muss für jedes implementierte F-Modul per FB-Eingang ‹Modulname›.S_ValuesToHostOK_DS zurückmelden,

– ob die aktuell an den F-Host geschickten Werte (Ausgänge der Sicherheitsapplikation = F-Inputs des F-Hosts) gültige Prozesswerte aus angeschlossenen Sensoren darstellen oder ob der F-Host Failsafe-Werte verwenden soll
und
– ob sie aktuell die untergeordneten Ausgänge mit Prozesswerten ansteuert oder auf Failsafe gesetzt hat.

Hinweis FDev_5 (undersampling)

WARNUNG!
Die Sicherheitsapplikation, die ein F-Eingangsmodul implementiert, muss für Prozesswerte an den F-Host (Ausgänge der Sicherheitsapplikation = F-Inputs des F-Hosts bei S_ValuesToHost_OK = TRUE), die einen gefährlichen Zustand darstellen bzw. eine Sicherheitsreaktion durch den F-Host erforderlich machen könnten, Folgendes sicherstellen: Die Prozesswerte müssen bei S_ValuesToHost_OK mindestens so lange anstehen, bis ihr Empfang im F-Host durch eine zweimalige Flanke am FB-Ausgang ‹Modulname›.S_ValuesToHost_tick_DC bestätigt wurde.

6.5 Einbindung von Feldgeräten

6.5.1 Zugriff auf Eingangssignale und Ausgangssignale

Der E/A-Zugriff erfolgt ausschließlich über die logischen E/As. Jede Mapping-Variable kann wie eine globale Variable als VAR_EXTERNAL deklariert und im Programm verwendet werden.

6.5.2 Anbindung digitaler 1oo1- und 1oo2-Eingangsmodule

VORSICHT!
Bei der Verwendung eines sicheren Eingangssignals ist die Auswertlogik des Eingangsmoduls zu beachten!

Regel P8 (Check:1oo1)

VORSICHT!
1oo1-Eingangssignale sind auf Plausibilität zu prüfen, zum Beispiel gegen ein zweites, redundantes Eingangssignal mittels Funktionsbaustein SF_Antivalent oder SF_Equivalent.

Zwei mögliche Auswertlogiken werden unterschieden:

- 1oo1
- 1oo2

1oo1-Auswertung bedeutet, dass nur ein Sensor vorhanden ist, der bei erwarteter Funktionsweise einen SAFEBOOL-Wert generiert.

1oo2-Auswertung bedeutet, dass einer von zwei Sensoren wie erwartet funktionieren muss, um den SAFEBOOL-Wert zu generieren. Wenn ein Sensor nicht das erwartete Verhalten zeigt, wird der Wert auf den fail-safe Wert gesetzt.

Bei der 1oo1-Auswertung wird die Fehlererkennung für jeden Kanal separat durchgeführt. In diesem Fall können zusätzliche Maßnahmen auf Ebene der Sicherheitsapplikation und bei der peripheren Schaltung notwendig sein.

Es gilt allgemein für PROFIsafe (d.h. PROFIBUS sowie PROFINET): Der Wert des F-Parameters „F_SIL“ zeigt die Auswertlogik an: "SIL2" bedeutet 1oo1-Auswertung, "SIL3" bedeutet 1oo2-Auswertung (Im Fall einfacher Eingangsklemmen muss das Anschließen von Sensoren abhängig von diesem Parameter erfolgen. Einzelheiten sind der Gerätedokumentation zu entnehmen). Bei FSoE-Eingangsmodulen muss deren Auswertlogik der Gerätedokumentation entnommen werden.

Zugriff auf 1oo2 Eingangssignale

Die Sicherheitsapplikation wird wie folgt mit Safety Eingangsmodule mit 1oo2-Auswertung verbunden:

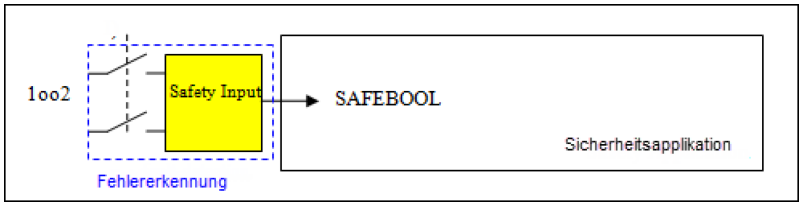


Abb. 74: Verbindung der Sicherheitsapplikation mit Safety Eingangsmodul mit 1oo2-Auswertung

Einige PLCopen-Funktionsbausteine, wie z.B. SF_GuardMonitoring, können sowohl mit Eingangsmodulen mit 1oo2-Auswertung (siehe folgende Abbildung) als auch mit Eingangsmodulen mit 1oo1-Auswertung (siehe Abb. 79) verbunden werden.

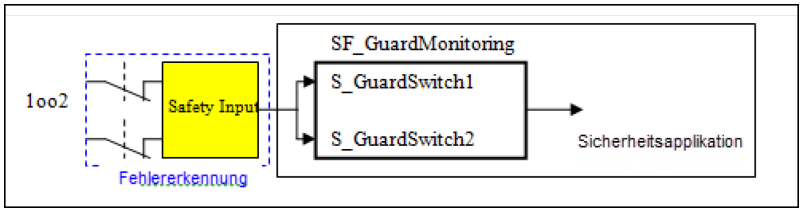


Abb. 75: 1oo2-Auswertung, SF_GuardMonitoring

Zugriff auf 1oo1 Eingangssignale

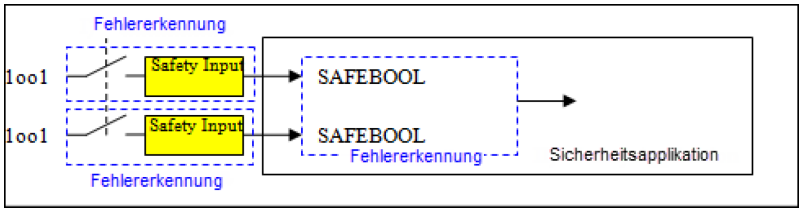


Abb. 76: Verbindung der Sicherheitsapplikation mit Safety Eingangsmodul mit 1oo1-Auswertung

Die Schalter können aus jeder Kombination aus normal geschlossen (NC) und normal offen (NO) bestehen, welche die Anforderungen und die Fähigkeit zur Fehlererkennung des Safety-Eingangs berücksichtigen. In Abb. 76 muss in der Sicherheitsapplikation eine zusätzliche Fehlererkennung implementiert werden. Diese wird entweder im signal-verarbeitenden Funktionsbaustein, oder separat über den Funktionsbaustein SF_Equivalent (siehe Abb. 78) oder den Funktionsbaustein SF_Antivalent (siehe Abb. 77) implementiert.

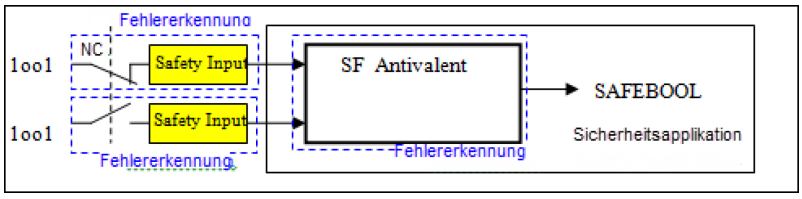


Abb. 77: 1oo1-Auswertung: Fehlererkennung mit SF_Antivalent

Bei Verwendung des Funktionsbaustein SF_Antivalent ist der NC-Schalter mit einem spezifischen Funktionsbaustein-Eingang zu verbinden. Für die Kombination von NC-Schalter mit NO-Schalter implementiert SF_Antivalent die notwendige Fehlererkennung. Die Schalter müssen dazu mit den entsprechenden Eingängen (S_ChannelNC und S_ChannelNO) verbunden werden.

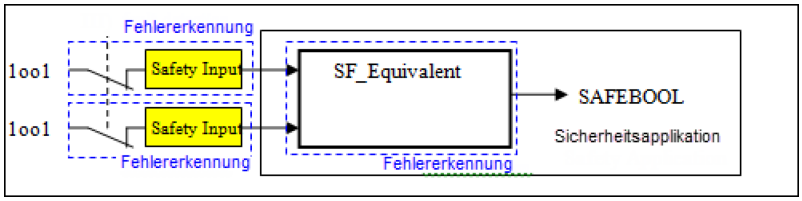


Abb. 78: 1oo1-Auswertung: Fehlererkennung mit SF_Equivalent

Für die Kombination zweier NC-Schalter oder zweier NO-Schalter implementiert SF_Equivalent die notwendige Fehlererkennung. Der SF_Equivalent verarbeitet zwei SAFEBOOL-Eingänge und überwacht, dass sich das Signal innerhalb einer spezifizierten Diskrepanzzeit äquivalent ändert.

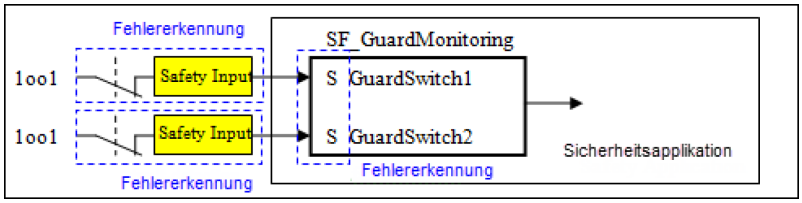


Abb. 79: 1oo1-Auswertung, SF_GuardMonitoring

Bei einigen PLCopen-Funktionsbausteinen , wie z.B. SF_Guard-Monitoring, ist die Fehlererkennung für die Kombination zweier NC- bzw. zweier NO-Schalter im Baustein selbst implementiert. D.h. die Schalter können ohne vorgeschaltetes SF_Equivalent direkt mit dem Baustein verbunden werden. (Wie solche Bausteine mit 1oo2-Eingangsmodulen verbunden werden können, zeigt Abb. 75).

6.5.3 Überwachung digitaler Eingangsmodule und Ausgangsmodule

Regel P9 (Check:Geräte)

VORSICHT!
Für Sicherheitsfunktionen verwendete Eingangsmodule (Sensoren, Taster, usw.) und Ausgangsmodule (Aktoren, Relays) müssen in ihrer Funktion überwacht werden.

Zur Überwachung der in der Maschinensicherheit üblichen Sicherheitsgeräte hat die PLCopen entsprechende Funktionsbausteine definiert. Es stehen in CODESYS Safety die folgenden PLCopen-Bausteine zur Verfügung:

- SF_Equivalent: Plausibilitätsüberwachung von zwei equivalenten Eingängen, welche zu einem logischen Ausgang verknüpft werden
- SF_Antivalent: Plausibilitätsüberwachung von zwei antivalenten Eingängen, welche zu einem logischen Ausgang verknüpft werden
- SF_ModeSelector: Plausibilitätsüberwachung eines aus 8 Schalter zur Auswahl der Betriebsart, wie z. B.. Manuell, Automatik
- SF_EmergencyStop: Auswertung Not-Halt-Schalter (Anlaufsperre)
- SF_ESPE (Electro-Sensitive Protective Equipment): Auswertung eines berührungslos wirkenden Sicherheitssensors (Anlaufsperre)
- SF_GuardMonitoring: Plausibilitätsüberwachung zweier Sicherheitstürschalter (Anlaufsperre)
- SF_TwoHandControlTypeII: Plausibilitätsüberwachung einer Zweihandbedienung Typ II nach EN 574 (ohne zeitliche Überwachung der 2 Eingangssignale).
- SF_TwoHandControlTypeIII: Plausibilitätsüberwachung einer Zweihandbedienung Typ III nach EN 574 (mit zeitlicher Überwachung der 2 Eingangssignale von fest vorgegebenen 500 Millisekunden)
- SF_GuardLocking (Safety Guard Interlocking with Locking): Schutztürüberwachung mit Zuhaltung (Anlaufsperre)
- SF_TestableSafetySensor: Baustein zur Überprüfung von berührungslos wirkenden Schutzeinrichtungen Typ 2 mit periodischen Tests
- SF_MutingSeq: Baustein zur temporären Unterdrückung der Schutzfunktion, um Material in die mit einem ESPE abgesicherte Gefahrenzone oder aus der Gefahrenzone zu transportieren. Anordnung mit 4 Sensoren mit Signalsequenz mit serieller Reihenfolge in einer vorgegebenen Reihenfolge.
- SF_MutingPar: Baustein zur temporären Unterdrückung der Schutzfunktion, um Material in die mit einem ESPE abgesicherte Gefahrenzone oder aus der Gefahrenzone zu transportieren. Anordnung mit 2 Sensor-Paaren in einer vorgegebenen Reihenfolge.
- SF_MutingPar_2Sensor: Baustein zur temporären Unterdrückung der Schutzfunktion, um Material in die mit einem ESPE abgesicherte Gefahrenzone oder aus der Gefahrenzone zu transportieren. Anordnung der 2 Sensoren, dass sich ihre Strahlen kreuzen.
- SF_EnableSwitch: Plausibilitätsüberwachung eines 3-stufigen Zustimmtasters (Anlaufsperre)
- SF_SafetyRequest: Baustein zur Plausibilitätsüberwachung der Sicherheitsfunktion eines generischen Aktors, wie z. B.. Antrieb oder Ventil.
- SF_OutControl: Zustimm-UND. Verknüpfung eines Prozesssignals mit Safety-Signal (Anlaufsperre)
- SF_EDM (External Device Monitoring): Überwachung von externen angeschlossenen Relais/Schützen mit zwangsgeführten Kontakten zur Kontrolle deren Schaltfunktion

Die detaillierte Beschreibung der Funktionsbausteine finden Sie in der Onlinehilfe.

6.5.4 Anbindung analoger Eingangsmodule

Die Verarbeitung analoger Eingangssignale erfordert eine Extended-Level POU. Bei Problemen in einem analogen Eingangsmodul oder einem Problem in der Kommunikation mit ihm setzt die zuständige Treiberinstanz `D` (siehe auch *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*) die entsprechenden analogen Eingangssignale in der Applikation auf den Wert 0 und einen entsprechenden Diagnoseausgang auf FALSE (bei FSoEMaster und NetVar-Receiver: `D.S_InReady`, bei PROFIsafeHost: `D.FV_activated_S)`.

Regel P10 (analoges Failsafe)

HINWEIS!
Ein analoges Eingangssignal im Extended-Level muss entweder so verknüpft werden, dass der Wert 0 als gefährlicher Wert behandelt wird und zu einer angemessenen Sicherheitsreaktion führt (0 als Failsafe). Oder der Diagnoseausgang (`D.S_InReady` bzw. `D.FV_activated_S)`) der entsprechenden Treiberinstanz muss so in die logische Verknüpfung einbezogen werden, dass der Wert FALSE zu einer angemessenen Sicherheitsreaktion führt.

Bei digitalen Eingängen ist so eine Prüfung nicht nötig, da in der Sicherheitstechnik FALSE als Failsafe-Wert global definiert ist.

6.6 Querkommunikation mit Netzwerkvariablen

Safety NetVars erfüllt die Funktion der sicheren Querkommunikation in CODESYS Safety. Es erlaubt die Konfiguration und den Betrieb eines sicheren Datenaustauschs zwischen CODESYS Safety Sicherheitssteuerungen in einem Projekt.

Das Prinzip: Ein CODESYS Projekt enthält mehrere Standardsteuerungen mit eingehängten Sicherheitssteuerungen. Der sichere Datenaustausch erfolgt nach folgendem Prinzip: Eine Sicherheitssteuerung veröffentlicht Daten durch einen Sender (Objekt vom Typ *„Safety NVL (Sender)“*) und andere Sicherheitssteuerungen des Projekts können die Daten mit Hilfe eines Empfängers (Objekt vom Typ *„Safety NVL (Empfänger)“*) lesen. Der Kommunikationsweg erfolgt über die Standardsteuerung.

Objekte

Der Zugriff auf Safety Netzwerkvariablen erfolgt über die Objekte Safety Netzwerkvariablenliste (Sender) und Safety Netzwerkvariablenliste (Empfänger).

Die Safety Netzwerkvariablen können wie globale Variablen als VAR_EXTERNAL deklariert in der Sicherheitsapplikation verwendet werden. Aber Netzwerkvariablen sind keine normalen globalen Variablen. Auf beiden Seiten sind Besonderheiten zu berücksichtigen:

Auf Senderseite sind Netzwerkvariablen Ausgänge der Applikation. Das heißt, ihnen muss genau einmal im Zyklus ein Wert zugewiesen werden. Allerdings dürfen nicht beliebige Werte zugewiesen werden, sonst droht gefährliches Undersampling, siehe ↳ *Kapitel 6.6.1 „Sampling-Rate und Undersampling“ auf Seite 153*

Auf Empfängerseite sind Netzwerkvariablen Eingänge der Applikation. Das heißt, sie dürfen gelesen, aber nicht beschrieben werden. Dabei muss allerdings berücksichtigt werden, welche Undersampling-Maßnahmen auf Senderseite ergriffen wurden und gegebenenfalls muss eine Behandlung für den Undersampling-Fall definiert werden, siehe ↳ *Kapitel 6.6.1 „Sampling-Rate und Undersampling“ auf Seite 153*

Wenn die Werte erfolgreich übertragen werden, enthalten die Netzwerkvariablen auf Empfängerseite die Werte der verbundenen Netzwerkvariablen auf Senderseite.

Netzwerkvariablen verwenden standardmäßig UDP Broadcasts im Maschinennetzwerk. Um die Netzlast zu reduzieren, können die IP-Adressen der gegenüberliegenden Standardsteuerung fix eingestellt werden (siehe Onlinehilfe, Kapitel "Objekt Safety Netzwerkvariablenliste (Sender)" und " Objekt Safety Netzwerkvariablenliste (Empfänger)", Abschnitt "Registerkarte SPS Netzwerk").

HINWEIS!

Wenn Netzwerkvariablen auf Senderseite veröffentlicht werden, hat jeder Bearbeiter des Projekts automatisch das Recht, die Verwendung dieser Variablen in der Empfängerapplikation zu konfigurieren, wenn er allgemein das Benutzerrecht hat, die Verwendung von Variablen zu konfigurieren.

HINWEIS!

Eine Empfängerapplikation kann mehrere Empfänger-NVLs haben, die mehrere Sender-NVLs der gleichen Senderapplikation abonnieren. Dann ist weder garantiert,

- dass die Werte aus verschiedenen NVLs aus dem gleichen Zyklus der Senderapplikation stammen (Zykluskonsistenz), noch
- dass bei erfolgreicher Übertragung der Werte einer NVL auch die Werte der anderen NVL übertragen werden (Kommunikationskonsistenz)

Wenn mehrere Empfänger-NVLs (in verschiedenen Empfänger-Applikationen) die gleiche Sender-NVL abonnieren, ist weder garantiert, dass jeder von ihnen die Werte aus dem gleichen Zyklus der Senderapplikation erhält (Zykluskonsistenz), noch, dass bei erfolgreicher Übertragung der Werte der Sender-NVL an eine Empfänger-NVL auch die anderen Empfänger-NVLs die Werte empfangen haben (Kommunikationskonsistenz).

Um eine Sender-Steuerung bei zukünftigen Erweiterungen der Maschine mit weiteren Empfängersteuerungen nicht ändern und aus diesem Grund neu abnehmen lassen zu müssen, kann das Empfängerlimit vorsorglich auf die Empfängeranzahl N im Maximalausbau (höher als die aktuelle Anzahl Empfänger in der Maschine) eingestellt werden und der Anwender kann sich im Vorhinein überzeugen (z.B. ausprobieren), dass die Senderapplikation auch im Maximalausbau mit N laufenden Variablenverbindungen zu N Empfängern ihre Zykluszeit nicht überschreitet.

Überwachung der Kommunikation

Bei Kommunikationsproblemen enthalten die Netzwerkvariablen auf Empfängerseite den Ersatzwert 0 bzw. FALSE.

Die Empfängerapplikation kann über die Treiberinstanz ‹NVL-Name› des NVL-Treiberbausteins "NetVarReceiver" den Status der Verbindung kontrollieren und Kommunikationsfehler bestätigen.

Die Senderapplikation kann über die Treiberinstanz ‹NVL-Name› des NVL-Treiberbausteins "NetVarSender" die Sendebereitschaft steuern bzw. Failsafe Ersatzwerte aktivieren.

Eine Übersicht zu den Treiberbausteinen finden Sie in ↳ *Kapitel 14.4.1 „Bibliothek SafetyNetVar“ auf Seite 288* und Allgemeines zu Treiberbausteinen finden Sie in ↳ *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*.

6.6.1 Sampling-Rate und Undersampling

Gefahr des Undersampling

Die Werte der Safety Netzwerkvariablen und Empfangsbestätigungen werden immer synchron am Ende eines Applikationszyklus (Ausgangsphase) verschickt und am Beginn eines Applikationszyklus (Eingangsphase) empfangen. Die Zykluszeiten der Applikationen von Sender und Empfänger und die Übertragungszeit beeinflussen die Sampling-Rate der Netzwerkvariablenwerte.

Wenn eine Netzwerkvariable im Sender kurzzeitig von Wert A auf Wert D wechselt und wieder zurück auf A, kann es sein, dass der Wert D nie an den Empfänger übertragen wird (Undersampling). In dem Fall, dass der Wert D dieser Variable im aktuellen Zustand der Applikation eine Situation darstellt, die die Auslösung einer Sicherheitsfunktion bedarf, kann es zu einer Gefährdung kommen.

Regel P11 (NvlSend)

VORSICHT!

Um zu vermeiden, dass zu kurze Signale nicht erkannt oder falsch übertragen werden, dürfen die Signale der Safety Netzwerkvariablen bei der Übertragung nicht zu kurz sein. Der Anwender muss auf Senderseite (analog zu einem Sensor bzw. Eingangsmodul) sicherstellen, dass ein gefahranzeigendes Signal D bzw. ein Signal D, das eine sichere Reaktion anfordert (Demand), im Sender

- entweder so lange ansteht, wie die Gefahr besteht bzw. die Reaktion erforderlich ist
- oder mindestens so lange ansteht, wie die größte Watchdog-Zeit aller Verbindungen zu diesem Signal, d.h., dass es lange genug ansteht, um von allen Empfänger-Steuerungen noch rechtzeitig verarbeitet zu werden.

Maßnahmen zur Vermeidung von Undersampling und Maßnahmen zur Erkennung von Undersampling sind weiter unten beschreiben.

- Wenn die Applikationen der Safety Netzwerkvariablenliste (Sender) und der Safety Netzwerkvariablenliste (Empfänger) die gleichen Zykluszeiten haben und die Übertragungszeit kleiner als die halbe Zykluszeit ist, muss eine Werteänderung in der Safety Netzwerkvariablenliste (Sender) 4 Zyklen stehen bleiben, damit sie garantiert in der Safety Netzwerkvariablenliste (Empfänger) ankommt.

  Sampling-Rate: 1 Werteänderung pro 4 Zyklen des NVL-Senders
- Wenn der Safety Netzwerkvariablenlisten (Empfänger) eine relativ kleine Zykluszeit hat, darf eine erneute Werteänderung in der Safety Netzwerkvariablenliste (Sender) nicht sofort im nächsten Zyklus vorgenommen werden. Eine Werteänderung muss mindestens 2 Zyklen im NVL-Sender stehen bleiben.

  Dabei ist die bestmögliche Sampling-Rate: 1 Werteänderung pro 2 Zyklen des NVL-Senders. Diese bestmögliche Sampling-Rate wird erreicht, wenn die Zykluszeit des NVL-Senders größer ist als (2 x Zykluszeit NVL-Empfängers + 2 x Übertragungszeit).

Vermeidung von gefährlichem Undersampling

Der Sender hat es in der Hand, gefährliches Undersampling durch die Belegung von Netzwerkvariablen mit Signalen mit den "richtigen" Eigenschaften zu vermeiden.

Die folgenden Programmierregeln sind ein Beispiel einer Maßnahme zur Undersampling-Vermeidung:

Regeln für SAFEBOOL Netzwerkvariablen

- Regel "Wiederanlaufsperre": Veröffentlichen Sie Netzwerkvariablen, die auf FALSE (nach Failsafe-Prinzip ein gefahranzeigendes oder reaktionsanforderndes Signal) stehen bleiben, bis der Operateur durch eine Signalflanke die Abwesenheit der Gefahr und die Rücknahme der Sicherheitsreaktion, bzw. das Wiederanlaufen durch eine Signalflanke bestätigt hat.
- Regel "Verriegelter Sensor": Veröffentlichen Sie Variablen, die mit dem Eingangssignal von einem Sensor mit mechanischer Verriegelung, z.B. ein Nothalt-Befehlsgerät, belegt sind. Denn mit dem Rücksetzen der Verriegelung bestätigt der Operateur das Ende der Gefahr.
- Regel "Physikalische Mindestdauer": Veröffentlichen Sie Variablen, die eine Prozesswert darstellen (z.B. Schutztür geöffnet), wenn dieser physikalisch bedingt erst ab einer gewissen Mindestdauer (Zeit zum Passieren der Schutztür) eine gefährliche Situation darstellt, und wenn diese Mindestdauer länger als die Watchdog-Zeit der Verbindung zum Sensor plus der längsten Watchdog-Zeit aller Empfänger ist.

Beispielregeln für SAFEINT Netzwerkvariablen

- Regel "Entscheidung im Sender": Veröffentlichen Sie analoge Prozesswerte (z.B. Geschwindigkeit, Position) nicht zu dem Zweck, dass Empfänger die Werte überwachen und gegebenenfalls eine Sicherheitsfunktion auslösen. Vielmehr überwachen Sie diese analogen Prozesswerte selbst und melden Sie das Erreichen bestimmter Werte an die Empfänger. Das heißt, veröffentlichen Sie statt dessen digitale Netzwerkvariablen, von denen die Empfänger ihre Reaktion abhängig machen können. Bei diesen Variablen kann Undersampling mit den Regeln für SAFEBOOL Netzwerkvariablen vermieden werden.
- Regel "Extremwertermittlung im Sender": Veröffentlichen Sie zu überwachende analoge Prozesswerte (Geschwindigkeit, Position, Temperatur, etc.) nicht direkt. Bestimmen Sie vielmehr deren größten und ggf. kleinsten erreichten Werte und veröffentlichen Sie nur diese Werte. Empfänger können dann selbständig gegen Überschreitungen von Grenzen (nach oben oder unten) prüfen und Sicherheitsfunktionen auslösen, ohne Extremwerte wegen Undersampling zu verpassen. Sehen Sie den Neustart der Extremwertermittlung durch den Operateur nach einer Sicherheitsreaktion des Empfängers wegen Grenzüberschreitung vor. Dann kann der Empfänger mit aktuellen Prozesswerten neu beginnen.
- Regel "Nonreaktive Signale": Veröffentlichen Sie Werte, die Empfängern nicht zur Erkennung von Gefahren dienen, d.h. keiner ihrer Werte stellt ein gefahranzeigendes oder reaktionsanforderndes Signal dar. Es gibt also keinen bestimmten Wert der Netzwerkvariable, der im Empfänger eine Sicherheitsfunktion auslösen würde, und damit keinen Wert, dessen Überspringen (Undersampling) das Auslassen einer erforderlichen Reaktion bewirken würde.

Erkennung von Undersampling (Sender und Empfänger)

Wenn Undersampling nicht durch Vermeidungsmaßnahmen ausgeschlossen wird, dann sollten Maßnahmen zur Erkennung von Undersampling und die Reaktion darauf implementiert werden.

Damit der Empfänger den Verlust einer kurzzeitigen Signaländerung erkennen kann, muss auf Senderseite entsprechende Vorarbeit geleistet werden.

Eine mögliche Umsetzung sieht wie folgt aus:

Senderseite

- Jeder Sender-NVL Nk mit Netzwerkvariablen `SO_xk` eine Variable `SO_changecount_k` vom Datentyp `SAFEINT` hinzufügen.
- Für jede Sender_NVL Nk ein Zählprogramm Pk anlgegen mit lokalen Merkervariablen `old_xi` für jede Netzwerkvariable `SO_xi`.
- Im Zählprogramm Pk durch Vergleich `SO_xi` gegen `old_xi` feststellen, ob sich der Wert irgendeiner Variablen geändert hat.
- Wenn sich der Wert einer Variablen geändert hat, `SO_changecount_k` um 1 erhöhen.

- Die Werte von `SO_xi` in `old_xi` merken.
- Pk in die Taskliste einfügen.

Empfängerseite

- Erkennung: Auf der Empfängerseite wird der gesendete aktuelle Zähler `S_changecount` der Safety Netzwerkvariablenliste (Empfänger) mit dem Zähler verglichen, der 1 Zyklus früher gesendet wurde. Wenn die beiden Zähler gleich sind, oder der aktuell gesendete um 1 höher ist, bedeutet dies, dass kein Signal verloren gegangen ist.
- Reaktion: Wenn der Zähler um mehr als 1 höher ist, ist ein Signal verloren gegangen. Die Reaktion könnte darin bestehen, alle empfangenen Netzwerkvariablen auf Failsafe-Werte zu setzen. Da die Applikation aber nicht die Variablen in der Empfänger-NVL überschreiben darf, müsste man für so eine Reaktion die Netzwerkvariablen in der NVL in `SI_raw_xi` umbenennen, eine GVL mit globalen Variablen (oder im gleichen Programm lokale Variablen) mit Namen `SI_xi` anlegen. Im Gutfall werden die Werte `SI_raw_xi` nach `SI_xi` kopiert, im Undersampling-Fall werden Failsafe-Werte in `SI_xi` kopiert.

6.7 Taskkonfiguration

Die Aufrufreihenfolge der Programme (POU-Typ *„PROGRAM“*) und die Zykluszeit werden im Task-Objekt, das in der Sicherheitsapplikation vorhanden sein muss, festgelegt. Die Programmliste, die mindestens einen Aufrufeintrag enthalten muss, wird immer nach Ablauf der Zykluszeit von Beginn ab verarbeitet. Die Überschreitung der Zykluszeit bei Abarbeitung der Programmliste führt zu einem Laufzeitfehler. Genauere Informationen zum Task-Objekt siehe *Kapitel 5.5.4.4 „Safety Task“ auf Seite 89*

6.8 Beispiele

6.8.1 Programmierbeispiel für Basic Level

Als CODESYS Safety Programmierbeispiel für ein Programm im Basic Level wird das Beispiel "2-Hand-Kontrolle mit EDM" des Dokuments "PLCopen - Technical Committee 5 Safety Software Technical Specification Part 2: User Examples Version 1.01 – Official Release" verwendet und als CODESYS Safety FUP-Implementierung dargestellt.

Für weitere Beispiele zur Programmierung im Basic-Level wird auf das Dokument "PLCopen - Technical Committee 5 Safety Software Technical Specification Part 2: User Examples Version 1.01 – Official Release" verwiesen.

Funktionale Beschreibung der Sicherheitsfunktionen

Folgende Sicherheitsfunktionen werden in diesem Beispiel verwendet:

- Bei Betätigen des Not-HaltTasters müssen alle gefahrbringenden Bewegungen gestoppt werden (über SF_EmergencyStop)

  Not-Halt hat die höchste Priorität. Nach Lösen des EStop-Druckknopfs wird ein Reset über S0-Reset benötigt.
- Der Sicherheitsausgang wird durch Drücken beider Drucktaster der Zwei-Hand-Kontrolle aktiviert. Das Lösen irgendeines der Zwei-Hand-Drucktaster deaktiviert den Sicherheits-Ausgang und stoppt die gefahrbringende Bewegung über die Schaltvorrichtungen K1 und K2 (über SF_TwoHandControlTypeII)
- Der Grundzustand und der Betriebszustand der verbundenen Schaltvorrichtungen werden überwacht. Falls ein Fehler entdeckt wird, kann der Sicherheitsausgang nicht funktionsbereit werden. (über SF_EDM)
- Nach Anschalten der Sicherheits- oder der funktionalen Anwendung, oder nach einer Not-Halt Bedingung muss die Zwei-Hand-Kontrolle gelöst und wieder betätigt werden, um den Sicherheits-Ausgang wieder zu aktivieren (über SF_OutControl). Um dies für den funktionalen Neustart zu gewährleisten, ist das Prozess-Signal der funktionalen Applikation mit dem Activate-Eingang des Zwei-Hand-Kontrolle Bausteins THC_S2_S3 verbunden. (Wenn die Anwendungs-Prozess wiedergestartet wird, während die Zwei-Hand-Kontrolle aktiviert ist, geht der Baustein in Status C0003, der den Fehler signalisiert, dass beide Taster bei Aktivierung gedrückt sind und einen Neustart verhindert.

In diesem Beispiel existiert nur ein Betriebszustand.

In Work

PROGRAM Programm (* Basic Level *)

Zeile	Gültigkeitsbereich	Name	Typ	Initialwert	Kommentar
1	VAR	EStop_S1	SF_EmergencyStop		Instanz des Bausteins SF_EmergencyStop
2	VAR	S1_S_EStopIn	SAFEBOOL	FALSE	Eingang, Not-Aus-Taster S1
3	VAR	S0_Reset	BOOL	FALSE	Eingang, Reset durch Benutzerschalter (abgeleitet aus der funktionalen Anwendung)
4	VAR	S_EStopOut	SAFEBOOL	FALSE	Ausgang von EStop_S1
5	VAR	THC_S2_S3	SF_TwoHan...ntrolTypeII		Instanz des Bausteins SF_TwoHandControlTypeII
6	VAR	Process	BOOL	FALSE	Eingang, FreigabeBewegung durch den Prozess (abgeleitet von der funktionlalen Anwendung)
7	VAR	S2_S_Switch1	SAFEBOOL	FALSE	Eingang, Schalter2 verbunden mit Druckknopf1 der 2-Hand-Kontrolle
8	VAR	S3_S_Switch2	SAFEBOOL	FALSE	Eingang, Schalter3 verbunden mit Druckknopf2 der 2-Hand-Kontrolle
9	VAR	S_TwoHandOut	SAFEBOOL	FALSE	Ausgang von THC_S2_S3
10	VAR	OC_K1_K2	SF_OutControl		Instanz des Bausteins EDM_K1_K2
11	VAR	EDM_K1_K2	SF_EDM		Instanz des Bausteins SF_EDM
12	VAR	K1_S_EDM1	SAFEBOOL	FALSE	Eingang, Rückmeldung des Standardgeräts K1
13	VAR	K2_S_EDM2	SAFEBOOL	FALSE	Eingang, Rückmeldung des Standardgeräts K2
14	VAR	S_EDM_Out_EDM_K1_K2	SAFEBOOL	FALSE	Ausgang, steuert einen Aktor über K1 und K2 an
15	VAR	Error_EStop_S1	BOOL	FALSE	Fehlerflag von EStop_S1
16	VAR	Diag_EStop_S1	WORD	0	Diagnosecode für EStop_S1, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in EStop_S1
17	VAR	Error_THC_S2_S3	BOOL	FALSE	Fehlerflag von TCH_S2_S3
18	VAR	Diag_THC_S2_S3	WORD	0	Diagnosecode fpr THC_S2_S3, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in THC_S2_S3
19	VAR	Error_OC_K1_K2	BOOL	FALSE	Fehlerflag von OC_K1_K2
20	VAR	Diag_OC_K1_K2	WORD	0	Diagnosecode für OC_K1_K2, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in OC_K1_K2
21	VAR	Error_EDM_K1_K2	BOOL	FALSE	Fehlerflag von EDM_K1_K2
22	VAR	Diag_EDM_K1_K2	WORD	0	Diagnosecode für EDM_K1_K2, 16#8xxx: regulärer Betrieb, 16#Cxxx bei Fehler in EDM_K1_K2

Abb. 80: Variablendeklaration für Programmierbeispiel: Zwei-Hand-Kontrolle mit EDM

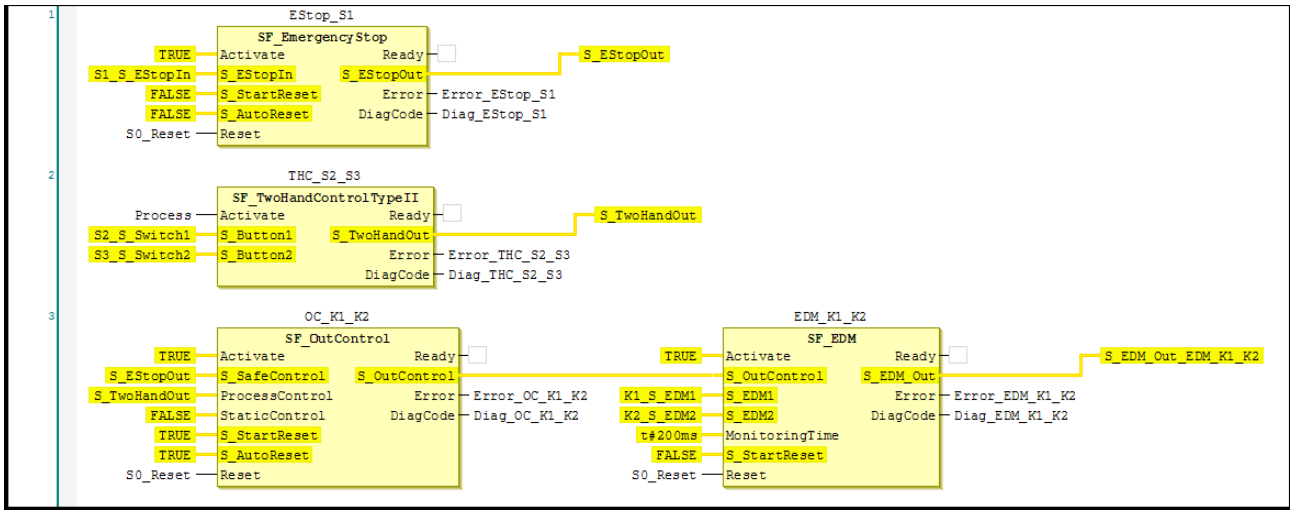


Abb. 81: Implementierung für Programmierbeispiel: Zwei-Hand-Kontrolle mit EDM

Tab. 12: Eingänge:

Name	Datentyp	Beschreibung
S1_S_EStopIn	SAFEBOOL	Not-Halt-Taster S1
S2_S_Switch1	SAFEBOOL	Schalter S2 verbunden mit Druckknopf 1 der Zwei-Hand-Kontrolle
S3_S_Switch2	SAFEBOOL	Schalter S3 verbunden mit Druckknopf 2 der Zwei-Hand-Kontrolle
K1_S_EDM1	SAFEBOOL	Rückmeldung externes Gerät K1
K2_S_EDM2	SAFEBOOL	Rückmeldung externes Gerät K2
S0_Reset	BOOL	Reset durch Entwickler mit Schalter S0 (abgeleitet von der funktionalen Anwendung)
Process	BOOL	Freigabe der Bewegung durch den Prozess (abgeleitet von der funktionalen Anwendung)

Tab. 13: Ausgänge:

Name	Datentyp	Beschreibung
S_EDM_Out_EDM_K1_K2	SAFEBOOL	steuert den Aktor über K1 und K2 an
Error_EStop_S1	BOOL	Fehlerflag von EStop_S1
Error_THC_S2_S3	BOOL	Fehlerflag von TCH_S2_S3
Error_OC_K1_K2	BOOL	Fehlerflag von OC_K1_K2

Name	Datentyp	Beschreibung
Diag_EStop_S1	WORD	Diagnosecode für EStop_S1, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in EStop_S1
Diag_THC_S2_S3	WORD	Diagnosecode für THC_S2_S3, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in THC_S2_S3
Diag_OC_K1_K2	WORD	Diagnosecode für OC_K1_K2, 16#8xxx: Regulärer Betrieb, 16#Cxxx bei Fehler in OC_K1_K2

Zusätzliche Anmerkungen

Dieses Beispiel kann auch mit SF_TwoHandControlTypeIII verwendet werden.

Der Eingang von *„Activate“* wurde der Einfachheit halber auf TRUE gesetzt. Dies kann in der Anwendung durch eine Variable ersetzt werden.

Tab. 14: Informationen zu verwendeten Bausteinparametern

Baustein	Eingang	Konstanter Wert	Beschreibung
EStop_S1	S_StartReset	FALSE	Kein automatischer Reset, wenn die S-SPS gestartet ist.
	S_AutoReset	FALSE	Kein automatischer Reset, Reset/Bestätigung durch Entwickler notwendig
OC_K1_K2	S_StartReset	TRUE	Automatischer Reset ist erlaubt, wenn die S-SPS gestartet ist.
	S_AutoReset	TRUE	Automatischer Reset, kein Reset/Bestätigung durch Entwickler notwendig
	Static Control	FALSE	Eine dynamische Änderung des Signals Appl_Control (steigende Flanke) wird gefordert nach Bausteinaktivierung oder eine getriggerte Sicherheitsfunktion (S_SafeControl auf FALSE)
EDM_K1_K2	S_StartReset	FALSE	Kein automatischer Reset, wenn die S-SPS gestartet ist
	MonitoringTime	T#200ms	Die maximale Antwortzeit der beiden Rückmeldesignale S_EDM1 und S_EDM2

7 Applikationserzeugung und Onlinebetrieb

7.1 Einführung

Analog zu CODESYS Standard unterstützt auch CODESYS Safety einen Online-Modus.

Die Funktionalitäten des Online-Modus dienen dem Debuggen und der Diagnose bei der Entwicklung und der Verifikation einer Sicherheitsapplikation.

Die Online-Befehle können auf dem selektierten Geräteobjekt, den Geräte-Editoren und auf dem selektierten bzw. aktiven Applikationsobjekt ausgeführt werden. Die Online-Funktionalität von CODESYS Safety unterscheidet sich nicht grundsätzlich von der Online-Funktionalität von Standard CODESYS. In diesem Kapitel werden die Besonderheiten der Online-Funktionalität von CODESYS Safety beschrieben.

Im ↳ *„Glossar“ auf Seite 313* finden Sie unter anderem die Begriffserklärungen zu den Begriffen

- Online
- Offline
- sicherer Betrieb
- Debug-Betrieb (unsicherer Betrieb)
- Downloadapplikation
- Bootapplikation
- bestätigte Verbindung
- Fernzugang

GEFAHR!

Der Entwickler ist dafür verantwortlich, dass die Sicherheit der Anlage/Maschine während des ganzen Zeitraums, in dem sich die Sicherheitssteuerung im Debug-Betrieb befindet, gewährleistet ist, z.B. durch Absperrungen um die Maschine (organisatorische Maßnahme).

HINWEIS!

Netzwerke, in der Applikationen und Maschinen mit Querkommunikation entwickelt werden, müssen von Netzwerken des Betriebs physikalisch getrennt sein, um eine Beeinflussung zuverlässig ausschießen zu können.

Damit die CODESYS Online-Funktionen und Eingabehilfen für die Sicherheitsapplikation funktionieren, müssen sie neben dem Safety-Sprachumfang auch der Standard-Compilerversion genügen. Wenn im Projekt eine neuere Compilerversion verwendet wird, können sich daraus zusätzliche Einschränkungen der Sicherheitsapplikation ergeben. Zum Beispiel kann es neue Schlüsselwörter geben, die dann nicht mehr als Bezeichner verwendet werden können.

Eine Verletzung solcher zusätzlicher Einschränkungen aus der Compilerversion erkennen Sie nicht beim Befehl „Erstellen ➔ Übersetzen“, sondern erst beim Einloggen. Es erscheint dann eine entsprechende Meldung und ein Einloggen ist nicht möglich. Einstellung der Compilerversion siehe Projektumgebung in der Standard Online-Hilfe

7.2 Verbindung zur Sicherheitssteuerung

Voraussetzungen für die Verbindung zur Sicherheitssteuerung

- Für den vollen Zugriff sind die Voraussetzungen eine Netzwerkverbindung und die Verbindungsbestätigung

  Vor jeder Verbindung zu einer Sicherheitssteuerung für den vollen Zugriff wird diese mittels einer eigenen Verbindungs-Id auf Konsistenz zwischen dem CODESYS Geräteobjekt und der Sicherheitssteuerung geprüft. Ist die Verbindungs-Id noch nicht vorhanden oder inkonsistent, so muss der Anwender die "neue" Verbindung bestätigen.

  Mit der Verbindungsbestätigung bestätigt der Anwender, dass die Netzwerkverbindung ihn mit der richtigen Steuerung verbunden hat.
- Für den Fernzugriff sind die Voraussetzungen eine Netzwerkverbindung, das Fernpasswort und die Freischaltung des Fernzugriffs. Fernzugang hat seinen Anwendungsfall im Betrieb und wird in *Kapitel 12 „Betrieb“ auf Seite 237* beschrieben (siehe *Kapitel 12.3.1 „Verbindung zur Sicherheitssteuerung für Fernzugang“ auf Seite 246*).

Für Entwicklung, Erzeugung und Debuggen von Sicherheitsapplikationen wird volle Online-Funktionalität zwischen CODESYS Safety und einer Sicherheitssteuerung benötigt. Diese erfordert eine bestätigte Verbindung zu der vom Entwickler bestimmten Sicherheitssteuerung innerhalb des CODESYS-Netzwerks.

HINWEIS!
Fernzugang ist für die Verifikation und die Abnahme nicht qualifiziert.

7.2.1 Kommunikationseinstellungen - allgemeine Informationen

Die Registerkarte *„Kommunikation“* enthält im rechten Teil die Daten und Informationen zu den aktuellen Einstellungen der Kommunikation zwischen dem Programmiersystem und der Sicherheitssteuerung. Diese Registerkarte entspricht CODESYS Standard. Für weitere Informationen wird auf die Online-Hilfe von CODESYS Standard verwiesen.

HINWEIS!
CODESYS bietet zwei alternative Ansichten der Kommunikationseinstellungen. Die neue grafische Ansicht ist nicht für die Verwendung mit CODESYS Safety freigegeben. Sie müssen vor Öffnen der Registerkarte *„Kommunikation“* im Dialog *„Tools ➔ Optionen ➔ Geräteeditor“* die Option *„Klassische Darstellung der Kommunikationseinstellungen verwenden “* aktivieren.

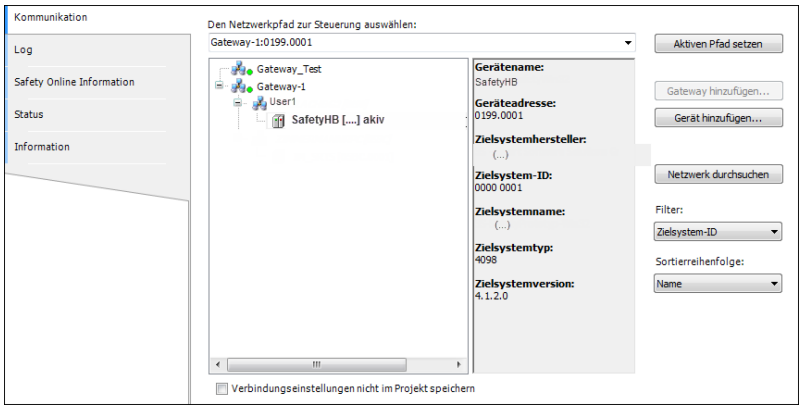


Abb. 82: Registerkarte 'Kommunikation' mit Kommunikationsbaum "Gateway-1"

7.2.2 Verbindungsaufbau

Netzwerkverbindung für die bestätigte Verbindung

1. In der Registerkarte *„Kommunikation“* der Sicherheitssteuerung den aktiven Pfad auf das gewünschte Gerät (Gerätename) setzen, siehe *„Netzwerkverbindung für die bestätigte Verbindung“ auf Seite 164* (Vorgehensweise wie in Standard CODESYS, für Details wird auf die Online Hilfe von CODESYS Standard verwiesen.)
2. Befehl *„Einloggen“* des Menüs *„Online“* aktivieren
3. Im erscheinenden Dialog *„Verbindung zur Sicherheitssteuerung“* die Verbindungsart *„Bestätigte Verbindung“* auswählen.
4. Die im Dialog beschriebene Aktion an der Sicherheitssteuerung durchführen (z.B. Knopf drücken)
5. Schaltfläche *„OK“* aktivieren
6. Je nachdem, ob sich keine Applikation, eine unveränderte oder eine veränderte Bootapplikation auf der Sicherheitssteuerung befinden, erscheinen entsprechende Dialoge, die zu prüfen und ggf. zu bestätigen sind
7. Im Dialog *„Authorisierung“* das gesetzte Bootapplikationspasswort (BA Passwort) eingeben. Das Passwort bei einer neuen Steuerung ist systemspezifisch, z. B. leer)

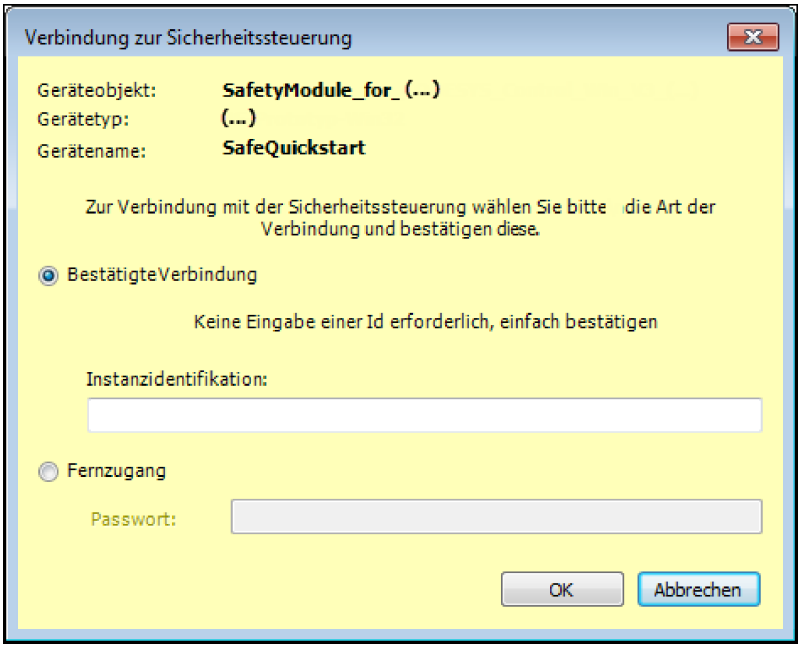


Abb. 83: Dialog 'Verbindung zur Sicherheitssteuerung'

Verbindungsbestätigung

Die Verbindungsbestätigung kann (systemspezifisch) mittels einer Aktion am Gerät oder per Eingabe einer eindeutigen Steuerungsinstanz-Id erfolgen. Die Art der auszuführenden Bestätigung wird von der jeweiligen Sicherheitssteuerung definiert.

Die Verbindung zur Sicherheitssteuerung erfolgt immer unter einem Benutzernamen. Ist der Entwickler nicht als Benutzer eingeloggt, so wird der aktuelle Benutzername des Betriebssystems (Windows) verwendet.

Varianten der Verbindungsbestätigung

- Aktion an der Sicherheitssteuerung
- Eingabe der Steuerungsinstanz-Id

Variante 1 : Aktion an der Sicherheitssteuerung

Zur Verbindung zur Steuerung muss direkt an der Steuerung eine Aktion vorgenommen werden, z.B. Betätigen eines Tasters. Der Entwickler wird in dem Dialog *„Verbindung zur Sicherheitssteuerung“* (*„Netzwerkverbindung für die bestätigte Verbindung“ auf Seite 164* zur Bestätigung der Verbindung durch eine Aktion an der Sicherheitssteuerung aufgefordert. Ohne die Aktion an der Sicherheitssteuerung kann keine Online-Verbindung zur Sicherheitssteuerung hergestellt werden.

Erfolgt die Bestätigung durch den Entwickler nicht innerhalb der systemspezifischen Wartezeit, so wird der entsprechende Dialog erneut geöffnet und der Entwickler kann die Aktion an der Sicherheitssteuerung wiederholen.

Variante 2: Steuerungsinstanz-Id

Die Verbindung zur Steuerung muss durch die Eingabe der Steuerungsinstanz-Id (*„Instanzidentifikation“*) dieser Steuerung bestätigt werden. Dazu wird der Entwickler in einem Dialog aufgefordert. Wird die Identifikation von der Steuerung abgelehnt, so wird der Dialog erneut geöffnet und der Entwickler kann die Engabe der Steuerungsinstanz-Id wiederholen.

Die Steuerungsinstanz-Id ist vom Hersteller eindeutig für die Steuerung vorgegeben (z.B. Seriennummer) und kann vom Anwender nicht geändert werden.

Eine erfolgreiche, bestätigte Verbindung wird nach folgenden Aktionen ungültig:

- Anwenderwechsel im CODESYS Projekt
- Kopieren des CODESYS Projekts auf einen anderen Rechner
- Wechsel der Steuerungs-Instanz (Knoten mit dem verbunden werden soll) innerhalb des CODESYS Projekts

HINWEIS!
Befinden sich in Projekten mit mehreren Sicherheitssteuerungen mehrere Sicherheitssteuerungen im Online-Modus, dann erscheint vor der Ausführung von Online-Befehlen ein Nachfrage-Dialog, auf welche Steuerung der Befehl gehen soll, so muss der Anwender den angezeigten Gerätenamen verifizieren, indem er den angezeigten Gerätenamen gegen die Erwartungshaltung prüft und im Fehlerfall den Online-Befehl abbricht.

7.2.3 Gerätename

Der Gerätename ist ein vom Entwickler vergebener Name für eine Steuerung in seinem Netzwerk. Durch ihn identifiziert sie die Steuerung bei Schreibbefehlen (siehe *Kapitel 7.5.2 „Debug-Modus und organisatorische Sicherheit“ auf Seite 176*) und in Info-Anzeigen (siehe *Kapitel 12.3.2 „Informationen zu Firmware und Bootapplikation “ auf Seite 247*)

Der Name der Steuerung (Gerätenamen) wird nicht im Projekt, sondern auf der Sicherheitssteuerung gespeichert.

GEFAHR!
Es müssen eindeutige Gerätenamen für die Sicherheitssteuerungen im Netzwerk vergeben werden.

Der Entwickler ist stets dafür verantwortlich, dass jeder Befehl an die richtige Steuerung geht.

Änderung des Gerätenamens

1. Registerkarte *„Kommunikation“* öffnen.
2. Gateway selektieren.
3. Schaltfläche *„Netzwerk durchsuchen“* aktivieren.
4. Knotenpunkt selektieren.
5. Kontextmenübefehl *„Gerätename ändern...“* aktivieren.
6. Im Dialog *„Verbindung zur Sicherheitssteuerung“* (siehe Abb. 84) einen neuen, eindeutigen Namen eingeben.
7. Die im Dialog *„Verbindung zur Sicherheitssteuerung“* beschriebene Aktion an der Sicherheitssteuerung durchführen, bzw. Steuerungsinstanz-Id (*„Instanzidentifikation“*) eingeben.
8. Schaltfläche *„OK“* aktivieren.

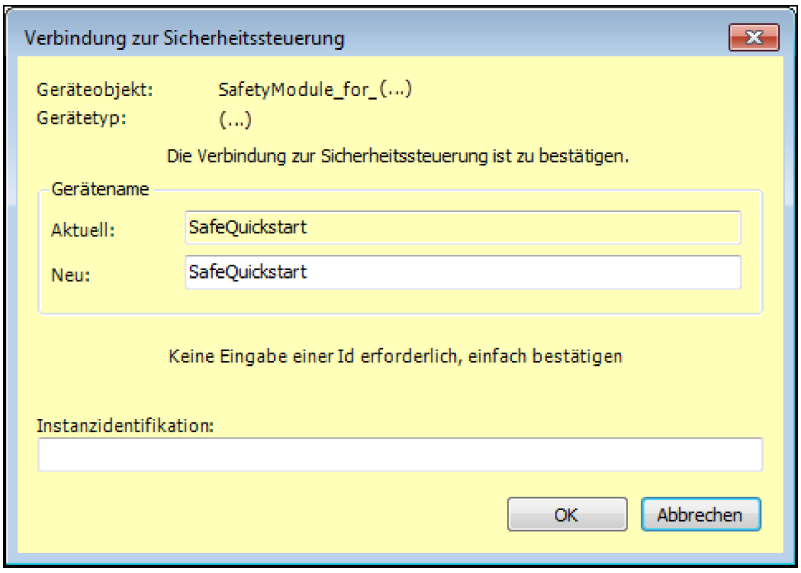


Abb. 84: Dialog zum Ändern des Gerätenamens

7.3 Einloggen auf der Steuerung und Wechsel in den Debug-Betrieb

Einloggen

Mit dem Befehl *„Einloggen“* des Online-Menüs wird eine Online-Verbindung zwischen der Applikation, für die der Befehl ausgeführt wird (aktive Applikation) und der Steuerung aufgebaut. Der Entwickler kann sich aus dem Projekt heraus nur auf die Applikation (Bootapplikation) auf der Steuerung einloggen, wenn die Applikation im Projekt und die Applikation auf der Steuerung übereinstimmen. Deshalb findet vor der Ausführung des Einlog-Vorgangs ein Vergleich zwischen der Applikation im Projekt und der, auf der Sicherheitssteuerung laufenden Applikation statt:

- Befindet sich keine Applikation auf der Steuerung, so wird der Entwickler in einem Dialog gefragt, ob die Applikation auf die Sicherheitssteuerung geladen werden soll (siehe Abb. 85). Wird die Applikation auf die Sicherheitssteuerung geladen (Download), so befindet sich die Sicherheitssteuerung im **Debug-Betrieb**. Die Applikation ist im Online-Modus.
- Haben die beiden Applikationen unterschiedliche Namen (d.h. es läuft eine andere Applikation auf der Sicherheitssteuerung), so wird in einem Dialog gefragt, ob die Applikation auf der Sicherheitssteuerung beendet und die aktuelle Applikation auf die Sicherheitssteuerung geladen werden soll . Wird diese aktuelle Applikation auf die Sicherheitssteuerng geladen (Download), so befindet sich die Sicherheitssteuerung im **Debug-Betrieb**. Die Applikation ist im Online-Modus.
- Wurde die aktuelle Applikation geändert, wird in einem Dialog gefragt, ob die Applikation auf der Sicherheitssteuerung beendet und die aktuelle Applikation auf die Sicherheitssteuerung geladen werden soll (siehe Abb. 86). Wird diese aktuelle Applikation auf die Sicherheitssteuerung geladen (Download), so befindet sich die Sicherheitssteuerung im **Debug-Betrieb**. Die Applikation ist im Online-Modus.

- Sind die aktuelle Applikation und die Applikation auf der Sicherheitssteuerung identisch und haben sie den gleichen Pin, erfolgt die Meldung, dass der Login auf eine gepinnte Applikation erfolgt. Die Sicherheitssteuerung befindet sich im **sicheren Betrieb**. Die Applikation ist im Online-Modus.
- Sind die aktuelle Applikation und die Applikation auf der Sicherheitssteuerung identisch und nicht gepinnt, wird ohne Dialog eingeloggt. Die Sicherheitssteuerung befindet sich im **sicheren Betrieb**. Die Applikation ist im Online-Modus.

In den Fällen, die zum Download der Applikation und in den Wechsel in den Debug-Betrieb führen, wird der Entwickler angewiesen, die Sicherheit organisatorisch herzustellen.

GEFAHR!
Der Entwickler ist dafür verantwortlich, dass die Sicherheit der Anlage/Maschine während des ganzen Zeitraums, in dem sich die Sicherheitssteuerung im Debug-Betrieb befindet, gewährleistet ist.

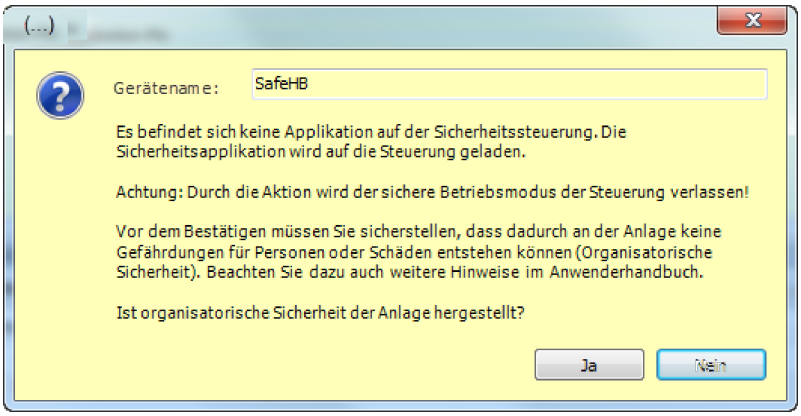


Abb. 85: Dialog beim Einloggen auf eine "leere" Steuerung

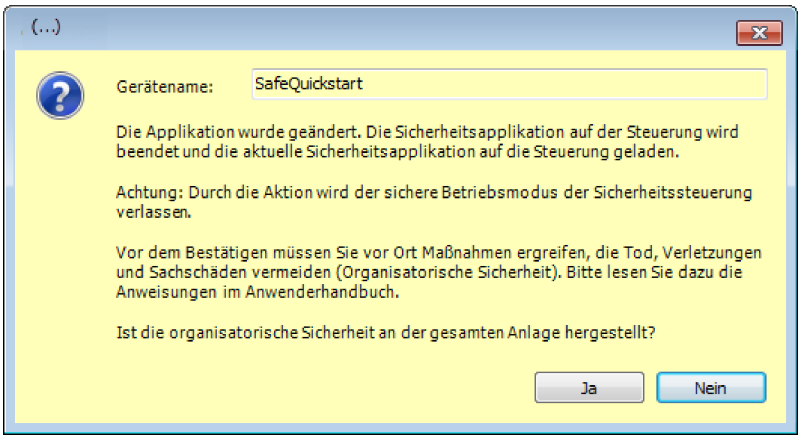


Abb. 86: Dialog beim Einloggen auf Steuerung mit gänderter Applikation

Applikationen mit gleichem Namen werden als unterschiedlich betrachtet, wenn die Liste der Objekte nicht identisch ist oder wenn wenigstens eine Prüfsumme der Objekte unterschiedlich ist oder wenn die Pin-Prüfsumme unterschiedlich ist. Siehe Safety Applikationsobjekt ↬ Kapitel 5.5.4.1 „Safety Applikationsobjekt“ auf Seite 61.

Download

Mit einem Download wird eine Sicherheitsapplikation temporär auf die Steuerung geladen. Ein Download erfolgt durch Aktivierung des Befehls *„Einloggen“* des Menüs *„Online“*. Damit die neue Sicherheitsapplikation wie konfiguriert mit Feldgeräten, Austauschvariablen und Netzwerkvariablen kommunizieren kann, muss auf ihrer Standardsteuerung ein dazu konsistenter Stand laufen (siehe ↬ *Kapitel 6.6 „Querkommunikation mit Netzwerkvariablen“ auf Seite 150*).

Vor dem Einloggen wird das Projekt auf Korrektheit überprüft. Ist das Projekt fehlerhaft, so ist ein Einloggen auf die Steuerung nicht möglich.

Solange sich eine temporär auf die Sicherheitssteuerung geladene Applikation auf der Sicherheitssteuerung befindet, ist die Sicherheitssteuerung im Debug-Betrieb!

Ausloggen

Der Online-Befehl *„Ausloggen“* beendet die bestehende Online-Verbindung zur Applikation. Es gibt folgende Varianten

- Der Entwickler ist auf die laufende Bootapplikation eingeloggt und diese ist im sicheren Betrieb:
  - die Online-Verbindung wird beendet
  - die Bootapplikation läuft weiter
- Der Entwickler ist auf die Applikation eingeloggt, die sich im Debug-Betrieb (Download-Applikation oder Bootapplikation) und auf der Sicherheitssteuerung ist eine Bootapplikation hinterlegt:
  - Applikation wird entladen
  - Der Entwickler wird in einem Dialog gefragt, ob die hinterlegte Bootapplikation geladen und gestartet werden soll (siehe Abb. 87).

    In diesem Fall wird die Debug-Freischaltung beendet, die Sicherheitssteuerung geht in den sicheren Betrieb.
- Der Entwickler ist auf die Applikation eingeloggt (siehe Abb. 88) und es befindet sich keine Bootapplikation auf der Sicherheitssteuerung:
  - Die Applikation wird entladen

Wird automatisch durch das LZS ausgeloggt (Fehlerfall), so wird die temporäre Downloadapplikation beendet und die Bootapplikation nicht automatisch gestartet.

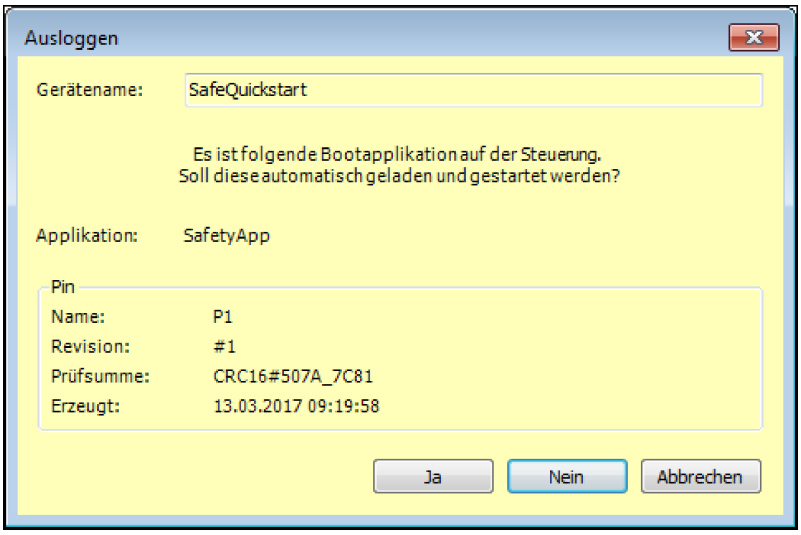


Abb. 87: Dialog: Ausloggen bei unterschiedlicher Bootapplikation und Downloadapplikation

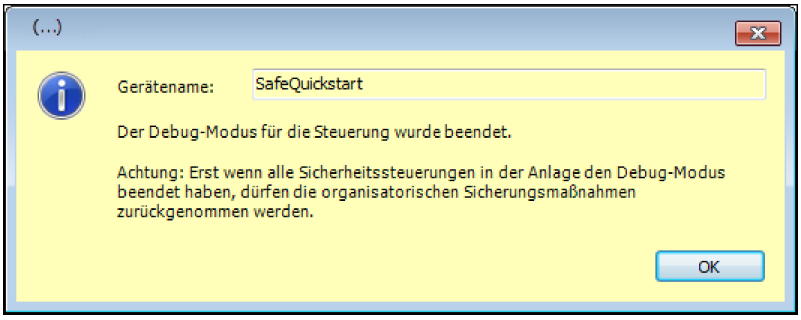


Abb. 88: Dialog beim Ausloggen mit Laden der Bootapplikation

7.4 Erzeugen und Neustart der Bootapplikation

Erzeugen der Bootapplikation

Eine laufende Applikation wird durch Aktivierung des Befehls *„Bootapplikation erzeugen“* des Menüs *„Online“* als Bootapplikation erzeugt und auf der Sicherheitssteuerung abgelegt. Der Befehl steht nur zur Verfügung, wenn der Entwickler auf der Sicherheitssteuerung eingeloggt ist.

Die Erzeugung einer Bootapplikation kann nur nach expliziter Bestätigung durch den Entwickler ausgeführt werden. In dem Bestätigungsdialog wird der Status der Applikation bzw. Pin-Kennung angezeigt.

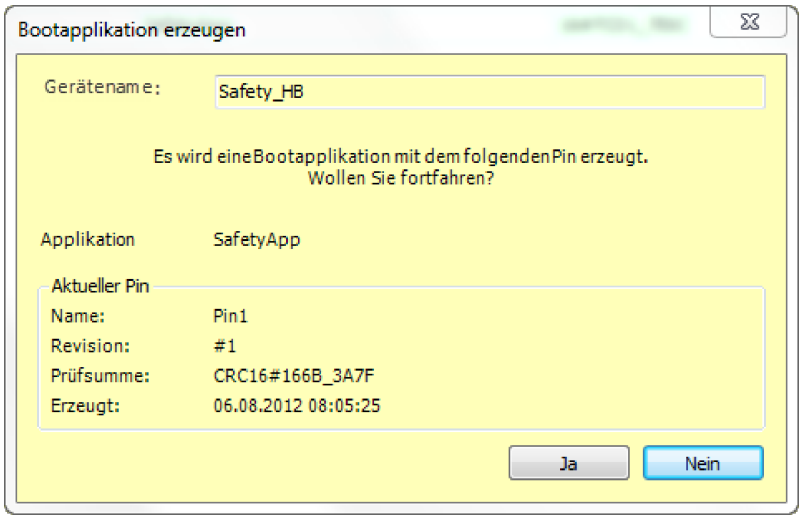


Abb. 89: Dialog: 'Bootapplikation erzeugen'

Mögliche Stati der Applikation:

- Applikation ist nicht konsistent mit dem Pin (In Work)
- Applikation ist konsistent mit dem Pin
- Applikation ist konsistent mit dem Pin, jedoch nicht mit dem Pin der zuletzt aus diesem Projekt erzeugten Bootapplikation (siehe Abb. 90)

Wird in diesen drei Fällen nach Bestätigung durch den Entwickler eine Bootapplikation erzeugt, so bleibt die Download-Applikation geladen und die Steuerung bleibt im Debug-Betrieb

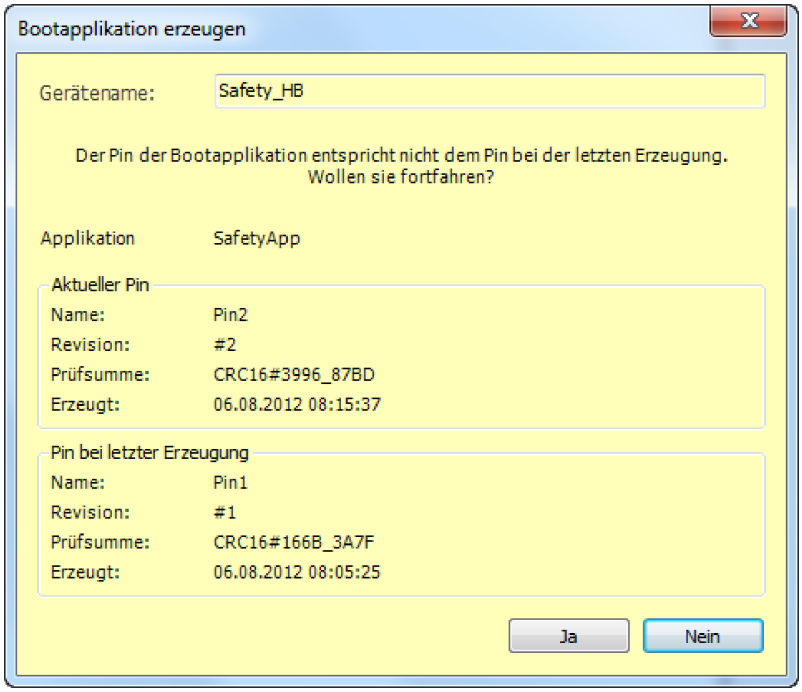


Abb. 90: Dialog: 'Bootapplikation erzeugen' bei geändertem Pin

GEFAHR!

Konnte der Befehl *„Bootapplikation erzeugen“* nicht erfolgreich ausgeführt werden, so kann es nach Neustart zu einer Gefährdung in der Anlage kommen, weil diese unter Umständen noch mit der alten Applikation anläuft.

Der Service-Mitarbeiter muss auf die Rückmeldung warten, ob der Befehl erfolgreich ausgeführt wurde. Bleibt diese aus, so ist die Steuerung so zu behandeln, als ob die Bootapplikation nach Neustart wieder aufstarten kann.

Neustart der Bootapplikation

Eine Bootapplikation auf der Steuerung wird durch Aktivierung des Befehls *„Neu starten“* (Registerkarte *„Safety Online Information“* der Steuerung) oder automatisch nach Einschalten der Steuerung neu gestartet.

Neustart bedeutet nicht, dass die Anlage losläuft. Der Entwickler legt in der Sicherheitsapplikation fest, ob die PLCopen Bausteine und die sicheren Ausgangsmodule automatisch anlaufen (Auto-Reset) oder erst durch ein Standardsignal (Reset).

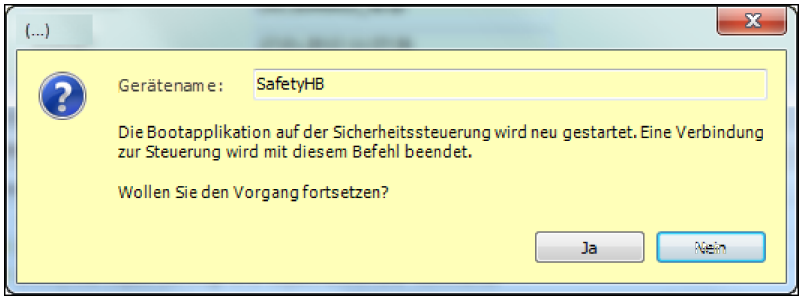


Abb. 91: Dialog beim Neustarten der Bootapplikation

7.5 Betriebsmodi

7.5.1 Betriebszustand und Applikationszustand

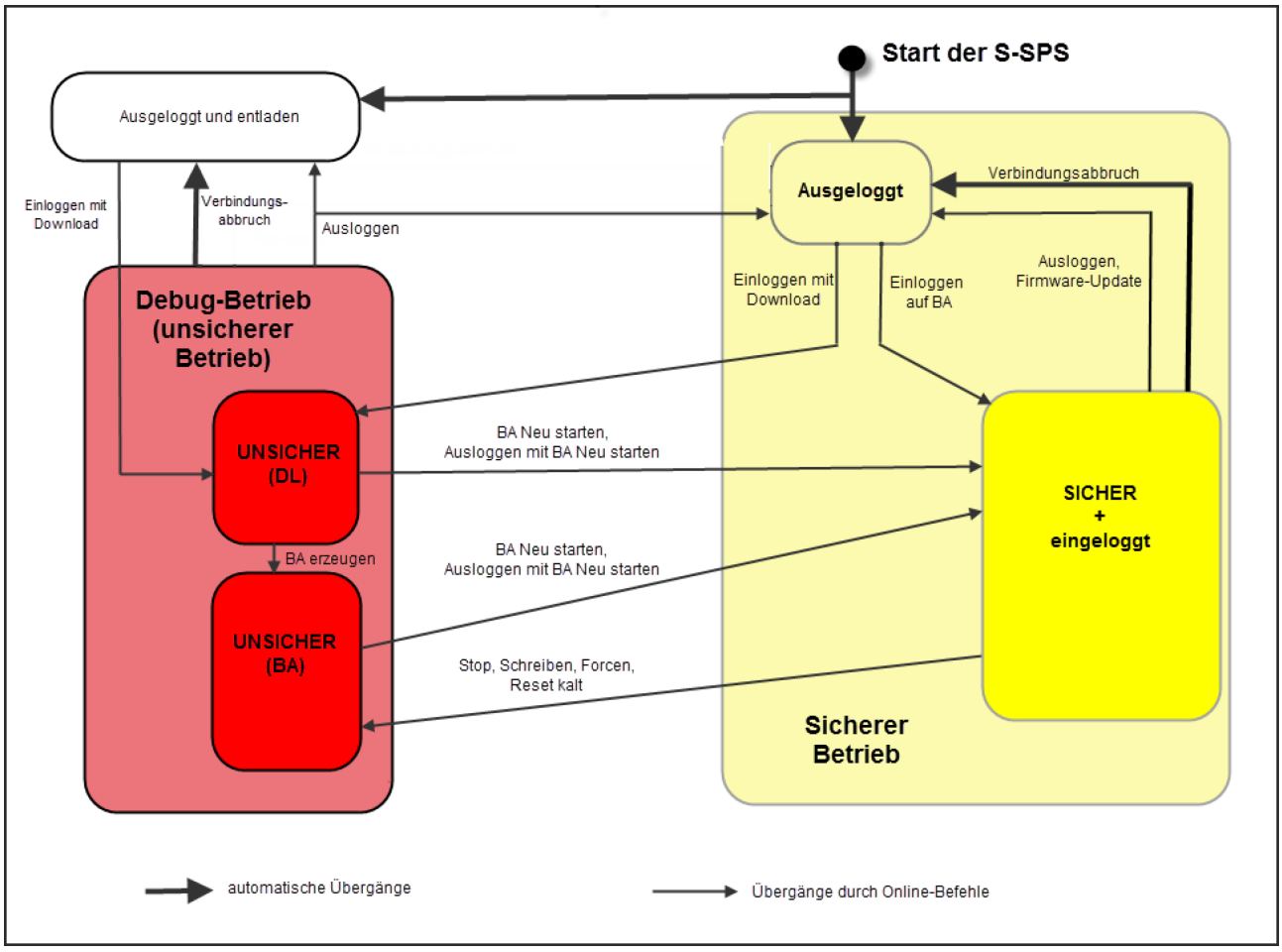


Abb. 92: Betriebsmodi der Sicherheitssteuerung

Betriebszustände der Sicherheits-steuerung: sicher und unsicher

Die beiden möglichen Betriebsmodi einer mit CODESYS Safety programmierten Sicherheitssteuerung sind **sicher** und **unsicher**.

Als sicherer Betrieb wird der Modus der Sicherheitssteuerung bezeichnet, in dem eine Bootapplikation geladen ist und die Steuerung nicht im Debug-Modus betrieben wird. Die Sicherheitssteuerung befindet sich so lange im sicheren Betrieb, wie die Bootapplikation läuft und der Entwickler nicht schreibend darauf zugreift. Sobald ein schreibender Zugriff stattfindet, wechselt die Steuerung in den Debug-Betrieb. Die Steuerung bleibt auch im sicheren Zustand, wenn ein Login auf die Steuerung erfolgt und Variablenwerte in CODESYS Safety angezeigt werden. Erst ein schreibender Dienst wie das Forcen eines Wertes veranlasst die Steuerung, in den Debug-Betrieb zu wechseln.

Der Zustand einer nicht geladenen Applikation ist zwar ebenfalls sicher, wird aber nicht als sicherer Betrieb bezeichnet

Befindet sich eine Bootapplikation auf der Sicherheitssteuerung, so läuft diese Bootapplikation mit dem Start der Steuerung an und die Steuerung befindet sich im sicheren Betriebszustand.

Wird die Steuerung dazu veranlasst vom sicheren in den unsicheren Zustand zu wechseln, so muss der Entwickler den Wechsel in den unsicheren Zustand bestätigen.

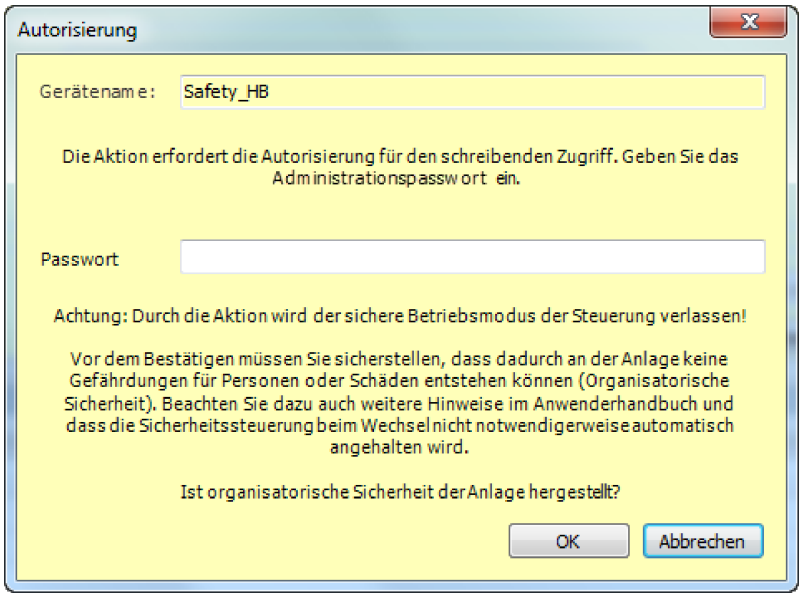


Abb. 93: Bsp. Dialog bei Wechsel in unsicheren Betrieb

Hinweise siehe ↳ *Kapitel 7.5.2 „Debug-Modus und organisatorische Sicherheit“ auf Seite 176*

Applikationszustand

Wird die Applikation mit einem Download auf die Steuerung geladen (siehe ↳ *Kapitel 7.3 „Einloggen auf der Steuerung und Wechsel in den Debug-Betrieb“ auf Seite 167*), ist die Sicherheitssteuerung immer im unsicheren Zustand. In diesem Applikationszustand kann Debuggen und Start/Stop auf der Steuerung durchgeführt werden.

Anzeige der Zustände der Sicherheitssteuerung

Ob sich die Sicherheitssteuerung im sicheren oder unsicheren Zustand und die Applikation im Stop- oder Läuft-Zustand befindet, wird in der in der allgemeinen Statuszeile von CODESYS am unteren Fensterrand angezeigt.

Es wird der Status der aktiven Applikation angezeigt, unabhängig von den geöffneten Editoren.

Beispiele für im Online-Modus angezeigte Informationen in der untersten Statuszeile

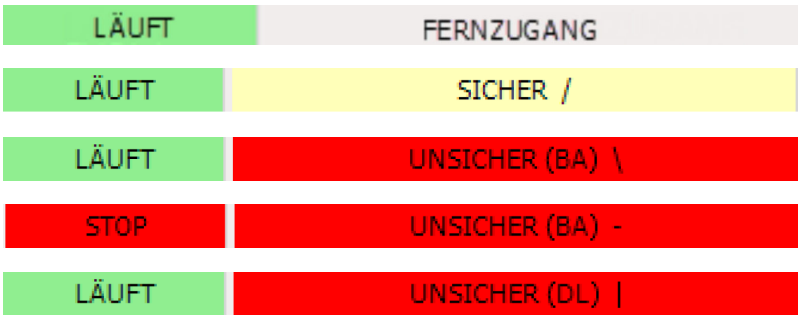


Abb. 94: links: Zustände der Sicherheitsapplikation, rechts: Zustände der Sicherheitssteuerung

Zustände der Sicherheitsapplikation

- *„LÄUFT“*, grün hinterlegt
- *„STOP“*, rot hinterlegt: die Applikation pausiert
- *„BEENDET“*, rot hinterlegt: die Applikation wurde beendet aufgrund eines Laufzeitfehlers

Der Zustand der Sicherheitsapplikation wird im eingeloggten Zustand auch im Projektbaum bei der aktiven Sicherheitsapplikation angezeigt.

Zustände der Sicherheitssteuerung:

- *„FERNZUANG“*, grau hinterlegt
  Zugriff auf Sicherheitssteuerung über Fernzugang
- *„SICHER“*, gelb hinterlegt, wenn die Bootapplikation läuft
- *„UNSICHER (BA)“*, rot hinterlegt:
  Bootapplikation in Debug-Modus
- *„UNSICHER (DL)“*, rot hinterlegt:
  Downloadapplikation in Debug-Modus
- *„ENTLADEN“*, grau hinterlegt:
  die laufende Applikation auf der Steuerung wurde entladen (kein Applikationsstatus mehr)
- *„AUSNAHME“* rot hinterlegt:
  zeigt unter speziellen Umständen, wenn Login bestehen bleibt, einen Systemfehler an (gewöhnlich erfolgt gleich ein Verbindungsabbruch).

„Force Aktiv“ wird zusätzlich zum Zustand der Sicherheitssteuerung angezeigt, falls momentan Werte geforct sind.

HINWEIS!
Außer bei Zustand "FERNZUGANG" wird hinter dem Zustand der Sicherheitssteuerung ein kreisender Balken angezeigt. Wenn er einfriert, kann sich der Status der Sicherheitssteuerung schon weiter verändert haben, ohne dass es angezeigt wurde.

7.5.2 Debug-Modus und organisatorische Sicherheit

Wechsel in den Debug-Betrieb

Folgende Befehle führen zum Wechsel in den Debug-Betrieb:

Befehle des Menüs Online

- *„Reset Kalt“* (siehe *Kapitel 7.6.5 „Debug-Befehle: Start/Stop und Applikation zurücksetzen“ auf Seite 184*)

Befehle des Menüs Debug

- *„Start“* (siehe *Kapitel 7.6.5 „Debug-Befehle: Start/Stop und Applikation zurücksetzen“ auf Seite 184*)
- *„Stop“* (siehe *Kapitel 7.6.5 „Debug-Befehle: Start/Stop und Applikation zurücksetzen“ auf Seite 184*)
- *„Werte schreiben“* (siehe *Kapitel 7.6.4 „Debug-Befehle: Schreiben/Forcen“ auf Seite 182*)
- *„Werte forcen“* (siehe *Kapitel 7.6.4 „Debug-Befehle: Schreiben/Forcen“ auf Seite 182*)
- *„Forcen für alle Werte aufheben“* (siehe *Kapitel 7.6.4 „Debug-Befehle: Schreiben/Forcen“ auf Seite 182*)

Der Befehl *„Start“* des Menüs *„Debug“* kann nur im Debug-Modus ausgeführt werden, wenn die Applikation im Stop-Zustand ist.

GEFAHR!
Bei jedem Wechsel vom sicheren in den unsicheren Betriebs der Sicherheitssteuerung muss der Anwender sicherstellen, dass die organisatorische Sicherheit der Anlage sichergestellt ist.

Für ein Netzwerk gibt es folgende Möglichkeiten:

– Alle Netzwerkkomponenten organisatorisch absichern
– Das Netzwerk physikalisch abtrennen, um nur eine kleineren Teil des Netzwerks zu debuggen und absichern zu müssen

GEFAHR!

Während des Debug-Betriebs befindet sich die Sicherheitssteuerung immer im unsicheren Zustand! Dabei liegt es in der Verantwortung des Entwicklers, dass die Sicherheit der Anlage durch organisatorische Maßnahmen sichergestellt wird. Diese Maßnahmen müssen vor Freischaltung der Debugdienste beginnen und solange aufrechterhalten bleiben, bis eines der folgenden Ereignisse eintritt:

- nach einem Online-Befehl wird die Zurücksetzung der Debug-Freischaltung rückgemeldet

- im Programmiersystem wird der sichere Status laufend dynamisiert angezeigt

- Reset der Sicherheitssteuerung

Wenn im Netzwerk Safety Netzwerkvariablen für die Querkommunikation verwendet werden, müssen die Maßnahmen so lange bestehen bleiben, bis auch die letzte Sicherheitssteuerung den Debug-Betrieb beendet hat. Die Rückmeldung von 1 Sicherheitssteuerung genügt nicht!

Hinweis FDev_12

GEFAHR!

Das Debuggen einer F-Device-Steuerung kann sich auch auf den F-Host auswirken. Die organisatorischen Sicherheitsmaßnahmen, die vom Beginn bis Ende des Debuggens einer Sicherheitsteuerung vorgenommen werden müssen, betreffen daher nicht nur den von dieser Sicherheitssteuerung überwachten Teil der Maschine. Die organisatorischen Sicherheitsmaßnahmen müssen auf die gesamte Maschine ausgedehnt werden, die über das PROFINET-Netzwerk verbunden ist. Dies bedeutet auch, dass sich keine Personen in den Gefahrenbereichen dieser Maschine aufhalten dürfen.

Diese gesamten organisatorischen Maßnahmen können vermieden werden, indem das PROFINET-Gerät (Standardsteuerung) physikalisch vom PROFINET-Netzwerk getrennt wird.

Die gemeinsamen organisatorischen Maßnahmen an allen Maschinen im PROFINET-Netzwerk müssen so lange bestehen bleiben, bis auch die letzte Sicherheitssteuerung den Debug-Betrieb beendet hat.

Schreiben auf mehreren SPSen

GEFAHR!
Bei Multi-SPS Projekten muss der Entwickler für die Freigabe von Debugdiensten, bzw. für die Befehlsausführung in einem Dialog den Gerätenamen bestätigen, damit der Befehl an die richtige Steuerung geht.

Rückkehr in den sicheren Zustand

Die Rückkehr einer Sicherheitssteuerung vom unsicheren in den sicheren Zustand kann erfolgen durch

- Neustarten der Bootapplikation (Befehl *„Neu starten“* der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung, siehe *„Neustart der Bootapplikation“ auf Seite 172*)
- Ausloggen aus der Sicherheitssteuerung (Befehl *„Ausloggen“* des Menüs *„Online“*, siehe *„Ausloggen“ auf Seite 169*)
- Ausschalten und anschließendes Einschalten der Sicherheitssteuerung
- Urlöschen der Bootapplikation (Befehl *„Urlöschen“* der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung, siehe *weitere Informationen auf Seite 258*)
- Löschen der Bootapplikation (Befehl *„Löschen“* der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung, siehe *„Bootapplikation löschen“ auf Seite 253*)

7.5.3 Beenden der Applikation

Die laufende Sicherheitsapplikation kann durch verschiedene Operationen im sicheren Betrieb und im Debugbetrieb beendet werden:

- Einloggen mit Download und Wechsel in Debugbetrieb (*Kapitel 7.3 „Einloggen auf der Steuerung und Wechsel in den Debug-Betrieb“ auf Seite 167*)
- Neustart der Bootapplikation (*Kapitel 7.4 „Erzeugen und Neustart der Bootapplikation“ auf Seite 170*)
- Urlöschen
- Firmware Update
- Laufzeitfehler in der Applikation

In all diesen Fällen wird noch ein letztes Ausgangsabbild erzeugt, in dem alle Ausgangskanäle (digital und analog) von sicheren Feldgeräten gemäß Protokoll auf Fail-safe Werte gesetzt und im Protokoll als Fail-safe markiert sind. Alle Ausgangskanäle (digital und analog) von nicht-sicheren Feldgeräte sind in diesem Ausgangsabbild genullt.

7.6 Monitoring und Debuggen

7.6.1 Monitoring

Monitoring

Beim Monitoring im Online-Modus werden, solange ein Login auf die Applikation besteht, der Online-Status und die gemonitorten Werte sichtbarer Variablen zyklisch aus der Steuerung geholt und im Implementierungs- und Deklarationsteils des jeweiligen Objekts angezeigt.

Das Monitoring funktioniert sowohl im sicheren Betrieb wie im Debug-Betrieb (bei lesendem Zugriff)

VORSICHT!

Der angezeigte Monitoringwert für eine Variable ist ein Wert, den diese Variable auf der verbundenen Steuerung hatte. Dieser Wert ist nicht unbedingt der aktuelle Wert, d.h. auf der Steuerung kann sich der Wert bereits wieder geändert haben. Die Monitoringanzeige im Safety-Editor im Online-Modus ist nur zum Nachweis geeignet, dass ein bestimmter Wert bzw. Zustand einmal eingenommen wurde. Alle in einem Safety-Editor im Online-Modus gleichzeitig angezeigten Werte haben in der angezeigten Kombination auch auf der Sicherheitssteuerung am Ende eines Applikationszyklus vorgelegen (zykluskonsistentes Monitoring). Deshalb kann die Monitoringanzeige als Hilfsfunktion eingesetzt werden, um die Zweigabdeckung von Tests anhand von Merker-Variablen zu verifizieren (siehe ↳ *„Nachweis der Zweigabdeckung “ auf Seite 211*)

VORSICHT!

Solange das Login besteht, dürfen die Objekte der Sicherheitsapplikation nicht geändert werden. Ob ein Objekt geändert ist, erkennt man bei einer gepinnten Applikation an der Anzeige *„In Work“* in der Online-Ansicht des Editors.

Das Monitoring der Variablen im Deklarationsfenster und im Implementierungsfenster der POU ist qualifiziert und für die Verifikation geeignet.

Überwachungsfenster

VORSICHT!

Monitoring im Überwachungsfenster (Watchfenster) ist nicht qualifiziert und damit nicht für die Verifikation einer Sicherheitsapplikation geeignet!

Für Detailinformationen zu den Themen Überwachungsfenster wird auf die Online Hilfe von CODESYS Standard verwiesen.

Monitoring bei Laufzeitfehler der Applikation

Das Monitoring dient auch zur Diagnose von Laufzeitfehlern. Im Falle eines Ausführungsfehlers wird die Applikation beendet, jedoch nicht entladen. Der Entwickler kann sich auf die Applikation einloggen und die aktuellen Werte der Variablen zum Zeitpunkt des Ausführungsfehlers monitoren.

7.6.2 Ablaufkontrolle

Bei aktivierter Ablaufkontrolle werden im FUP-Implementierungsteil von POUs nicht mehr die Werte von Variablen am Ende des Zyklus dargestellt. Es werden nun die Werte von Variablen, die Ergebnisse von Aufrufen von Operatoren und die Operationen an der jeweiligen Abarbeitungsposition und zum jeweiligen Abarbeitungszeitpunkt im Implementierungsteil des Safety FUP-Editors angezeigt. In der Statuszeile wird dabei *„Ablaufkontrolle aktiviert“* angezeigt.

Genauere Informationen finden Sie in der CODESYS Safety Online-Hilfe.

HINWEIS!
Die Ablaufkontrolle ist nicht qualifiziert und damit nicht für die Verifikation einer Sicherheitsapplikation geeignet.

Voraussetzung für die Funktionalität "Ablaufkontrolle" ist eine aktuelle Sicherheitssteuerung. Bei einer älteren Firmware-Version kann es notwendig sein, dass Sie die Firmware aktualisieren, siehe ↳ Kapitel 12.6.2 „Firmware-Update aufspielen“ auf Seite 256.

7.6.3 Debug-Betrieb der Sicherheitssteuerung

HINWEIS!
Während des Debug-Betriebs (Debug-Modus) befindet sich die Sicherheitssteuerung immer im **unsicheren Betrieb**.

Debuggen kann auf der Sicherheitssteuerung auf der Download-Applikation und auf der Bootapplikation durchgeführt werden. Das Debuggen dient dem Erkennen und Finden von Fehlern der Sicherheitsapplikation.

Zum Debuggen der Sicherheitsapplikation muss der Entwickler aus dem Projekt heraus auf der Sicherheitssteuerung eingeloggt sein!

GEFAHR!

Der Entwickler ist dafür verantwortlich, dass die Sicherheit der Anlage/Maschine während des ganzen Zeitraums, in dem sich die Sicherheitssteuerung im Debug-Betrieb (unsicheren Betrieb) befindet, gewährleistet ist.

GEFAHR!

Wenn der Entwickler die Sicherheitsapplikation mit schreibenden Diensten debuggen möchte, während die Sicherheitssteuerung in der Maschine eingebaut ist, muss er an der Maschine/Anlage sicherstellen, dass sich keine Person im Gefährdungsbereich befindet.

Diese Maßnahmen müssen vor Freischaltung der Debugdienste beginnen und müssen aufrechterhalten werden

- bis nach einem Online-Befehl rückgemeldet wird, dass die Debugbefehl-Freischaltung zurückgesetzt wurde,
  oder
- bis im PS die Statusanzeige den Status "sicher“ laufend dynamisiert anzeigt,
  oder
- bis zum Reset der Steuerung

Wenn die Bootapplikation läuft, werden schreibende Debugdienste (inkl. Download einer temporären Applikation, aber ohne Stop) nur ausgeführt, wenn der Entwickler die Verwendung von Debugdiensten (d.h. Debug-Betrieb) explizit freigeschaltet hat (siehe ↬ *Kapitel 7.5.2 „Debug-Modus und organisatorische Sicherheit“ auf Seite 176*). Es ist dabei sichergestellt, dass eine erteilte Freigabe nur für eine Debug-Sitzung gültig ist und dass sie nach einem Logout nicht mehr gültig ist.

7.6.4 Debug-Befehle: Schreiben/Forcen

GEFAHR!
Das Ausführen schreibender Debugdienste ändert das aktuelle Verhalten der gerade laufenden Applikation!

Befindet sich die Steuerung im sicheren Betrieb und die Befehle *„Werte schreiben“* bzw. *„Werte forcen“*, oder *„Reset kalt“* werden aktiviert, so geht die Steuerung in den Zustand *„DEBUG (BA), d.h. in den Debug-Betrieb (unsicherer Betrieb)“*

„Schreiben“ und *„Forcen“* in CODESYS Safety funktionieren wie in CODESYS Standard. Es wird deshalb auch auf die Online-Hilfe von CODESYS Standard verwiesen.

Während der Entwicklung der Sicherheitsapplikation können schreibende Debugdienste uneingeschränkt verwendet werden.

HINWEIS!
Schreibende Debugdienste dürfen nicht für funktionale Tests der Sicherheitsapplikation verwendet werden. Ausnahme siehe ↳ *Kapitel 9.4.1 „Dynamische Verifikation und Validierung“ auf Seite 208*

Werte schreiben

Der Debug-Befehl *„Werte schreiben“* bewirkt, dass einmalig vor dem Applikationszyklus alle zum Schreiben vorbereiteten Werte der aktiven Sicherheitsapplikation aus allen Editoren von Variablendeklarationen und allen Überwachungsfenstern auf einmal in die Steuerung geschrieben werden.

Werte Forcen

Forcen funktioniert analog zum Schreiben, nur dass hier die zum Schreiben vorbereiteten Werte in der Steuerung hinterlegt werden.

Vor und nach jedem Zyklus werden die Werte der entsprechenden Variablen mit den Werten aus der Schreibliste überschrieben (geforct), bis das Forcen aufgehoben wird oder die Steuerung den Debug-Modus verlässt.

Angezeigt in Zustand als *„Force aktiv“* siehe ↳ *„Anzeige der Zustände der Sicherheitssteuerung“ auf Seite 174*

Da die Variablen nur am Anfang und Ende von jedem Zyklus auf den Forcewert gesetzt werden,

- können die Variablen während des Zyklus durch andere Werte überschrieben werden

- kann nur in die Ursprungswerte einer Berechnung eingegriffen werden und ist es nicht sinnvoll Zwischenvariablen zu forcen

- werden nur die Ausgänge der Applikation am Ende des Zyklus mit den Forcewerten überschrieben

Die Vorbereitung von Werten erfolgt wie in CODESYS, siehe Abb. 95, Abb. 96, Abb. 97

Über den Dialog *„Wert vorbereiten“* (Abb. 96) kann bestehendes Forcen für diese Variable aufgehoben werden.

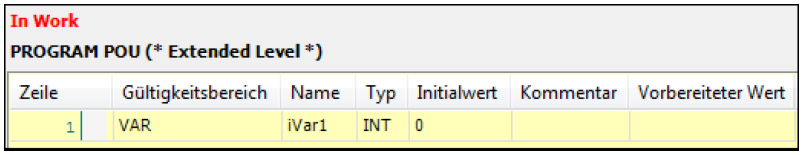


Abb. 95: Deklarationsteil der POU mit zusätzlichen Spalten: Wert und Vorbereiteter Wert

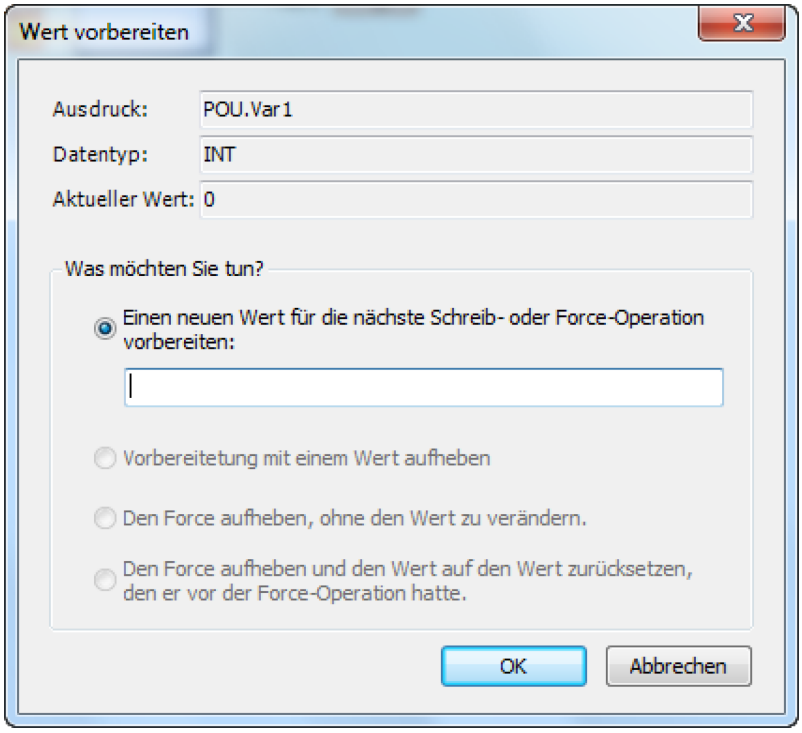


Abb. 96: Dialog zum Vorbereiten der Werte für Schreiben/Forcen

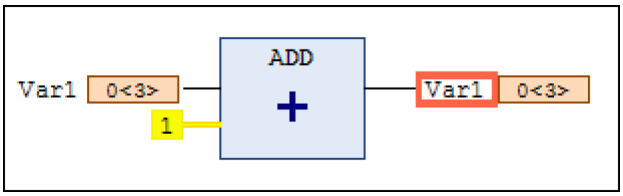


Abb. 97: Bsp. Variablen Var1 mit aktuellem Wert 0 und vorbereitetem Wert 3

Forcen für alle Werte aufheben

Dieser Debug-Befehl beendet das Forcen in der Sicherheitssteuerung. In der Oberfläche werden die Markierungen der geforcten Variablen zurückgesetzt.

7.6.5 Debug-Befehle: Start/Stop und Applikation zurücksetzen

Programmablaufkontrolle Start/ Stop und Applikation zurücksetzen

Die Befehle *„Start“* und *„Stop“* der Kategorie *„Debug“* sind nur verfügbar, wenn der Entwickler aus dem Projekt heraus auf der Steuerung eingeloggt ist.

Mit dem Debug-Befehl *„Start“* startet die aktive Applikation auf der Sicherheitssteuerung und setzt sie in den Zustand *„RUN“*. *„Stop“* hält die aktive Applikation der Sicherheitssteuerung an, die Applikation wird nicht mehr durchlaufen, die Stacks der sicheren Protokolle werden weiterhin ausgeführt.

Wird mit dem Befehl „Stop“ die Bootapplikation, die auf der Sicherheitssteuerung im sicheren Betrieb läuft, angehalten, so geht die Sicherheitssteuerung in den Debug-Betrieb.

Im Zustand STOP werden alle Ausgangswerte an Feldgeräten (digital und analog, sicher und nicht-sicher) automatisch genullt und im Fall von sicheren Feldgeräten, sofern das Safety-Protokoll es unterstützt, werden die Failsafe-Werte aktiviert. Bei sicheren Ausgangsmodulen zwingt das die Maschine in den sicheren Zustand (das heißt in der Regel, dass sie die Maschine anhalten). Die Ausgangswerte eines gestoppten NVL-Senders bleiben auf dem letzten Wert vor dem Stopp stehen und werden weiterhin an seine NVL-Empfänger gesendet.

Und im Zustand STOP werden gültige Eingangswerte von Feldgeräten und NVL-Sendern weiterhin in die gemappten Variablen kopiert.

VORSICHT!
Mit *„Start“* gelangen sofort wieder die aktuellen Werte aus der Applikation an die Feldgeräte und die Maschine läuft gegenfalls weiter (je nach Zustand der Applikation). (Anders als bei einem Kommunikationsfehler, der in der Regel erst quittiert werden muss, bevor wieder Werte übertragen werden)

Für die Debug-Befehle „Start“ und „Stop“ wird auf die Online-Hilfe von CODESYS Standard verwiesen.

Applikation zurücksetzen

Der Befehl *„Reset kalt“* der Kategorie *„Online“* ist nur im Online-Betrieb verfügbar. Durch Aktivierung dieses Befehls geht die Applikation bzw. bleibt in *„STOP“*, wird zurückgesetzt und neu initialisiert (gleich der Initialisierung nach dem Laden der Applikation).

7.7 Online Informationen aus der Sicherheitssteuerung

Die Sicherheitssteuerung stellt dem Anwender eine Reihe von Informationen zu Diagnose- und Debugzwecken zur Verfügung. Diese Informationen sind in einem Editor auf verschiedenen, über Tabs auswählbaren Registerkarten hinterlegt. Der Editor kann durch einen Doppelklick auf die Sichherheitssteuerung im Projektbaum oder über Selektion der Sicherheitssteuerung im Projektbaum und Aktivierung des Kontextmenübefehls *„Objekt bearbeiten“* geöffnet werden.

Der Editor enthält folgende Registerkarten:

- *„Kommunkationseinstellungen“*
  siehe *Kapitel 7.2.1 „Kommunikationseinstellungen - allgemeine Informationen“ auf Seite 163*
- *„Log“*
  siehe *Kapitel 12.3.3 „Logbuch: Diagnose von System- und Laufzeitfehlern“ auf Seite 248*
- *„Safety Online Information“*
  siehe *Kapitel 12.3.2 „Informationen zu Firmware und Bootapplikation “ auf Seite 247*
- *„Status“*
  siehe *Kapitel 12.3.4 „Status: Diagnose der Kommunikation“ auf Seite 250*
- *„Information“*
  Anzeige von allgemeinen Informationen (siehe CODESYS Onlinehilfe)

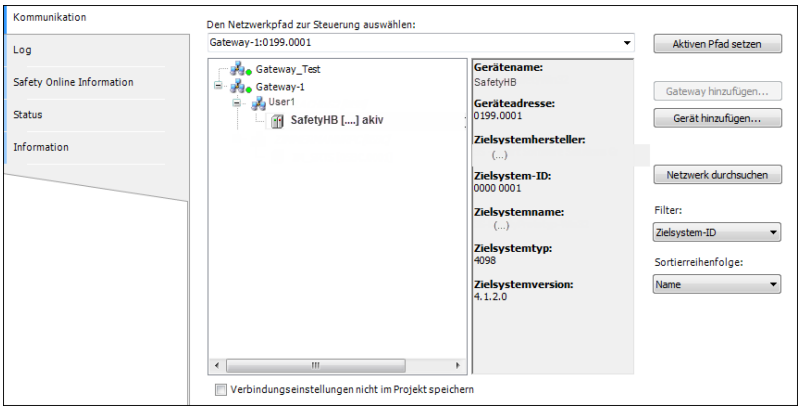


Abb. 98: Editor der Sicherheitssteuerung, Registerkarte 'Kommunikation'

7.8 Abstimmung mit der Standardsteuerung

Konsistenz der Applikationen

HINWEIS!

Werden in der Sicherheitssteuerung physikalische Geräte und Variablen der Standardsteuerung verwendet (als Logisches E/A-Objekt), so muss sowohl ein Download der Applikation auf die Standardsteuerung, als auch ein Download der Sicherheitsapplikation auf die Sicherheitssteuerung durchgeführt werden, damit die physikalischen Geräte und die GVLs für logischen Austausch der Standardsteuerung und die logischen E/As der Sicherheitssteuerung mit aktuellen Werten versorgt werden.

HINWEIS!

Werden in der Sicherheitssteuerung Netzwerkvariablen verwendet, so muss sowohl ein Download der Applikation auf die Standardsteuerung, als auch ein Download der Sicherheitsapplikation auf die Sicherheitssteuerung durchgeführt werden. Wenn auf Seiten der verbundenen Sicherheitsteuerung der Stand der Applikationen nicht mehr aktuell ist, muss auch dort sowohl ein Download der Applikation auf die Standardsteuerung, als auch ein Download der Sicherheitsapplikation auf die Sicherheitssteuerung durchgeführt werden. Nur wenn die vier Applikationen auf den beiden Sicherheitsteuerungen und den beiden Standardsteuerungen konsistent sind, werden die Netzwerkvariablen mit aktuellen Werten versorgt.

Konfigurationsunterschiede

Sind die physikalischen Geräte und die Austauschvariablen anders konfiguriert als die entsprechenden Objekte der Sicherheitsapplikation, so wird dies mit dem Warnsymbol für Konfigurationsfehler von CODESYS Standard im Projektbaum bei der Sicherheitssteuerung angezeigt.

Die Konfigurationsunterschiede beziehen sich auf Anzahl, Id oder E/A-Größe der E/A-Module und Austauschvariablen für Standard- und Sicherheitssteuerung.

Für Bussysteme, bei denen die Übertragung der sicheren Konfiguration über die Standardkonfiguration erfolgt, wird die unterschiedliche Parametrierung eines E/A-Moduls als Konfigurationsunterschied angezeigt.

Wie Konfigurationsunterschiede angezeigt, gemeldet und geloggt werden, ist in der Online-Hilfe im Kapitel "Abstimmung mit der Standardsteuerung" beschrieben.

Unterbrechungen durch die Standardsteuerung

HINWEIS!

Durch Unterbrechen der funktionalen Applikation (zum Beispiel durch Ausschalten oder zum Beispiel durch Online-Befehle *„Stop“*, *„Reset“*, *„Bootapplikation löschen “*, Download) oder wenn die Aktualisierung der funktionalen Applikation zu einem Mismatch der E/A-Liste mit der Sicherheitssteuerung führt, wird die Verbindung der Sicherheitssteuerung mit ihren Feldgeräten und mit anderen Sicherheitssteuerungen in der Regel unterbrochen(*), was nach Ablauf der Überwachungszeiten der Verbindungen zu Failsafe-Reaktionen führt:

- Die Feldgeräte bringen die Maschine bzw. den Maschinenabschnitt der Sicherheitssteuerung in den "sicheren Zustand" (das heißt in der Regel, die Felgeräte halten sie an).
- Die Eingangswerte der Sicherheitsapplikation und ihre Empfänger-NVLs nehmen Failsafe-Werte ein.
- Die mit der Sicherheitsapplikation ausgetauschten Variablen nehmen in den anderen Empfänger-Sicherheitssteuerungen Failsafe-Werte ein, was zu weiteren Failsafe-Reaktionen **in anderen Maschinenabschnitten** führen kann.

(*) Je nach Feldbus und Implementierung seines Treibers kann es auch sein, dass die Kommunikation auch ohne die funktionale Applikation weiter läuft.

Wenn die funktionale Applikation danach wieder läuft (zum Beispiel nach Online-Befehl „Run“), muss in der Regel erst der Kommunikationsfehler quittiert werden, bevor die Failsafe-Reaktionen der sicheren Ausgangsmodule und der anderen Sicherheitssteuerungen zurückgenommen werden, und die Maschine wieder anlaufen kann.

Variablenaustausch mit der Standardsteuerung

Können, solange die Applikation nicht beendet ist, keine aktuellen Werte ausgetauscht werden, dann gilt für den Variablenaustausch:

Ersatzwerte für Austauschvariablen

- Hat der Entwickler die Sicherheitsapplikation gestoppt, oder die Applikation der Standardsteuerung läuft nicht, dann
  - werden die Werte der Standardsteuerung übertragen, auch wenn die Standardsteuerung gestoppt wurde.
  - Außer bei Austausch-Mismatch werden weiterhin die Werte der anderen Applikation (Variable*„...Out“*) in die eigenen Variablen (Variable *„...In)“* kopiert.
- Wenn der Variablenaustausch wegen Mismatch eingestellt wurde, werden Null-Werte auf die gemappten Lesevariablen (Logisches Austauschobjekt *„... In“*) der Sicherheitsapplikation geschrieben.
- Wenn die Sicherheitsapplikation beendet wird (z. B. durch Download einer neuen Applikation), werden zuvor alle Schreib-Variablen (Variable *„... Out“* der anderen Applikation noch einmal genullt.

8 Pinnen der Software

Vorbereitende Maßnahme für die Verifikation

Für die Verifikation der Sicherheitsapplikation muss der Entwickler vorbereitende Maßnahmen treffen. Ein wesentlicher Aspekt hierbei ist es, den zur Verifikation vorgesehenen Stand der Sicherheitsapplikation festzusetzen und somit sicherzustellen, dass immer nur genau dieser Stand der Sicherheitsapplikation für die Verifikation, Validierung und anschließende Abnahme verwendet wird.

Speziell hierfür bietet CODESYS Safety die Funktion **Pinnen**.

HINWEIS!
Bevor die programmierte oder geänderte Sicherheitsapplikation verifiziert wird, ist sie neu zu pinnen. Verifikation und Abnahme darf nur bei gepinnten Applikationen durchgeführt werden. Kein Objekt der Applikation darf den Status "In Work" aufweisen. Die Prüfung auf "In Work" muss vor der Verifkation und vor der Abnahme erfolgen.

Was ist Pinnen

Pinnen bedeutet, dass ein Referenzpunkt auf den aktuellen Stand einer Sicherheitsapplikation gesetzt wird, der den konkreten Stand der Sicherheitsapplikation und der dazugehörenden Objekte identifiziert. Mittels des Pins ist es möglich, einen bestimmten Stand der Applikation im Projekt, eines Objekts im Editor und einer Bootapplikaton auf der Sicherheitssteuerung zu identifizieren. Außerdem kann der Verifizierende - basierend auf dem Pin - jederzeit Änderungen in der Applikationsstruktur, im Inhalt ihrer Objekte und in den angezogenen Bibliotheksbausteinen erkennen.

Durch das Setzen eines Pins wird ein konkreter Stand identifizierbar gemacht, es wird dabei jedoch keine Kopie des konkreten Standes erzeugt!

Objektliste

Die Pin-Funktionen sind im Editor des Applikationsobjekts zu finden. Dazu wird im Projektbaum das Safety Applikationsobjekt selektiert und über den Kontextmenü-Befehl *„Objekt bearbeiten“* geöffnet. Die Registerkarte *„Objekte“* zeigt die Objektliste, welche die Version und die Prüfsumme der Objekte des aktuellen Projekts und des gepinnten Projekts darstellt.

Eine genaue Beschreibung zu den Informationen und zur Objektliste befindet sich im ↳ *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)

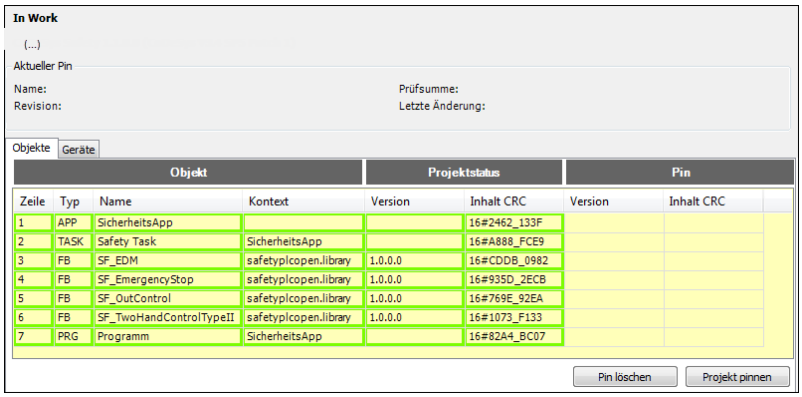


Abb. 99: Objektliste einer noch nicht gepinnten Sicherheitsapplikation

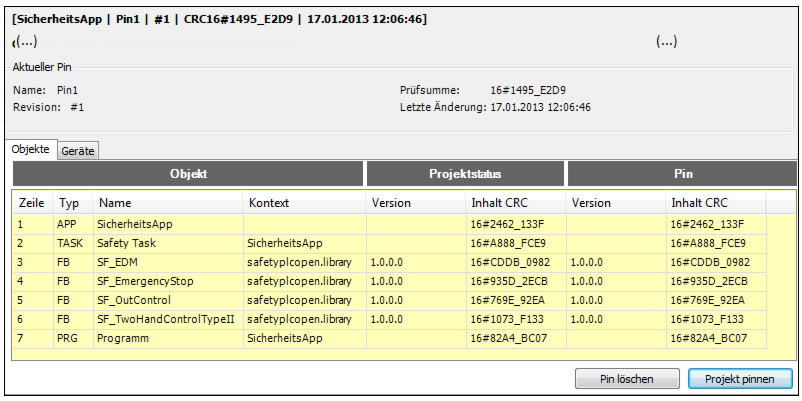


Abb. 100: Objektliste einer gepinnten Sicherheitsapplikation

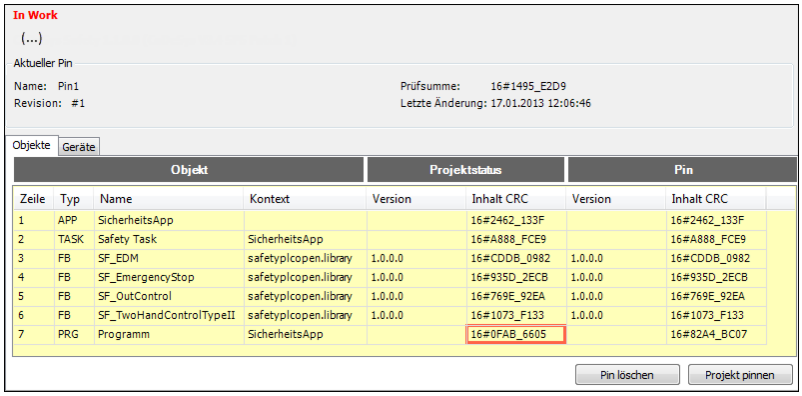


Abb. 101: Objektlsite einer gepinnten Sicherheitsapplikation mit geänderter POUBa der aktuellen Applikation

Sicherheitsapplikation pinnen

Mit dem Befehl *„Projekt pinnen“* wird auf den aktuellen Stand der Applikation ein Pin gesetzt, wodurch nicht der Inhalt, sondern die Prüfsumme und die Version festgehalten (gemerkt) werden. Es kann ein Pin-Name eingegeben werden. Die Revisionsnummer des Pins wird mit jedem "neuen" Pinnen automatisch um 1 erhöht.

Durch den Befehl *„Pin löschen“* wird der aktuelle Pin gelöscht und alle Objekte befinden sich wieder *„In Work“*.

Der gepinnte Stand Sicherheitsapplikation umfasst:

- Umfang der Sicherheitsapplikation:
  - welche Safety-Objekte gehören zur Applikation
  - welche Bibliotheksbausteine benötigt die Applikation
- Ausführungsrelevanter Stand der Objekte und Bibliotheksbausteine im Applikationsumfang:
  - Code jedes Objekts der Applikation
  - Konfiguration und Geräteparameter jedes logischen E/A-Objekts der Applikation
  - Schnittstelle der externen Implementierung jedes verwendeten Bibliotheksbausteins
  - Versionsbezeichnungen der Objekte

Nicht zum ausführungsrelevanten Stand gehören die Kommentare der Objekte. Diese werden nicht mitgepinnt und können somit am Ende und während der Verifikation aktualisiert werden!

Der Verifzierende identifiziert einen gepinnten Stand durch eine Pin-Kennung, die an verschiedenen Stellen im Programmiersystem angezeigt wird. Die Pin-Kennung enthält:

- Name
- Revisionszähler, der beim Pinnen automatisch um 1 erhöht wird.
- Prüfsumme: ein CRC32 über den gepinnten Stand der Applikation

Zusätzlich wir die Zeit des Pinnens festgehalten. Sie ist jedoch nicht Bestandteil der Pin-Kennung.

Anzeige der Pin-Informationen und deren Abweichungen

Die Pin-Informationen zur Applikation zu einer Sicherheitsapplikation werden im Editor des Safety Applikationsobjekts angezeigt.

Die Informationen zum Pin der Sicherheitsapplikation sind:

- *„Name“*
  Name des Pins
- *„Revision“*
- *„Prüfsumme“*
  die Prüfsumme wird über die gesammte gepinnte Applikation erstellt.
- *„Letzte Änderung“*
  Zeitpunkt der Pin-Erzeugung

Zusätzlich wird in der Objektliste des Safety Applikationsobjekts dargestellt, wie der aktuelle Projektstand vom aktuell gepinnten Stand der Applikation abweicht. Dabei werden folgende Abweichungen angezeigt:

- neue Objekte
- gelöschte Objekte
- bezüglich Code, Konfiguration oder Parameter modifizierte Objekte
- neu aus Bibliotheken angezogene Bausteine

- nicht mehr angezogene Bibliotheksbausteine
- bezüglich Schnittstelle oder Implementierungsversion abweichende Bibliotheksbausteine

Abweichungen sind deutlich farblich gekennzeichnet, so dass sie der Verifizierende leicht erkennnen kann:

- grün: im Projekt neue Objekte oder Bibliotheksbausteine
- rot: inhaltliche Änderung/Abweichung im Objekt bzw. Geräteparametersatz bzw. Bibliotheksbaustein
- blau : im Projekt gelöschte oder nicht mehr verwendetete Objekte bzw. Bibliotheksbausteine

Pinnen in der Projekt- und in der Objektansicht

Ist die Sicherheitsapplikation gepinnt, so enthält die Vergleichsansicht die Pin-Informationen und im Projektbaum sind der Knotenpunkt SafetyApp und seine Kindobjekte mit dem Symbol gekennzeichnet. Der Knotenpunkt *„SafetyApp“* gilt als gepinnt, , wenn das Objekt selbst und alle seine Kindobjekte dem im Pin gemerkten Objektstand entsprechen. .

Ist die Applikation noch nicht gepinnt oder der Pin wurde gelöscht, so erscheint in der obersten Zeile nur der Status *„In Work“* und im Projektbaum werden der Knotenpunkt SafetyApp und seine Kindobjekte nicht markiert. Befindet sich ein Kindobjekt der Sicherheitsapplikation *„In Work“*, so ist auch die Sicherheitsapplikation *„In Work“*.

Die Informationen zum Pin bzw. *„In Work“* werden in der Objektansicht und im Ausdruck des Projekts angezeigt.

Vorgehen bei der Verifikation

Das Setzen eines Pins erlaubt, ausführungsrelevante Änderungen während und nach der Verifikation zu erkennen. Mit dem Setzen eines Pins wird der Stand bzgl. der ausführungsrelevanten Teile für Verifikationstätigkeiten und Abnahme identifizierbar gemacht.

HINWEIS!

- Vor Verifikationstätigkeiten und Abnahme ist der Stand per Pin identifizierbar zu machen.
- Der Entwickler muss bei Reviews, Whitebox Testentwicklung und Testdurchführung prüfen, dass die vorliegende Applikation gepinnt wurde, dass die angezeigte Pin-Kennung den aktuell zu verifizierenden Applikationstand darstellt, und dass keine Abweichung des Projektstandes vom gepinnten Stand gemeldet wird.
- Er muss vor Änderungen an einer abgenommenen oder in Verifikation befindlichen Applikation prüfen, dass im Ausgangsstand seiner Änderung die vorliegende Applikation gepinnt ist, die richtige Pin-Kennung angezeigt wird, und dass keine Abweichung des Projektstandes vom gepinnten Stand gemeldet wird.

Pinnen der Software

9 Software-Verifikation

9.1 Einführung

Mit dem Prozess der Software-Verifikation wird nachgewiesen, dass die Spezifikation und die Programmierrichtlinien erfüllt und damit verifiziert sind. Die Software-Verifikation erfolgt durch die Kombination von statischen und dynamischen Verifikationsmethoden (toolbasierte Checks, Codereview, Tests).

Der Software-Verifikation folgt als weiterer Verifikationsschritt die Validierung der programmierten Sicherheitsfunktionen in der aufgebauten Maschine.

Beides sind Voraussetzungen für die Abnahme von Maschine und Sicherheitsapplikation.

Als Voraussetzung muss der Anwender in früheren Entwicklungsphasen spezifiziert haben, was sein gewünschtes Verhalten der Software-Applikation ist (Software-Spezifikation) und was seine gewünschten Regeln von Programmierung und Kommentierung (Programmierrichtlinien) sind.

Prüfung R1 (Installation)

HINWEIS!

Verifizieren Sie die Korrektheit der CODESYS-Installation gemäß Hinweis Installation1, Hinweis Installation2, Hinweis Installation3 und Hinweis Installation4 (siehe ↳ *Kapitel 2.6 „Richtige Version und Konfiguration des Programmiersystems CODESYS Safety“ auf Seite 13*).

Prüfung R2 (BibFB Versionen)

HINWEIS!

Verifizieren Sie, dass die von Ihrer Applikation verwendeten Versionen von Bibliotheksbausteinen mit den in diesem Dokument beschriebenen Versionen übereinstimmen. Sie ermitteln die Version eines verwendeten Bibliotheksbausteins in der Registerkarte *„Objekte“* des Editors des Safety Applikationsobjekts (siehe ↳ *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*. Wie sie lauten muss, steht im Kapitel ↳ *Kapitel 15.1.2 „Applikative Bibliotheken“ auf Seite 293*. (Im Fall von Bibliotheksbausteinen anderer Hersteller muss gegen die in der entsprechenden Begleitdokumentation beschriebenen Version verglichen werden.)

HINWEIS!

Das Bibliotheks-Repository und der Bibliotheksverwalter sind für den Nachweis der in der Sicherheitsapplikation verwendeten Bibliotheksbausteine bei der Verifikation und der Abnahme nicht geeignet. Der Nachweis der ausführungsrelevanten Stände muss über die Objektliste erfolgen, siehe *Kapitel 8 „Pinnen der Software“ auf Seite 189*. Abnahmedokumentation siehe *Kapitel 10.1 „Einleitung “ auf Seite 217*.

Prüfung R3 (Gepinnt)

HINWEIS!

Verifikation und Abnahme darf nur auf gepinnten Sicherheitsapplikationen durchgeführt werden. Kein Objekt der Applikation darf im Objektbaum und in den Ansichten den Status "In Work" aufweisen.

HINWEIS!

Der CODESYS Projektnavigator ist für die Verifikation und Abnahme einer Sicherheitsapplikation **nicht geeignet**. Zum Nachweis, welche Objekte zur Sicherheitsapplikation gehören muss der Editor des Safety Applikationsobjekts verwendet werden (siehe *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)

HINWEIS!

Das Lupenwerkzeug  des Safety FUP-Editors darf für die Verifikation und die Abnahme nicht verwendet werden.

HINWEIS!

Der CODESYS Standard Projektvergleich ist für die Verifikation und Abnahme von Änderungen gegenüber einer bereits vorher verifizierten/abgenommenen Sicherheitsapplikation **nicht geeignet**. Er kann lediglich als Hilfsfunktion verwendet werden, um die Vergleichsansicht des gepinnten Stands der Applikation zu öffnen. Die Vergleichsansicht wird durch einen Doppelklick auf das Safety Applikationsobjekt im Standard Projektvergleich geöffnet.

Um den gesamten Prozess der Verifikation und Validierung zu vereinfachen und damit zu beschleunigen, stellt CODESYS Safety die bereits validierten und zertifizierten PLCopen-Bausteine und die Bausteine der SafetyStandard Bibliothek zur Verfügung.

9.2 Anforderungen an Verifikation/Validierung

9.2.1 PL-e Sicherheitsapplikationen

Verifikation,Validierung und Abnahme für Sicherheitsapplikaton nach PL-e

Für Sicherheitsapplikationen mit **PL-e** sollen gemäß der Norm Tätigkeiten zur Verifikation und Validierung dürchgeführt werden, z.B.:

- Verifikation durch Kontroll- und Datenflussanalyse
- Validierung der funktionalen Leistungen mit Black-Box Tests.
- Validierung des Leistungsverhaltens (z.B. zeitliches Leistungsverhaltens) mit Black-Box Tests.
- Empfehlung: Testfallausführung auf Basis von Grenzwertanalysen
- E/A Tests müssen die korrekte Verwendung der sicherheitsbezogenen Signale sicherstellen.
- Nach Änderungen in der Sicherheitsapplikation muss mithilfe einer Einflussanalyse sichergestellt werden, dass die Spezifikation erfüllt ist.

9.2.2 SIL3 Sicherheitsapplikationen

Verifikation, Validierung und Abnahme für Sicherheitsapplikaton nach SIL3

Für Sicherheitsapplikationen nach SIL3 muss gemäß der Norm die Software dahingegen geprüft werden, ob sie mit dem spezifizierten Entwurf, den Programmierrichtlinien und den Anforderungen der Sicherheitsplanung übereinstimmt. Für Sicherheitsapplikationen nach SIL3 schließen Verifikation und Validierung strukturelle Tests der Anwendungssoftware als White-Box Tests, funktionale Tests des Anwendungsprogramms als Black-Box Tests und Schnittstellentests als Grey-Box Tests (Wechselwirkungen mit Sicherheitssteuerung und anwendungsspezifischer Hardwarekonfiguration) ein. Es müssen gemäß der Norm Tätigkeiten für Verifikation und Validierung dürchgeführt werden, wobei Testen die hauptsächlich verwendete Verifikationsmethode sein soll. Dies beinhaltet unter anderem:

- Überprüfung der E/A Konfiguration durch Überprüfung, Test oder Simulation
- Geeignete funktionale Tests aller Softwaremodule durch Überprüfung, Test oder Simulation
- Geeignete Tests der Module mit
  - Zweigabdeckung
  - Testen mit Grenzdaten
  - Überprüfung der Implementierung der Abläufe, einschließlich relevanter Synchronisierungsbedingungen

9.3 Statische Verifikation

9.3.1 Statische Verifikation

Die statische Verifikation umfasst die Überprüfung der Übereinstimmung mit

- den Codier- bzw. Programmierrichtlinien
  Diese Richtlinien werden zum Teil automatisch durch den Checker (siehe *Kapitel 9.3.3 „Automatische Prüfung der Programmierrichtlinien“ auf Seite 199*) und zum Teil durch Review (siehe *Kapitel 9.3.4 „Manuelle Prüfung der Programmierrichtlinien“ auf Seite 200*) überprüft.
- der Dokumentation verwendeter Bausteine
- den vorhandenen Spezifikationen
  Diese Überprüfung findet durch einen Review statt.
- der Planung des Gesamtsystems (*Kapitel 4 „Planung des Gesamtsystems“ auf Seite 33*)

Die gewünschten Check-Optionen können im Safety Applikationsobjekt eingestellt werden (siehe Kapitel 9.3.3 „Automatische Prüfung der Programmierrichtlinien“ auf Seite 199).

Beim Review kann der Stand eines Objekts durch Vergleich der angezeigten Pin-Kennung gegen die geforderte Pin-Kennung verifiziert werden.

HINWEIS!
Bei Multi-SPS Projekten muss während der ganzen statischen Verifikation durch Prüfen der Pin-Kennung in der Objektansicht jedes Objekts der Sicherheitsapplikation verifiziert werden, dass das Objekt zur richtigen Sicherheitssteuerung gehört.

9.3.2 Gerätekonfiguration und Kommunikationsschnittstelle

Bei der Verifikation gegen den Systemplan prüft der Verifizierende, dass die Safety-Adressen, Connection IDs und Watchdog-Zeiten konfiguriert sind wie geplant.

Der Verifizierende muss prüfen, dass die Empfänger-Netzwerkvariablenlisten mit den richtigen Sender-Netzwerkvariablenlisten verknüpft sind.

Prüfung R4 (Safety-Adressen)

HINWEIS!
Bei der Überprüfung, dass die NVL-Konfiguration die richtigen Sender- und Empfänger-Netzwerkvariablenlisten miteinander verknüpft, müssen alle Safety-Adressen überprüft werden, die in der Netzwerkvariablenlisten (Empfänger) und Netzwerkvariablenlisten (Sender) angezeigt werden, bzw. für eine Steuerung oder Maschine dokumentiert sind.

Prüfung R5 (Gerätebeschreibungen)

HINWEIS!
Bei der statischen Verifikation gegen die Spezifikation muss geprüft werden, ob die im CODESYS Safety Projekt verwendeten Gerätebeschreibungen der Feldgeräte den in der Hardware-Spezifikation der Maschine vorgesehenen Feldgeräte entsprechen.

Dabei sollten geprüft werden:

- die richtige Beschreibung für die sicherheitsgerichtete Konfiguration
- die richtige Beschreibung für die Geräteparametrierung

Der Verifizierende sollte dabei folgendermaßen vorgehen:

1. Jedes logische E/A, das in der gepinnten Objektliste im Applikationseditor aufgelistet ist, öffnen.

   Es muss sichergestellt werden, dass alle Geräte geprüft werden.
2. Bei jedem logischen E/A die Pin-Kennung (*„I/O Abbild“*-Seite), den Eintrag gegen die Maschine (Registerkarte *„Information“*) und Originator-Info (aktuelle CODESYS Safety-Version) überprüfen.
3. Bei jedem logischen E/A die *„Sichere Konfiguration“*-Registerkarte (falls vorhanden) und Pin-Kennung, DeviceInfo und Orginator-Info (aktuelle CODESYS Safety-Version) prüfen.

9.3.3 Automatische Prüfung der Programmierrichtlinien

Einstellungen für die automatische Prüfung von Programmierrichtlinien

Die automatische Prüfung der Programmierrichtlinien findet explizit über den Befehl „Übersetzen“ im Erstellen-Menü und automatisch bei jedem Erzeugen einer Downloadapplikation (Befehl „Einloggen“ des Online-Menüs) oder bei jedem Erzeugen einer Bootapplikation (Befehl „Bootapplikation erzeugen“ des Online-Menüs) statt.

Automatisch geprüfte Programmierrichtlinien siehe Kapitel "Programmierung" ↳ *Kapitel 6.2.6 „Automatisch geprüfte Programmierrichtlinien“ auf Seite 116*

Prüfung R6 (Autom. Checks)

HINWEIS!

Der Verifizierende muss prüfen, dass die in den Programmierrichtlinien gewünschten Checker-Optionen eingestellt sind. Über den Einstellungen muss die Pin-Kennung des zu verifizierenden Applikationsstandes stehen!

Entweder ist die Option *„Warnungen als Fehler behandeln“* aktiviert, oder die Überprüfung der Applikation muss gestartet werden und, falls Warnungen ausgegeben werden, müssen diese bewertet werden.

Alle in diesem Dialog ausgewählten optionalen Programmierregeln und optionalen Begrenzungen werden beim Übersetzen der Sicherheitsapplikation überprüft. Für jeden Verstoß wird eine Warnung im Meldungsfenster ausgegeben.

Es gibt zahlreiche weitere, nicht optionale Regeln, die in jedem Fall überprüft werden. Diese sind bei den einzelnen Sprachelementen in ↳ *Kapitel 6 „Programmierung“ auf Seite 99* und im Kapitel "Fehlermeldungen" in der CODESYS Safety Online-Hilfe als Erläuterungen aufgeführt.

9.3.4 Manuelle Prüfung der Programmierrichtlinien

Es liegt in der Verantwortung und Entscheidung des Verifizierenden, eine manuelle Prüfung von Programmierrichtlinien vorzunehmen.

Der Checker prüft optional, ob bei der Applikation und bei den POUs Kommentare vorhanden sind. Ob diese oder die anderen Kommentare den gewählten Programmierrichtlinien genügen, muss manuell verifiziert werden.

Prüfung R7 (allgem. Regeln)

HINWEIS!

Alle manuellen Programmierregeln aus ↳ *Kapitel 6 „Programmierung“ auf Seite 99* müssen manuell verifiziert werden: ↳ *„Regel P1(Doku)“ auf Seite 103*, ↳ *„Regel P2 (Namen)“ auf Seite 105*, ↳ *„Regel P3 (Check:Plausibel)“ auf Seite 106*, ↳ *„Regel P8 (Check:1oo1)“ auf Seite 146*, ↳ *„Regel P9 (Check:Geräte)“ auf Seite 148*, ↳ *„Regel P11 (NvlSend)“ auf Seite 153*.

Prüfung R8 (Extended Regeln)

HINWEIS!
In Extended-Level POUs müssen zusätzlich geprüft werden: *„Regel P4 (Check:Num)“ auf Seite 106*, *„Regel P5 (NOT/XOR)“ auf Seite 135*, *„Regel P6 (Sprung)“ auf Seite 140*, *„Regel P7 (Return)“ auf Seite 140*, *„Regel P12 (Operatoren)“ auf Seite 136*, *„Regel P13 (Bitcodes)“ auf Seite 134*.

Bedingte Sprünge sollten daraufhin geprüft werden, dass sie gemäß den PLCopen-Regeln nur für Zustandsmaschinen verwendet werden.

Bedingte Returns sollten daraufhin geprüft werden, dass sie gemäß den PLCopen-Regeln nur als Fehleraussprung verwendet werden.

Es wird empfohlen, bei der Prüfung auf Zustandsmaschinen die Netzwerke, welche einen Zweig bilden, im Netzwerkkommentar zu markieren, um diese für den späteren Schritt "Zweigabdeckung" wieder zu finden.

Die Prüfung der Extended-Level Regeln (*„Regel P5 (NOT/XOR)“ auf Seite 135*, *„Regel P6 (Sprung)“ auf Seite 140*, *„Regel P7 (Return)“ auf Seite 140*, *„Regel P10 (analoges Failsafe)“ auf Seite 150*) erfordert eine Datenflussanalyse bzw. eine Kontrollflussanalyse. Diese werden durch die Safety Querverweisliste unterstützt. Dabei sind die Hinweise in *Kapitel 9.3.6.2 „Verwendung von Querverweisliste und Gehe zur Definition“ auf Seite 203* zu beachten.

9.3.5 Manuell Prüfung der Verwendung von Bausteinen

Wenn die Dokumentation oder Spezifikation eines Bausteins Einschränkungen oder spezielle Anforderungen für der Verwendung bzw. Parametrierung des Bausteins enthält, so muss die Einhaltung dieser Einschränkungen bzw. Anforderungen bei der Verifikation manuell geprüft werden.

Prüfung R9 (Verwendung FBs)

HINWEIS!

Die FBs müssen darauf geprüft werden, dass sie gemäß ihrer Dokumentation verwendet werden. Siehe allgemeine Regeln zur Verwendung PLCopen-konformer FBs in *Kapitel 6.2.5 „Regeln zur Verwendung PLCopen-konformer Funktionsbausteine“ auf Seite 115* und spezifische Regeln in *Kapitel 15.2 „Spezifische Sicherheitshinweise für applikative Bibliotheksbausteine“ auf Seite 298*

Zum Beispiel muss man in einer Extended-Level POU prüfen, dass die TIME-Eingägne von PLCopen-Bausteinen ihre Werte bei den Aufrufen nicht ändern (Hinweis Lib5 (TIME-Eingänge) siehe *Kapitel 15.2 „Spezifische Sicherheitshinweise für applikative Bibliotheksbausteine“ auf Seite 298*.)

9.3.6 Applikationsspezifische Prüfungen

9.3.6.1 Prüfung gegen die Spezifikation

Die Überprüfung gegen die Spezifikation der Sicherheitsapplikation wird durch Prüfung der Schnittstellen und durch die Analyse des Kontroll- und Datenflusses durchgeführt.

Prüfung R10 (Parameter)

HINWEIS!

Die Prametrierung der verwendeten Bausteine und Geräte muss verifiziert werden: Überwachungszeiten, Diskrepanzzeiten, Auto-Quittierungen etc. müssen sinnvoll und gemäß der Spezifikation gesetzt sein.

Prüfung R11 (E/A-Wirkung)

HINWEIS!

Es muss verifiziert werden, welche Eingänge auf welche Ausgänge wirken: Auf einen Ausgang müssen all Eingänge wirken, welche die Spezifikation fordert, und keine anderen. (Siehe *Kapitel 9.3.6.5 „Datenflussanalyse“ auf Seite 205*)

9.3.6.2 Verwendung von Querverweisliste und Gehe zur Definition

Zur Unterstützung der Analyse des Kontrollflusses und des Datenflusses in einer Sicherheitsapplikation bietet CODESYS Safety die Ansicht *„Safety-Querverweisliste“* und den Befehl *„Gehe zur Definition“*. Genauere Informationen zu diesen Funktionalitäten finden Sie in der Onlinehilfe.

Für Querverweise in Sicherheitsapplikationen müssen Sie die Ansicht „Safety Querverweisliste“ verwenden. (In der Ansicht „Querverweisliste“ werden Sicherheitsapplikationen nicht angezeigt.)

HINWEIS!

Die *„Safety-Querverweisliste“* und der Befehl *„Gehe zur Definition“* sind möglicherweise nicht mehr für die Kontroll- und Datenflussanalyse geeignet, wenn zusätzliche Packages in CODESYS installiert wurden (siehe *„Hinweis Installation2“ auf Seite 14*.

VORSICHT!

Verwendung der Safety Querverweisliste

Bei Verwendung der Safety Querverweisliste zur Kontrollflussanalyse oder zur Datenflussanalyse der Sicherheitsapplikation sind folgende Punkte zu beachten:

1. **Richtiges Namensformat**. Im Feld *„Name“* dürfen nur "unqualifizierte" Bezeichner eingegeben werden. D.h. nach einer globalen Variable wird durch Eingabe von "‹*Variablenname*›" gesucht, nicht durch Eingabe von "‹*GVL-Name*›.‹*Variablenname*›". Nach FB-Eingängen und FB-Ausgängen wird durch Eingabe von "‹*Ein/Ausgangsname*›" gesucht; die instanzbezogene Suche mittels "‹*FB Instanz-Name*›.‹*Ein/Ausgangsname*›" wird nicht unterstützt.

2. **Richtiger Abschluss der Eingabe / Start der Suche**. Nach Auswahl des Gültigkeitsbereichs und Eingabe des Bezeichners im Feld *„Name“* müssen diese Eingaben durch Betätigen der *[Enter]*-Taste abgeschlossen werden. Die *[Enter]*-Taste aktiviert die Auflistung aller Querverweise in der darunter liegenden Tabelle. Die Verwendung der Lupen-Schaltfläche sucht nur alle Verwendungsstellen im Gültigkeitsbereich *„Aktive Applikation“*.

9.3.6.3 Globale Kontrollflussanalyse

Kontrollflussanalyse

Bei der Kontrollflussanalyse wird der Programmablauf, d.h. die Abarbeitungsreihenfolge eines Programms und seiner Funktionsbausteine überprüft.

Die Kontrollflussanalyse wird von PLCopen zur Verifikation einer Sicherheitsapplikation gefordert.

In CODESYS Safety ist die Kontrollflussanalyse bereits dadurch vereinfacht, dass Sprünge nur im Extended-Level erlaubt werden, und hier wiederum nur Vorwärtssprünge zulässig sind. Zusätzliche Kontrollflussanalyse für den Extended Level siehe *Kapitel 9.3.6.4 „Lokale Kontrollflussananlyse im Extended Level“ auf Seite 205*

Für die Durchführung der Kontrollflussanalyse stehen die Funktionen *„Querverweise ausgeben“* und *„Gehe zur Definition“* zur Verfügung. Besonders geeignet ist hierzu die Funktion *„Gehe zur Definition“*. Genauere Informationen zu diesen Funktionen finden Sie in der CODESYS Safety Online-Hilfe.

Die Hinweise in *Kapitel 9.3.6.2 „Verwendung von Querverweisliste und Gehe zur Definition“ auf Seite 203* sind zu beachten.

Kontrollflussanalyse mit Gehe zur Definition

Der Verifizierende möchte

- von einer Task aus das aufgerufene Programm öffnen: Im Task-Editor muss die Zeile mit dem entsprechenden Programmaufruf selektiert und der Befehl *„Gehe zur Definition“* aktiviert werden.
- von einer POU in einen aufgerufenen Funktionsbaustein wechseln: Die Funktionsbaustein-Box im Implementierungsteil der POU selektieren und den Befehl *„Gehe zur Definition“* aktivieren.

Kontrollflussanalyse mit Querverweisliste

Der Verifizierende möchte

- zu einer POU (Programm oder Funktionsbaustein) alle Aufrufstellen finden: Safety Querverweisliste öffnen (Menü *„Ansicht“*, Auswahl *„Safety Querverweisliste“*) und in die Combobox *„Name:“* den Namen der POU eingeben und mit *[Enter]* abschließen. Es werden alle Verwendungsstellen der POU aufgelistet. Durch einen Doppelklick auf ein Element der Safety Querverweisliste, kann dieses geöffnet und ggf. als Aufruf identifiziert werden.
- alle Aufrufe für eine FB-Instanz finden: Safety Querverweisliste öffnen (Menü *„Ansicht“*, Auswahl *„Safety Querverweisliste“*) und in die Combobox *„Name:“* den Namen der FB-Instanz eingeben und mit *[Enter]* abschließen. Es werden alle Verwendungsstellen der Instanz aufgelistet. Durch einen Doppelklick auf ein Element der Safety Querverweisliste, kann dieses geöffnet und ggf. als Aufruf identifiziert werden.

HINWEIS!
Die Deklaration einer POU selbst wird in der Safety Querverweisliste nicht als Verwendungsstelle aufgeführt.

9.3.6.4 Lokale Kontrollflussananlyse im Extended Level

Bedingte Sprünge (nur Vorwärtssprünge erlaubt) und Returns stehen nur für POUs im Extended Level zur Verfügung. Werden bedingte Sprünge und Returns verwendet, so muss für diese Extended-POUs eine zusätzliche Kontrollflussanalyse der Sprünge durchgeführt werden. Für die Kontrollflussanalyse stehen folgende Funktionen zur Verfügung:

- von einem bedingten Sprung zum Sprungziel wechseln: Sprung selektieren und den Befehl *„Gehe zur Definition“* aktivieren.
- von einem Sprungziel alle Sprünge finden, die zu diesem Sprungziel führen: Sprungziel selektieren und den Befehl *„Ansicht ➔ Safety Querverweisliste“* aktivieren. Durch einen Doppelklick auf einen in der Safety Querverweisliste aufgelisteten Sprung kann an die entsprechende Stelle gewechselt werden.

Die Hinweise in *Kapitel 9.3.6.2 „Verwendung von Querverweisliste und Gehe zur Definition“ auf Seite 203* sind zu beachten.

9.3.6.5 Datenflussanalyse

Datenflussanalyse

Bei der Datenflussanalyse wird untersucht und aufgezeigt, wo und wie Ein-/Ausgänge und Variablen in einem CODESYS Safety Projekt verwendet werden.

Die von der PLCopen zur Verifikation geforderte Datenflussanalyse wird in CODESYS Safety erleichtert durch:

- Verwendung von FUP
- Keine Sprünge und Returns im Basic Level
- Trennung von sicheren und unsicheren Signalen
- Keine globalen Variablen in Funktionsbausteinen

Die Datenflussanalyse wird insbesondere mit Hilfe der Safety Querverweisliste durchgeführt. Es werden in dieser Liste alle Positionen im Projekt, in eingebundenen Bibliotheken oder optional in der aktuellen POU aufgeführt, an denen die Variable verwendet wird. Genauere Informationen zu dieser Funktion finden Sie in der CODESYS Safety Online-Hilfe.

Die Hinweise in *Kapitel 9.3.6.2 „Verwendung von Querverweisliste und Gehe zur Definition“ auf Seite 203* sind zu beachten.

Ermittlung, worauf ein Eingang wirkt

1. Die Variable ermitteln, worauf der Eingang wirkt:

   die Variable ist in der Registerkarte *„I/O Abbild“* des logischen E/As des entsprechenden physikalischen Geräts, bzw. der GVL für logischen Austausch (Doppelklick auf logisches E/A im Projektbaum und Öffnen der Registerkarte *„I/O Abbild“*) festgelegt.

2. Positionen der Variablen im Projekt ermitteln:

   Die Safety Querverweisliste (*„Ansicht ➔ Safety Querverweisliste“* öffnen, in die Combobox *„Name:“* den Namen der Variablen eingeben und mit *[Enter]* abschließen. In einer Spalte der Safety Querverweisliste wird die Zugriffsart auf die Variable aufgeführt. Durch einen Doppelklick auf eine Zeile der Safety Querverweisliste gelangt der Verifizierende zu der entsprechenden Stelle im Implementierungsteil der POU bei Zugriffsart *„Lesen“* oder *„Schreiben“*, bzw. zu der entsprechenden Stelle im Deklarationsteil der POU bei der Zugriffsart *„Deklaration“*.

3. Den Datenfluss der Variablen innerhalb der POUs, in denen sie verwendet wird, ermitteln:

   die Datenflusslinien der Variablen überprüfen

4. Ermitteln, auf welchen Ausgang die Variable zugewiesen wird:

   Mithilfe der Datenflusslinien und Safety Querverweisliste

5. Ermitteln, ob die Variable anderen Variablen zugewiesen wird. In diesem Fall muss der Datenfluss auch für diese Variablen gemäß den Schritten 2 bis 5 überprüft werden.

Ermittlung, was auf einen Ausgang wirkt

1. Die Variable ermitteln, die auf den Ausgang wirkt:

   die Variable ist in der Registerkarte *„I/O Abbild“* des logischen E/As des entsprechenden physikalischen Geräts, bzw. der GVL für logischen Austausch (Doppelklick auf logisches E/A im Projektbaum und Öffnen der Registerkarte *„I/O Abbild“*) festgelegt.

2. Positionen der Variablen im Projekt ermitteln:

   Die Safety Querverweisliste (*„Ansicht ➔ Safety Querverweisliste“* öffnen, in die Combobox *„Name:“* den Namen der Variablen eingeben und mit *[Enter]* abschließen. Es werden in der Safety Querverweisliste alle Positionen im Projekt, oder optional in der aktuellen POU aufgeführt, an denen die Variable verwendet wird. In einer Spalte der Safety Querverweisliste wird die Zugriffsart auf die Variable aufgeführt. Durch einen Doppelklick auf eine Zeile der Safety Querverweisliste gelangt der Verifizierende zu der entsprechenden Stelle im Implementierungsteil der POU (bei Zugriffsart Lesen oder Schreiben) oder im Deklarationsteil der POU (bei Zugriffsart Deklaration) .

3. Den Datenfluss der Variablen in den POUs, in denen sie verwendet wird, mithilfe der Datenflusslinien überprüfen.

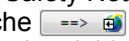
4. Ermitteln, welche Eingänge auf die Variable wirken: Mithilfe der Datenflusslinien, Safety Querverweisliste und Registerkarte *„I/O Abbild“* der logischen E/As

5. Ermitteln, ob der Variablen anderen Variablen zugewiesen werden. In diesem Fall muss der Datenfluss auch für diese Variablen gemäß den Schritten 2 bis 5 überprüft werden.

Ermittlung, worauf eine Sender-Variable wirkt

→ Im Editor der Safety Netzwerkvariablenliste (Sender) auf die Schaltfläche *„Empfänger anzeigen“* oder im Kontextmenü der Safety Netzwerkvariablenliste (Sender) im Gerätebaum den Befehl *„Alle zugehörigen Empfänger anzeigen“* wählen.

Ermittlung, was auf eine Empfänger-Variable wirkt

→ Im Editor der Safety Netzwerkvariablenliste (Empfänger) auf die Schaltfläche  klicken oder im Kontextmenü der Safety Netzwerkvariablenliste (Empfänger) im Gerätebaum den Befehl *„Gehe zu Sender“* wählen.

9.4 Dynamische Verifikation

9.4.1 Dynamische Verifikation und Validierung

VORSICHT!
Während der gesamten Verifikation liegt die Sicherheit in der Verantwortung des Verifizierenden! Dies bedeutet, dass die Sicherheit organisatorisch auch dann hergestellt werden muss, wenn die Sicherheitssteuerung "sicherer Betrieb" meldet oder der Verifizierende nicht explizit zur Herstellung der organisatorischen Sicherheit aufgefordert wird.

Durch die Verwendung der bereits zertifizierten Bausteine der Bibliotheken PLCopen und SafetyStandard und der Programmiersprache FUP mit den Sprachteilmengen Basic und Extended wird die dynamische Verifikation wesentlich vereinfacht.

Die dynamische Verifikation ist die Überprüfung des Programms mithilfe von funktionalen Tests an der laufenden Applikation oder in einer speziell dafür entwickelten Testumgebung. Dieser Teil der Verifikation ist mit White-Box und Black-Box Tests durchführbar.

HINWEIS!
Vor der Testdurchführung muss der Verifizierende prüfen, dass er mit der richtigen Steuerung verbunden ist und dass die Bootapplikation auf der Sicherheitssteuerung die richtige Pin-Kennung hat.

Bei Online-Tests (*Kapitel 9.4.2.1 „Monitoring von Variablen“ auf Seite 209*) prüft er dazu am einfachsten beim Einloggen den angezeigten Gerätenamen und die angezeigt Pin-Information. Bei Offline-Tests (*Kapitel 9.4.3 „Vollständiger Funktionstest der Applikation“ auf Seite 212*, *Kapitel 9.4.4 „Verifikation in der fertigen Maschine“ auf Seite 214*) kann er es genauso machen (und sich wieder ausloggen), oder er prüft auf in der Registerkarte *„Safety Online Information“* den angezeigten Gerätenamen und die Pin-Information.

HINWEIS!
Während des gesamten Testdesigns muss durch Prüfen der Pin-Kennung in der Objektansicht jedes Objekts der Sicherheitsapplikation verifiziert werden, dass das Objekt zur richtigen Sicherheitsapplikation gehört und dass es sich seit dem Einloggen nicht geändert hat.

9.4.2 Onlinetests

Vor jedem Test muss der Befehl *„Neu starten“* aktiviert werden.

Durch Aktivieren des Befehls *„Neu starten“* in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung wird die SPS in den Zustand für den nächsten Test gebracht, das heißt, die Bootapplikation auf der Steuerung startet.

9.4.2.1 Monitoring von Variablen

Monitoring von Variablen bei funktionalen Tests

Das Monitoring der Variablen im Deklarationsfenster und Implementierungsfenster der Objekte ist qualifiziert und damit für die Verifikation einer CODESYS Safety Sicherheitsapplikation geeignet.

HINWEIS!

Bei Verifikationstätigkeiten dürfen nur bestätigte Verbindungen verwendet werden. Monitoring per Fernzugang ist nicht für die Verifikation qualifiziert.

Wenn Sie bei bestehender Onlineverbindung verifizieren wollen, ob die Verbindung bestätigt ist, setzen Sie die zu verifizierende Applikation als aktive Applikation. In der Statuszeile muss als Betriebsmodus *„SICHER“* mit animiertem Balken zu sehen sein (siehe *Kapitel 7.5.1 „Betriebszustand und Applikationszustand“ auf Seite 173*).

HINWEIS!

Monitoring im Überwachungsfenster ist nicht qualifiziert und damit nicht für Verifikation einer Sicherheitsapplikation geeignet!

VORSICHT!

Wenn Ablaufkontrolle aktiviert ist, dann sind die im Implementierungsfenster angezeigten Werte nicht für die Verifikation einer Sicherheitsapplikation geeignet.

Dass es sich bei Werten im Implementierungsfenster um Verifikations-geeignete Monitoringwerte handelt, erkennen Sie an der Hintergrundfarbe: sie darf nicht grün sein!

VORSICHT!

Verifizieren Sie, dass der angezeigte Werte für eine Variable, der für die Verifikation verwendet wird, der qualifizierte Wert ist, der am Ende eines Applikationszyklus vorgelegen hat (zykluskonsistentes Monitoring), und nicht der nicht qualifizierte Wert der Ablaufkontrolle.

Erkennen der qualifizierten Werte

– Alle angezeigten Werte im Deklarationsteil eines Editors sind Werte, die am Ende eines Applikationszyklus vorgelegen haben und sind damit qualifiziert.
– Im Implementierungsteil des FUP-Editors sind Werte, die am Ende eines Applikationszyklus vorgelegen haben, nicht grün hinterlegt. Dabei darf der Befehl *„Online ➔ Ablaufkontrolle“* nicht aktiviert sein.

VORSICHT!

Der angezeigte Monitoringwert für eine Variable ist ein Wert, den diese Variable auf der verbundenen Steuerung hatte. Dieser Wert ist nicht unbedingt der aktuelle Wert, d.h. auf der Steuerung kann sich der Wert bereits wieder geändert haben. Die Monitoringanzeige im Safety-Editor im Online-Modus ist nur zum Nachweis geeignet, dass ein bestimmter Wert bzw. Zustand einmal eingenommen wurde. Alle in einem Safety-Editor im Online-Modus gleichzeitig angezeigten Werte haben in der angezeigten Kombination auch auf der Sicherheitssteuerung am Ende eines Applikationszyklus vorgelegen (zykluskonsistentes Monitoring). Deshalb kann die Monitoringanzeige als Hilfsfunktion eingesetzt werden, um die Zweigabdeckung von Tests anhand von Merker-Variablen zu verifizieren (siehe ↯ *„Nachweis der Zweigabdeckung “ auf Seite 211*)

Die Funktionalität "Schreiben/Forcen" darf nur für Tests einzelner Funktionsbausteine und nur zur Erzwingung eines Zustandes eingesetzt werden, der über Black-Box nicht erreichbar ist, z.B. um den Zweig mit der Behandlung illegaler Werte zu testen. Wenn es für die Testbeurteilung entscheidend ist, welcher Wert in die Anwendung geschrieben wurde, muss der Verifizierende über das qualifizierte Monitoring unter Berücksichtigung der Hinweise des Kapitels ↯ *Kapitel 7.6.1 „Monitoring“ auf Seite 179* verifizieren, dass der richtige Wert in die Variable geschrieben wurde.

Variablen werden für das qualifizierte Monitoring bei funktionalen Tests nach folgender Vorgehensweise geschrieben oder geforct:

1. Befehl *„Einloggen“* der Kategorie *„Online“*
2. Geräte-Ansicht (Projektbaum) öffnen mit Befehl *„Geräte“* der Kategorie *„Ansicht“*
3. Instanzauswahl durch Doppelklick auf das entsprechende Objekt Sicherheitsapplikation *„SafetyApp“*
4. Im Deklarationsteil der POU oder GVL durch Klick auf die Spalte *„Vorbereiteter Wert“* den Wert zum Schreiben vorbereiten
5. Befehl *„Werte schreiben“* oder *„Werte forcen“* der Kategorie *„Debug“* aktivieren
6. Die gemonitorten Variablenwerte im Implementierungs- oder Deklarationsfenster des entsprechenden Objekts anschauen.

9.4.2.2 Onlinetest im Extended Level

Nachweis der Zweigabdeckung

Bei der Programmierung mit CODESYS Safety können Zweige ausschließlich in Extended-POUs, durch die nur hier erlaubten Sprünge und Returns vorkommen. Nur in POUs im Extended Level mit bedingten Sprüngen und Returns ist auch ein Nachweis der Zweigabdeckung durchzuführen.

Beispielhaftes Verfahren

1. Der zu verifizierende Programmcode steht zur Verfügung.
2. Die Liste der Zweige der POU aus der Kontrollflussanalyse steht zur Verfügung.
3. In der POU in jeden Zweig Z ein neues Netzwerk einfügen und darin einer (über Autodeclare neu deklarierten) booleschen Variablen bZ (mit Initialwert FALSE) den Wert TRUE zuweisen.
4. Für jeden Testfall für die POU: *„Neu starten“* der Bootapplikation, Testlauf, wenn Variable bZ = TRUE, den Zweig Z in der Zweigliste abhaken.
5. Nach dem Durchlauf aller Tests, müssen alle Zweige in der Liste abgehakt sein.

Die Merker-Variablen müssen nach dem Nachweis der Zweigabdeckung im Deklarations- und im Implementierungsfenster manuell entfernt werden. Sie können auch auskommentiert werden, aber dann sollte die entsprechende Warnung deaktiviert werden.

HINWEIS!

Funktionale Tests, die mit Instrumentierung (Merkern) durchgeführt wurden, müssen nach Entfernung der Instrumentierung oder Auskommentierung wiederholt werden!

HINWEIS!
Die Anzeige von Variablen in einem Überwachungsfenster ist für die Verifikation einer Sicherheitsapplikation nicht geeignet!

Test der Grenzwerte

Sollen die Grenzwerte einer POU im Extended Level überprüft werden, so werden durch den bereitgestellten Sprachumfang die notwendigen Mittel zur Verfügung gestellt.

Der Test der Funktion erfolgt mit einem Wert der innerhalb der Grenze liegt. Ein weiterer Test wird mit einem Wert oberhalb der Grenze durchgeführt.

Der Test der Grenzwerte der FBs erfolgt im FB oder wird selbst programmiert (siehe *Kapitel 6.2.3 „Defensives Programmieren“ auf Seite 106*).

Beispielhaftes Verfahren

1. Die Grenzwerte aus der bereitstehenden Spezifikation oder dem Programmcode ermitteln.
2. Test schreiben und beschalten werden. Dabei sollte pro Grenzwert eine Instanz des Bausteins aufgerufen werden.
3. Pro Testfall überprüfen, ob Ausgang des Funktionsbausteins der Erwartung entspricht.
4. Pro Testfall einen Befehl *„Neu starten“* der Bootapplikation ausführen.

9.4.3 Vollständiger Funktionstest der Applikation

Mithilfe von funktionalen Tests wird die Sicherheitsapplikation auf der Steuerung dahingegen getestet, ob die Ausgänge auf die Beschaltungen der Eingänge entsprechend der Spezifikation reagieren. Es handelt sich hierbei um einen vollständigen Test des Input-Output-Verhaltens gegen die Spezifikation der Applikation. Dabei kann die Applikation noch weitere zu testende Eingänge und Ausgänge haben.

HINWEIS!

Es muss nicht nur die Reaktion an den Ausgängen zu Feldgeräten getestet werden, sondern auch die Reaktion an versendeten Netzwerkvariablen und an Austauschvariablen zur Standardsteuerung. Auch für zukünftige Erweiterungen (vgl. *Kapitel 13.4.2 „Änderungen in Projekten mit Querkommunikation“ auf Seite 267*) vorsorglich veröffentlichte Netzwerkvariablen ohne aktuelle Empfänger müssen getestet werden! Und die Reaktionen müssen nicht nur unter verschiedenen Beschaltungen der Eingänge von Feldgeräten getestet werden, sondern auch unter verschiedenen Beschaltungen der Netzwerkvariablen und Austauschvariablen, die die Applikation von Safety NVL-Sendern bzw. der Standardsteuerung empfängt.

Sie können einen vollständigen Funktionstest der Applikation für jede Sicherheitsapplikation der Maschine separat durchführen.

Um die Sicherheitssteuerung in den Zustand für den nächsten Test zu bringen, muss ein Reset der Hardware durchgeführt werden.

Der Test kann entweder auf der endgültigen Sicherheitssteuerung durchgeführt werden oder auf einem baugleichen Modell wie die endgültige Sicherheitssteuerung, wenn anschließend die Bootapplikation auf die endgültige Sicherheitssteuerung transferiert werden, wie im Kapitel "Hardware-Tausch" beschrieben ist. (Siehe *Kapitel 12.6.3 „Hardware-Tausch“ auf Seite 256*)

VORSICHT!

Der Transfer darf nur so erfolgen, wie es im Kapitel "Hardeware-Tausch" beschrieben ist, nicht durch die Programmiersystemfunktion *Bootapplikation erzeugen*.

Wenn eine Bootapplikation mit *„Bootapplikation erzeugen“* erzeugt wird, dann muss dies Bootapplikation auf dieser Sicherheitssteuerung neu getestet werden.

9.4.4 Verifikation in der fertigen Maschine

VORSICHT!

Die im Folgenden erläuterten Prüfungen der Eindeutigkeit der Adressen sind zwingend erforderlich, um mit SIL3-Sicherheit auszuschließen, dass durch einen Fehler auf der Kommunikationsstrecke eine Sicherheitssteuerung mit dem falschen sicheren Feldgerät verbunden werden könnte bzw. mehrere Sicherheitssteuerungen mit dem selben sicheren Feldgerät verbunden werden könnten.

In der fertigen, kompletten Maschine, in der die separat funktionsgetesteten Sicherheitsapplikationen installiert sind, ist zu verifizieren:

- Eindeutigkeit der Adressen:
  - Die an den sicheren Feldgeräten in der Maschine eingestellten Safety-Adressen sind maschinenweit eindeutig.
  - Die in allen Sicherheitsapplikationen der Maschine konfigurierten Safety-Adressen. Das heißt bei PROFIsafe-Geräten sind "F_Source_Add" und "F_Dest_Add" eindeutig. Bei FSoE-Geräten sind "FSoE Adress" und "Connection ID" eindeutig. Die Safety-Adresse jeder Sender-Netzwerkvariablenliste und die Connection ID jeder Empfänger-Netzwerkvariablenliste sind maschinenweit eindeutig.
- Richtige Gerätebeschreibungen: Für jedes Feldgerät mit eingestellter Safety-Adresse X in der Maschine verwendet das entsprechende Geräteobjekt mit konfigurierter Safety-Adresse X im Projekt die passende Beschreibungsdatei (Registerkarte *„Sichere Konfiguration“*

  Diese Verifikation kann durch Testen gegen die Spezifikation, Schnittstellen (siehe *Kapitel 9.3.6 „Applikationsspezifische Prüfungen“ auf Seite 202* ersetzt werden.)
- Richtige Prozesssignale: An die Feldgeräte mit eingestellter Safety-Adresse X sind in der Maschine Sensoren und Aktoren so angeschlossen, dass sie zur Verwendung des Abbilds des Geräteobjekts mit konfigurierter Safety-Adresse X im Projekt passen.
- Sicherheitsvalidierung: Im Rahmen der Sicherheitsvalidierung der Maschine wird die Realisierung der spezifizierten Sicherheitsfunktionen durch die Gesamtheit der (kommunizierenden) Sicherheitssteuerungen in der Maschine verifiziert.

HINWEIS!

Vor der Validierung muss sichergestellt werden, dass die Sicherheitsapplikation, die sich auf der Steuerung befindet, gepinnt ist und dass der Pin und die Pin-Kennung mit der zu verifizierenden Sicherheitsapplikation übereinstimmt.

Bei der Validierung werden alle Sicherheitsfunktionen, die in der Sicherheitsapplikation programmiert worden sind, direkt an der Anlage getestet. Es wird getestet, ob die Sicherheitsfunktionen tatsächlich funktionieren und die Sicherheitsapplikation für die Anlage geeignet ist.

10 Software-Abnahme und Dokumentation

10.1 Einleitung

In diesem Kapitel werden die Voraussetzungen, die erforderlichen Nachweise und die Funktionen zur Archivierung des Projekts und Ausdruck der Abnahmedokumentation beschrieben.

HINWEIS!
Während des gesamten Prozesses von der Entwicklung bis zur Abnahme einer Sicherheitsapplikation muss von Anwenderseite dafür gesorgt werden, dass die richtigen Objektstände, die richtigen Bibliotheken und richtigen Gerätebeschreibungsdateien verwenden werden (siehe *Kapitel 8 „Pinnen der Software“ auf Seite 189*)

HINWEIS!
Der Anwender ist dafür verantwortlich, dass bei der Abnahme der Sicherheitsapplikation alle dafür relevanten Normen eingehalten werden.

HINWEIS!
Für die Beurteilung der richtigen Verwendung der Schnittstelle von externen FBs ist die zu dieser Steuerung gehörende Anwenderdokumentation für diesen FB in der nachgewiesenen Version heranzuziehen.

HINWEIS!
Die CODESYS Standard Projektansicht ist für die Verifikation und Abnahme einer Sicherheitsapplikation **nicht geeignet**. Zum Nachweis, welche Objekte zur Sicherheitsapplikation gehören muss die Vergleichsansicht verwendet werden (siehe *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)

HINWEIS!
Der CODESYS Standard Projektvergleich ist für die Verifikation und Abnahme einer Sicherheitsapplikation **nicht geeignet**. Er kann lediglich als Hilfsfunktion verwendet werden, um die Vergleichsansicht des gepinnten Stands der Applikation zu öffnen. Die Vergleichsansicht wird durch einen Doppelklick auf das Safety Applikationsobjekt im Standard Projektvergleich geöffnet.

10.2 Voraussetzungen und Nachweise für die Abnahme

Für die Abnahme muss eine CODESYS-Version und CODESYS Safety-Version zur Verfügung stehen, die das Format und die Ausführungsversion der Version unterstützt, mit der die Sicherheitsapplikation verifiziert wurde.

Prüfung A1

HINWEIS!
Die vorliegende CODESYS Safety-Version wurde bei Entwicklung und Testen der Sicherheitsapplikation verwenden. Diese CODESYS Safety-Version ist in der aktuellen Revisionsliste in der Zertifikatsdatenbank des TÜV zugelassen (siehe *„Hinweis Installation1“ auf Seite 13*). Gegebenenfalls bekannte Probleme wurden als nicht relevant für die Applikation bewertet.

Die vorliegende CODESYS Safety Version lässt sich mit dem Befehl *„Hilfe ➔ Safety Versionsinformation anzeigen...“* anzeigen.

Prüfung A2

HINWEIS!
Das Projekt liegt als CODESYS-Projektarchiv auf einem Speichermedium zum Archivieren vor (siehe *Kapitel 10.3.1 „Archivierung“ auf Seite 223*).

Das Projektarchiv mit der passenden CODESYS-Version öffnen.

Prüfung A3

HINWEIS!
Das Projekt im Projektarchiv muss eine Benutzerverwaltung besitzen (das in CODESYS Safety bereitgestellte Benutzerverwaltungs-Template oder eine vom Anwender eigens erstellte Benutzerverwaltung).

In der Benutzerverwaltung unter einem anderen Benutzernamen anmelden, als der, welcher bei Entwicklung und Verifikation benutzt wurde.

Prüfung A4

HINWEIS!
Die Applikation im Projektarchiv ist gepinnt. Es werden keine Abweichungen des Projektstands vom gepinnten Stand gemeldet (nicht "In Work")

Dazu das Projektarchiv öffnen. Die Pin-Information im Editor des Safety Applikationsobjekts im Projekt mit dem Ausdruck und mit den Review- und Testberichten vergleichen.

Prüfung A5

HINWEIS!
Nur gültige Versionen vordefinierter Bausteine werden von der Applikation verwendet.

Welche Bausteine in welcher Version verwendet werden sieht man im Editor des Safety Applikationsobjekts, Registerkarte *„Objekte“*. Ob eine Version gültig ist, steht in ↳ *Kapitel 15.2 „Spezifische Sicherheitshinweise für applikative Bibliotheksbausteine“ auf Seite 298*

HINWEIS!
Für den Nachweis, welche Bibliotheken oder Bibliotheksbausteine in welchen Versionen verwendet werden, ist der Bibliotheksverwalter nicht geeignet. Für diesen Nachweis muss die Vergleichsansicht des Safety Applikationsobjekts verwendet werden (siehe ↳ *„Editor des Safety Applikationsobjekts mit Objektliste“ auf Seite 66*)

Prüfung A6

HINWEIS!
Der Stand der Applikation im Projektarchiv stimmt mit dem gereviewten und getesteten Stand überein oder es gibt eine Erklärung und Bewertung für Abweichungen.

Für diese Prüfung sollte die im Applikationsobjekte angezeigte Pin-Information mit der in Reviewberichten und Testberichten vermerkten Pin-Information abgeglichen werden.

Prüfung A7

HINWEIS!
Wenn sich mehrere Sicherheitssteuerungen im Projekt befinden, muss den entsprechenden Steuerungen in der Maschine ein eindeutiger Gerätename gegeben worden sein.

Welchen Gerätenamen eine Steuerung in der Maschine hat, verifiziert man, wenn man sich unter einem neuen Benutzernamen das erste Mal mit der Steuerung verbindet (siehe ↳ *Kapitel 7.2.2 „Verbindungsaufbau“ auf Seite 164*). Dabei wird ihre Identität durch eine Aktion an der Steuerung oder durch eine Seriennummer oder ähnliches bestätigt, während ihr Gerätenamen angezeigt wird. Für die Eindeutigkeit muss man sich mit jeder Sicherheitssteuerung des Projekts einmal verbinden. Dies muss ohnehin getan werden, wenn man (der Reihe nach) alle Sicherheitssteuerungen des Projekts abnimmt.

Prüfung A8

HINWEIS!
Der Stand der Applikation im Projektarchiv stimmt mit dem Stand auf der Steuerung überein.

Für diese Prüfung verbindet man sich aus dem Projekt heraus über eine bestätigte Verbindung mit der Steuerung: Entweder man loggt sich in die Applikation ein. Dann muss der Dialog beim Einloggen den Gerätenamen der abzunehmenden Steuerung anzeigen und eine Übereinstimmung des Pins melden. Alternativ kann man sich auf der Registerkarte *„Safety Online Information“* den Gerätenamen und die Pin-Information der Steuerung anzeigen lassen und mit der Pin-Information im Editor des Safety Applikationsobjekts vergleichen.

HINWEIS!
Für die Abnahmeprüfungen darf nur eine bestätigte Verbindung verwendet werden. Der Fernzugang ist nicht für Abnahmeprüfungen qualifiziert.

Entweder Sie haben die Verbindung zur Steuerung beim ersten Onlinedienst bestätigt oder Sie verifizieren die Angabe auf der Registerkarte *„Safety Online Information“*, dass die Verbindung eine bestätigte Verbindung ist.

HINWEIS!
Dabei muss darauf geachtet werden, dass man mit der richtigen, der abzunehmenden Steuerung verbunden ist.

Entweder hat man beim Verbinden die Identität der Steuerung bestätigen müssen (siehe *„Verbindungsbestätigung“ auf Seite 165*). Oder man prüft den angezeigten Gerätenamen beim Einloggen bzw. auf der Registerkarte *„Safety Online Information“*.

Prüfung A9

HINWEIS!
Die Firmware auf der Steuerung hat einen gültigen Stand.

Welcher Stand sich auf der Steuerung befindet, wird auf der Registerkarte *„Safety Online Information“* angezeigt. Ob der Stand gültig ist, erfahren Sie vom Gerätehersteller.

Prüfung A10

HINWEIS!
Für die Abnahme muss geprüft werden, ob im Projekt tatsächlich die zum Gerät in der Maschine passende Gerätebeschreibung zum Einsatz kommt.

Die im Eintrag Device Info in Konfiguratoren und Geräteparametern logischer Geräte identifizierten Gerätebeschreibungsdateien müssen die richtigen Beschreibungen für die damit in der Maschine verbundenen Feldgeräte sein.

Prüfung A11

HINWEIS!
Die Konfiguration und der Systemaufbau erfüllen die spezifischen Systemanforderungen für die eingesetzten sicheren Feldbus- bzw. Kommunikationstechnologien, siehe *Kapitel 14.2.3 „PROFIsafe-spezifische Nachweise für die Abnahme“ auf Seite 282* bzw. *Kapitel 14.3.3 „FSoE-spezifische Nachweise für die Abnahme“ auf Seite 286* bzw. *Kapitel 14.4.3 „Safety NetVar-spezifische Nachweise für die Abnahme“ auf Seite 290*.

Prüfung A12

HINWEIS!

Die Dokumentation der Maschine muss Betreiber und Integratoren über die Sicherheitshinweise für den Betrieb von Sicherheitssystemen mit CODESYS Safety aus *Kapitel 12 „Betrieb“ auf Seite 237* informieren (außer sie sind für die anzunehmende Maschine nicht anwendbar).

Prüfung A13

HINWEIS!

Bei Verwendung von Netzwerkvariablen muss die Dokumentation der Maschine die Betreiber und Integratoren über alle verwendeteten Safety-Adressen und Connection IDs von Netzwerkvariablenlisten informieren. Das ist nötig, wenn das Maschinennetzwerk einmal erweitert werden soll, oder um die Sicherheitssteuerung bzw. Teilmaschine mit anderen Maschinenteilen zur einer Gesamtmaschine kombinieren zu können (siehe auch *Kapitel 10.4 „Dokumentation für Betreiber und Integratoren“ auf Seite 226*).

Prüfung A14

HINWEIS!

Die Informationen der Sicherheitsapplikation und der Geräte der Sicherheitsapplikation stehen als Ausdruck zur Verfügung. Drucken siehe *Kapitel 10.3.2 „Ausdruck Projektdokumentation“ auf Seite 225*.

Prüfung A15

HINWEIS!

Der Gerätename der Steuerung und der Stand der Firmware sind dokumentiert.

Der Gerätename und der Stand der Firmware, der auf der Steuerung läuft, werden auf der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung angezeigt)

HINWEIS!

Gerätename und Firmwarestand der Steuerung sind nicht in dem durch den Befehl *„Dokumentieren“* der Kategorie *„Projekt“* erstellten Ausdruck enthalten und müssen deshalb extra erzeugt werden.

Zur Abnahmedokumentation gehören:

- Projektausdruck (*Kapitel 10.3.2 „Ausdruck Projektdokumentation“ auf Seite 225*)
- Verifikationsberichte (*Kapitel 9 „Software-Verifikation“ auf Seite 195*)
- Gerätenamen der Steuerung und Stand der Firmware

10.3 Funktionen für die Abnahme

10.3.1 Archivierung

Vor der Archivierung muss das Projekt gespeichert werden (Befehl *„Projekt speichern“* des Datei-Menüs).

Mit Hilfe der Archivierung werden sämtliche zu einer Sicherheitsapplikation gehörenden Objekte in eine einzige Datei zusammengefasst.

Diese Datei kann als Datensicherung abgelegt und im Bedarfsfall wieder entpackt werden.

Das Archivieren des Projekts für die Abnahme der Sicherheitsapplikation kann über die Funktionalität der Projektarchivierung erfolgen. Dies erfolgt mit dem Befehl „Archiv speichern/versenden“ des Untermenüs „Projektarchiv“ der Kategorie „Datei“. Im Dialog „Projektarchiv“ müssen alle für die Abnahme relevanten Informationen selektiert werden.

Folgende Informationen müssen im Dialog *„Projektarchiv“* zur Archivierung der Sicherheitsapplikation ausgewählt werden:

- Bibliotheksprofil
- Optionen
- Referenzierte Bibliotheken
- Referenzierte Geräte

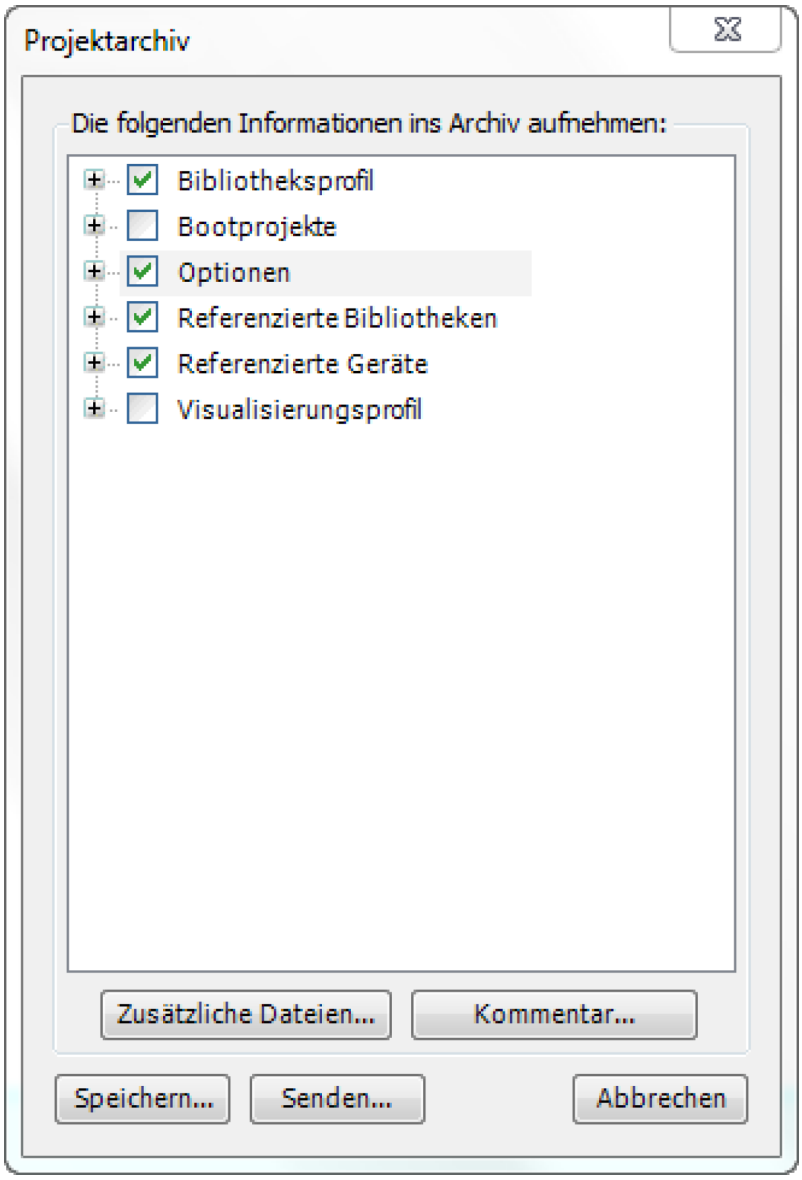


Abb. 102: Dialog 'Projektarchiv'

Ins Projektarchiv mit aufgenommen werden durch korrekt erfolgte Selektion im Auswahldialog

- alle Sicherheitsapplikationen mit Kanalmappings und Austauschlisten
- alle Feldbuskonfigurationen und Geräteparametrierungen
- alle verwendeten Bibliotheken

10.3.2 Ausdruck Projektdokumentation

Zum Ausdruck der Projektdokumentation wird in CODESYS Safety der Befehl *„Projekt ➔ Dokumentieren“* verwendet. Es wird dabei ein Ausdruck aller markierten Objekte, inklusiv der Objektinformationen, Prüfsumme und Pin-Information (Pin-Kennung) erstellt. Mit den Objekt-Ansichten (Objektinhalt) wird auch zu jedem Objekt die Pin-Kennung und falls das Objekt nicht gepinnt ist die Information *„In Work“* mit ausgedruckt. Der Ausdruck enthält optional zusätzlich ein Deckblatt mit den Informationen: Datei, Datum und Profil sowie ein Inhaltsverzeichnis.

Die Projektdokumentation enthält folgende, für die Abnahme relevante Informationen:

- die Version von CODESYS Safety
- die verwendeten Quellen
- wiederverwendete vordefinierte Bausteine und ihre Version
- die sicherheitsgerichtete Konfiguration und Geräteparameter sicherer Feldgeräte

Für den Ausdruck der Abnahmedokumentation werden im Dialog „Projekt dokumentieren“ die Sicherheitssteuerung und alle ihre Unterknotenpunkte und Objekte markiert.

Detailinformationen zur Dokumentierung eines Projekts siehe Onlinehilfe von CODESYS Standard

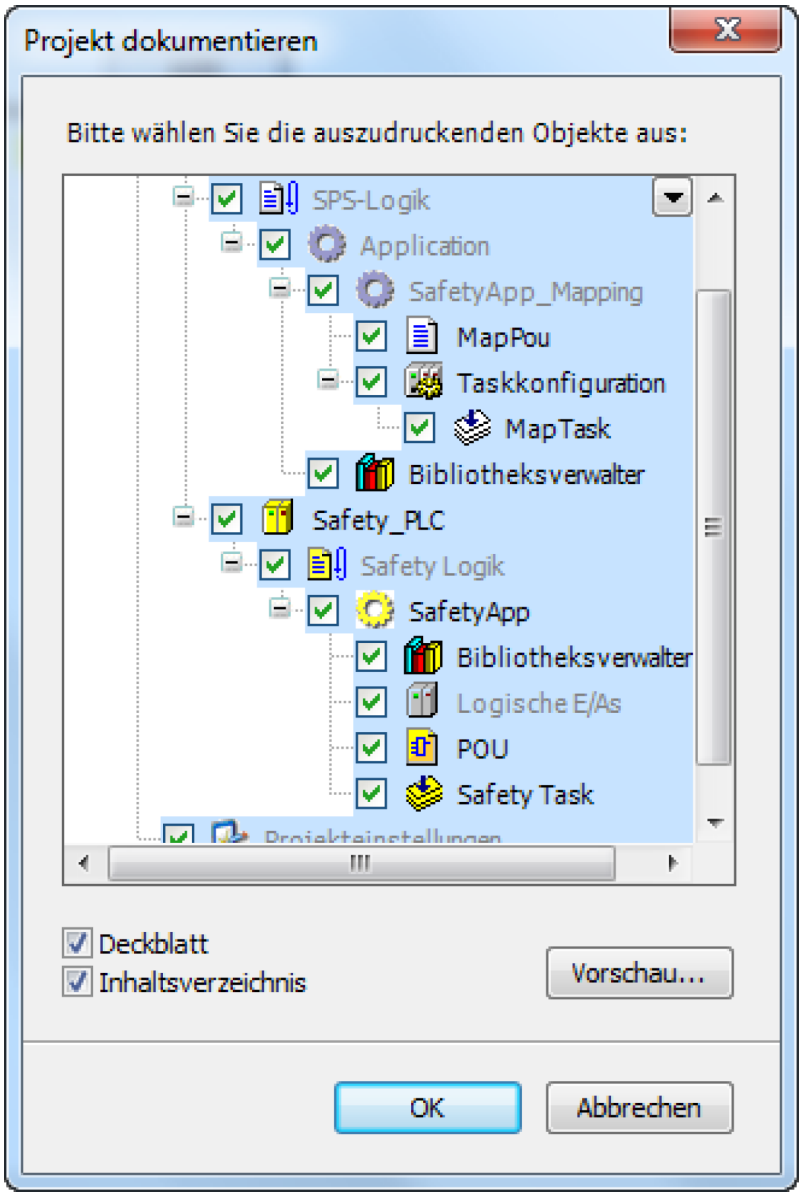


Abb. 103: Dialog 'Projekt dokumentieren'

10.4 Dokumentation für Betreiber und Integratoren

Dokumentation von Safety-Adressierungen bei Verwendung von Safety Netzwerkvariablen

Die Safety-Adressierungen, die auf Eindeutigkeit bzw. korrekte Verbindung geprüft werden müssen, müssen den CODESYS-Projekten entnommen werden, mit denen die Steuerungen programmiert wurden. Da dieses mehrere Projekte sein können, muss der Inbetriebnehmer oder Betreiber die Safety-Adressierungen aus allen Projekten kennen.

Folgende Safety-Adressierungen sind relevant:

- Safety-Adressierungen, die für Querkommunikation zwischen Projekten verwendet werden sollen
- Safety-Adressierungen, die für projekt-lokale NVL-Verbindungen verwendet werden sollen

HINWEIS!

In der Dokumentation der Sicherheitsapplikation sind alle verwendeten Safety-Adressierungen von NVLs und in der Dokumentation einer Maschine alle verwendeten Safety-Adressierungen aller eingebauten Sicherheitssteuerungen aufzuführen:

– für alle enthaltenen Sender-NVLs: Auflistung der Safety-Adresse, die sie belegen bzw. unter der sie erreichbar sind, ggf. mit einem Vermerk, dass sie intern sind.
– für alle enthaltenen Empfänger-NVLs: Auflistung der Connection ID, die sie belegen.
– Auflistung der Safety-Adressen der verknüpften Sender-NVL, mit der sie eine Verbindung aufbauen können.

Dokumentation implementierter F-Module für die Host-Programmierung

Hinweis FDev_6 (Doc. WCDT)

HINWEIS!

Der F-Device-Programmierer muss für den F-Host-Programmierer die maximale Bearbeitungszeit WCDT_IN und WCDT_OUT für jedes Signal zwischen F-Host und Sensor bzw. Aktor dokumentieren (vergleiche Formel zur Gesamtreaktionszeit-Obergrenze in ↬ *Kapitel 4.3.3 „Reaktionszeiten bei F-Device-Steuerungen“ auf Seite 43*).

Hinweis FDev_7 (Doc. device fault)

HINWEIS!

Es muss dokumentiert werden, ob die Applikation nach Gerätefehler (PROFIsafe-Statussignal Device_fault) einen automatischen Wiederanlauf der Prozesskommunikation implementiert oder ob ein Neustart erforderlich ist (vergleiche ↬ *Kapitel 6.4 „Implementierung von F-Modulen“ auf Seite 143*), damit der F-Host-Programmierer das Verhalten des F-Moduls in seiner Applikation berücksichtigen kann.

Hinweis FDev_8

HINWEIS!

Damit der F-Host-Programmierer das programmierte F-Device erfolgreich konfigurieren kann, muss der F-Device-Programmierer zu jedem F-Modul dokumentieren (zum Beispiel in einem Gerätehandbuch für das programmierte F-Device) welches Modul es ist: Modulname in der Gerätebeschreibungsdatei GSDML, Abbildstruktur und unterstützte PROFIsafe-Funktionen bzw. Parameter.

Hinweis FDev_9

HINWEIS!

Damit der F-Host-Programmierer das programmierte F-Device erfolgreich konfigurieren kann, muss der F-Device-Programmierer zu jedem F-Modul dokumentieren (zum Beispiel in einem Gerätehandbuch für das programmierte F-Device), welche Werte er in der Sicherheitsapplikation für die Parameter *„Source Address “* und *„Destination Address“* für jedes F-Moduls voreingestellt hat (vergleiche ↳ *Kapitel 14.5.2 „F-Modul Parameter“ auf Seite 292*).

Hinweis FDev_10

HINWEIS!

Die für die PROFIsafe notwendige Eindeutigkeit der F_Adressen "Codenames" (F_Source_Add und F_Dest_Add) muss der F-Host-Programmierer nicht nur für alle F-Module im PROFINET-Netzwerk seines F-Hosts gewährleisten; zusätzlich müssen diese auch eindeutig gegenüber den F-Adressen aller F-Module in den untergeordneten Feldbussen angeschlossener CODESYS-F-Devices sein. Dies gilt insbesondere beim Einbau mehrerer Klone des gleichen programmierten F-Devices.

Hinweis FDev_11

HINWEIS!

Die für Safety-Protokolle im untergeordneten Feldbus notwendige Eindeutigkeit entsprechender Safety-Adressen kann nicht lokal pro F-Device garantiert werden. Um mehrere F-Devices in einem PROFINET-Netzwerk sicher zu betreiben, müssen die Safety-Adressen jedes untergeordneten Feldbusses (z. B. FSoE) eindeutig gegenüber den Safety-Adressen in den untergeordneten Feldbussen anderer angeschlossener CODESYS-F-Devices sein. Welche Konfigurationsparameter konkret die eindeutige Safety-Adressen enthält, ist abhängig vom Safety-Protokoll im untergeordneten Feldbus. Dies gilt insbesondere beim Einbau mehrerer Klone des gleichen programmierten F-Devices.

11 Software-Aktualisierung

11.1 Überblick Versionierung

In diesem Kapitel werden die mögliche Auswirkungen der Installation einer neuen Geräte-, Firmware-, oder CODESYS-Version auf ein bereits verifiziertes bzw. abgenommenes Projekt und Folgen der Installation eines nicht für CODESYS Safety freigegebenen Packages beschrieben.

11.2 Aktualisieren der Geräteversion

Ein geänderter Inhalt einer Gerätebeschreibung kann dazu führen, dass die Sicherheitsapplikation nicht mehr gepinnt, sondern wieder "In Work" ist. Die in der Vergleichsansicht des Safety Applikationsobjekts aufgelisteten, von der geänderten Gerätebeschreibung betroffenen Safety-Objekte mit geänderter Prüfsumme müssen einer Einflussanalyse unterzogen und anschließend neu gepinnt und verifiziert, und gegebenenfalls neu abgenommen werden.

Steht eine neue Version einer Gerätebeschreibung zur Verfügung, so sollte sich der Service-Mitarbeiter vor dem Installieren der neuen Gerätebeschreibungsversion genau darüber informieren, um welche der möglichen Varianten einer neuen Version es sich handelt und welche Auswirkungen dies auf das Projekt und damit auch auf die Verfikation und die Abnahme haben wird:

- Das neue Gerät ist kompatibel zum bisherigen Gerät:
  Nach Installieren der neuen Gerätebeschreibung im Geräte-Repository und Aktualisieren (über Kontextmenu im Gerätebaum) des Geräts, treten beim Übersetzen der Sicherheitsapplikation keine Fehler auf. Es müssen keine weiteren Tätigkeiten ausgeführt werden.
- Es handelt sich um ein Update eines Safety-Gerätes:
  Das Update kann geänderte Geräte-Parameter oder eine geänderte E/A-Struktur enthalten, die Änderungen an Safety-Objekten der Sicherheitsapplikation nach sich ziehen.
  Die in der Vergleichsansicht des Safety Applikationsobjekts aufgelisteten, von der geänderten Gerätebeschreibung betroffenen Safety-Objekte mit geänderter Prüfsumme müssen einer Einflussanalyse unterzogen und anschließend neu gepinnt und verifiziert werden.
- Es handelt sich um ein Update eines Standard-Feldgeräts:
  – Werden keine E/A-Daten des Standard-Feldgerätes über die logischen E/As in die Sicherheitsapplikation gemappt, ist in der Sicherheitsapplikation nichts zu tun. Die Sicherheitsapplikation ist nicht betroffen.
    Die Vergleichsansicht der Sicherheitsapplikation muss geprüft werden: Sie bleibt unverändert.
  – Werden E/A-Daten des Standard-Feldgeräts über logischen E/As in die Sicherheitsapplikation gemappt, so kann das Update Änderungen an Safety-Objekten der Sicherheitsapplikation nach sich ziehen.
    Die in der Vergleichsansicht des Safety Applikationsobjekts aufgelisteten, von der geänderten Gerätebeschreibung betroffenen Safety-Objekte mit geänderter Prüfsumme müssen einer Einflussanalyse unterzogen und anschließend neu gepinnt und verifiziert werden.

Neue Gerätebeschreibungsdateien müssen zuerst im Geräte-Repository installiert werden (Detailinformationen siehe ↳ Kapitel 5.3 „Geräteverwaltung “ auf Seite 56). Zur Aktualisierung des Geräts muss das physikalische Gerät im Projektbaum selektiert und der Befehl „Gerät aktualisieren...“ des Kontextmenüs aktiviert werden.

Die Detailinformationen zu den installierten Gerätebeschreibungen sind im Geräte-Repository verfügbar. Hierzu muss das entsprechende Gerät selektiert und die Schaltfläche „Details...“ aktiviert werden.

11.3 Aktualisieren der Firmware, die Ausführungsversion

Ein Update der Firmware führt in der Regel nicht zum Verlust der Abnahme Ihrer Sicherheitsapplikation. In diesem Kapitel wird beschrieben, unter welchen Umständen Sie Ihre Abnahme verlieren und wann eine bereits abgenommene Sicherheitsapplikation Ihre Abnahme behält.

Wie geschieht ein Firmware-Update

Expliziter Firmaware-Update: Aktivieren der Schaltfläche *„Firmware-Update“* in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung.

Impliziter Firmware-Update: Verwendung einer neueren Sicherheitssteuerung mit einem anderen Firmware-Stand als dem, mit dem die Sicherheitsapplikation abgenommen wurde.

Firmwareversion und Ausführungsversion

Welcher Firmwarestand sich auf Ihrer Hardware befindet, können sie mit Hilfe der Registerkarte *„Safety Online Information“* auslesen. Im untersten Ausgabefeld wird der Firmwarestand ausgegeben. Der Firmwarestand hat keine direkte Auswirkung auf die Bearbeitung Ihrer Sicherheitsapplikation. Änderungen in der Abarbeitung der Sicherungsapplikation werden getrennt von der Firmwareversion in der sogenannten **Ausführungsversion** sowie in den Versionen der Bibliotheken angezeigt. Die Ausführungsversion findet sich im Eigenschaften-Dialog der Sicherheitsapplikation in der Registerkarte *„Safety“*. Gleiche Ausführungsversion bei gleicher Sicherheitsapplikation garantiert eine identische Bearbeitung. Eine Abnahme der Sicherheitsapplikation bleibt erhalten.

Die Ausführungsversion der Bootapplikation auf der Steuerung ist im Online-Zustand in der Registerkarte „Safety Online Information“ des Editors der Sicherheitssteuerung ersichtlich.

Firmware und Gerätebeschreibung

Alle für CODESYS relevanten Informationen zur Firmware und Hardware eines Geräts sind in der sogenannten Gerätebeschreibung hinterlegt (siehe Standard CODESYS). Passt die Gerätebeschreibung nicht zum Gerät, mit dem sie verbunden wird, erscheint eine Fehlermeldung wie in CODESYS Standard. Ein Verbinden mit der falschen Gerätebeschreibung ist somit unmöglich. In der Gerätebeschreibung ist auch hinterlegt, welche Ausführungsversionen das Gerät unterstützt und welche Versionen der Bibliotheken sich auf dem Gerät befinden.

Ausführungsversion ändern

In der Regel wird ein neues Gerät alte Ausführungsversionen unterstützen. Durch Beibehalten der Ausführungsversion bleibt das Laufzeitverhalten Ihrer Sicherheitsapplikation erhalten. Wird eine Ausführungsversion allerdings nicht mehr unterstützt, so sollte die neueste Version eingestellt werden. In diesem Fall verliert die Sicherheitsapplikation ihren Pin.

Änderung der Ausführungsversion

Die Ausführungsversion kann im Dialog „Eigenschaften“ des Applikationsobjekts unter dem Tab „Safety“ ausgewählt werden.

Änderung einer Bibliotheksversion: Durch Aktualisieren der Gerätebeschreibung kann es passieren, dass neuere Bibliotheksversionen angezogen werden. Dies erfolgt automatisch so, dass die eingestellten Bibliotheken immer zum verwendeten Gerät passen. Wenn sich Bibliotheksbausteine geändert haben, haben diese eine andere Prüfsumme und der Anwender muss seine Applikation neu pinnen, verifizieren und abnehmen.

Ein Update der Gerätebeschreibung der Sicherheitsapplikation kann zum Verlust der Abnahme der Sicherheitsapplikation führen. Informieren Sie sich bei Ihrem Gerätehersteller, ob ein Update problemlos möglich ist.

Verwenden einer anderen Bibliotheksversion ändert unter Umständen den Pin eines verwendeten Bibliotheksbausteins. Ob dies der Fall ist, ist der Vergleichsansicht der Sicherheitsapplikation zu entnehmen.

In diesem Fall muss die Applikation neu gepinnt, verifiziert und abgenommen werden.

Mit der Gerätebeschreibung der Sicherheitssteuerung werden automatisch die richtigen Bibliotheken zu dieser Sicherheitssteuerung in der richtigen Version angezogen. Der Bibliotheksverwalter dient lediglich der Übersicht über die vorhandenen Bibliotheken und Ihre Bausteine. Es wird empfohlen, Bibliotheken nicht manuell zu entfernen und hinzuzufügen.

11.4 Aktualisieren der CODESYS-Version

Für CODESYS Safety sind 2 Versionen von Bedeutung: Die Version von CODESYS, sowie die Version der Erweiterung CODESYS Safety (beide Informationen sind im Menü *„Hilfe“*, Befehl *„Safety-Versionsinformation anzeigen...“* zu finden)

CODESYS und CODESYS Safety werden so weiterentwickelt , dass eine Softwareaktualisierung ohne Verlust der Abnahme einer Sicherheitsapplikation möglich ist.

Neuere Versionen von CODESYS und CODESYS Safety sollten über den Hersteller der Sicherheitssteuerung bezogen werden. Nur dieser kann die Auswirkungen einer Software-Aktualisierung bewerten. Die beiden Softwarepakete sollten stets gemeinsam aktualisiert werden, außer der Hersteller der Sicherheitssteuerung gibt Empfehlungen.

11.5 Erweiterung von CODESYS durch Packages

Mit den Funktionen des Package Managers (siehe CODESYS) kann ein um CODESYS Safety erweitertes CODESYS Profil um weitere Plug-ins erweitert werden.

Erweiterung des CODESYS Safety durch Packages

HINWEIS!
Mit dem Installieren zusätzlicher Packages kann die CODESYS-Installation ihre Qualifizierung verlieren. Siehe . *„Hinweis Installation2“ auf Seite 14*

HINWEIS!
Ein nachinstalliertes Package könnte eine gültige Kombination von CODESYS und CODESYS Safety in unzulässiger Weise verändern. Dann meldet sich nach dem Neustart spätestens bei der ersten Verwendung von Safety Funktionen die Safety Installationsprüfung mit der Meldung, dass die aktuelle CODESYS Safety Installation ungültig ist und die Applikation beendet wird (vgl. *Kapitel 2.7 „Umgang mit Fehlermeldungen aus CODESYS Safety“ auf Seite 16*).

Software-Aktualisierung

Erweiterung von CODESYS durch Packages

12 Betrieb

Im Betrieb der Maschine ist die Sicherheit durch die abgenommenen Sicherheitsfunktionen gegeben. Im Betrieb läuft die Maschine für sich oder wird gewartet (offline), oder eine CODESYS-Instanz ist mit einer Steuerung der Maschine verbunden, zum Beobachten oder zur Administration (online).

Dieses Kapitel beschreibt Maßnahmen und Verfahren, die während des Betriebs der Maschine (offline und online) beachtet werden müssen, unter anderem um die Sicherheitsfunktionen zu erhalten.

12.1 IT-Sicherheit im Betrieb

Ohne Security kein Safety!

VORSICHT!
Das Konzept zur Sicherheit der Maschine geht von bestimmten betrieblichen Rahmenbedingungen und Risiken aus. Das Betriebskonzept der Maschine muss die IT-Sicherheit (Security) der "Standard"-Steuerungseinrichtungen gewährleisten. Ansonsten könnten Angreifer die Maschine außerhalb dieser Rahmenbedingungen laufen lassen, so dass zusätzliche oder vergrößerte Risiken bestehen, welche die abgenommenen Sicherheitsfunktionen nicht (mehr) beherrschen (Safety).

VORSICHT!
Das Betriebskonzept der Maschine muss die IT-Sicherheit (Security) der Sicherheitssteuerung gewährleisten. Ansonsten könnten Angreifer die Sicherheitsfunktionen so kompromittieren, dass sie selbst bei unveränderten betrieblichen Rahmenbedingungen keine funktionale Sicherheit (Safety) mehr bieten.

Für den Einsatz im Betrieb sollte der Zugriffsschutz der Sicherheitssteuerung eingerichtet werden. Insbesondere sollten eventuell während der Applikationsentwicklung verwendete Passwörter (siehe ↬ *Kapitel 5.2.4 „Einrichten des Admin-Passworts auf der Steuerung“ auf Seite 56*) durch Passwörter für den Betrieb ersetzt werden. Bei Bedarf kann auch ein lesender Fernzugriff ohne Bestätigung ermöglicht werden.

Während des Betriebs sollten Zugriffsschutz und Fernzugriffsmöglichkeit zu geeigneten Zeitpunkten neu bewertet und gegebenenfalls angepasst werden.

Um den Zugriffsschutz der Standardsteuerung und der Sicherheitssteuerung zu gewährleisten, müssen die im Folgenden aufgeführten Hinweise beachtet und die Maßnahmen durchgeführt werden.

HINWEIS!
Auf jeder Ebene kann eine Verletzung der Schutzmechanismen gegen unerlaubte Zugriffe von Außen eine Bedrohung für das Sicherheitssystem darstellen!

Überblick über die Schutzebenen

- Barriere 1: Zugriffsschutz für die Maschine
- Barriere 2: Zugriffsschutz für die Standardsteuerung
- Barriere 3: Grundsätzlicher Selbstschutz der Sicherheitssteuerung
  - Barriere 3a. Identifizierung der Sicherheitssteuerung
  - Barriere 3b. Fernpasswort
- Barriere 4: Operationsabhängiger Schutz der Sicherheitssteuerung
  - Barriere 4a. Administrationspasswort
  - Barriere 4b. Benutzerverwaltung im Projekt

12.1.1 Schutzmaßnahmen im Umfeld der Sicherheitssteuerung

Barriere 1: Zugriffsschutz für die Maschine

Das Steuerungssystem einer Maschine kann aus mehreren Steuerungen bestehen. Allgemein gilt, dass ein zwischen ihnen bestehendes Netzwerk (Maschinennetzwerk) geschützt werden sollte.

Wenn das Maschinennetzwerk für Safety Netzwerkvariablen verwendet wird (Topologie T3), reicht eine Zugriffschutz auf jeder einzelnen Steuerung nicht mehr aus. Es kann immer noch die sichere Querkommunikation gestört werden oder es können sogar falsche Daten eingeschmuggelt werden.

Aufgrund von Sicherheitsanforderungen (vgl. *Kapitel 14.4 „Netzwerkvariablen“ auf Seite 286*) **muss** dann das Netzwerk mit allen teilnehmenden Steuerungen **physikalisch** von der Außenwelt getrennt sein. Damit ergibt sich für den Zugriffsschutz automatisch das höchste Schutzniveau.

Wenn keine Sicherheitssteuerung in der Maschine Safety Netzwerkvariablen verwendet (Topologien T1 oder T2) und das Maschinennetzwerk mit der Außenwelt verbunden werden soll, ist ein Gateway zu verwenden:

- Es wird als eigenes Netzwerk ausgelegt, das nur über ein Gateway mit der Außenwelt (dem Firmennetz) verbunden ist
- Auf diesem Gateway den Zugriffsschutz auf das Steuerungsnetzwerk der Maschine einrichten
- Gateways sollten die Kommunikationsprotokolle mit der Außenwelt auf das absolut Notwendige einschränken, und z.B. keine Standard CODESYS Netzwerkvariablen-Kommunikation von und zu der Maschine durchlassen.

Barriere 2: Zugriffsschutz für die Standardsteuerung

VORSICHT!

Zugriffsmöglichkeiten auf Steuerung prüfen!

Steuerungen sollten nicht vom Internet oder nicht vertrauenswürdigen Netzen aus zugreifbar sein!

Im Speziellen dürfen die Programmier-Ports der Steuerung unter keinen Umständen ungeschützt aus dem Internet zugreifbar sein (meist UDP-Ports 1740..1743 und TCP-Ports 1217 + 11740 bzw. die steuerungsspezifischen Ports)!

Wenn ein Zugriff aus dem Internet dennoch ermöglicht werden muss (z.B. für Fernzugriff auf die Sicherheitssteuerung, vgl. *Kapitel 12.3.1 „Verbindung zur Sicherheitssteuerung für Fernzugang“ auf Seite 246*), dann muss zwingend ein sicheres Verfahren gewählt werden, um sich mit der Steuerung zu verbinden (z.B. VPN).

HINWEIS!

Um das Risiko von Datensicherheitsverletzungen zu minimieren, empfehlen wir die folgenden organisatorischen und technischen Maßnahmen für das System, auf dem Ihre Applikationen laufen:

Vermeiden Sie soweit als möglich, die SPS und Steuerungsnetzwerke offenen Netzwerken und dem Internet auszusetzen. Verwenden Sie zum Schutz zusätzliche Sicherungsschichten wie ein VPN für Remote-Zugriffe und installieren Sie Firewall-Mechanismen. Beschränken Sie den Zugriff auf autorisierte Personen, ändern Sie vorhandene Standard-Passwörter bei der ersten Inbetriebnahme und auch weiterhin regelmäßig. Wenn Sie trotz allem Ihre Web-Visualisierung veröffentlichen möchten, wird dringend empfohlen, sie zumindest mit einem einfachen Passwortschutz zu versehen, um zu verhindern, dass jemand über Internet auf Ihre Steuerungsfunktionalität zugreifen kann (sehen Sie ein Beispiel im Projekt "SimpleWebvisuLogin.project", das mit der Standardinstallation des Programmiersystems bereitgestellt wird).

Die IT-Security-Analyse sollte prüfen, dass über Angriffe auf das Maschinennetzwerk und die Standardsteuerungen keine Risiken in der Maschine hervorgerufen werden können, die außerhalb dessen liegen, wofür das Sicherheitssteuerungssystem ausgelegt wurde.

12.1.2 Schutzmaßnahmen in der Sicherheitssteuerung

Barriere 3: Grundsätzlicher Selbstschutz der Sicherheitssteuerung

Jede Onlineverbindung zur Sicherheitssteuerung unterliegt einer grundsätzlichen Schutzmaßnahme.

Die zwei Anwendungsfälle einer Onlineverbindung (einerseits Entwicklung und Administration vor Ort, andererseits Diagnose aus der Ferne) werden durch zwei verschiedene Barrieren geschützt.

Barriere 3a. Identifizierung der Sicherheitssteuerung

Jeder Zugriff auf die Sicherheitssteuerung zum Zweck der Entwicklung oder Administration erfordert zuvor mindestens einmal die Identifizierung der Sicherheitssteuerung.

Folgende Maßnahme aus dem Security Level 1 der IEC 62443 ist implementiert:

Gerätespezifisch eine von beiden Maßnahmen, um das Einloggen auf die falsche Steuerung zu vermeiden:

- Eingabe der Seriennummer der Sicherheitssteuerung
- Drücken eines Knopfes (bzw. Schalters) an der Sicherheitssteuerung

Um sich erfolgreich auf die Sicherheitssteuerung einzuloggen muss, abhängig von der Sicherheitssteuerung, eine dieser beiden Aktionen durchgeführt werden (siehe ↬ *„Verbindungsbestätigung“ auf Seite 165*). Diese Maßnahmen können als relativ sicher eingestuft werden, da das Drücken eines Knopfes manuell und direkt an der Sicherheitssteuerung durchgeführt werden muss und die Eingabe der Seriennummer Insiderwissen über die Maschine erfordert.

Barriere 3b. Fernpasswort

Zum Zweck der Diagnose (vgl. ↬ *Kapitel 12.3.1 „Verbindung zur Sicherheitssteuerung für Fernzugang“ auf Seite 246* ist auch ein Zugriff auf die Sicherheitssteuerung ohne Id der Steuerung möglich. Dieser Zugriff ist durch folgende Maßnahmen geschützt.

- 1. Er wird von der Steuerung nur akzeptiert, wenn der Fernzugriff zuvor durch einen Anwender vor Ort unter Identifikation der Steuerung (Barriere 3.1a) und Angabe des Administrationspassworts (Barriere 3.2a) freigeschalten (erlaubt) wurde.
- 2. Er erfordert das Fernpasswort.
- 3. Er ist auf lesenden Zugriff beschränkt.

Befehle, die bei Fernzugang möglich sind:

- *„Einloggen“*
- *„Ausloggen“*
- *„Aktualisieren“*, Schaltfläche in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung
- Logbuch anzeigen und abspeichern, Schaltflächen in der Registerkarte *„Log“* der Sicherheitssteuerung

HINWEIS!
Um einen unerlaubten Zugriff aus dem Netz auf die Sicherheitssteuerung zu verhindern, sollte ein möglichst "starkes" Passwort gewählt werden.

Barriere 4: Operationsabhängiger Schutz der Sicherheitssteuerung

Auch wenn grundsätzlich der Zugriff auf die Sicherheitssteuerung per Identifizierung oder Fernpasswort erlangt wurde, können die dann möglichen Operationen beschränkt sein.

Barriere 4a. Administrationspasswort

Jeder Eingriff in den Ablauf und jede Änderung an der identifizierten Sicherheitssteuerung über Onlinedienste setzen die Identifikation der Sicherheitssteuerung (Barriere 3a) voraus und erfordern das Administrationspasswort.

Die Bootapplikation kann mit dem Administrationspasswort (Admin-Passwort) vor nicht autorisiertem schreibendem Zugriff geschützt werden (siehe *Kapitel 12.1.2 „Schutzmaßnahmen in der Sicherheitssteuerung“ auf Seite 240*).

Befehle, die durch das Administrationspasswort geschützt sind:

- *„Einloggen“*
- *„Bootapplikation erzeugen“*
  „Bootapplikation löschen“
- *„Bootapplikation neu starten“*
- *„Administrationspasswort setzen“*
- *„Firmware aktualisieren“*
- *„Fernzugang konfigurieren“*
- *„Reset Kalt“*
- *„STOP“*
- *„START“*
- *„Werte schreiben “*
- *„Werte forcen“*
- *„Forcen für alle Werte aufheben“*
- *„Gerätenamen ändern“*

HINWEIS!
Um eine Manipulation einer einmal erfolgreich identifizierten Sicherheitssteuerung zu verhindern, sollte ein möglichst "starkes" Passwort gewählt werden.

Barriere 4b. Benutzerverwaltung im Projekt

Jeder Zugriff auf die Inhalte der laufenden Applikation (Variablen lesen, schreiben, stoppen etc.) erfordert ein Projekt mit dem gleichen Stand der Applikation wie auf der Steuerung. Über die Benutzerverwaltung des Projekts kann ein Schutz vor unerlaubten Zugriffsmöglichkeiten erreicht werden.

Es können in der Projektbenutzerverwaltung Rechte so vergeben werden, dass Safety-Objekte eines Projekts nur von bestimmten Benutzergruppen (z.B. "Safety-Entwickler") erstellt bzw. modifiziert werden können.

Jeder Teil einer Sicherheitsapplikation eines CODESYS Projekts kann vor nicht autorisiertem Zugriff durch entsprechende Einstellungen in der CODESYS Benutzerverwaltung geschützt werden.

HINWEIS!

Um den Zugriffsschutz der Sicherheitsapplikation zu gewährleisten, muss der Anwender für jedes Projekt eine entsprechende CODESYS Benutzerverwaltung einrichten oder die in CODESYS Safety integrierte Safety Benutzerkonfiguration (siehe *Kapitel 5.2.3 „Einrichten der Benutzerverwaltung im Projekt“ auf Seite 53*) verwenden.

Ohne Benutzerverwaltung im Projekt besitzt jede Person, die an dem Projekt mitarbeitet, alle Rechte an den Standard- und Sicherheitsapplikationen im Projekt!

12.1.3 Schutz der Sicherheitssteuerung vor Schreibzugriff

Administrationspasswort setzen

Mithilfe des Administrationspassworts kann die Sicherheitssteuerung vor nicht-autorisiertem schreibendem Zugriff geschützt werden.

HINWEIS!

Um eine Manipulation einer einmal erfolgreich identifizierten Sicherheitssteuerung zu verhindern, sollte ein möglichst "starkes" Passwort gewählt werden.

Die Schaltfläche für den Befehl *„Admin-Passwort setzen“* befindet sich im Editor der Sicherheitssteuerung in der Registerkarte *„Safety Online Information“*. Diese wird nach Aktivierung des Befehls *„Objekt bearbeiten“* des Kontextmenüs der Sicherheitssteuerung durch Klicken des Tabs *„Safety Online Information“* geöffnet.

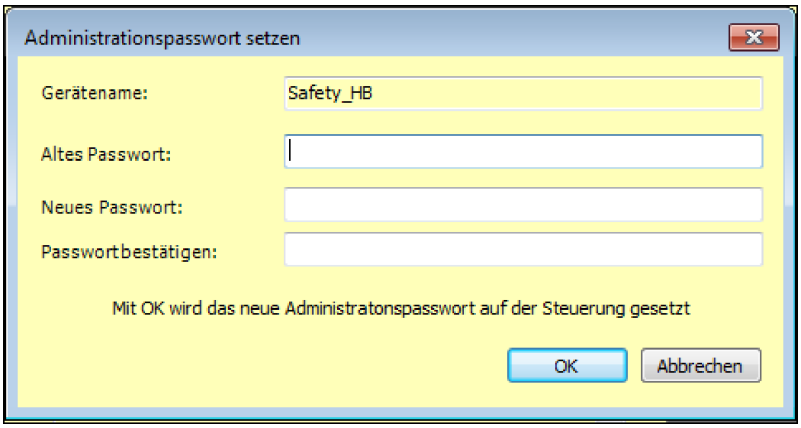


Abb. 104: Dialog 'Administrationspasswort setzen'

Bei einem Austausch der SPS wird das Admin-Passwort mit der Sicherheitsapplikation auf die neue Steuerung übertragen.

Bei der Serienfertigung einer Sicherheitssteuerung mit Bootapplikation wird das Admin-Passwort mit kopiert, was zur Folge hat, dass alle Serienmodelle das gleiche Admin-Passwort haben!

Um das Admin-Passwort ändern zu können, muss sich der Service-Mitarbeiter zuerst mit dem bereits existierenden Admin-Passwort autorisieren!

Admin-Passwort Abfrage

Das Admin-Passwort wird vor der Ausführung des ersten schreibenden Online-Befehls nach Öffnen des Projekts, bevor also die Steuerung in den unsicheren Zustand geht, abgefragt. Ein eingegebenes Passwort bleibt in CODESYS bis zum Schließen des Projekts aktiv und muss nicht erneut eingegeben werden.

Abb. 105: Dialog zur Admin-Passwort Abfrage

12.1.4 Schutz der Sicherheitssteuerung vor Fernzugriff

Fernzugang konfigurieren

Anfänglich ist kein Fernzugang möglich. Er muss erst freigeschalten werden. Dabei muss ein Fernpasswort vergeben werden. Mit Hilfe des Fernpassworts kann die Sicherheitssteuerung vor nicht autorisiertem Fernzugriff (nur lesend) geschützt werden.

HINWEIS!

Um die Ausspähung einer über die Standardsteuerung erreichten Sicherheitssteuerung zu verhindern, sollte der Fernzugang nur freigeschalten werden, wenn und solange er für den Betrieb benötigt wird, und es sollte dabei ein möglichst starkes Passwort vergeben werden.

Die Schaltfläche für den Befehl *„Fernzugang konfigurieren“* befindet sich im Editor der Sicherheitssteuerung in der Registerkarte *„Safety Online Information“*. Über den Dialog kann der Fernzugang freigeschalten werden und wieder blockiert werden, sowie das Fernpasswort jederzeit geändert werden. Die Beschreibung des Dialogs finden Sie in der Onlinehilfe unter "Fernzugang konfigurieren".

Um über den Fernzugang auf die Sicherheitssteuerung zugreifen zu können, wird das Fernpasswort benötigt. Zusätzlich muss auch die ganze Kommunikationsstrecke von außen bis in die Sicherheitssteuerung für den Fernzugang möglich sein. Für den Kommunikationsweg muss ein sicheres Verfahren gewählt werden, um sich mit der Steuerung zu verbinden, zum Beispiel VPN, siehe auch *Kapitel 12.1.1 „Schutzmaßnahmen im Umfeld der Sicherheitssteuerung“ auf Seite 238*.

12.1.5 Beobachtung Security-relevanter Ereignisse

Änderungen an den Schutzmaßnahmen lassen sich wie folgt nachvollziehen: Für jede Passwortänderung erfolgt ein Logbucheintrag.

Ob und wann ein Fernzugriff erfolgt ist, lässt sich wie folgt nachvollziehen: Für jeden Fernzugriff erfolgt ein Logbucheintrag.

12.2 Beobachtung von Fehlern im Betrieb

12.2.1 Erhöhte Kommunikationsfehler-Häufigkeit

FSoE-Geräte in der Maschine

VORSICHT!

Restfehlerrate

Von der Treiberinstanz FSoEMaster gemeldete Kommunikationsfehler dürfen nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt.

Safety Netzwerkvariablen in der Maschine

VORSICHT!

Restfehlerrate

Von der Treiberinstanz NetVarReceiver gemeldete Kommunikationsfehler dürfen nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt.

12.2.2 Anwenderverhalten bei Fehlermeldungen

Anwenderverhalten bei undefiniertem oder sicherheitsrelevantem Fehler

HINWEIS!

Der Anwender ist dazu verpflichtet, sicherheitsrelevante Fehler unverzüglich an den jeweiligen Gerätehersteller zu melden.

HINWEIS!

Das Auftreten eines undefinierten Fehlers muss unverzüglich an den Gerätehersteller gemeldet werden!

12.3 Diagnose von Fehlern im Betrieb

Eine Diagnose der Sicherheitsfunktion kann auf vier Ebenen erfolgen:

- Applikative Diagnose: Funktionen zur Diagnose, die in die Sicherheitsapplikation und funktionale Applikation einprogrammiert wurden, sollten die typischen Diagnosefälle abdecken. Zum Beispiel könnte eine Visualisierung auf der Standardsteuerung die Fehlercodes aus Treiberbausteinen und PLCopen Funktionsbausteinen auswerten (wenn diese per Austauschvariablen in die Standardsteuerung übertragen werden).
- Diagnose mit Standard-CODESYS Mitteln: Die Verbindung zur Sicherheitsteuerung und die von der Sicherheitssteuerung mit verwendeten Feldbusse und Netzwerkvariablen können bei bestehender Onlineverbindung mit der Standardsteuerung über die Status-Icons im Gerätebaum und die Registerkarte *„Status“* des Geräteeditors der Standardsteuerung oder Sicherheitssteuerung diagnostiziert werden.

- Diagnose mit CODESYS im sicheren Betrieb: nur lesend, entweder vor Ort mit bestätigter Verbindung (*Kapitel 7.2.2 „Verbindungsaufbau“ auf Seite 164*) oder mit Fernzugriff (*Kapitel 12.3.1 „Verbindung zur Sicherheitssteuerung für Fernzugang“ auf Seite 246*).

  Möglich sind:
  - Registerkarte *„Log“* der Sicherheitssteuerung (siehe *Kapitel 12.3.3 „Logbuch: Diagnose von System- und Laufzeitfehlern“ auf Seite 248*)
  - Registerkarte *„Safety Online Information“* der Sicherheitssteuerung (siehe *Kapitel 12.3.2 „Informationen zu Firmware und Bootapplikation “ auf Seite 247*)
  - Registerkarte *„Status“* der Sicherheitssteuerung (siehe *Kapitel 12.3.4 „Status: Diagnose der Kommunikation“ auf Seite 250*)
  - Nach dem Einloggen: Monitoren in Editoren und Überwachungsliste (siehe *Kapitel 7.6.1 „Monitoring“ auf Seite 179*), die Ablaufkontrolle ist bei Fernzugriff nicht möglich.
- Nur bei bestätigter Verbindung möglich: Debuggen mit CODESYS: Wenn die genannten Möglichkeiten zur Diagnose nicht ausreichen, besteht als letzte Möglichkeit, die Applikation durch Eingriff in den Ablauf zu debuggen (siehe *Kapitel 7.6 „Monitoring und Debuggen“ auf Seite 179*). Diese stellt eine Form der Wartung dar, bei welcher der sichere Betrieb temporär verlassen wird und die Maschine organisatorisch zu sichern ist (siehe *Kapitel 7.5.2 „Debug-Modus und organisatorische Sicherheit“ auf Seite 176*.

12.3.1 Verbindung zur Sicherheitssteuerung für Fernzugang

Die Voraussetzungen für den Fernzugriff sind eine Netzwerkverbindung, das Fernpasswort und die Freischaltung des Fernzugriffs. Während des Fernzugriffs ist nur eingeschränkter Zugriff (lesender Zugriff) auf die Sicherheitssteuerung möglich.

1. In der Registerkarte *„Kommunikation“* der Sicherheitssteuerung den aktiven Pfad auf das gewünschte Gerät (Gerätename) setzen, siehe *„Netzwerkverbindung für die bestätigte Verbindung“ auf Seite 164* (Vorgehensweise wie in Standard CODESYS, für Details wird auf die Online Hilfe von CODESYS Standard verwiesen.)
2. Befehl *„Einloggen“* des Menüs *„Online“* aktivieren
3. Im erscheinenenden Dialog *„Verbindung zur Sicherheitssteuerung“* die Verbindungsart *„Fernzugang “* auswählen.

Die Option ist nur verfügbar, wenn der Fernzugang für die Steuerung freigeschalten wurde (siehe Kapitel 12.1.4 „Schutz der Sicherheitssteuerung vor Fernzugriff“ auf Seite 244)

4. Das Fernpasswort eingeben.
5. Schaltfläche *„OK“* aktivieren

HINWEIS!
Fernzugang ist für die Verifikation und die Abnahme nicht qualifiziert.

12.3.2 Informationen zu Firmware und Bootapplikation

Informationen zur Firmware und zur Bootapplikation

Die Informationen zur geladenen Bootapplikation und zur Firmware und werden in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung angezeigt.

HINWEIS!
Bei Verwendung der Registerkarte *„Safety Online Information“* für Verifikationstätigkeiten oder Analysen muss anhand des Gerätenamens geprüft werden, dass die Informationen von der gewünschten Steuerung stammen und dass die Informationen über eine bestätigte Verbindung geholt wurden (*„aktuelle Verbindung: Identität des Geräts bestätigt“*).

Durch Aktivierung des Befehls „Aktualisieren“ werden alle aktuellen Informationen für die Bootapplikation und Firmware mit aktuellen Werten von der angezeigten Geräteinstanz angezeigt. Es wird dem Entwickler gemeldet, wenn die von der Steuerung hochgelesenen Informationen korrupt sind.

Zur Bootapplikation werden folgende Daten angezeigt:

- *„Name “*
  Name des Safety Applikationsobjekts
- *„Kommentar“*
- *„Ausführungsversion“*
- *„ Erzeugt“*
  Zeitpunkt der Erzeugung der Bootapplikation
- *„Bootapplikation-Pin“*
  Informationen zum Pin der Bootapplikation
  – *„Name“*
  – *„Revision“*
  – *„Prüfsumme“*
  – *„Erzeugt“*

Genauere Informationen siehe *Kapitel 12.3.2 „Informationen zu Firmware und Bootapplikation “ auf Seite 247*

Information zur Sicherheitssteuerung

- *„Firmware“*

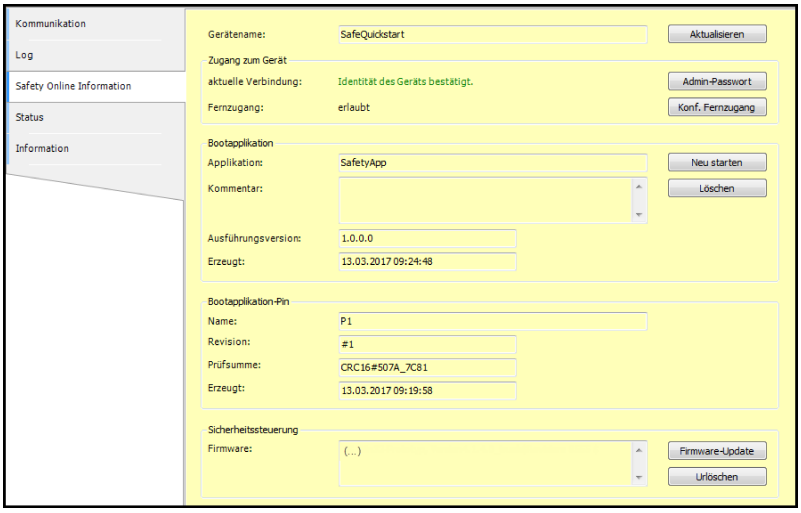


Abb. 106: Registerkarte 'Safety Online Information'

12.3.3 Logbuch: Diagnose von System- und Laufzeitfehlern

Logbuch

Die Logbuch-Ansicht von CODESYS Safety entspricht dem Logbuch von CODESYS Standard. Für weitere Detailinformationen wird deshalb auf die CODESYS Onlinehilfe verwiesen.

Das Logbuch wird in der Registerkarte *„Log“* der Sicherheitssteuerung angezeigt und dient als Protokoll und zur Diagnose von Laufzeitfehlern der Applikation und Systemfehlern. Es kann helfen, die Ursache eines Fehlers in der Steuerung oder in der Applikation zu finden.

HINWEIS!

Das Logbuch ist für die Verifikation, also für den Nachweis der Fehlerfreiheit und den Nachweis der Erfüllung der Anforderungen nicht zulässig!

In CODESYS Safety stehen zwei Logbücher zur Verfügung:

- Gerätelogbuch
- Applikationslogbuch

Das Gerätelogbuch gehört zum Gerät und es ist für Einträge gedacht, die das Gerät betreffen, z.B. Systemfehler, Erzeugen neuer Bootapplikationen.

Das Applikationslogbuch gehört zur IEC-Applikation und es ist für Einträge gedacht, die die Applikation betreffen, z.B. Laufzeitfehler, Fehler beim Laden der Bootapplikation, Fehler bei der Online-Kommunikation.

Das Applikationslogbuch ist das Default-Logbuch der Sicherheitssteuerung. Das Geräte-Logbuch steht erst dann zur Verfügung, wenn das Default-Logbuch (Applikationslogbuch) geladen ist.

Durch Betätigen der Schaltfläche werden alle zur Verfügung stehenden Logbücher (Geräte- und Applikationslogbuch) zyklisch von der Steuerung geladen und können in Fenster *„logger“* selektiert werden.

Die in den Logbüchern angezeigten Informationen sind folgendermaßen strukturiert:

- Gewichtung (Information, Warnung, Fehler, Ausnahme),
- Zeitstempel
- Fehlerbeschreibung
- Komponente Erzeuger

Bei Einträgen von Änderungen aufgrund von Online-Zugriffen wird der Name des Entwicklers mitgeloggt. Da es keine Benutzerverwaltung auf der Steuerung gibt, wird dabei der Name des Entwicklers aus der Benutzerverwaltung des Programmiersystems verwendet. Wenn der Entwickler im Programmiersystem nicht als ein spezifischer Benutzer, sondern als einer der vordefinierten Benutzer angemeldet ist, wird stattdessen der Name des Anwenders aus der Benutzerverwaltung von Windows verwendet.

Die Logbücher können als XML-Datei exportiert und importiert werden.

Erzeugung der Logbucheinträge

Logbucheinträge werden insbesondere dann erstellt, wenn die Bootapplikation im Offline-Betrieb aus irgendeinem Grund nicht geladen werden kann und ein Systemfehler auftritt.

Ein Systemfehler wird auch dann erzeugt, wenn bei dem Versuch des Laufzeitsystems, die Bootapplikation beim Hochfahren zu laden, das Logbuch nicht beschrieben werden kann.

Zeigt das Logbuch die Ursache eines Systemfehlers nicht an, so kann der Grund für diesen Systemfehler sein, dass das Logbuch vom sicherheitsgerichteten Laufzeitsystem nicht beschrieben werden konnte.

HINWEIS!

Bei Fehlern der Hardware konnten eventuell nicht alle Einträge ins Logbuch geschrieben werden.

Ein Logbucheintrag wird erstellt,

- immer wenn das Laufzeitsystem im Offline-Betrieb (d.h. beim Booten ohne Online-Verbindung) gemäß einer Sicherheitsanforderung auf einen Fehler in der Applikation mit dem Abbruch des Ladens reagiert.
- immer wenn das Laufzeitsystem im Offline-Betrieb auf einen Fehler der Applikation mit dem Beenden der Ausführung der Applikation reagiert.
- bei Vertauschung der Bootapplikation gegenüber dem letzten Bootladen bzw. der letzten Erzeugung einer Bootapplikation. Dies wird durch das Laufzeitsystem erkannt.
- bei verfälschter oder unpassender Ausführungsversion des Laufzeitsystems.
- Protokollierung von Änderungen der Bootapplikation
- Protokollierung Firmware-Update
- Protokollierung der Erzeugung neuer Bootapplikationen

Logbucheintrag bei Laufzeitfehler

Wird die Applikation wegen eines Laufzeitfehlers beendet, so enthält der erzeugte Logbucheintrag folgende Informationen:

- Name der fehlerhaften POU
- Nummer des fehlerhaften Netzwerks der POU
- bei einer FB-POU: FB-Instanz, in welcher sich der Fehler zeigte

12.3.4 Status: Diagnose der Kommunikation

In der Registerkarte *„Status “* werden die Diagnosemeldungen der Sicherheitssteuerung und die Diagnosemeldungen über den Verbindungsstatus der Sicherheitssteuerung zu übergeordneten bzw. unterlagerten Geräten angezeigt.

Die Registerkarte steht nur im Online-Modus der Sicherheitssteuerung oder der Standardsteuerung zur Verfügung.

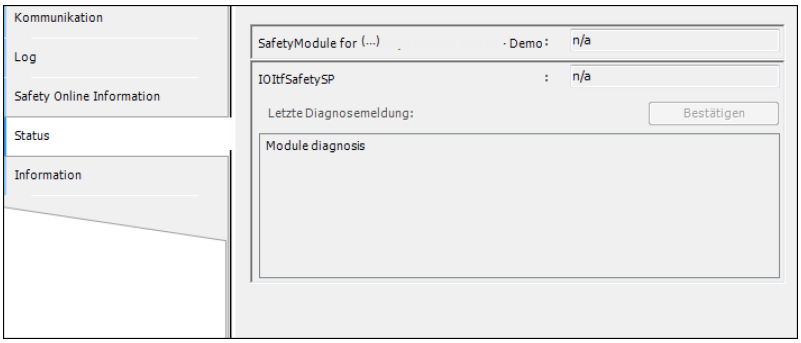


Abb. 107: Registerkarte 'Status'

Meldungen von der Sicherheitssteuerung, welche ihren internen Status betreffen werden im Feld für das Gerät angezeigt.

Meldungen bzgl. des Verbindungsstatus der Sicherheitssteuerung zu übergeordneten bzw. unterlagerten Geräten, werden im Feld für das entsprechende Modul angezeigt.

Es werden folgende Meldungen angezeigt:

- unterschiedliche Konfiguration der E/A-Module bzgl. Anzahl, Id oder E/A-Größe auf Standard und Safety
- Unterschiedliche Konfiguration eines E/A-Moduls bezüglich der Protokollparametrierung auf Standard und Safety.
  Diese Anzeige erfolgt nur für Bussysteme, bei denen die Übertragung der sicheren Konfiguration über die Standardkonfiguration erfolgt (z.B. PROFIsafe)

12.4 Administration mit CODESYS

Safety Online Information

Die Registerkarte *„Safety Online Information“* ist die Einstiegsseite für die Administration der Sicherheitssteuerung mit CODESYS

Dieser Registerkarte (siehe Abb. 23) bietet Informationen zu der Sicherheitssteuerung, mit der CODESYS Safety aktuell verbunden ist.

Bei bestätigter Verbindung zur Sicherheitssteuerung können Passwörter, Bootapplikation und Firmware der Sicherheitssteuerung verwaltet werden.

Ist CODESYS Safety mit keiner Steuerung verbunden, sind die Textausgabefelder leer.

Befehle und Textausgaben:

- Befehl *„Aktualisieren“*: Damit die aktuellen Informationen zur Bootapplikation in der Steuerung ausgegeben werden, muss zunächst dieser Befehl ausgeführt werden.
- *„Gerätename“*: Der Name der Sicherheitssteuerung, auf der eingeloggt ist.

Informationen und Befehle zum Zugang zum Gerät:

- *„Akutelle Verbindung“*: Zeigt an, ob die angezeigten Informationen von einem bestätigten Gerät oder aus einem Fernzugang stammen.
- *„Fernzugang“*: Zeigt an, ob der Fernzugang auf die Steuerung erlaubt ist.
- Befehl *„Admin-Passwort“*: Setzt ein Passwort für alle schreibenden Zugriffe auf die Steuerung. Dies ist initial leer. Schreibende Zugriffe auf die Sicherheitssteuerung (z.B. Projekt laden) müssen mit diesem Passwort bestätigt werden.
  (siehe ↬ *„Administrationspasswort setzen“ auf Seite 242*)
- Befehl *„Fernzugang konfigurieren“*: Öffnet einen Dialog, der dazu dient den Fernzugang für die Sicherheitssteuerung zu erlauben, das Fernpasswort zu ändern, oder den Fernzugang zu verbieten (siehe ↬ *Kapitel 12.1.4 „Schutz der Sicherheitssteuerung vor Fernzugriff“ auf Seite 244*).

Informationen und Befehle zur Bootapplikation: (Diese Information kann nur ausgegeben werden, wenn auf der Steuerung ein Bootprojekt hinterlegt ist.)

- *„Name:“* Name des aktuell geladenen Safety Applikationsobjekts
- *„Kommentar:“* Kommentar, wie er im Eigenschaften-Dialog des Safety Applikationsobjekts (Registerkarte *„Safety“*, Abschnitt *„Kommentar“*) hinterlegt wurde.
- *„Ausführungsversion:“* Die Ausführungsversion, wie sie im Eigenschaften-Dialog der Safety Applikationsobjekts eingestellt wurde.
- *„Erzeugt:“* Datum und Uhrzeit, zu der das Bootprojekt erzeugt wurde.
- Befehl *„Neu starten“*: Entlädt die aktuelle Applikation, lädt und startet das Bootprojekt.
  (siehe ↳ *„Neustart der Bootapplikation“ auf Seite 172*)
- Befehl *„Löschen“*: Löscht das Bootprojekt in der Sicherheitssteuerung und entlädt die laufende Applikation.
  (siehe ↳ *„Bootapplikation löschen“ auf Seite 253*)

Im Abschnitt *„Bootapplikation Pin“* wird die Pin-Kennung (siehe ↳ *Kapitel 8 „Pinnen der Software“ auf Seite 189*) zum in der Sicherheitssteuerung abgelegten Bootprojekt ausgegeben. Sie entspricht den Informationen, wie sie im Editorfenster der Safety Applikationsobjekts angezeigt werden.

Wurde das Bootprojekt zu einer nicht gepinnten oder nach dem Pinnen veränderten Bootapplikation erzeugt, sind die Ausgabefelder leer.

- *„Name:“* Die Bezeichnung des gepinnten Standes
- *„Revision:“* Die Revisionsnummer
- *„Prüfsumme:“* Die Prüfsumme über die gepinnte Applikation, zu der das Bootprojekt erzeugt wurde. (Pin-Prüfsumme)
- *„Letzte Änderung:“* Datum und Uhrzeit

Informationen und Befehle zur Sicherheitssteuerung

- *„Firmware“*: Aus der Sicherheitssteuerung wird die genaue Bezeichnung der Firmware ausgelesen.
- Befehl *„Firmware-Update“*: Lädt eine (andere) Firmware-Release aus einer Datei auf die Steuerung, um eine dort bereits vorhandene Firmware zu ersetzen.
  (siehe ↳ *Kapitel 12.6.2 „Firmware-Update aufspielen“ auf Seite 256*)
- Befehl *„Urlöschen“*: Erlaubt das Entladen der Applikation, Löschen der Bootapplikation und Rücksetzen des Passworts. Dieser Befehl kann ohne Passwort ausgeführt werden und erlaubt das Rücksetzen der Steuerung, auch wenn ein Passwort vergessen wurde.
  (siehe ↳ *weitere Informationen auf Seite 258*)

Identifikation der Sicherheitsapplikation

Eine Sicherheitsapplikation in der Steuerung wird über die Pin-Kennung der Bootapplikation identifiziert (in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung). Diese Pin-Kennung muss mit der Pin-Kennung der Sicherheitsapplikation in CODESYS Safety (in der Registerkarte *„Objekte“* des Safety Applikationsobjekts) übereinstimmen.

Bootapplikation starten

Neustart der Bootapplikation

Eine Bootapplikation auf der Steuerung wird durch Aktivierung des Befehls *„Neu starten“* (Registerkarte *„Safety Online Information“* der Steuerung) oder automatisch nach Einschalten der Steuerung neu gestartet.

Neustart bedeutet nicht automatisch, dass die Anlage in jedem Fall losläuft. Der Entwickler legt in der Applikation fest, ob die PLCopen Bausteine und die sicheren Ausgangsmodule automatisch anlaufen (AutoReset) oder erst durch ein Standardsignal (Reset).

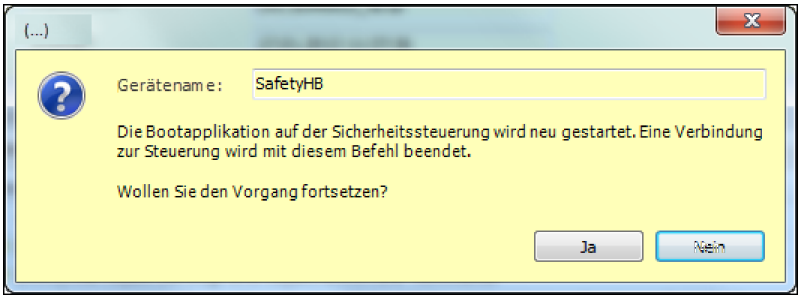


Abb. 108: Dialog: "Neu starten" der Bootapplikation

Bei Fehlererkennung während des Ladens der neuen Bootapplikation wird der Ladevorgang abgebrochen und es wird ein Logbucheintrag erzeugt.

Bootapplikation löschen

Mit der Schaltfläche *„Löschen“* der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung wird die aktuelle Applikation entladen und die Bootapplikation auf der Sicherheitssteuerung gelöscht. Das Löschen der Bootapplikation ist nur nach bereits erfolgter Autorisierung durch das Admin-Passwort möglich. Eine bestehende Online-Verbindung zur Sicherheitssteuerung wird nach Bestätigung durch den Service-Mitarbeiter durch die Ausführung des Befehls beendet.

Dieser Befehl ist nur aktivierbar, wenn der Befehl *„Aktualisieren“* mindestens einmal ausgeführt wurde und auf der Steuerung BA-Informationen verfügbar waren. Die Schaltfläche *„Aktualisieren“* befindet sich in der Registerkarte *„Safety Online Information“*.

Der Service-Mitarbeiter erhält immer eine Rückmeldung, ob der Befehl erfolgreich ausgeführt wurde oder nicht.

VORSICHT!

Wird der Befehl *„Löschen“* der Bootapplikation nicht vollständig ausgeführt, so kann es nach Neustart zu einer Gefährdung in der Anlage kommen, weil diese noch mit der alten Applikation anläuft.

Der Service-Mitarbeiter muss auf die Rückmeldung warten, ob der Befehl erfolgreich ausgeführt wurde. Bleibt diese aus, so ist die Steuerung so zu behandeln, als ob die Bootapplikation nach Neustart wieder aufstarten kann.

Der Sevice-Mitarbeiter kann sich zusätzlich in der Anzeige der Applikationsinformationen zur Applikation auf der Steuerung in der Registerkarte „Safety Online Information“ der Sicherheitssteuerung vergewissern, dass sich keine Bootapplikation mehr auf der Steuerung befindet.

Anzeige der Applikationsinformation

1. Sicherheitssteuerung im Gerätebaum (Projektbaum) auswählen
2. Befehl *„Objekt bearbeiten“* des Kontextmenüs aktivieren
3. Registerkarte *„Safety Online Information“* öffnen
4. Schaltfläche *„Aktualisieren“* betätigen
5. Überprüfen, ob keine Bootapplikation mehr existiert

Das Admin-Passwort wird bei der Ausführung des Befehls „Löschen“ nicht gelöscht. Das Admin-Passwort kann nur durch „Urlöschen“ der Bootapplikation gelöscht werden (siehe ↳ weitere Informationen auf Seite 258)

12.5 Verfahren für die Wartung

12.5.1 Temporärer Wechsel in den unsicheren Betrieb

Das Debuggen einer laufenden Applikation (vgl. ↳ *Kapitel 12.3 „Diagnose von Fehlern im Betrieb“ auf Seite 245*) und das Aktualisieren von Bootapplikation (↳ *Kapitel 12.6.1 „Neue Bootapplikation aufspielen“ auf Seite 255* oder Firmware (↳ *Kapitel 12.6.2 „Firmware-Update aufspielen“ auf Seite 256*) bedingen, dass die Sicherheitssteuerung den sicheren Betrieb verlässt. Debuggen und Aktualisieren erfordern eine bestätigte Verbindung zur Sicherheitssteuerung (Fernzugriff reicht nicht) und organisatorische Sicherungsmaßnahmen.

Verlassen des Betriebsmodus - Debuggen der Maschine

VORSICHT!

Solange sich in einer Maschine eine Sicherheitssteuerung, die eine Safety Netzwerkvariablenliste (Sender) enthält, im Debug-Betrieb befindet, muss die ganze Maschine organisatorisch abgesichert sein.

Dies bedeutet auch: Es dürfen sich keine Personen in Gefahrenbereichen aufhalten, die durch Sicherheitssteuerungen überwacht werden, die über die sichere Querkommunikation mit der Sicherheitssteuerung, die sich im Debug-Betrieb befindet, verbunden sind.

HINWEIS!

Die Hinweise des Kapitels *Kapitel 7.5.2 „Debug-Modus und organisatorische Sicherheit“ auf Seite 176* und *Kapitel 7.6.3 „Debug-Betrieb der Sicherheitssteuerung“ auf Seite 180* müssen beachtet werden.

12.6 Wartung und Service

12.6.1 Neue Bootapplikation aufspielen

Vorgehen im Betrieb

VORSICHT!

Der Systemintegrierende in der Maschinenproduktion muss beim Download einer abgenommenen Bootapplikation auf die Steuerung prüfen, dass die Erzeugung einer gepinnten Bootapplikation ohne Abweichungen und mit der richtigen Pin-Kennung gemeldet wird. Wird die Erzeugung einer Bootapplikation eines "In Work"-Standes gemeldet, so muss der Vorgang abgebrochen werden.

HINWEIS!

Wird vor Ort eine abgenommene Bootapplikation zum Software-Update auf die Steuerung gespielt, muss der Service-Mitarbeiter prüfen, ob eine gepinnte Bootapplikation ohne Abweichungen und mit der richtigen Pin-Kennung gemeldet wird.

HINWEIS!
Der Service-Mitarbeiter muss bei Übertragung einer abgenommenen Applikation durch ein Speichermedium auf eine andere Steuerung verifizieren, dass die Bootapplikation auf der "neuen" Steuerung, die richtige Pin-Kennung hat (in der Registerkarte *„Safety Online Information“* der Sicherheitssteuerung).

12.6.2 Firmware-Update aufspielen

Bevor ein Firmware-Update durchgeführt wird, sollte sich der Service-Mitarbeiter über den Grund und die Folgen des Firmware-Updates im Klaren sein. Siehe Kapitel 11.3 „Aktualisieren der Firmware, die Ausführungsversion“ auf Seite 233

Die Firmware auf der Sicherheitssteuerung kann durch einen Firmware-Update ausgetauscht werden. Hierzu dient die Schaltfläche *„Firmware-Update“* im Geräte-Editor *„Safety Online Information“*. Dabei wird die neue Firmware aus einer Datei heraus auf die Sicherheitssteuerung geladen und die bisherige Firmware der Sicherheitssteuerung ersetzt. Der Befehl ist nur aktivierbar, wenn mindestens einmal der Befehl *„Aktualisieren“* ausgeführt wurde. Zur Ausführung des Befehls ist die Autorisierung mit dem Admin-Passwort erforderlich.

Ob ein Firmware-Update zur Verfügung steht und wie genau dieser durchzuführen ist, hängt von der verwendeten Sicherheitssteuerung ab.

Für nähere Informationen zum Firmware-Update kontaktieren Sie bitte den Gerätehersteller.

12.6.3 Hardware-Tausch

Tausch von Sicherheitssteuerungen

In der Maschine kann man eine defekte oder alte Sicherheitssteuerung gegen eine Neue austauschen. Die alte Bootapplikation kann auf der neuen Sicherheitssteuerung weiterlaufen,

1. wenn auf der neuen Steuerung keine Inkompatibilität durch einen anderen Firmwarestand entsteht (siehe *Kapitel 11.3 „Aktualisieren der Firmware, die Ausführungsversion“ auf Seite 233*).

2. die Bootapplikation (abhängig vom Gerätehersteller) außerhalb der ausgetauschten Hardware gespeichert war oder auf einem Datenträger gespeichert ist und dieser in die neue Steuerung übertragen werden kann.

VORSICHT!

Sie müssen sicherstellen, dass tatsächlich die gleiche Bootapplikation wie bisher auf der neuen Hardware läuft.

Fall 1: Ihre Sicherheitssteuerung speichert die Bootapplikation auf einem eingesteckten Datenträger. Dann müssen Sie sicherstellen, dass der Datenträger der alten Sicherheitssteuerung in die neue Sicherheitssteuerung gesteckt wurde.

Fall 2: Ihre Sicherheitssteuerung speichert die Bootapplikation, z.B. auf der Standardsteuerung. Dann müssen Sie nach Tausch der Sicherheitssteuerung mit CODESYS Safety Verbindung zur Sicherheitssteuerung aufnehmen und auf der Registerkarte *„Safety Online Information“* (↳ *Kapitel 12.3.2 „Informationen zu Firmware und Bootapplikation “ auf Seite 247*) den Pin der Bootapplikation überprüfen: Es muss der Pin sein, der für die ausgetauschte Sicherheitssteuerung in der Maschine abgenommen wurde.

Das Admin-Passwort geht bei einem Austausch des Datenträgers zusammen mit der Bootapplikation auf die neue Steuerung.

Für die alte, ausgebaute Steuerung sind die Schritte zum SPS-Ausbau anzuwenden (siehe ↳ Kapitel 12.8 „Verfahren zur Außerbetriebnahme und zum Ausbau der Sicherheitssteuerung“ auf Seite 258.

Tausch von Standardsteuerungen

HINWEIS!

Verwendung von Safety Netzwerkvariablenlisten

Beim Tausch von Standardsteuerungen oder Austausch ihrer Applikationen muss darauf geachtet werden, dass ihre Applikationen passend zu den Applikationen auf den Sicherheitssteuerungen unter dieser Standardsteuerung erzeugt wurden. Ansonsten kann der Tausch zu Mismatch und folglich zur Unterbrechung der Querkommunikation führen.

Verwendung von Safety Netzwerkvariablenlisten

Beim Tausch von Standardsteuerungen sollte darauf geachtet werden, dass sie die gleiche IP-Konfiguration, insbesondere gleiche IP-Adresse, hat.

Grund: Wenn der Tausch einer Standardsteuerung deren IP-Adresse ändert, muss der Anwender die IP-Adresskonfiguration aller Netzwerkvariablenlisten mit Verbindung zur Sicherheitssteuerung unter der Standardsteuerung (der Sicherheitssteuerung) anpassen und die Applikationen der entsprechenden Standardsteuerung neu erzeugen.

12.7 Änderungen in Netzwerken und Feldbussen

Bei allen Änderungen am Netzwerk auf Maschinenebene und an Feldbussen müssen die entsprechenden Systemanforderungen eingehalten werden. Siehe entsprechende Abschnitte in ↳ *Kapitel 14 „Feldbusse und Netzwerkvariablen“ auf Seite 271*.

12.8 Verfahren zur Außerbetriebnahme und zum Ausbau der Sicherheitssteuerung

Um eine Sicherheitssteuerung außer Betrieb zu nehmen, muss Klarheit darüber bestehen, ob sich die Bootapplikation auf der Standardsteuerung (z. B. SD-Karte) oder auf der Sicherheitssteuerung befindet.

Urlöschen

VORSICHT!

Bevor eine Sicherheitssteuerung außer Betrieb genommen und aus der Anlage ausgebaut wird, muss die sich auf der Sicherheitssteuerung befindende Bootapplikation "urgelöscht" werden. Dadurch wird verhindert, dass es bei einem Wiederverwenden der Sicherheitssteuerung auf einer anderen Anlage zum unbeabsichtigten Starten der "alten" Bootapplikation kommt.

VORSICHT!

Eine nicht vollständige Ausführung des Befehls*„Urlöschen“* kann dazu führen, dass die Bootapplikation auf der Steuerung bleibt und beim nächsten Starten der Steuerung wieder aktiv wird.

Bei Ausführen des Befehls *„Urlöschen“* wird die laufende Applikation entladen, die Bootapplikation gelöscht, alle Passwörter zurückgesetzt (bzw. die systemspezifischen Defaultpasswörter reaktiviert) und die Freischaltung des Fernzugangs zurückgesetzt. Die Logbücher werden dabei nicht gelöscht. Zur Ausführung des Befehls ist keine Autorisierung durch das Administrationspasswort erforderlich.

Ist der Service-Mitarbeiter aktuell auf der Applikation der Geräteinstanz eingeloggt, wird der Befehl nur nach durch den Service-Mitarbeiter bestätigter Rückfrage ausgeführt und die Online-Verbindung beendet.

Der Service-Mitarbeiter erhält immer eine Rückmeldung, ob der Befehl erfolgreich ausgeführt wurde oder nicht. Diese Rückmeldung muss durch den Service-Mitarbeiter bestätigt werden.

Verfahren "Urlöschen"

1. Befehl *„Objekt bearbeiten“* des Kontextmenüs der Sicherheitssteuerung im Gerätebaum aktivieren
2. Registerkarte *„Safety Online Information“* öffnen
3. Mit Klick auf die Schaltfläche *„Urlöschen“* den Befehl aktivieren

Steuerung mit eingebauter Netzwerkadresse

VORSICHT!

Wird eine Steuerung mit eingebauter Netzwerkadresse nach der Zweit-Verifikation ausgebaut und an anderer Stelle wieder eingebaut, ist eine Gefährdung möglich, wenn sich der Service-Mitarbeiter auf der organisatorisch nicht abgesicherten "neuen" Anlage oder Anlagenteil ohne Schutz durch Verbindungsverifikation einloggt, debuggt oder die Bootapplikation aktualisiert.

Um eine Gefährdung zu vermeiden muss der Service-Mitarbeiter direkt vor oder nach dem Ausbau

- die Steuerung urlöschen

oder

- sich einmal mit einem neuen Projekt auf die Steuerung einloggen.

Löschen des Admin-Passworts

Bei verlorengegangenem Admin-Passwort ermöglicht der Befehl „Urlöschen“ das Löschen des alten Admin-Passworts.

Anschließend muss erst die Bootapplikation neu erzeugt werden, um dann ein neues Admin-Passwort vergeben zu können.

Verfahren zur Außerbetriebnahme und zum Ausbau der Sicherheitssteuerung

13 Verfahren bei Änderungen und Wiederverwendung der abgenommenen Software

13.1 Verfahren bei Änderungen und Wiederverwendung der Software

Bei Änderungen und Wiederverwendung sind folgende Fälle zu unterscheiden:

- Unveränderte Wiederverwendung einer Sicherheitsapplikation in unverändertem Systemkontext (siehe *Kapitel 13.2 „Wiederverwendung eines abgenommenen Safety-Projekts“ auf Seite 262*). Hier wird kein Software-Entwicklungsprozess durchlaufen.
- Unveränderte Wiederverwendung einer Sicherheitsapplikation in einem neuem Systemkontext (siehe *Kapitel 13.2 „Wiederverwendung eines abgenommenen Safety-Projekts“ auf Seite 262*). Hier wird erfolgt zwar keine eigentliche Softwareentwicklung, aber die systemweite Planung von Adresseindeutigkeit und Querkommunikation zwischen Sicherheitssteuerung (siehe *Kapitel 4 „Planung des Gesamtsystems“ auf Seite 33*) und die Verifikation in der Maschine (siehe *Kapitel 9.4.4 „Verifikation in der fertigen Maschine“ auf Seite 214*) müssen wiederholt werden.
- Unveränderte Wiederverwendung einzelner Funktionsbausteine (siehe *Kapitel 13.3 „Wiederverwendung von Funktionsbausteinen“ auf Seite 263*) innerhalb des Software-Entwicklungsprozesses für eine neue Sicherheitsapplikation
- Änderung an einer Sicherheitsapplikation, die abgenommen ist (siehe *Kapitel 13.4.1 „Änderungen im Projekt“ auf Seite 264*). Dieses bedingt den Start eines Software-Entwicklungsprozesses für diese neue, geänderte Sicherheitsapplikation.
- Änderung an einer Sicherheitsapplikation, die in der Verifikation ist (siehe *Kapitel 13.4.1 „Änderungen im Projekt“ auf Seite 264*). Dieses bedingt einen Rücksprung des Software-Entwicklungsprozesses für diese Sicherheitsapplikation von der Verifikation in die Programmierung.

Wiederverwendung von Software kann sich auf ein komplettes CODESYS Safety-Projekt oder nur auf einen oder mehrere Funktionsbausteine beziehen.

Bei Änderungen gibt es die Varianten, dass Änderungen während der Verifikation einer Sicherheitsapplikation oder an einer bereits abgenommenen Sicherheitsapplikationen vorgenommen werden.

Hinweise zu Änderungen und Wiederverwendung

Grundsätzlich gilt für alle Sicherheitsapplikationen die mit CODESYS Safety erstellt wurden:

HINWEIS!
Es liegt in der Verantwortung des Service-Mitarbeiters sich zu vergewissern, dass die erzeugte Bootapplikation dem gewünschten Applikations-Pin entspricht.

HINWEIS!
Verifikation und Abnahme darf nur auf gepinnten Sicherheitsapplikationen durchgeführt werden. Kein Objekt der Applikation darf den Status "In Work" aufweisen (siehe *Kapitel 8 „Pinnen der Software“ auf Seite 189*).

13.2 Wiederverwendung eines abgenommenen Safety-Projekts

Die Abnahme bzw. die Verifikationsergebnisse eines CODESYS Safety Projekts können auf weitere Sicherheitssteuerungen übertragen werden, ohne die Applikation erneut verifizieren und abnehmen zu müssen, z.B. zur Serienfertigung von Maschinen.

VORSICHT!
Eine bereits abgenommene Sicherheitsapplikation kann nur auf folgende Weise auf eine andere Steuerung gebracht werden, ohne die Abnahme zu verlieren: Duplikation der Bootapplikation aus der ursprünglichen Steuerung durch Kopieren des Datenträgers. Andernfalls ist die Sicherheitsapplikation auf der Zweitsteuerung erneut zu verifizieren und abzunehmen.

Soll eine abgenommene Sicherheitsapplikation auf einer weiteren Sicherheitssteuerung verwendet werden, folgen Sie den Anweisungen des Herstellers der Sicherheitssteuerung. Unter Umständen stellt er einen eigenen sicheren Mechanismus zum Kopieren der Applikation zur Verfügung.

Bei Serienfertigung wird das Admin-Passwort mitkopiert! Dies bedeutet, dass alle Serienmodelle das gleiche Admin-Passwort haben.

Das Kopieren auf den Datenträger einer anderen Steuerung führt bei jedem kompatiblen Firmwarestand zum gleichen Verhalten der Bootapplikation.

13.3 Wiederverwendung von Funktionsbausteinen

Ein bereits validierter Funktionsbaustein einer abgenommenen Sicherheitsapplikation kann in einem anderen Projekt unter bestimmten Bedingungen wiederverwendet werden. Dabei können die Validierung und die Testergebnisse des Funktionsbausteins auf die neue Applikation übertragen werden.

Die Identifikation eines wiederverwendeten Baustein muss nachgewiesen werden

- wenn der Funktionsbaustein als Teil einer Applikation validiert wurde
- wenn der Funktionsbaustein in die Applikation kopiert und wiederverwendet wird
- wenn der Funktionsbaustein aus einer Bibliothek heraus wiederverwendet wird

HINWEIS!
Der Nachweis von wiederverwendeten Bausteinen (IEC-Bausteine und externe Bausteine) muss über die Objektliste des Applikationsobjekts erfolgen.

HINWEIS!
Wird ein Funktionsbaustein in einem anderen Projekt wiederverwendet, so muss die Prüfsumme des FBs und die Prüfsummen der FBs, die von diesem FB aufgerufen werden, den Prüfsummen der Abnahme entsprechen! Ist dies nicht der Fall, so muss der Funktionsbaustein neu verifiziert und validiert werden (siehe *Kapitel 9.1 „Einführung“ auf Seite 195*)

In CODESYS Safety führt ein Baustein bei jedem kompatiblen Firmwarestand immer wieder zum gleichen Verhalten auf der Steuerung wenn er

- *unverändert wiederverwendet wird und*
- *keine globalen Variablen enthält und*
- *keinen FB verwendet, dessen Datenlayout sich geändert hat.*

Versionsänderung externer FBs

Bei externen FBs dient die Version dazu, die Implementierung der Sicherheitssteuerung zu identifizieren.

In der Gerätebeschreibung der Sicherheitssteuerung ist hinterlegt, welche Version der externen FBs (PLCopen, Standard) zu dieser Sicherheitssteuerung gehört. Diese Bibliotheken werden automatisch geladen.

Ändert sich die Version einer Bibliothek, so wechselt der Pin-Status der Sicherheitsapplikation auf *„In Work“*. In der Vergleichsansicht des Safety Applikationsobjekts ist deutlich erkennbar, dass sich die Applikationen (Projektstatus und gepinnter Status) nur in diesem externen FB unterscheiden. Es muss eine Auswirkungsanalyse gemacht werden. Eventuell muss nicht die gesamte Applikation neu getestet werden, sondern nur die von dem entsprechen externen FB betroffenen Objekte.

13.4 Änderungen im Projekt

13.4.1 Änderungen im Projekt

HINWEIS!

Für den Nachweis von Strukturänderungen der Applikation, bzw. welche Objekte unverändert sind, und für die Auswirkungsanalyse von Änderungen oder für die Differenzabnahme muss der Vergleichseditor des Applikationsobjekts verwendet werden. Der Standard-Projektvergleich ist hierfür nicht geeignet.

Der Standard Projektvergleich kann lediglich als Hilfsfunktion verwendet werden, um die Vergleichsansicht des gepinnten Stands der Applikation zu öffnen. Die Vergleichsansicht wird durch einen Doppelklick auf das Safety Applikationsobjekt im Standard Projektvergleich geöffnet.

HINWEIS!

Es muss mittels der Pin-Kennung verifiziert werden, dass der Strukturvergleich (Vergleichseditor) die gewünschten Applikationsstände vergleicht.

HINWEIS!

Bei Erzeugung einer neuen Bootapplikation zur Verifikation nach einer Änderung muss sich der Service-Mitarbeiter vergewissern, dass die erzeugte Bootapplikation dem gewünschten Applikations-Pin entspricht. Er muss verifizieren, dass beim Download nicht die Erzeugung eines "In Work"-Standes, sondern eines gepinnten Standes der Applikation mit der richtigen Pin-Kennung angekündigt wird. Siehe ↳ *Kapitel 8 „Pinnen der Software“ auf Seite 189*.

Änderungen während der Verifikation

HINWEIS!
Werden während der Verifikation Änderungen an Objekten der Sicherheitsapplikation oder an der Projektstruktur der Sicherheitsapplikation vorgenommen, so muss die Sicherheitsapplikation neu gepinnt werden und alle von den Änderungen betroffenen Safety-Objekte anschließend neu verifiziert werden.

Ein Abgleich der Pin-Stände der Applikation muss mithilfe des Vergleichseditors der Sicherheitsapplikation (Doppelklick auf das Safety Applikationsobjekt im Projektvergleich) durchgeführt werden.

HINWEIS!
Es muss mittels der Pin-Kennung verifiziert werden, dass der Strukturvergleich (Vergleichseditor) die gewünschten Applikationsstände vergleicht.

HINWEIS!
Alle im Vergleichseditor aufgelisteten geänderten Safety-Objekte (ihre aktuelle Prüfsumme entspricht nicht ihrer gepinnten Prüfsumme) müssen neu gepinnt und verifiziert werden.

Zur Einflussanalyse bei Änderungen können auch die Kontrollfluss- und Datenflussanalyse verwendet werden: ↬ *„Kontrollflussanalyse“ auf Seite 204* und ↬ *Kapitel 9.3.6.5 „Datenflussanalyse“ auf Seite 205*. Dabei kann man zu jeder Variablen in einer geänderten GVL oder einem geänderten Mapping die Liste der Verwendungsstellen erhalten. Zu einer geänderten POU kann dabei eine Liste der Verwendungsstellen (betroffenen POUs) erzeugt werden.

Im CODESYS Standard Projektvergleich wird der aktuelle Stand des geöffneten Projekts mit einem früheren Projektstand aus einer Projektdatei oder in einer Quellcodeverwaltung verglichen, um die geänderten Objekte und die darin geänderten Stellen zu identifizieren. Für genauere Informationen wird auf die Online-Hilfe von CODESYS Standard verwiesen.

HINWEIS!
Wenn bei Multi-SPS Projekten die Verifikation nach einer Änderung fortgesetzt wird, muss beim Erzeugen der Bootapplikation in einem Bestätigungsdialog die Pin-Kennung geprüft werden, damit die richtige Bootapplikation auf der richtigen Steuerung erzeugt wird.

Dokumentation der Änderungen siehe *Kapitel 13.4.1 „Änderungen im Projekt“ auf Seite 264*.

Änderungen eines bereits abgenommenen Sicherheitsprojekt

Änderungen im Gerätebaum (physikalische Geräte) des Projektsbaums der Standardapplikation und Änderungen der Standardapplikation haben keinerlei Auswirkung auf die bereits abgenommene Sicherheitsapplikation.

HINWEIS!

Wenn die Änderung im Projekt die Sicherheitsapplikation inhaltlich ändert, dass sie "In Work" ist, dann muss ein Teil des Entwicklungsprozesses erneut durchgeführt werden:

1. erneut pinnen (*Kapitel 8 „Pinnen der Software“ auf Seite 189*), und

2. entweder vollständig verifizieren (*Kapitel 9 „Software-Verifikation“ auf Seite 195*), oder nach Einflussanalyse der gepinnten Änderungen teilweise nach-verifizieren (siehe unten), und

3. erneut abnehmen (*Kapitel 10 „Software-Abnahme und Dokumentation“ auf Seite 217*).

HINWEIS!

Bei Multi-SPS Projekten muss beim Erzeugen der Bootapplikation in einem Bestätigungsdialog die Pin-Kennung geprüft werden, damit die richtige Bootapplikation auf der richtigen Steuerung erzeugt wird.

HINWEIS!

Alle im Vergleichseditor des Safety Applikationsobjekts aufgelisteten, geänderten Safety-Objekte (Prüfsumme des aktuellen Projekts entspricht nicht ihrer gepinnten Prüfsumme) müssen neu verifiziert werden.

Zur Einflussanalyse bei Änderungen können auch die Kontrollfluss- und Datenflussanalyse verwendet werden: *Kapitel 9.3.6.3 „Globale Kontrollflussanalyse“ auf Seite 204* und *Kapitel 9.3.6.5 „Datenflussanalyse“ auf Seite 205*. Dabei kann der Service-Mitarbeiter zu jeder Variablen in einer geänderten GVL oder einem geänderten Mapping die Liste der Verwendungsstellen erhalten. Zu einer geänderten POU kann dabei eine Liste der Verwendungsstellen (betroffenen POUs) erzeugt werden.

Bei erforderlicher, erneuter Verifikation, bzw. Teilverifikation der Sicherheitsapplikation siehe *Verifikation*

Dokumentation von Änderungen

HINWEIS!
Die für die Einflussanalyse und die Dokumentation von Änderungen qualifizierte Ansicht ist der Vergleichseditor des Safety Applikationsobjekts.

Die Änderung im Programm können durch Datenfluss und Kontrollfluss Auswirkungen auf unveränderte Teile des Programms haben. Um mögliche bzw. untersuchte Auswirkungen zu dokumentieren, kann die Querverweisliste zu Variablen, die nun anders beschrieben werden, oder FB-Instanzen die nun anders aufgerufen werden, ausgedruckt werden.

Der Ausdruck der Querverweisliste erfolgt durch Aktivierung der Schaltfläche im Editor der Safety Querverweisliste.

Des Weiteren kann für Dokumentation von Änderungen der Vergleichseditor des Safety Applikationsobjekts ausgedruckt werden.

Der Ausdruck erfolgt in diesem Fall durch Aktivierung des Befehls *„Vergleich drucken...“* im Vergleichseditor.

13.4.2 Änderungen in Projekten mit Querkommunikation

HINWEIS!
Bei Änderung des Senders, die sich auf die Variablenwerte der Netzwerkvariablenliste (Sender) auswirkt, muss der Anwender nicht nur Auswirkungen auf die Empfänger im gleichen Projekt sondern die Auswirkungen auf alle Empfänger dieser Netzwerkvariablenliste (Sender) im UDP-Netzwerk aus gegebenenfalls verschiedenen Projekten betrachten.

Wenn die betroffenen Empfänger gar nicht bekannt sind (z.B. bei Entwicklung von Teilmaschinen, welche andere zu einer vollständigen Maschine integrieren, oder von Maschinen, die in einer Maschinenhalle miteinander kommunizieren sollen), wird folgendes Vorgehen von CODESYS unterstützt: Der Anwender ändert die Objektversion der Sender-Netzwerkvariablenliste (im Eigenschaftendialog, Registerkarte *„Safety)“*. Damit erreicht er, dass kein alter, unveränderter Empfänger im Netzwerk die neuen Werte empfangen kann. Er erzwingt, dass der Integrator der Gesamtmaschine bzw. der Betreiber der Maschinenhalle alle Empfänger im Netzwerk neu erzeugen muss, wodurch er zum Analysieren der Auswirkung der Senderänderung auf den Empfänger gebracht wird. Eine Auswirkungsanalyse der semantischen Änderung der Variablen auf den Empfänger mag dann immer noch ergeben, dass keine funktionale Anpassung des Empfängers erforderlich ist und nur der Safety NVL-Empfänger aktualisiert werden muss, um die geänderte Objektversion des Senders zu übernehmen (was dann als Dokumentation der Durchführung der Analyse verstanden werden kann).

HINWEIS!
Eine modulare Erweiterung einer Maschine um eine Steuerung mit eigenen für andere bestehende Steuerungen relevanten Signale (zum Beispiel ein weiteres Not-Halt-Gerät), die von bestehenden Safety NVL-Empfängern mit berücksichtigt werden sollen, wird nicht unterstützt!

HINWEIS!
Änderung der Empfänger - Auswirkung auf Sender
Wenn ein neuer Safety NVL-Empfänger hinzugefügt, oder ein bestehender Safety NVL-Empfänger beendet wird, kann dies einen zeitlichen Einfluss auf die Standardsteuerung des Empfängers, die Standardsteuerung des Senders, und den Sender der Variablen haben. Wenn durch das Hinzufügen von Safety NVL-Empfängern die Zykluszeit des Senders überschritten wird, könnte eine manuelle Anpassung des Senders mit anschließendem Download nötig werden.

HINWEIS!
Bei Änderung von Applikation oder Signallängen im Prozess muss die Änderungsanalyse auch Auswirkungen in Form eines möglichen Undersamplings analysieren. Vergleiche *„Gefahr des Undersamplings“ auf Seite 41*.

Projektänderung und Auswirkungsanalyse

HINWEIS!
Wenn sich durch Änderungen in der Sendersteuerung oder an ihren Eingangsmodulen die Eigenschaften der gesendeten Werte ändern (andere Wertebereiche, andere Wertverläufe, andere zeitliche Eigenschaften der Werte, andere Bedeutung in der Maschine), wirkt sich das auf die empfangenen Werte in den Empfängersteuerungen aus. Das muss bei der Auswirkungsanalyse der Änderung berücksichtigt werden.

Änderungen in oder an der Empfängersteuerung oder Hinzufügen von Empfängersteuerungen wirken sich nicht funktional auf die Sendersteuerung aus; es besteht also keine funktionale Abhängigkeit des Senders von seinen Empfängern; Ausnahme: Die Zykluszeit kann überschritten werden.

Änderungen in oder an einer Empfängersteuerung oder Hinzufügen von Empfängersteuerungen können höchstens zu Kommunikationsunterbrechungen führen (z. B. bei Überschreitung verfügbarer Resourcen oder bei doppelter Adressvergabe). Ansonsten wirken sich diese Änderungen nicht funktional auf die anderen Empfängersteuerungen aus.

Verfahren bei Änderungen und Wiederverwendung der abgenommenen Software

14 Feldbusse und Netzwerkvariablen

14.1 Allgemeiner Teil

Begriffsdefinitionen

Ausgangstelegramm ist das protokollspezifische Telegramm (Safety-PDO) von der Sicherheitssteuerung in das sichere Feldgerät. Dieses Telegramm beinhaltet die Ausgangsdaten an das sichere Feldgerät.

Eingangstelegramm ist das protokollspezifische Telegramm (Safety-PDO) vom sicheren Feldgerät in die Sicherheitssteuerung. Dieses Telegramm beinhaltet die Eingangsdaten vom sicheren Feldgerät.

Treiberinstanz: für jedes konfigurierte logische E/A wird implizit Code für eine Treiberinstanz des unterstützten Protokolltyps (z. B.+ für ein FSoE-Feldgerät eine Treiberinstanz vom Typ FSoE-Master angelegt. Für die Standardmodule und Austauschvariablen wird eine Treiberinstanz vom Typ NonSafeIO (siehe ↳ *Kapitel 5.5.4.2.2.2 „Logisches E/A eines Standardfeldgeräts“ auf Seite 75*, ↳ *Kapitel 5.5.4.2.2.3 „Logisches E/A für Datenaustausch mit der Standardsteuerung“ auf Seite 77*) angelegt. Ersatzwerte siehe ↳ *„Unterbrechungen durch die Standardsteuerung“ auf Seite 187*

Interface Beschreibung

Die Sicherheitssteuerung überwacht die Übertragung der E/A-Daten vom, bzw. zum sicheren Feldgerät. Für jedes sichere Feldgerät wird ein logisches E/A vom Programmiersystem erzeugt. Pro logisches Gerät wird automatisch eine Treiberinstanz angelegt.

Mit dem Anlegen der Treiberinstanz wird impliziter Code generiert, der für Ein- bzw. Ausgangsvariablen folgendes bewirkt:

- Phase 1 (Eingangsphase): Bearbeitung der Eingangstelegramme (implizit)
  Die Treiberinstanz empfängt das Eingangstelegramm und überprüft dies gemäß dessen Protokollspezifikation, z. B. PROFIsafe
  Die Eingangsdaten oder im Fehlerfall die Ersatzwerte werden in die gemappten Eingangsvariablen der Applikation kopiert.
- Phase 2: Bearbeitung der Anwenderapplikation
  Die Ausgangsdaten werden in Abhängigkeit der Eingangsdaten und des Zustands der Applikation generiert.
- Phase 3 (Ausgangsphase): Bearbeitung der Ausgangstelegramme (implizit)
  Die gemappten Ausgangsdaten der Applikation werden der Treiberinstanz übergeben. Die Treiberinstanz generiert das Ausgangstelegramm gemäß dessen Protokollspezifikation und schickt dieses an das sichere Feldgerät.

Default-Verhalten des Treibers

Hinweis Drv_1 (Start der Applikation)

VORSICHT!

Mit Start der Applikation kann es ab dem ersten Zyklus zum Austausch von Prozessdaten zwischen Feld und Applikation kommen. Durch die Verwendung von FBs mit Anlaufsperre (S_StartReset = FALSE) oder ansonsten durch die Implementierung anderer System- und Applikationsmaßnahmen (siehe Safety-Anwenderhandbuch ↳ *„Regel FB1 (S_StartReset)“ auf Seite 115*) ist sicherzustellen, dass es beim Start der Applikation nicht zu einem unerwarteten (oder unbeabsichtigten) Starten der Maschine kommt.

Hinweis Drv_2 (Kommunikationsfehler am Start)

VORSICHT!

Die üblichen Kommunikationsfehler am Start verhindern per Default nicht den automatischen Beginn der Prozessdatenkommunikation, sondern verzögern ihn nur. Siehe Safety Anwenderhandbuch ↳ *„Eingang für automatische Quittierung von Anlauffehler“ auf Seite 273* (auto-Quitt-Anlauffehler).

Die Wiederaufnahme der Prozessdatenübertragung nach einem Kommunikationsfehler ist nicht automatisch

Das Default-Verhalten des Treibers für das Anlaufen nach einem Reset bzw. für das Wiederanlaufen nach einem Kommunikationsfehler wird von dem Initialwert des jeweiligen Eingangs definiert. Um das Default-Verhalten zu überschreiben, muss der Baustein in der Applikation aufgerufen werden und der entsprechende Eingang entsprechend belegt werden:

Der automatische Beginn der Prozessdatenübertragung nach dem Start wird programmatisch verhindert, indem der Baustein in der Applikation aufgerufen und der Eingang *auto-Quitt-Anlauffehler* auf FALSE gesetzt wird (siehe ↳ *„Eingang für automatische Quittierung von Anlauffehler“ auf Seite 273*).

Die automatische Wiederaufnahme der Prozessdatenübertragung wird programmatisch freigegeben, indem der Baustein in der Applikation aufgerufen und der Eingang *auto-Quitt-Unterbrechung* auf TRUE gesetzt werden (siehe ↳ *„Eingang für automatische Quittierung nach Unterbrechung“ auf Seite 273*).

Eingang für automatische Quittierung von Anlauffehler

Name	Datentyp	Initialwert	Beschreibung, Parameterwerte
‹auto-Quitt-Anlauffehler›	BOOL	TRUE	Anlaufverhalten nach einem Reset (Befehle: *Reset kalt Applikation zurücksetzen (- 185)* und *„Neustart der Bootapplikation“ auf Seite 172*) der Applikation, z. B. PowerON. TRUE: Automatische Quittierung von Fehlern während der Anlaufphase der sicheren Kommunikation bis einmal die sichere Übertragung aufgenommen worden ist. FALSE: Explizite, applikative Quittierung von aufgetretenen Fehlern während der Anlaufphase der sicheren Kommunikation erforderlich.

HINWEIS!

Beachten Sie *„Hinweis Drv_2 (Kommunikationsfehler am Start)“ auf Seite 272*: Der Initialwert für den Eingang für das Anlaufverhalten nach einem Reset ist TRUE. Es werden sämtliche Fehler für das Anlaufverhalten nach einem Reset bestätigt.

Folglich setzt die Prozessdatenkommunikation ein, sobald der initiale Kommunikationsfehler verschwunden ist.

HINWEIS!

Beachten Sie *„Hinweis Drv_1 (Start der Applikation)“ auf Seite 272*: Mit Setzen von *auto-Quitt-Anlauffehler* auf FALSE kann man das automatische Anlaufen der Prozessdatenkommunikation nicht unterbinden. Treten beim Anlauf keine Fehler auf, so kann der Stack automatisch anlaufen, selbst wenn *auto-Quitt-Anlauffehler* auf FALSE ist. Dies muss der Anwender in der Applikation berücksichtigen.

Eingang für automatische Quittierung nach Unterbrechung

Name	Datentyp	Initialwert	Beschreibung, Parameterwerte
‹auto-Quitt-Unterbrechung›	BOOL	FALSE	TRUE: Automatische Quittierung nach einem Kommunikationsfehler FALSE: Explizite, applikative Quittierung von Kommunikationsfehler erforderlich.

Hinweis Drv_3 (Eingang für automatische Quittierung nach Unterbrechung)

VORSICHT!

Sind *auto-Quitt-Anlauffehler* und *auto-Quitt-Unterbrechung* TRUE, so werden sämtliche Fehler bestätigt. Dies ist nur in absoluten Ausnahmefällen sinnvoll.

Wiederanlaufverhalten nach einem Kommunikationsfehler

Der Initialwert für den Eingang für das Wiederanlaufverhalten nach einem Kommunikationsfehler ist FALSE, es wird der Fehler bzgl. Kommunikationsübertragung nicht automatisch bestätigt.

Es ist ein expliziter Aufruf der Baustein-Instanz mit entsprechender Beschaltung der Eingänge erforderlich.

Eingang zur Quittierungsflanke (manuelle Quittierung)

Fehler können mit einer positiven Flanke am Eingang zur Quittierung bestätigt werden, sofern der Ausgang für die Quittierungsanforderung (*„Ausgang zur Quittierungsanforderung“ auf Seite 275*) gesetzt ist.

Wenn sämtliche Fehler automatisch bestätigt werden (was nur in Ausnahmefällen sinnvoll ist), dann wird dieser Eingang nicht benötigt und kann unbelegt bleiben.

Wenn aktuell keine Quittierung angefordert wird, wird der Eingang ignoriert: Dehalb kann das gleiche Signal mit dem Eingang zur Quittierungsflanke aller Treiberinstanzen verbunden werden, um eine unspezifische Bestätigung von Kommunikationsproblemen zu realisieren.

Name	Datentyp	Initialwert	Beschreibung, Parameterwerte
‹Quitt-Flanke›	BOOL	FALSE	Mit einer steigenden Flanke des Eingangs wird die Wiederaufnahme der Sicherheitsfunktion nach einem Fehler bestätigt.

Hinweis Drv_4 (Quitt-Flanke)

VORSICHT!

Quitt-Flanke ist ein Eingang zur Bestätigung durch den Anwender. Er soll nicht programmatisch gesetzt, sondern mit einem Eingangssignal verbunden werden.

Der Eingang Quitt-Flanke benötigt eine steigende Flanke. Er sollte, nachdem der Anwender die Bestätigung gestoppt hat, wieder auf FALSE gehen, um die Quittierung abzuschließen. Erst dann wird beim nächsten Kommunikationsfehler am Ausgang Quitt-Anf die nächste Quittierung angefordert.

Ausgang zur Quittierungsanforderung

Der Ausgang steht auf TRUE, wenn ein Kommunikationsfehler aufgetreten ist (Anlauffehler oder Unterbrechung) und dieser nur manuell bestätigt werden muss, um die Kommunikation wieder aufzunehmen.

Ob die Verbindung eine Bestätigung durch den Anwender benötigt, wird durch den Ausgang *Quitt-Anf* = TRUE angezeigt. Normalerweise wird es dem Anwender gemeldet, dass seine Bestätigung gebraucht wird (zum Beispiel mithilfe einer Austauschvariablen zur Standardsteuerung und Anzeige in der Visualisierung)

Wenn der Ausgang gesetzt ist, kann der Fehler am Eingang zur Quittierung bestätigt werden.

Name	Datentyp	Initialwert	Beschreibung, Parameterwerte
‹Quitt-Anf›	BOOL	FALSE	

Die Anzeige am Ausgang Quitt-Anf kann nur erfolgen:

1. wenn der Kommunikationsfehler nicht automatisch quittiert wird, d.h. wenn auto-Quitt-Anlauffehler bzw. auto-Quitt-Unterbrechung FALSE ist.

2. wenn der Eingang „Quitt-Flanke“ momentan FALSE ist, d.h. wenn eine manuelle Quittierung begonnen, aber noch nicht abgeschlossen wurde.

Falls also ein neues Kommunikationsproblem auftritt, während der Anwender ein vorher aufgetretenes Problem noch bestätigt, so wird das neue Kommunikationsproblem erst angezeigt und bearbeitet, nachdem der Anwender die Bestätigung des vorherigen Problems beendet hat.

Explizites Aufrufen des Bausteins

Wird der Baustein vom Anwender in der Applikation explizit aufgerufen (Phase 2), dann können die FB-Eingänge gesetzt und die FB-Ausgänge gelesen werden, wobei der FB selbst keine Operationen ausführt. Zu jeder Treiberinstanz kann es in der Applikation höchstens einen Aufruf geben.

Die FB-Ausgänge der Treiberinstanz können unabhängig von einem Aufruf ausgelesen werden.

14.2 PROFIsafe

Anwendbare Sicherheitsstandards

PROFIsafe ist eine international genormte (IEC 61784-3-3) Technologie für funktional sichere Kommunikation.

Grundlage: [N3.1.2] *PROFIsafe – Profile for Safety Technology on PROFIBUS DP and PROFINET IO*, Version 2.4, March 2007, Order No. 3.192b

Zusätzlich: [N3.1.5] *PROFIsafe - Profile for Safety Technology on PROFIBUS DP and PROFINET IO*, Version 2.5, December 2012, Order No.: 3.192b

Begriffsdefinitionen

F-Device ist ein sicheres, dezentrales Feldgerät, das die PROFIsafe-Kommunikation unterstützt.

Systemanforderungen

HINWEIS!

Der Anwender muss die feldbus-spezifischen Systemanforderungen (siehe *Kapitel 14.2.3 „PROFIsafe-spezifische Nachweise für die Abnahme“ auf Seite 282* und weitere, allgemeine Systemanforderungen beachten, z.B. die Installationsrichtlinien der IEC 61918 [N3.1.2-Kap. 9.2] und die Einschränkung auf Ethernet-Switches mit Industrietauglichkeit [N3.1.2-Kap.9.5.3].

HINWEIS!

Wenn PROFIsafe auf PROFINET verwendet wird: In dem dazu verwendeten Ethernet sind gemäß PROFIsafe Norm IEC 61784-3-3 keine Regler erlaubt, die das Verlassen des Netzwerks ermöglichen und keine Single-Port-Router.

Begrenzung Geräteanzahl, die an einer Sicherheitsfunktion beteiligt sind

VORSICHT!

Kommunikationsbeziehungen pro Sicherheitsfunktion

Für SIL3 Sicherheitsfunktionen erlaubt die PROFIsafe Norm maximal 100 PROFIsafe-Geräte, damit die durchschnittliche Wahrscheinlichkeit gefährlicher Fehler pro Stunde (PFH) unter dem SIL3-Grenzwert von 10^{-9} bleibt.

Für SIL2 Sicherheitsfunktionen erlaubt die PROFIsafe Norm bis zu 1000 PROFIsafe-Geräte, da die Wahrscheinlichkeit gefährlicher Fehler pro Stunde (PFH) um 4 x 10^{-12} für jedes zusätzliche Gerät ansteigt.

14.2.1 Bibliothek Safety PROFIsafeHost

Für PROFIsafe-Geräte wird der Treiberbaustein PROFIsafeHost aus der Bibliothek SafetyProfisafeHost verwendet. Die allgemeine Beschreibung von Treiberbausteinen finden Sie in ↳ *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*. Die detaillierte Beschreibung des Bausteins (Schnittstelle, Verhalten, Diagnose) finden Sie in der Onlinehilfe.

Die Version des hier beschriebenen Bausteins entspricht der neuesten Version des Bausteins in der Versionsliste für ↳ Kapitel 15.1.3 „Treiberbibliotheken“ auf Seite 297

VORSICHT!

Bei Verwendung von PROFIsafe-Geräten sind die allgemeinen Sicherheitshinweise für Bibliotheksbausteine (↳ *Kapitel 15.1.1 „Hinweise zu den Versionslisten“ auf Seite 293*) und die Sicherheitshinweise zu Treiberbausteinen im Allgemeinen zu beachten (↳ *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*). Dabei entspricht der Eingang *„StartOA“* dem Eingang *‹auto-Quitt-Anlauffehler›* und der Eingang*„ AutoOA“* dem Eingang *‹auto-Quitt-Unterbrechung›*. Der Eingang *„OA_C“* entspricht dem Eingang *‹Quitt-Flanke›* für manuelle Quittierung. Siehe ↳ *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*.

Erkennung Loopback-Fehler

Loopback-Fehler werden erkannt und mit Diagnosecode 16#0C101 angezeigt.

HINWEIS!

SIL Monitor

Diese Implementierung unterstützt die Variante B des SIL Monitor. Jeder CRC-Fehler über das empfangene Telegramm führt zu einer Fehlerreaktion: Fehlerzustand des PROFIsafeHost 0C103, Fehlerzustand des F-Device: 16#C2XX Bit 2.

Wurde die durch eine Diagnosemeldung verursachte Anfrage zum manuellen Operator Acknowledgement mehr als einmal innerhalb von 100 Stunden durchgeführt, dann sollte der verantwortliche Servicetechniker hinzugezogen werden.

Für Betreiber und Servicetechniker: Dies stellt eine starke Beeinträchtigung der Datenübertragung innerhalb des Feldbussystems dar. Gründe für diese Störungen können sein: Änderungen in der Installation, Korrosion von Buskabelabschirmungen mit Stecker und extreme elektromagnetische Interferenzen. Die Übereinstimmung mit den entsprechenden Installationsrichtlinien sollte geprüft, oder es sollte ein EMC Experte (für weitere Anleitungen siehe Annex der PROFIsafe Specification, version 2.5, December 2012) hinzugezogen werden.

Verwendung der FB-Instanz

In der Applikation wird der PROFIsafeHost-Baustein verwendet für

- Änderung der Default-Werte
- manuellen Bestätigung von Fehlern
- Diagnose der Verbindung zu einem F-Device

Dazu muss die entsprechende Instanz des ProfisafeHost-Bausteins in einem Programm mittels `VAR_EXTERNAL ‹Gerätename›:ProfisafeHost` sichtbar gemacht werden.

Abb. 109: Baustein PROFIsafeHost

14.2.2 PROFIsafe Parameter: F-Parameter und i-Parameter

Die Parameter von PROFIsafe werden in zwei Kategorien eingeteilt:

- **F-Parameter**: Protokollparameter von PROFIsafe.
- **i-Parameter**: sicherheitsgerichteten Geräteparameter von PROFIsafe-Geräten

In einer Sicherheitsapplikation befinden sich die Parameter im Editor der entsprechenden logischen E/As:

- i-Parameter in der Registerkarte *„Sichere Parametrierung“*
- F-Parameter in der Registerkarte *„Sichere Konfiguration“*

HINWEIS!

Die allgemeinen Hinweise zur sicheren Parametrierung und sicheren Konfiguration in *Kapitel 5.5.4.2.3.2 „Sichere Parametrierung und Sichere Konfiguration “ auf Seite 84* sind zu beachten.

Es ist geräteabhängig, welche der im Folgenden aufgeführten F-Parameter tatsächlich verwendet werden.

- *„F_Check_SeqNr“*
  - V2 mode: immer in die CRC2-Generation einzubeziehende fortlaufende Nummer
  - nicht editierbar
- *„F_Check_iPar“*
  - Hersteller-spezifische Verwendung innerhalb eines homogenen Systems
  - nicht editierbar
- *„F_SIL“*
  - SIL, den der Anwender vom jeweiligen F-Device benötigt. Er wird mit der Angabe des Herstellers verglichen.
  - editierbar
- *„F_CRC_Length“*
  - CRC Länge
  - Wert: 3 Byte CRC
- *„F_Par_Version“*
  - Versionsnummer von F-Parametern/FSCT 3/1 Betriebsmodus
  - Voreingestellter Wert: V2 mode (Nur V2 wird unterstützt)
  - editierbar
- *„F_Block_ID“*
  - Identifikation des Parameterblock-Typs
  - nicht editierbar

Die folgenden 3 Parameter dienen dem Einstellen der bei der Planung des Gesamtsystems (*Kapitel 4 „Planung des Gesamtsystems“ auf Seite 33*) vergebenen Quell- und Zieladressen und der für die Reaktionszeiten der Sicherheitsfunktionen nötigen Watchdog-Zeiten:

- *„F_Source_Add“*
  - editierbar
  - Durch eine Änderung von *„F_Source_Add“* ändert sich der *„F_Par_CRC“*

VORSICHT!

Der Anwender muss organisieren, dass der F-Host innerhalb des PROFINET-Netzwerks bzw. PROFIBUS-Feldbusses eine eindeutige Quelladresse hat (auch bei mehreren Sicherheitssteuerungen im gleichen Feldbus).

Bei Topologie T2, wenn mehrere Sicherheitssteuerungen unter dem gleichen PROFIBUS/ PROFINET-Master geachtet werden: Die Eindeutigkeit der F_Source_Adr im ganzen PROFINET/ PROFIBUS-Netzwerk muss gewährleistet sein.

Der Anwender muss organisieren, dass die Quelladresse „F_Source_Add“ aller F-Devices eines F-Hosts exakt der Quelladresse ihres F-Hosts entspricht.

- *„F_Dest_Add“*
  - Eindeutige Zieladresse des F-Devices im PROFIsafe-Netzwerk.

    - Kommen zur Steuerung einer Anlage mehrere F-Hosts zum Einsatz, wird empfohlen, allen F-Devices der Anlage eine eindeutige *„F_Dest_Add“* zu geben. Hierzu wird empfohlen, dass der Anwender eine Aufstellung mit den für die einzelnen Hosts zulässigen Wertebereichen für die*„F_Dest_Add“* macht.

    Dies muss auch realisiert werden, wenn die Voraussetzung, dass jedes F-Device über die Verkabelung mit genau einem F_Host verbunden ist und somit seine F-Parameter nur von diesem F-Host erhält, nicht gegeben ist.
  - editierbar
  - Durch eine Änderung von *„F_Dest_Add“* ändert sich der *„F_Par_CRC“*

VORSICHT!
Der Anwender muss organisieren, dass alle F-Devices innerhalb eines PROFINET-Netzwerks bzw. PROFIBUS-Feldbusses eine eindeutige Zieladresse *„F_Dest_Add“* haben (auch bei mehreren Sicherheitssteuerungen im gleichen Netzwerk).

Der Anwender muss darauf achten, dass die „F_Dest_Add“ in der Registerkarte „Sichere Konfiguration“ mit der Einstellung am Adressschalter des F-Devices (in der Regel DIP-Schalter) übereinstimmt.

- *„F_WD_Time“*
  - Zeitangabe in Millisekunden für Wachtdog-Timer, der die Dauer bis zum Empfang der nächsten gültigen PROFIsafe-Nachricht überwacht.
  - Default-Wert der GSD-Datei: Maximale Bearbeitungszeit des F-Devices
  - edtierbar
  - durch eine Änderung von *„F_WD_Time“* ändert sich der *„F_Par_CRC“*
- *„F_iPar_CRC“*
  - Prüfsumme, die aus den i-Parametern des F-Devices berechnet wird.
  - Durch eine Änderung von *„F_iPar_CRC“* ändert sich der *„F_Par_CRC“*

Als Wert von F_iPar_CRC muss manuell der Wert der Prüfsumme (CRC) über die i-Parameter des F-Geräts eingetragen werden. Der CRC über die i-Parameter befindet sich in der Registerkarte „Sichere Parametrierung“ des logischen E/As des F-Devices oder in dem jeweiligen F-Device-Hersteller-spezifischen Tool.

- *„F_Par_CRC“*
  - Prüfsumme über alle F-Parameter.
  - Berechnung durch sichere Editoren
  - Damit wird die fehlerfreie Übertragung der F-Parameter sichergestellt.
  - nicht editierbar

14.2.3 PROFIsafe-spezifische Nachweise für die Abnahme

Bei der Abnahme einer Sicherheitsapplikation muss ein Nachweis erbracht werden für:

- Eindeutige Quelladresse jedes F-Hosts innerhalb einer Anlage/ eines Netzwerks
- Die Quelladressen aller F-Devices eines Hosts müssen der Quelladresse ihres F-Hosts innerhalb einer Anlage/eines Netzwerks entsprechen. (*„F_Source_Add“* in der Registerkarte *„Sichere Konfiguration“* des logischen E/As des sicheren Feldgeräts (F-Devices))
- Eindeutige Zieladresse aller F-Devices aller F-Hosts. (*„F_Dest_Add“* in der Registerkarte *„Sichere Konfiguration“* des logischen E/As des sicheren Feldgeräts (F-Devices))
- Maximal 100 PROFIsafe-Geräte pro SIL3-Sicherheitsfunktion
- Maximal 1000 PROFIsafe-Geräte pro SIL2-Sicherheitsfunktion

14.3 FSoE

Anwendbare Sicherheitsstandards

FSoE (Fail Safe over EtherCAT) ist ein für EtherCAT entwickeltes Safety-Protokoll für SIL3 nach IEC 61508.

Grundlagen

- [N3.5.4] ETG: *Safety over EtherCAT Protocol specification (ETG.5100)*, Version 1.2.0 , 11.03.2011
- [N3.5.5] ETG: *Safety over EtherCAT Implementation Guide (ETG.5101)*, Version 1.1.1, 14.05.2010"

Begriffsdefinitionen

Safety Device ist ein sicheres, dezentrales Feldgerät, das die FSoE Kommunikation unterstützt.

Modulare EtherCAT-Geräte: Es sind EtherCAT-Geräte möglich, die mehrere parallele FSoE-Verbindungen gleichzeitig besitzen. In dem Fall gibt es 1 EtherCAT-Geräteobjekt im Gerätebaum des CODESYS-Projekts, das mit mehreren logischen Geräten der Sicherheitsapplikation verbunden ist. Über jedes logische Gerät wird eine der FSoE-Verbindungen des EtherCAT-Geräts konfiguriert und verwaltet.

Systemanforderungen

HINWEIS!

Der Anwender muss die feldbus-spezifischen Systemanforderungen (siehe *Kapitel 14.3.3 „FSoE-spezifische Nachweise für die Abnahme“ auf Seite 286* und weitere, allgemeine Systemanforderungen beachten, z. B. an die angeschlossenen Geräte (elektrische Sicherheit z. B. nach IEC 60204-1) [N3.5.4-Kap.9.2], an Buskoppler und andere Geräte in der Kommunikationsstrecke [N3.5.4-Kap. 9.5.1].

HINWEIS!

Wenn das für EtherCAT verwendete Ethernet nicht auf die Maschine begrenzt ist, müssen gemäß FSoE Norm IEC 61784-3-12 die Connection IDs im gesamten erreichbaren Kommunikationsnetzwerk eindeutig sein.

VORSICHT!

Restfehlerrate

Die FSoE-Spezifikation fordert, dass von der Treiberinstanz gemeldete Kommunikationsfehler nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt.

Begrenzung Verbindungsanzahl pro Feldbus

In einem EtherCAT-Feldbus sind, unabhängig von der Anzahl der Sicherheitssteuerungen in diesem Feldbus maximal 65 535 FSoE-Verbindungen möglich. (Diese Zahl ergibt sich aus dem Wertebereich der eindeutigen Connection ID) Die maximale Anzahl von FSoE-Geräten im Feldbus ist niedriger, wenn ein FSoE-Gerät gleich mehrere FSoE-Verbindungen belegt.

14.3.1 Bibliothek Safety FSoEMaster

Für FSoE-Geräte wir der Trieberbaustein FSoEMaster aus der Bibliothek SafetyFSoEMaster verwendet. Die allgemeine Beschreibung von Treiberbausteinen finden Sie in *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*. Die detaillierte Beschreibung des Bausteins (Schnittstelle, Verhalten, Diagnose) finden Sie in der Onlinehilfe.

Die Version des hier beschriebenen Bausteins entspricht der neuesten Version des Bausteins in der Versionsliste für Kapitel 15.1.3 „Treiberbibliotheken“ auf Seite 297

VORSICHT!

Bei Verwendung von FSoE-Geräten sind die allgemeinen Sicherheitshinweise für Bibliotheksbausteine (*Kapitel 15.1.1 „Hinweise zu den Versionslisten“ auf Seite 293*) und die Sicherheitshinweise zu Trieberbausteinen im Allgemeinen zu beachten (*Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*). Dabei entspricht der Eingang StartReset dem Eingang *‹auto-Quitt-Anlauffehler›* und der Eingang AutoReset dem Eingang *‹auto-Quitt-Unterbrechung›*.

Der Eingang *„Reset“* entspricht dem Eingang *‹Quitt-Flanke›* zur manuellen Quittierung. Siehe *„Eingang zur Quittierungsflanke (manuelle Quittierung)“ auf Seite 274*.

Verwendung der FB-Instanz (Treiberinstanz)

In der Applikation wird der FSoEMaster-Baustein verwendet für

- Änderung der Default-Werte
- manuelle Bestätigung von Fehlern
- Diagnose der Verbindung zu einem Safety Device

Dazu muss die entsprechende Instanz des FSoEMaster-Bausteins in einem Programm mittels `VAR_EXTERNAL ‹Gerätename›: FSoEMaster` sichtbar gemacht werden.

Abb. 110: Baustein FSoEMaster

14.3.2 FSoE Parameter

Die FSoE Parameter befinden sich in der Registerkarte *„Sichere Konfiguration“*) des jeweiligen logischen E/As des Safety Devices.

HINWEIS!

Die allgemeinen Hinweise zur sicheren Parametrierung und sicheren Konfiguration in *Kapitel 5.5.4.2.3.2 „Sichere Parametrierung und Sichere Konfiguration “ auf Seite 84* sind zu beachten.

VORSICHT!

Bei EtherCAT-Geräten mit mehreren parallelen FSoE-Verbindungen muss jeder eine eigene FSoE-Adresse gegeben werden. Diese verschiedenen FSoE-Adressen des gleichen EtherCAT-Geräts dürfen nicht verwechselt werden.

Einstellen der bei der Planung des Gesamtsystems (*Kapitel 4 „Planung des Gesamtsystems“ auf Seite 33*) vergebenen Adressen und Verbindungs-IDs und der für die Reaktionszeiten der Sicherheitsfuntkonen nötigen Watchdog-Zeiten:

- *„FSoE address“*
  - Adresse des Safety Devices (FSoE Slave) (Stellung des DIP-Schalters),
  - Feld *„Wert“* ist editierbar

VORSICHT!

Adresse (*„FSoE address“*) muss innerhalb des EtherCAT-Feldbusses eindeutig sein (auch über mehrere Sicherheitssteuerungen im gleichen Feldbus hinweg)

- *„Connection ID“*
  - Feld *„Wert“* ist editierbar

VORSICHT!

Verbindungs-ID (*„Connection ID“*) zum Safety-Device muss innerhalb des EtherCAT-Feldbusses eindeutig sein (auch über mehrere Sicherheitssteuerungen im gleichen Feldbus hinweg).

Bei Topologie T2, wenn mehrere Sicherheitssteuerungen unter dem gleichen EtherCAT-Master verwendet werden: Die Eindeutigkeit der FSoE Connections IDs im ganzen EtherCAT-Netzwerk des EtherCAT-Masters muss gewährleistet sein.

- *„WatchdogTime“*
  - Feld *„Wert“* ist editierbar

Die hier aufgelisteten Parameter sind die FSoE Communication Parameter. Nach diesen können noch gerätespezifische Parameter folgen (FSoE Application Parameter).

Die Variante von FSoE Application Parametern, die nicht über die Sicherheitssteuerung sondern über FoE-Dienste in das Gerät übertragen werden, wird nicht unterstützt.

14.3.3 FSoE-spezifische Nachweise für die Abnahme

Bei der Abnahme einer Sicherheitsapplikation muss ein Nachweis erbracht werden für:

- Eindeutige FSoE-Adresse jedes FSoE-Geräts (*„FSoE address“* in der Registerkarte *„Sichere Konfiguration“* des logischen E/As des sicheren Feldgeräts) in der Anlage
- Eindeutige Connection ID in allen logischen Geräten in der Anlage (Eingabefeld *„Connection ID“* in der Registerkarte *„Sichere Konfiguration“* des logischen E/As des sicheren Feldgeräts.
- Die Betriebsanleitung für die Maschine verpflichtet den Betreiber die Kommunikationsfehlerrate im Betrieb zu überwachen: Von der Treiberinstanz gemeldete Kommunikationsfehler dürfen nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt.

14.4 Netzwerkvariablen

Systemanforderungen

VORSICHT!

Restfehlerrate

Von der Treiberinstanz gemeldete Kommunikationsfehler dürfen nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt.

Das Netzwerk, in welchem eindeutige Safety-Adressierungen zu gewährleisten sind, muss eindeutig eingrenzbar sein und darf nicht ohne erneute Eindeutigkeitsprüfung verändert werden:

Inbetriebnehmer und Betreiber dürfen zwischen dem Maschinennetz N1, über das Sicherheitssteuerungen kommunizieren, und einem anderen Netzwerk N2 nur dann eine physikalische Verbindung oder Verbindung durch Netzwerkgeräte (Switches, Router, PCs) herstellen, wenn für das gesamte via N2 per UDP/IP prinzipiell erreichbare Netzwerk N* (in welchem N1 und N2 enthalten sind) folgende Voraussetzungen erfüllt sind:

- Inbetriebnehmer und Betreiber kennen alle Sicherheitssteuerungen in N*.
- Inbetriebnehmer und Betreiber kennen alle von ihnen verwendeten Safety-Adressierungen – und diese sind alle und zusammen mit denen in N1 eindeutig .
- Inbetriebnehmer und Betreiber haben durch organisatorische Maßnahmen für das Netzwerk N* dafür gesorgt, dass bei zukünftigen Änderungen in oder an N* die Eindeutigkeit bewahrt wird. Dies betrifft Änderungen an einer querkommunizierenden Sicherheitssteuerung (auch außerhalb N1 und N2), Hinzufügen einer querkommunizierenden Sicherheitssteuerung unter Standardsteuerungen in N* und die Verbindung von N* mit weiteren Maschinennetzen.

VORSICHT!

Netzwerke, bei denen die genannten Voraussetzungen nicht erfüllt sind, müssen vom Maschinennetzwerk N1 getrennt bleiben. Insbesondere darf das Maschinennetzwerk nie mit dem Internet bzw. an ein mit dem Internet verbundenes Firmennetzwerk verbunden werden (neben der Eindeutigkeit der Adresse spricht auch die geforderte Industrietauglichkeit der Netzwerkgeräte dagegen). Und auch die Maschinennetzwerke baugleicher Seriemaschinen mit Querkommunikation dürfen niemals verbunden werden, oder am gleichen übergeordneten Netzwerk hängen, da sie die identische Sicherheitsapplikationen enthalten und somit die Safety-Adressierungen nicht innerhalb N* eindeutig wären.

VORSICHT!

Wenn Querkommunikation in der Maschine verwendet wird (T3), dann muss das dazu verwendete Ethernet von anderen Netzwerken getrennt sein. Die Trennung muss physikalisch sein. Eine bloße Filterung des IP-Verkehrs zwischen N1 und N2, z.B. durch entsprechend konfigurierte Router oder Firewalls, stellt keine für SIL3-Anwendungen ausreichende Trennung dar.

VORSICHT!

Die Eindeutigkeit muss nicht nur einmal während der Entwicklung oder Inbetriebnahme der Maschine geprüft werden. Die Prüfung muss jedesmal wiederholt werden, wenn neue Steuerungen mit querkommunizierenden Sicherheitssteuerungen an das UDP-Netzwerk angeschlossen werden oder wenn das UDP-Netzwerk mit einem übergeordneten oder untergeordneten UDP-Netzwerk verbunden wird.

HINWEIS!
Der Anwender muss die spezifischen Systemanforderungen und weitere, allgemeine Systemanforderungen beachten.

Begrenzung NVL-Anzahl im Projekt

Die Anzahl von Netzwerkvariablenlisten in einem Projekt ist durch CODESYS auf 100 Safety Netzwerkvariablenlisten (Sender) und 20 Safety Netzwerkvariablenlisten (Empfänger) pro Sender beschränkt.

HINWEIS!
Man bewegt sich außerhalb der Spezifikation, wenn die Zusammenfassung aller NVL-Verbindungen zwischen den gleichen zwei Sicherheitssteuerungen mehr als 100 Kommunikationsbeziehungen zwischen Sicherheitssteuerungen ergibt.

14.4.1 Bibliothek SafetyNetVar

Für Safety NVL-Empfänger wir der Treiberbaustein NetVarReceiver und für Safety NVL-Sender der Treiberbaustein NetVarSender aus der Bibliothek SafetyNetVar verwendet. Die allgemeine Beschreibung von Treiberbausteinen finden Sie in *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*. Die detaillierte Beschreibung des Bausteins (Schnittstelle, Verhalten, Diagnose) finden Sie in der Onlinehilfe.

Die Version des hier beschriebenen Bausteins entspricht der neuesten Version des Bausteins in der Kapitel 15.1.2 „Applikative Bibliotheken“ auf Seite 293.

VORSICHT!
Safety NVL-Empfänger
Bei Verwendung von Safety NVL-Empfängern sind die allgemeinen Sicherheitshinweise für Bibliotheksbausteine (*Kapitel 15.1.1 „Hinweise zu den Versionslisten“ auf Seite 293*) und die Sicherheitshinweise zu Treiberbausteinen im Allgemeinen zu beachten (*Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*). Dabei entspricht der Eingang *„StartReset“* dem Eingang *‹auto-Quitt-Anlauffehler›* zur automatischen Quittierung von Fehlern beim Anlauf und der Eingang *„AutoReset“* dem Eingang *‹auto-Quitt-Unterbrechung›* zur automatischen Quittierung nach Unterbrechungen.

Der Eingang *„Reset“* entspricht dem Eingang *‹Quitt-Flanke›* zur manuellen Quittierung. Siehe *„Eingang zur Quittierungsflanke (manuelle Quittierung)“ auf Seite 274*.

VORSICHT!
Safety NVL-Sender
Bei Verwendung von Safety NVL-Sendern sind die allgemeinen Sicherheitshinweise für Bibliotheksbausteine (*Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*) zu beachten.

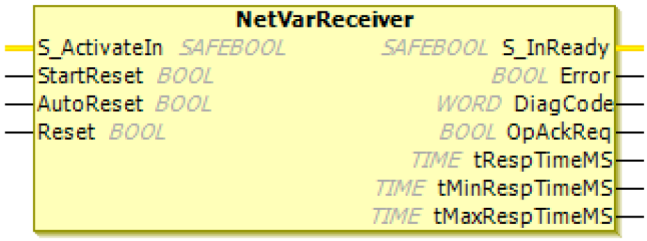


Abb. 111: Baustein NetVarReceiver

Abb. 112: Baustein NetVarSender

14.4.2 Safety NetVar Parameter

Einstellen der bei der Planung des Gesamtsystems vergebenen Adressen und Connection IDs und der für die Reaktionszeiten der Sicherheitsfunktionen nötigen Watchdog-Zeiten:

- *„Safety Address“*
  Adresse der Safety Netzwerkvariablenliste (Sender)
  Eingabefeld *„Safety-Adresse dieser Variablenliste“* im Objekt *„Safety Netzwerkvariablenliste (Sender)“*, Registerkarte *„Safety Konfiguration“*

VORSICHT!
Die Adresse (*„Safety Address“*) jeder Safety Netzwerkvariablenliste muss projektweit, im gesamten Netzwerk und in allen per UDP/IP erreichbaren Steuerungen eindeutig sein.

- *„Max. Receivers“*
  Maximale Anzahl von Safety Netzwerkvariablenlisten (Empfänger), die gleichzeitig von dieser Safety Netzwerkvariablenlisten (Sender) Werte erhalten können
- *„Connection ID“*
  Safety NVL Verbindungs-ID
  Im Objekt *„Safety Netzwerkvariablenliste (Empfänger)“*, Registerkarte *„Safety-Konfiguration“*

VORSICHT!
Verbindungs-ID (*„Connection ID“*) zur Safety Netzwerkvariablenliste muss projektweit eindeutig sein.

Bei Topologie T3: Die Eindeutigkeit der Netzwerkvariablenlisten Connection IDs aller Sicherheitssteuerungen im Netzwerk der ganzen Maschine muss gewährleistet sein.

- *„Watchdog Time“*

14.4.3 Safety NetVar-spezifische Nachweise für die Abnahme

Bei der Abnahme einer Sicherheitsapplikation muss ein Nachweis erbracht werden für:

- Eindeutige Safety-Adresse jeder Sender-Netzwerkvariablenliste (Eingabefeld *„Safety-Adresse dieser Variablenliste“* im Objekt *„Safety Netzwerkvariablenliste (Sender)“*, Registerkarte *„Safety Konfiguration“*) im gesamten Netzwerk
- Eindeutige Connection ID jeder Empfänger-Netzwerkvariablenliste (im Objekt *„Safety Netzwerkvariablenliste (Empfänger)“*, Registerkarte *„Safety Konfiguration“*) im gesamten Netzwerk

- Die Betriebsanleitung für die Maschine verpflichtet den Betreiber die Kommunikationsfehlerrate im Betrieb zu überwachen: Von der Treiberinstanz NetVarReceiver gemeldete Kommunikationsfehler dürfen nicht öfter als 1 Mal in 5 Stunden auftreten, damit die Restfehlerrate pro Stunde für die sicherheitsbezogenen Signale unter dem SIL3-Grenzwert von 10^{-9} bleibt
- Die Betriebsanleitung für die Maschine verpflichtet den Betreiber bei Update von Sicherheitsapplikationen oder Erweiterungen des Netzwerks erneut die Eindeutigkeit von Netzwerkvariablenlisten Safety-Adressen und Connection IDs nachzuweisen

14.5 PROFIsafe F-Device

Anwendbare Sicherheitsstandards

[N61784.3.3] IEC 61784-3-3: Industrial communication networks - Profiles - Part 3-3: Functional safety fieldbuses - Additional specifications for CPF 3 (2016)

14.5.1 Bibliothek SafetyProfisafeDevice

Diese Bibliothek wird verwendet, wenn die Sicherheitssteuerung als F-Device einer übergeordneten Sicherheitssteuerung (F-Host) verwendet wird.

Die detaillierte Beschreibung des Bausteins finden Sie in der CODESYS PROFIsafe F-Device-Hilfe.

Die Version des hier beschriebenen Bausteins entspricht der neuesten Version des Bausteins in der Versionsliste *Kapitel 15.1.3 „Treiberbibliotheken“ auf Seite 297.*

VORSICHT!

Bei der Verwendung sind die allgemeinen Sicherheitshinweise für Bibliotheksbausteine (*Kapitel 15.1.1 „Hinweise zu den Versionslisten“ auf Seite 293*) und die Hinweise für die *Kapitel 6.4 „Implementierung von F-Modulen“ auf Seite 143* zu beachten.

Verwendung der FB-Instanz

Die Eingänge des Funktionsbausteins PSDeviceModule sind Statussignale, die vom F-DeviceModule zum F-Host gesendet werden. Die Ausgänge des Bausteins sind Steuersignale, die vom F-Host an das F-DeviceModule gesendet werden.

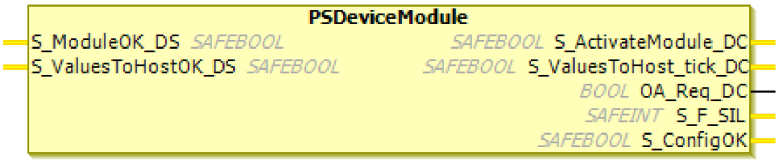


Abb. 113: Baustein PSDeviceModule

14.5.2 F-Modul Parameter

HINWEIS!
Die allgemeinen Hinweise zur sicheren Parametrierung und sicheren Konfiguration in ↳ *Kapitel 5.5.4.2.3.2 „Sichere Parametrierung und Sichere Konfiguration “ auf Seite 84* sind zu beachten.

Die Parameter „Source Address“ und „Destination Address“ stellen gemeinsam die F-Adresse, den "Codename" des F-Moduls dar, die im PROFINET eindeutig vergeben sein muss. Der F-Host kann nur eine Verbindung zum F-Modul aufbauen, wenn dieser "Codename" in seine F-Parametern für dieses F-Modul als „F_Source_Add“ und „F_Dest_Add“ eingetragen sind. Vergleiche ↳ „Hinweis FDev_9“ auf Seite 228

- *„Source Address“*
  Der hier gewählte Wert gibt vor, was im Host-Projekt in der Konfiguration des F-Moduls als Wert des F-Parameters *„F_Source_Add“* eingetragen werden muss, damit sich der F-Host mit diesem F-Modul verbinden kann.
- *„Destination Address“*
  Der hier gewählte Wert gibt vor, was im Host-Projekt in der Konfiguration des F-Moduls als Wert des F-Parameters *„F_Dest_Add“* eingetragen werden muss, damit sich der F-Host mit diesem F-Modul verbinden kann.

15 Vordefinierte Bausteine

Die vordefinierten Bausteine von CODESYS Safety sind in Bibliotheken organisiert. Die Beschreibung der Bausteine finden Sie in der Onlinehilfe. Bei ihrer Verwendung muss auf die im Anschluss genannte korrekte Version und die Sicherheitshinweise geachtet werden.

15.1 Versionsliste der Bausteine

15.1.1 Hinweise zu den Versionslisten

Hinweis Lib_1 (Version)

VORSICHT!
Die Version jedes in die Applikation eingebundenen Bibliotheksbausteins muss mit der dokumentierten Version des Bausteins übereinstimmen (siehe Onlinehilfe und neuester Eintrag in der Spalte *„Version“*)

Hinweis Lib_2 (Gültig)

VORSICHT!
Wenn bei einem Baustein zu einer bestimmten Version in der Spalte *„Gültig“* "Nein" steht, so darf diese Version des Bausteins nicht mehr verwendet werden.

15.1.2 Applikative Bibliotheken

Zu jedem Baustein werden die zu beachtenden Sicherheitshinweise aufgelistet. Sie finden allgemeine Regeln für sicherheitsgerichtete Funktionsbausteine in *Kapitel 6.2.4 „Gestaltungsregeln für PLCopen-konforme Funktionsbausteine“ auf Seite 107* und die spezifischen Sicherheitshinweise für Funktionsbausteine in *Kapitel 15.2 „Spezifische Sicherheitshinweise für applikative Bibliotheksbausteine“ auf Seite 298*

Sicherheitsbibliothek SafetyStandard

Baustein	Version	Gültig	Bemerkung
SF_CTD	1.0.0.0	Ja	
SF_CTU	1.0.0.0	Ja	
SF_CTUD	1.1.1.0	Ja	Eine steigende Flanke an einem der Eingänge CU bzw. CD bewirkt den entsprechenden Zählvorgang, auch wenn der andere Eingang bereits gleich TRUE gesetzt ist.

Baustein	Version	Gültig	Bemerkung
	1.0.0.0	Nein	*Hinweis Lib_2 (Gültig)*
SF_F_TRIG	1.0.0.0	Ja	*Hinweis Lib_4 (SF_F_TRIG)*
SF_R_TRIG	1.0.0.0	Ja	
SF_RS	1.0.0.0	Ja	
SF_SR	1.0.0.0	Ja	
SF_TOF	1.0.0.0	Ja	
SF_TON	1.0.0.0	Ja	
SF_TP	1.0.0.0	Ja	

Sicherheitsbibliothek SafetyPLCopen

Hinweis Lib_3 (Produktnormen)

HINWEIS!
Beim Einsatz jedes Bausteins zur Überwachung oder Steuerung eines Sicherheitsbauteils müssen auch die Anforderungen der entsprechenden Produktnormen beachtet werden.

Baustein	Version	Gültig	Bemerkung
SF_Antivalent	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB5 (Reset)* *Hinweis Lib_5 (TIME-Eingänge)*
SF_EDM	1.0.0.0	Ja	*Hinweis Lib_5 (TIME-Eingänge)*
SF_EmergencyStop	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)*
SF_EnableSwitch	1.0.0.0	Ja	*Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)*
SF_Equivalent	1.0.0.0	Ja	*Hinweis Lib_5 (TIME-Eingänge)*

Baustein	Version	Gültig	Bemerkung
SF_ESPE	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)*
SF_GuardLocking	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)*
SF_GuardMonitoring	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)* *Hinweis Lib_5 (TIME-Eingänge)*
SF_ModeSelector	1.1.0.0	Ja	Der FB wechselt bei einer Modusänderung innerhalb von Zustand 8004 und anschließendem Setzen von S_Unlock auf TRUE nicht mehr in Zustand 8000 (Ausgänge neu setzen), sondern in Zustand 8005 (Warten auf Aktivierung) Die abgelaufene Zeit zur Prüfung gegen ModeMonitorTime wird in Zustand 8005 mit jeder Modusänderung zurückgesetzt. *Hinweis Lib_5 (TIME-Eingänge)* *Hinweis Lib_6 (AutoSetMode)* *Regel FB5 (Reset)*
	1.0.0.0	Nein	*Hinweis Lib_2 (Gültig)*

Baustein	Version	Gültig	Bemerkung
SF_MutingPar	1.1.0.0	Ja	S_AOPD_In gleich FALSE in den Zuständen 8014, 8114, 8314 und 8414 bedingt keinen Fehler (siehe auch entsprechender Hinweis in der Beschreibung des Funktionsbausteins.) MutingEnable gleich FALSE in Zustand 8000 bedingt keinen Fehler *Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB5 (Reset)* *Hinweis Lib_5 (TIME-Eingänge)* *Hinweis Lib_8 (Not S_AOPD_In)*
	1.0.0.0	Nein	*Hinweis Lib_2 (Gültig)*
SF_MutingPar_2Sensor	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Hinweis Lib_5 (TIME-Eingänge)* *Hinweis Lib_9 (Muting_Sensorsignale)*
SF_MutingSeq	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB5 (Reset)* *Hinweis Lib_5 (TIME-Eingänge)* *Hinweis Lib_7 (Muting-Sensorsignale)*
SF_OutControl	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB2 (S_Start_Reset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)* *Hinweis Lib_11 (StaticControl)*
SF_SafetyRequest	1.0.0.0	Ja	*Regel FB5 (Reset)* *Hinweis Lib_5 (TIME-Eingänge)* *Hinweis Lib_10 (Sicherheitsfunktion)*
SF_TestableSafetySensor	1.0.0.0	Ja	*Regel FB1 (S_StartReset)* *Regel FB3 (S_AutoReset)* *Regel FB4 (S_AutoReset)* *Regel FB5 (Reset)* *Hinweis Lib_3 (Produktnormen)* *Hinweis Lib_5 (TIME-Eingänge)*

Baustein	Version	Gültig	Bemerkung
SF_TwoHandControlTypeII	1.0.0.0	Ja	
SF_TwoHandControlTypeIII	1.0.0.0	Ja	

15.1.3 Treiberbibliotheken

Zu jedem Baustein werden die zu beachtenden Sicherheitshinweise aufgelistet. Sie finden diese in ↳ *Kapitel 14.1 „Allgemeiner Teil“ auf Seite 271*.

Sicherheitsbibliothek SafetyProfisafeHost

Baustein	Version	Gültig	Bemerkung
ProfisafeHost	1.0.0.0	Ja	↳ *Hinweis Drv_1 (Start der Applikation)* ↳ *Hinweis Drv_2 (Kommunikationsfehler am Start)* ↳ *Hinweis Drv_3 (Eingang für automatische Quittierung nach Unterbrechung)* ↳ *Hinweis Drv_4 (Quitt-Flanke)*

Sicherheitsbibliothek SafetyFSoEMaster

Baustein	Version	Gültig	Bemerkung
FSoEMaster	1.1.1.0	Ja	↳ *Hinweis Drv_1 (Start der Applikation)* ↳ *Hinweis Drv_2 (Kommunikationsfehler am Start)* ↳ *Hinweis Drv_3 (Eingang für automatische Quittierung nach Unterbrechung)* ↳ *Hinweis Drv_4 (Quitt-Flanke)*

Sicherheitsbibliothek SafetyNetVar

Baustein	Version	Gültig	Bemerkung
NetVarReceiver	1.2.0	Ja	*Hinweis Drv_1 (Start der Applikation)* *Hinweis Drv_2 (Kommunikationsfehler am Start)* *Hinweis Drv_3 (Eingang für automatische Quittierung nach Unterbrechung)* *Hinweis Drv_4 (Quitt-Flanke)*
NetVarSender	1.2.0	Ja	
NetVarSenderStack	1.2.0	Ja	Interner Baustein, in der Applikation nicht zugreifbar

Sicherheitsbibliothek SafetyProfisafeDevice

Baustein	Version	Gültig	Bemerkung
PSDeviceModule	1.5.0	Ja	*Hinweis FDev_1 (Einsatzbereich)* *Hinweis FDev_2 (device fault)* *Hinweis FDev_4 (safe state)* *Hinweis FDev_3 (process values)* *Hinweis FDev_5 (undersampling)*

15.2 Spezifische Sicherheitshinweise für applikative Bibliotheksbausteine

Hinweis Lib_4 (SF_F_TRIG)

Betrifft: SF_F_TRIG

VORSICHT!

Besonderes Verhalten des Bausteins im 1. Zyklus

Im ersten Zyklus erkennt SF_F_TRIG eine fallende Flanke bei anliegendem FALSE und setzt den Ausgang Q auf TRUE. Dieses Verhalten ist konform zum F_TRIG von IEC 61131-3 ab edition2. Es weicht somit vom Verhalten des Bausteins F_TRIG von Standard CODESYS ab, welcher konform zu IEC 61131-3 edition1 ist.

Hinweis Lib_5 (TIME-Eingänge)

Betrifft: SF_Equivalent, SF_Antivalent, SF_GuardMonitoring, SF_ModeSelector, SF_SafetyRequest, SF_EDM, SF_TestableSafetySensor, SF_MutingSeq, SF_MutingPar, SF_MutingPar_2Sensor

VORSICHT!

TIME-Eingänge

Für Entwickler im Extended Level gilt: Alle TIME-Eingänge dürfen nur mit konstanten Werten beschaltet werden. D.h. der Wert darf bei den Aufrufen nicht verändert werden.

Betroffene Eingänge: DiscrepancyTime, DiscTimeEntry, DiscTime11_12, DiscTime21_22, MonitoringTime, ModeMonitoringTime, MaxMutingTime, TestTime

Hinweis Lib_6 (AutoSetMode)

Betrifft: SF_ModeSelector

VORSICHT!

AutoSetMode

Der AutoSetMode-Eingang soll nur aktiviert werden, wenn sichergestellt ist, dass keine Gefährdung auftreten kann, wenn die Sicherheitssteuerung gestartet wird.

Hinweis Lib_7 (Muting-Sensorsignale)

Betrifft: SF_MutingSeq, SF_MutingPar

VORSICHT!

Ein Kurzschluss bei den Muting-Sensorsignalen, oder ein funktionaler Applikationsfehler beim Unterstützen dieser Signale wird von diesem Baustein nicht unterstützt, aber als inkorrekte Mutingsequenz interpretiert (Datentyp BOOL (nicht sicher), versorgt durch die funktionale Anwenderhardware oder - software). Es muss sichergestellt sein, dass so ein Fall nicht zu einem ungewollten Muting führt. Der Anwender sollte dies in seiner Risikoanalyse berücksichtigen.

Hinweis Lib_8 (Not S_AOPD_In)

Betrifft: SF_MutingPar

HINWEIS!

Die Implementierung des Zustandsmodells weicht von der PLCopen Spezifikation 1.0 [N2.1.1] in einem Fall ab: Mit FALSE am Eingang S_AOPD_In wird auch bei aktivem Muting (Zustände 8012, 8021, 8112, 8121) in den Zustand SafetyDemand AOPD gewechselt.

Hinweis Lib_9 (Muting_Sensorsignale)

Betrifft: SF_MutingPar_2Sensor

HINWEIS!

Leitungskontrolle der Muting-Sensorsignale muss in der Sicherheitsschleife aktiv sein.

Hinweis Lib_10 (Sicherheitsfunktion)

Betrifft: SF_SafetyRequest

VORSICHT!

Die Sicherheitsfunktion wird von der angeschlossenen, sicheren Peripherie selbständig durchgeführt (z.B. bei einem Antrieb: harter Stopp, Torque OFF, begrenzte Position, begrenzte Geschwindigkeit, usw.). Der Baustein SF_SafetyRequest initiiert hierfür nur die Anforderung und überwacht sie. Der Baustein definiert nicht die Parameter der angeschlossenen, sicheren Peripherie. Das Verhalten der sicheren Peripherie im Fall der Anforderung muss auf anderem, gerätespezifischem Wege definiert werden (z.B. i-Parameter von PROFIsafe-Geräten). Es ist darauf zu achten, dass die von SF_SafetyRequest ausgelöste Sicherheitsfunktion der aktuellen Situation angemessen ist.

Hinweis Lib_11 (StaticControl)

Betrifft: SF_OutControl

VORSICHT!

StaticControl

Der Eingang StaticControl, soll nur aktiviert werden, wenn sichergestellt ist, dass keine Gefährdungssituation entstehen kann, wenn die Sicherheitssteuerung startet.

16 Liste zulässiger oder geänderter Funktionen

HINWEIS!
Die mit * markierten Funktionen sind kritisch und dürfen in einer angepassten, nicht valierten Installation nicht verwendet werden.

16.1 Zulässige Befehle

Zulässige Menübefehle

Befehle der Kategorie 'Safety Applikation und Task'
alle Befehle

Befehle der Kategorie 'Safety Deklarationen'
alle Befehle

Befehl der Kategorie 'Safety FUP'
alle Befehle

Befehle der Kategorie 'Safety Online' *
alle Befehle

Befehle der Kategorie 'Online' *	
„Einloggen“	*„Ausloggen“*
„Bootapplikation erzeugen“	*„Reset kalt “*
„Start (aktive Applikation)“	*„Stop (aktive Applikation)“*
„Start (selektierte Applikation)“	*„Stop (selektierte Applikation)“*
„Werte schreiben (aktive Applikation)“	*„Werte forcen (aktive Applikation)“*
„Forcen für alle Werte aufheben“	*„Alle Forces zur Überwachungsliste hinzufügen“*
„Forceliste aufheben“	*„Ablaufkontrolle (aktive Applikation)“*
„Binär“	*„Zur Überwachungsliste hinzufügen“*
„Hexadezimal“	*„Dezimal“*

Liste zulässiger oder geänderter Funktionen

Zulässige Befehle

Befehle der Kategorie 'Datei'	
„Vergleichen“	*„Archiv extrahieren“*
„Archiv speichern/versenden“	*„Beenden“*
„Neues Projekt“	*„Projekt öffnen“*
„Projekt schließen“	*„Projekt speichern“*
„Projekt speichern unter“	*„Projektinformationen“*
„Projekteinstellungen“	

Befehle der Kategorie 'Drucken' *	
„Dokumentieren“	*„Drucken“*

Befehle der Kategorie 'Quelltextnavigation' *	
„Querverweise ausgeben“	*„Gehe zur Definition“*

Befehle der Kategorie 'Übersetzen'	
„Übersetzen“	*„Bereinigen“*
„Alles bereinigen - Achtung!“	*„Bibliothekskompatibilität prüfen“*
„Überprüfe alle Pool-Objekte“	*„Neu übersetzen“*

Befehle der Kategorie 'Hilfe'	
„Safety-Versionsinformation anzeigen“	*„Inhalt“*
„Suchen“	*„Index“*
„Informationen“	*„3S Homepage“*

Befehle der Kategorie 'Intelligente Code-Bearbeitung'	
„Eingabehilfe“	*„Variable deklarieren“*

Befehle der Kategorie 'Lesezeichen'	
„Lesezeichen ein-/ausschalten“	*„Lesezeichen löschen“*
„Nächstes Lesezeichen“	*„Vorheriges Lesezeichen“*

Befehle der Kategorie 'Zwischenablage'	
„Alles selektieren“	*„Ausschneiden“*
„Einfügen“	*„Kopieren“*
„Löschen“	

Befehle der Kategorie 'Rückgängig/Wiederherstellen'	
„Rückgängig“	*„Wiederherstellen“*

Befehle der Kategorie 'Suchen/Ersetzen'	
„Ersetzen“	*„Suchen“*
„Vorheriges suchen“	*„Vorheriges suchen (Auswahl)“*
„Weitersuchen“	*„Weitersuchen (Auswahl)“*

Befehle der Kategorie 'Meldungsfenster'	
„Gehe zur Quelltextposition“	*„Nächste Meldung“*
„Vorherige Meldung“	

Befehle der Kategorie 'Einstellungen'	
„Anpassen“	*„Optionen“*

Befehle der Kategorie 'Benutzerverwaltung'	
„Benutzer anmelden“	*„Benutzer abmelden“*
„Rechte“	

Befehle der Kategorie 'Ansicht'	
„Eigenschaften“	*„Geräte“*
„Meldungen“	*„POUs“*
„Querverweisliste“	*„Safety Querverweisliste“*
„Überwachungsliste 2“	*„Überwachungsliste 1“*
„Überwachungsliste 4“	*„Überwachungsliste 3“*
„Alle Forces anzeigen“	*„Werkzeuge“*

Liste zulässiger oder geänderter Funktionen

Zulässige Ansichten

Befehle der Kategorie 'Objekte'	
„Objekt bearbeiten“	*„Objekt hinzufügen“*
„Exportieren“	*„Importieren“*
„Aktive Applikation setzen“	*„Objekt bearbeiten mit“*
„Objekt offline bearbeiten“	*„Ordner hinzufügen“*

Befehle der Kategorie 'Gerätekommunikation'
„Gerätenamen ändern“

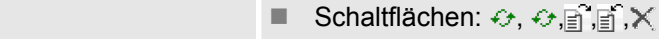
Befehle der Kategorie 'Geräte'	
„Geräte-Repository“	*„Gerät aktualisieren“*

Befehle der Kategorie 'Installation'	
„Bibliotheks-Repository“	*„Package Manager“*

16.2 Zulässige Ansichten

Zulässige Editoren, ihre Register-karten und Steuerelemente

Tab. 15: ***Editor der Sicherheitssteuerung***

Registerkarte	Steuerelement
„Kommunikation“ Hinweis: (Die Option *„Klassische Darstellung der Kommunikationseinstellungen verwenden“* in *„Tools ➔ Optionen“*, Kategorie *„Geräteeditor“* muss aktiviert sein)	Stammt aus CODESYS Basis. Es sind nur die im Folgenden aufgelisteten Steuerelemente zulässig: ■ Auswahllisten – *„Netzwerkpfad zur Steuerung auswählen“* – *„Filter“* – *„Sortierreihenfolge“* ■ Schaltflächen – *„Aktiven Pfad setzen“* – *„Gateway hinzufügen“* – *„Geräte hinzufügen“* – *„Netzwerk durchsuchen“* ■ Optionen – *„Verbindungseinstellungen nicht im Projekt speichern“* – *„Bestätigter Online-Betrieb“*
„Log“	Stammt aus der CODESYS Basis und kann deshalb variieren. Es sind nur die im Folgenden aufgelisteten Steuerelemente zulässig: ■ Option *„Offline-Aufzeichnung“* ■ Option *„UTC Zeit“* ■ Auswahlliste für Komponenten ■ Auswahlliste *„Logger“* ■ Schaltflächen: 
„Safety Online Information“ *	alle Steuerelemente
„Status“	Stammt aus der CODESYS Basis und kann deshalb variieren. Es sind nur die im Folgenden aufgelisteten Steuerelemente zulässig: Steuerelement: Schaltfläche *„Bestätigen“*
„Information“	Stammt aus der CODESYS Basis und kann deshalb variieren. Enthält nur Informationen

Liste zulässiger oder geänderter Funktionen

Zulässige Ansichten

Tab. 16: ***Objekt Bibliotheksverwalter***

Fenster	Steuerelement
Befehlsleiste oberhalb der Fenster	Schaltflächen: ■ *„Bibliothek hinzufügen“* ■ *„Bibliothek löschen“* ■ *„Eigenschaften“* *„Details“* *„Platzhalter“* ■ *„Bibliotheks-Repository“*
Liste der eingebundenen Bibliotheken	
Liste der Bausteine der selektierten Bibliothek	Sortier- und Suchfunktionen
Details der Bausteine mit den Registerkarten *„Eingänge/Ausgänge“*, *„Grafisch“*, *„Dokumentation“*	

Tab. 17: ***Vergleichseditor ****

Steuerelement: Schaltfläche *„Vergleich drucken“*

Tab. 18: ***Editoren von Safety Applikationsobjekt, Safety Task, Safety Globale Variablenliste***

Alle Steuerelemente

Tab. 19: ***Editoren von Safety POU ****

Alle Steuerelemente mit Ausnahme der Lupenfunktion

Tab. 20: Geräteeditor der logischen E/As

Registerkarte	Steuerelement
*„Sichere Konfiguration“ **	alle Steuerelemente
*„Sichere Parametrierung“ **	alle Steuerelemente
*„I/O Abbild“ **	alle Steuerelemente
„Information“	Stammt aus der CODESYS Basis und kann deshalb variieren. Enthält nur Informationen.

*Tab. 21: Editor der Safety Netzwerkvariablenliste (Empfänger) **

Registerkarte	Steuerelement
Bereich oberhalb der Registerkaten	alle Steuerelemente
„Safety Konfiguration“	alle Steuerelemente
„SPS Netzwerk“	alle Steuerelemente

*Tab. 22: Editor der Safety Netzwerkvariablenliste (Sender) **

Registerkarte	Steuerelement
Bereich oberhalb der Registerkaten	alle Steuerelemente
„Safety Konfiguration“	alle Steuerelemente
„SPS Netzwerk“	alle Steuerelemente

Ansichten (Menü 'Ansicht')

Tab. 23: Ansicht 'Geräte'

Stammt aus der CODESYS Basis und kann variieren.

Tab. 24: Ansicht 'POUs'

Stammt aus der CODESYS Basis und kann variieren.

Tab. 25: Ansicht 'Meldungen'

Steuerelemente	■ ☒ ■ Auswahlliste mit Meldungskategorien ■ Meldungstypen: – : Fehler – : Warnung – : Information

Tab. 26: Ansicht 'Safety Querverweisliste'

alle Steuerelemente

Tab. 27: Ansicht 'Eigenschaften'

Registerkarten
„Allgemein“: Eingabefeld zur Änderung des Objektbezeichners
„Zugriffskontrolle“: alle Steuerelemente
„Safety“: alle Steuerelemente

Tab. 28: Ansicht 'Überwachungsliste'

„Überwachungsliste 1“
„Überwachungsliste 2“
„Überwachungsliste 3“
„Überwachungsliste 4“
„Alle Forces anzeigen“

Liste zulässiger oder geänderter Funktionen

Zulässige Ansichten

Zulässige Werkzeuge

Zulässige FUP-Werkzeuge (FUP, Ansicht *„Werkzeuge“*)	
„Generelles“	*„Zuweisung“* *„Eingang“* *„Sprung“* *„Return“*
„Boolesche Operatoren“	*„AND (2 Eingänge)“* *„AND (3 Eingänge)“* *„OR (2 Eingänge)“* *„OR (3 Eingänge)“* *„XOR“* *„NOT“*
„Mathematische Operatoren“	*„ADD (2 Eingänge)“* *„ADD (3 Eingänge)“* *„SUB“* *„MUL“* *„DIV“* *„EQ“* *„NE“* *„LT“* *„LE“* *„GT“* *„GE“*
„Sonstige Operatoren“	*„SEL“* *„MUX“*
Safety Funktionsbausteine	alle
Safety *„Standard“* FBs	alle

Dialoge des Menüs 'Tools'

Tab. 29: Dialog 'Optionen'

Alle Kategorien sind zulässig. **In folgenden Kategorien müssen bestimmte Optionen aktiviert sein:**	
Kategorie *„Geräteeditor“*	Die Option *„Klassische Darstellung der Kommunikationseinstellungen verwenden“* muss aktiviert sein!

Tab. 30: Dialog 'Anpassen'

alle Steuerelemente

16.3 Geänderte Standardfunktionen

Befehl 'Querverweise ausgeben'

Wenn der Befehl auf nonsafety-Objekte angewendet wird, zeigt der Befehl normales Verhalten, gemäß CODESYS : Querverweise werden in der Ansicht *„Querverweisliste“* aufgelistet. Wenn der Befehl auf Safety-Objekte angewendet wird, werden die Querverweise in der Ansicht *„Safety Querverweisliste“* aufgelistet.

Ansicht 'Querverweisliste'

In dieser Ansicht können keine Querverweise in Safety-Objekten gefunden werden.

Im Gegensatz dazu können in der Ansicht *„Safety Querverweisliste“* nur Querverweise in Safety-Objekten gefunden werden, keine aus Standard-Objekten. Informationen zur Funktionsweise finden Sie in der CODESYS Safety Onlinehilfe.

Befehl 'Dokumentieren'

Wenn das Projekt mindestens 1 Safety-Objekt enthält, wird eine bei Freigabe von CODESYS Safety fixierte Version des CODESYS Druck-Rahmenwerks verwendet. Es kann also zum Beispiel sein, dass neue Funktionen des Dokumentieren-Auswahldialogs aus der CODESYS Basis nun nicht mehr zur Verfügung stehen. Die Funktion entspricht dann nicht mehr der Beschreibung in der Online-Hilfe CODESYS, sondern nur noch der Beschreibung in der Online-Hilfe CODESYS Safety.

Befehle 'Archiv speichern/versenden'

Wenn das Projekt mindestens 1 Safety-Objekt enthält, wird eine bei Freigabe von CODESYS Safety fixierte Version des Archivieren-Auswahldialogs verwendet. Es kann also zum Beispiel sein, dass neue Funktionen des Archivieren-Auswahldialogs aus der CODESYS Basis nun nicht mehr zur Verfügung stehen. Die Funktion entspricht dann nicht mehr der Beschreibung in der Online-Hilfe CODESYS, sondern nur noch der Beschreibung in der Online-Hilfe CODESYS Safety.

Wenn bei dem Befehl *„Datei ➔ Projektarchiv ➔ Archiv speichern /versenden“* ein logisches Gerät der Sicherheitsapplikation ausgewählt ist, werden alle entsprechenden sicherheitsbezogenen Gerätebeschreibungen mit archiviert. Wenn bei der Safety-Variante der Bibliotheksverwalter mit ausgewählt ist, werden alle referenzierten Bibliotheken mit archiviert.

Liste zulässiger oder geänderter Funktionen

Geänderte Standardfunktionen

17 Literaturverzeichnis

[N1.1.3]	IEC 61131-3, Programmable controllers, Part 3: Programming languages, Edition 3 (2013-02)
[N1.2.1]	IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements, Edition 2 (2010-04)
[N1.3d]	DIN EN IEC 62061: Sicherheit von Maschinen – Funktionale Sicherheit sicherheitsbezogener elektrischer, elektronischer und programmierbarer elektronischer Steuerungssysteme (2005-10) mit Corrigendum:2010, Amendment 1:2013 und Amendment 2:2015
[N1.4.1d]	DIN EN ISO 13849-1: Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen, Teil 1 (2016-06)
[N2.1.1]	PLCopen, TC 5: Safety Software (Technical Specification), Part 1: Concepts and Function Blocks, Version 1.0 (2006-1)
[N3.1.2]	PROFIsafe – Profile for Safety Technology on PROFIBUS DP and PROFINET IO, Version 2.4, March 2007, Order No. 3.192b
[N3.1.5]	PROFIsafe – Profile for Safety Technology on PROFIBUS DP and PROFINET IO, Version 2.5, December 2012, Order No. 3.192b
[N4.3192b]	PROFIsafe – Profile for Safety Technology on PROFIBUS DB and PROFINET IO, Version 2.6, October 2013, Order No. 3.192b
[N3.5.4]	ETG: Safety over EtherCAT Protocol specification (ETG. 5100), Version 1.2.0, 11.03.2011
[N3.5.5]	ETG: Safety over EtherCAT Implementation Guide (ETG.5101), Version 1.1.1, 14.05.2010
[N61784.3.3]	IEC 61784-3-3: Industrial communication networks - Profiles - Part 3-3: Functional safety fieldbuses - Additional specifications for CPF 3 (2016)

Literaturverzeichnis

18 Glossar

Abnahme der Applikation

Ein Fall von Softwareabnahme. Teil der **Maschinenabnahme**. Allgemein: Die Abnahme einer **Applikation**, d. h. der ablauffähigen Einheit auf der Steuerung. Die Geräteparameter gehören nicht dazu (obwohl sie auch Software sind). Die Abnahme der Applikation kann aber abhängig sein von der Abnahme der Geräteparameter der benützten E/A-Geräte (**Geräteparameterabnahme**), bzw. organisatorisch zusammen mit ihr erfolgen. Die Abnahme der Applikation (im Gegensatz zur Abnahme der Geräteparameter) ist typischerweise abhängig vom Typ der Steuerung und von der Version der Firmware auf der Steuerung.

Anwender

(von CODESYS Safety): Entwickler, der Sicherheitsapplikationen bearbeitet und/oder auf Sicherheitssteuerungen ausführt.

Anwenderprogramm-Fehlerreaktion

Vom **Anwender** in der **Applikation** programmierte Fehlerreaktion.

API

Protokollunabhängige Applikationsschnittstelle

Mithilfe API erhält der Entwickler auf der Programmierebene Zugang für jedes sichere E/A-Modul auf Ebene der Safety-Kommunikation

Applikation

Ablauffähige Einheit auf einer Steuerung aus Programmcode und zugehöriger E/A-Konfiguration

Applikationsstatus des LZS

Zustände, in denen sich eine Applikation auf dem LZS befinden kann, z. B. Applikation läuft, Applikation steht usw. Vgl. **Betriebsmodus**

Ausführungsversion

Kennung mittels welcher die kontrollierte Kompatibilität zwischen (abgenommener) Bootapplikation und LZS-Kern geregelt wird. Kompatibilität mit externen POUs, siehe **Implementierungsversion**

Aus Anwendersicht entspricht die Ausführungsversion der Compilerversion aus CODESYS.

Eine geänderte Versionsnummer bedeutet, dass sich an der Abarbeitung der Applikation etwas ändern kann.

Austauschvariablen

Variablen, deren Werte zwischen einer Sicherheitsapplikation (auf einer Sicherheitssteuerung) und einer oder mehreren Standardapplikationen (auf verbundenen Standardsteuerungen) ausgetauscht werden

Basic-Level

Gemäß PLCopen ein **Programmierlevel** mit einfachem, definiert eingeschränktem Sprachumfang. Der Level soll dem Anwender entsprechen, der bislang die Verdrahtung von sicherheitsgerichteten Modulen vorgenommen hat.

Programmierlevel, der auf die Verwendung zertifizierter (oder validierter) Funktionsbausteine in Sicherheitsapplikation ausgerichtet ist

Bedienschnittstelle (User Interface)

Schnittstelle zwischen Mensch und Software. Insbesondere die grafische Bedienoberfläche des **Programmiersystems** zum **Anwender**. Zu unterscheiden von **Programmierschnittstelle**.

Glossar

Bestätigte Verbindung	Für jede Onlinefunktionalität vor Ort zwischen CODESYS Safety und einer Sicherheitssteuerung ist eine bestätigte Verbindung erforderlich. Mit der Verbindungsbestätigung bestätigt der Anwender, dass die Netzwerkverbindung ihn mit der richtigen Steuerung verbunden hat. Eine bestätigte Verbindung gibt es im sicheren Betrieb und im Debug-Betrieb.
Betriebsmodus des sicherheitsgerichteten LZS	Zustand, in denen sich das sicherheitsgerichtete LZS befinden kann, welche für den Anwender wichtig sind, z. B. Debug, System-Fehlerreaktion usw. Je nach Modus sind unterschiedliche Aktionen aktiviert bzw. deaktiviert. Vgl. **Applikationsstatus**
Bibliotheksrepository	Datenablage von CODESYS, in der Bibliotheken versioniert abgelegt werden
Bibliotheksverwalter	Objekt zur Verwaltung der eingebundenen Bibliotheken in CODESYS
Bootapplikation	Eine Bootapplikation wird im Onlinebetrieb aus der Downloadapplikation erzeugt und auf der Sicherheitssteuerung abgelegt. Sie bleibt nach dem Ausloggen auf der Steuerung und startet nach dem Neustarten der Sicherheitssteuerung.
CODESYS	(Controller Development System) Programmiersystem der Firma 3S-Smart Software Solutions, zur Programmierung von **SPS**en. Liegt derzeit in der zweiten Generation vor und trägt die Versionsbezeichnung V3.x
CODESYS Control Safety	Produktname für das sicherheitsgerichtete LZS von 3S-Smart Software Solutions
CODESYS Safety	Erweiterung von CODESYS V3.x um Sicherheitsfunktionalität. Technisch gesehen ist dies eine CODESYS-Instanz mit geladenem Safety-Profil, das die Safety-Komponenten lädt.
CRC (Cyclic Redundancy Check)	engl. Begriff für **Prüfsumme**
Datentypen	**Typen**
Debug-Betrieb (unsicherer Betrieb)	Zustand, in dem entweder statt der Bootapplikation eine Downloadapplikation läuft oder in dem der Lauf der Bootapplikation durch Debug-Befehle beeinflusst wurde bzw. beeinflusst werden kann.
Differenzabnahme	Erleichterte Abnahme einer Applikation, die eine Änderung einer bereits verifizierten Applikation ist
Differenzverifikation	Erleichterte Verifikation einer Applikation, die eine Änderung einer bereits verifizierten Applikation ist
Downloadapplikation	Aktuelle, fehlerfrei übersetzte Sicherheitsapplikation, die beim Einloggen auf die Steuerung geladen wird. Die Downloadapplikation ist nach dem Ausloggen nicht mehr auf der Steuerung.
DP-...	Decentralized Peripherals ... Im Zusammenhang mit PROFIBUS zu sehen und dort als Abgrenzung zu anderen PROFIBUS-Varianten gemeint.
DP-Master (Decentralized Peripherals-Master)	Master auf PROFIBUS-DP (pollt zyklisch die angeschlossenen Geräte)

DP-Slave (Decentralized Peripherals-Slave)	Teilnehmer auf PROFIBUS-DP (wird zyklisch vom DP-Master gepollt); Koppler am PROFIBUS-DP
E/A...	die Ein-/Ausgabe betreffend (englisch I/O für Input/Output)
E/A-Konfiguration	**Konfiguration**
E/A-Modul	Die Einheit oder Einheiten eines Feldgeräts, welche einem unabhängigen Abschnitt (PDO) in seinem Prozessabbild entsprechen. Ein E/A-Modul liefert ein **PDO** mit Eingangssignalen und/oder empfängt ein PDO mit Ausgangssignalen. Es wird unterschieden zwischen **-** Hardware-Modul (als Eingangs- bzw. Ausgangsbaugruppe, Klemme oder Scheibchen bezeichnet) hinter einem modularen Feldgerät (auch als Koppler bezeichnet) mit Eingangskanälen und/oder Ausgangskanälen **-** E/A-Modul, das selbst Feldbusteilnehmer ist, z. B. der typische Antrieb. In diesem Fall bildet das **PDO** des E/A-Moduls das gesamte Prozessabbild des Feldbusteilnehmers. **-** Softwaredefiniertes Modul einer SPS, die Feldgerät ist
E/A-Parameter	**Konfigurationsdaten**
Editor	Komponente des PS zum Bearbeiten von Programmteilen
Extended-Level	Gemäß PLCopen ein **Programmierlevel** mit Sprachumfang zwischen **Basic-Level** und System-Level
Externe POU	Im Laufzeitsystem implementierte Bausteine
F-	Präfix, verwendet mit PROFIsafe-Begriffen Kurz für failsafe (dt. sicherheitsgerichtet)
failsafe	Werte von Variablen, die die Sicherheit der Anlange gewährleisten. Failsafe-Wert für Variablen des Datentyps SAFEBOOL muss FALSE sein. Der Anwender muss sicherstellen, dass alle Variablen vom Datentyp SAFEBOOL die Sicherheit der Anlage gewährleisten. Alle Variablen vom Datentyp SAFEBOOL müssen sowohl für die Initialisierung als auch für eine Sicherheitsanforderung auf den Wert FALSE gesetzt werden. Sicherheitsgerichtete Systeme basieren auf "negativer" Logik. Zum Beispiel ist ein physikalischer Not-Halt-Schalter normalerweise geschlossen, so dass Strom fließt. Wird der Schalter betätigt, öffnet der Kontakt und der Stromfluss wird gestoppt ("Ruhestrom-Prinzip").
FB	Funktionsbaustein, Funktionsblock
FBD	Function Block Diagram, **Funktionsplan**
Fehlerreaktion	Reaktion des LZS, die die Anlage in den sicheren Zustand bringt. Unterschieden werden die **Anwenderprogramm-Fehlerreaktion** und die **System-Fehlerreaktion**.
Feldbusteilnehmer	Gerät, das ein Feldbusprotokoll (z. B. PROFIBUS-DP oder PROFINET) implementiert, über welches das Prozessabbild ausgetauscht wird. Relevant sind insbesondere **-** modulare Feldgeräte, das heißt Feldgeräte mit untergeordneten **E/A-Modulen** (auch Koppler genannt)

	- Antriebe, d. h. **E/A Module**, die typischerweise selbständige Feldbusteilnehmer sind. **-** SPS als Feldgerät, d. h. die SPS ist ein modulares Gerät, das die Slaveseite des PROFINET-Protokolls implementiert
Feldgeräte	**Feldbusteilnehmer**
Fernzugang	Der Fernzugang dient der Diagnose der Steuerung und ist nur im sicheren Betrieb möglich. Folgende Funktionalitäten stehen während des Fernzugangs zur Verfügung: ■ Logbuch anzeigen und abspeichern ■ Anzeige der SPS-Info und Firmware-Info ■ Einloggen bei Gleichheit zum Projekt (kein Download) ■ Monitoren
F-Host	Sicherheitssteuerung, welche die Masterseite des PROFIsafe-Protokolls implementiert
Firmware	Systemspezifische Software für den Betrieb des Laufzeitsystems (z.B. Betriebssystem, HW-spezifische Anpassungen) plus das Laufzeitsystem selbst. Auf der Sicherheitssteuerung dementsprechend das **sicherheitsgerichtete LZS**.
F-Modul	Sicheres E/A-Modul eines PROFINET oder PROFIBUS-Feldgeräts, welches die Slaveseite des PROFIsafe Protokolls implementiert
F-Parameter	Sicherheitsgerichtete **Konfigurationsdaten** bei PROFIsafe, auch als **Safety-Protokollparameter** bezeichnet
FSoE	FailSafe over EtherCAT
Funktionale Applikation	Steuert die operativen Funktionen der Maschine, läuft auf der Standardsteuerung
Funktionsbaustein	**POU** nach **IEC 61131-3**, kann andere Funktionsbausteine aufrufen und von einer Applikation und anderen Funktionsbausteinen aufgerufen werden.
Funktionsplan	Ein/e in der IEC 61131-Funktionsbausteinsprache (FBS) geschriebene/s Programm/Funktion. Die Sprache ist zulässig für die Sicherheitsprogrammierung in **Basic-Level**, **Extended- Level** und **System-Level**.
FUP	**Funktionsplan**
Gerätebeschreibungsdatei	Beschreibung eines oder mehrerer Geräte in bussystemspezifischen Format (ESI für EtherCAT, GSD für PROFIbus, GSDML für PROFINET).
Geräteparameterabnahme	Ein Fall von Softwareabnahme. Teil der **Maschinenabnahme**. Die Abnahme der **Safety-Geräteparameter** eines Feldgeräts. Die Abnahme der Geräteparameter aller verwendeter Feldgeräte kann Voraussetzung für die Abnahme der Sicherheitsapplikation sein (**Applikationsabnahme**). Die Abnahme der Geräteparameter ist typischerweise abhängig vom Gerätetyp.
GVL	Globale Variablenliste. Eine (benannte) Liste von globalen Variablen innerhalb einer Applikation. Es kann mehrere GVL innerhalb einer Applikation geben.

HW	Hardware
IEC 61131-3	Internationale Norm für Speicherprogrammierbare Steuerungen
IEC-Code	Programmquelltext, konform zu IEC 61131-3, also AWL, KOP, FUP, AS oder ST
Implementierungsversion	Kennung einer externen POU, mittels welcher die kontrollierte Kompatibilität zwischen (abgenommener) Bootapplikation und der Implementierung der externen POU geregelt wird. Kompatibilität mit LZS-Kern ist die **Ausführungsversion**.
i-Parameter	Bezeichnung der **Safety-Geräteparameter** in PROFIsafe
Kanalvariablen	Repräsentieren einen Eingangskanal (Eingangskanal-Variable) oder einen Ausgangskanal (Ausgangskanal-Variable) eines bestimmten E/A-Moduls
Klemme	Steckbares E/A-Modul eines modularen Feldgeräts zum Anschluss von Sensoren bzw. Aktoren
Komponente	**SW-Komponente**
Konfiguration	Beschreibung des Systems aus Steuerungen und E/A-Modulen (sowohl Standard als auch sicherheitsgerichtet); Festlegung der Projektstruktur
Laufzeitsystem	SW-Komponente der **SPS**. Das LZS ist als Teil der **Firmware** verantwortlich für die Implementierung der Steuerungslogik.
LZS	**Laufzeitsystem**
Maschinenabnahme	Die Abnahme einer Maschine oder Anlage bestehend aus Hardware und Software. Aufbauend auf der Abnahme der **Sicherheitsapplikation** und der **Safety-Geräteparameter** sind dabei die Verdrahtung, die Standardapplikation (z. B. Reset) und die Gesamt-Reaktionszeiten der Sicherheitsfunktionen zu beurteilen. Vgl. **Applikationsabnahme**
Master	Allgemein eine HW, die die (steuernde) Kontrolle über andere HW (Slave) hat. Hier im Zusammenhang mit Feldbussystemen die Ansteuerung des Busses.
Netzwerkvariablen	Variablen, deren Werte zwischen 2 oder mehreren Sicherheitssteuerungen ausgetauscht werden
NVL	Netzwerkvariablenliste. Der Austausch von Netzwerkvariablen zwischen Sicherheitssteuerungen erfolgt über die Objekte Netzwerkvariablenliste (Sender) und Netzwerkvariablenliste (Empfänger).
NVMem	(Non Volatile Memory) Nicht-flüchtiger Speicher beliebigen Typs, z.B. FLASH, NVRAM, EEPROM, MemoryCard, Festplatte, usw.
Objekt	Ein Element der Projektstruktur, welches eine Strukturierungseinheit des Projekts repräsentiert.

Glossar

OEM	(Original Equipment Manufacturer) Ein Hersteller, der sein Gerät zusammen mit Software von 3S-Smart Software Solutions als Gesamtsystem anbietet. Im Gegensatz zu einem Endkunden, der Software von 3S-Smart Software Solutions direkt nutzt, verkauft ein OEM die Software lediglich weiter.
Offline	Im Offlinebetrieb ist das Programmiersystem nicht mit der Steuerung verbunden.
Online	Im Onlinebetrieb ist das Programmiersystem mit der Sicherheitssteuerung verbunden. Dies bedeutet, dass entweder eine Verbindung zur aktuellen, auf der Sicherheitssteuerung geladenen Applikation oder zur Bootapplikation auf der Sicherheitssteuerung besteht. Im Onlinebetrieb können Debug-Befehle ausgeführt werden.
Parametrierung	Konfiguration der Parameter des Bussystems und der E/A-Module
PDO	Process Data Object Zyklisch ausgetauschte Daten eines **E/A-Moduls** (Gerätemoduls)
Pinnen einer Applikation	(Begriff aus Visual SourceSafe, deutsch „festhalten“) Der aktuelle Stand der Applikation wird festgehalten: Änderung an der Applikation ist zwar noch möglich – so dass der aktuelle Stand im Projekt vom gepinnten Stand abweichen kann – aber die abweichenden Objekte sowie der Ausgangsstand (Pin) für die Abweichung können identifiziert werden. Pinnen ist kombiniert mit der Vergabe einer Bezeichnung für diesen Stand.
PLC	Programmable Logic Controller (**SPS**)
PLCopen	Die PLCopen ist ein firmen- und produktunabhängiger weltweiter Interessenverband zur Verbreitung der Norm **IEC 61131-3**.
PLCopen TC5	Technisches Komitee TC5 der PLCopen, das Standards für die Programmierung von sicherheitsgerichteten Steuerungen definiert
POU	**Programmorganisationseinheit** (deutsch: POE)
PROFIBUS-DP (Process Field Bus mit Decentralized Peripherals)	Feldbussystem zum Anschluss von E/A-Modulen (entweder direkt als PROFIBUS-Teilnehmer oder als Klemme eines modularen PROFIBUS-Teilnehmers)
PROFIsafe	Protokoll zur sicherheitsgerichteten Kommunikation auf PROFIBUS DP und PROFINET
Programm	POU nach IEC 61131-3
Programmierlevel	Es gibt drei verschiedene Programmierlevel für unterschiedliche Sicherheitseinschränkungen während der Entwicklung: **Basic-Level**, **Extended-Level** und **System-Level**. Basic- und Extended-Level sind von der PLCopen bzw. den entsprechenden Spezifikationen von CODESYS Safety definiert.

Programmierschnittstelle
Programmierschnittstelle (API) – Schnittstelle zwischen Softwarebausteinen: Ein Baustein exportiert eine Schnittstelle; Bausteine, die diesen benützen sollen, werden gegen diese Schnittstelle programmiert. Im Fall eines **Funktionsbaustein** der IEC 61131-3 besteht die exportierte Programmierschnittstelle aus seinen Ein- und Ausgabebariablen. Zu unterscheiden von **Bedienschnittstelle**

Programmiersystem
Entwicklungsumgebung auf einem Windows-PC für das Erstellen von Programmen nach **IEC 61131-3** sowie SW-technischer Zugang zu Steuerungen mit CODESYS-LZS

Programmorganisationseinheit
Objekt in der Projektstruktur, welches Software repräsentiert (und nicht Geräte/Module) POU

Projekt
Ein Projekt kann mehrere Steuerungen mit ihren untergeordneten Objekten beinhalten. Dies können Sicherheits- und/oder Standardsteuerungen sein, die eine Anlage oder Maschine steuern. Wird in CODESYS V3.x in der **Projektdatei** gespeichert.

Projektdatei
Datei mit den Daten des Projekts, aber ohne Bibliotheken und Gerätebeschreibungen (xxx.project)

Prozesswerte
Werte, die aus dem gesteuerten Prozess heraus bestimmt wurden, zum Beispiel durch Sensoren (Eingangswerte), oder Werte, die auf den gesteuerten Prozess wirken, zum Beispiel durch Aktoren (Ausgangswerte).

Prüfsumme
Eine Prüfsumme über einen Datenstrom, die durch eine bestimmet Methode bestimmt wird. Die Methode ist parametriert durch einen Startwert und ein Polynom. Prüfsummen dienen der Absicherung übertragener oder gespeicherter Daten.

PS
Programmiersystem

Quellcodeverwaltung
Externes Programm zur Verwaltung von Objektdaten außerhalb von CODESYS. Diese können neben den eigentlichen, für den Anwender sichtbaren Quelldaten auch Daten zur Verwaltung dieser Quellen innerhalb von CODESYS enthalten. Synonym mit dem Begriff Versionsverwaltung verwendet.

Rückwirkungsfreiheit
Im Kontext der CODESYS Safety-Dokumentation die Fähigkeit, mögliche Rückwirkungen von sicherheitsgerichteten SW-Funktionen zu erkennen und zu beherrschen

Safety-Adresse
(des E/A-Moduls), die über Parametrierung eingestellte Adressierung des E/A-Moduls innerhalb seines Feldbusprotokolls (im Fall von PROFIsafe: die "F-Adresse" `F_Dest_Add`).

Safety-Gerätebeschreibung
Beschreibung von Parametersätzen, mit denen ein Gerät konfiguriert werden kann. Im Rahmen von CODESYS Safety kommen vor: **Safety-Protokollparameter** und **Safety-Geräteparameter**.

Safety-Geräteparameter
Bearbeiteter Parametersatz, der nicht in die Sicherheitssteuerung, sondern nur ins Gerät geladen wird. Im PROFIsafe-Kontext als **I-Parameter** bekannt.

Glossar

Safety-Konfigurationsdaten	Zusammenfassender Begriff für diejenigen Konfigurationsdaten eines E/A-Moduls, die für die Sicherheitsapplikation bzw. Sicherheitssteuerung relevant sind. Die Safety-Geräteparameter werden nicht als Safety-Konfigurationsdaten bezeichnet.
Safety-PDO	Gemäß sicherem E/A-Protokoll abgesicherte **PDO** eines sicheren **E/A-Moduls** Die Kodierung der Werte der E/A-Kanäle eines einzelnen, sicheren E/A-Moduls zusammen mit Zusatzinformation als Abschnitt im Prozessabbild eines Standard-Feldbusteilnehmers (z. B. ein PROFIBUS-DP oder PROFINET-Gerät)
Safety-Protokollparameter	Parameter eines **sicheren E/A-Protokolls**
SAFExxx	Familie von Datentypen mit Präfix SAFE gemäß PLCopen, z. B. SAFEBOOL, SAFEINT
Sicherer Betrieb	Zustand, in dem die Bootapplikation ungestört läuft
Sicheres E/A-Modul	Ein **E/A-Modul**, das ein sicheres E/A-Protokoll implementiert und in der Sicherheitsprogrammierung verwendet werden kann. In einer Sicherheitsapplikation sind die Nutzdaten der E/A-Kanäle von sicheren E/A-Modulen von einem der Datentypen **SAFExxx**.
Sicheres E/A-Protokoll	Ein Protokoll zur Verwendung von **Safety-PDO**s über Standardfeldbusse, sodass eine eindeutige Identifikation des E/A-Moduls möglich ist (z.B. PROFIsafe V1.3)
Sicherheitsapplikation	Sicherheitsgerichtete **Applikation** zur Ausführung auf einer **sicherheitsgerichteten** Steuerung
Sicherheitsgeräte	Sicherheitsgerichtete Geräte
Sicherheitsgerichtet	(en: fail-safe), Fähigkeit eines Systems, beim Auftreten eines Ausfalls im sicheren Zustand zu bleiben bzw. unmittelbar in den sicheren Zustand überzugehen.
Sicherheitsgerichtete Firmware	Firmware für den Betrieb eines sicherheitsgerichteten Laufzeitsystems. Besteht aus CODESYS Control Safety und betriebssystem-ähnlichen Basisfunktionen
Sicherheitsgerichtetes Objekt	Objekt im **Sicherheits-PS**, das eine Sicherheitssteuerung, ein **sicheres E/A-Modul** oder eine Einheit einer Sicherheitsapplikation repräsentiert (die beiden Erstgenannten sind **sicherheitsgerichtete Geräte**, letztere sind sicherheitsgerichtete Logikobjekte).
Sicherheitsgerichtetes Programmiersystem	Programmiersystem auf einem Arbeitsplatzrechner (Windows-PC) für die Sicherheitsprogrammierung
Sicherheits-PS, Sicherheitsprogrammierung	Gesamtheit der Aktivitäten zur Erstellung von Sicherheitsapplikationen; Programmierung von **sicherheitsgerichteten** Steuerungen
Sicherheitssteuerung	Eine **SPS**, die (üblicherweise) durch spezielle Hardware-Erweiterungen (z.B. zwei Prozessoren mit gegenseitiger Prozessbeobachtung) und entsprechend angepasster Software eine erhöhte Fehlersicherheit bietet.
Slave	HW, die einem Master untergeordnet ist und von diesem gesteuert wird bzw. diesen mit Daten beliefert

Speicherprogrammierbare Steuerung	(en: PLC) bezeichnet ein Gerät, das mittels spezieller Programmiersoftware zur automatisierten Steuerung von industriellen Maschinen und Anlagen verwendet wird
SPS	**Speicherprogrammierbare Steuerung**
S-SPS	Sicherheitsgerichtete **SPS**
Standard-...	Präfix: **-** als Gegensatz zu **sicherheitsgerichtet** **-** als Gegensatz zu safety-spezifisch, z. B. Standard-SW, Standardkomponente, Standardansicht
Standard CODESYS	CODESYS ohne Zusätze, wie es ausgeliefert wird. Technisch gesehen eine CODESYS-Instanz, in deren Profil insbesondere die Safety-Erweiterung NICHT geladen wird
Standardsteuerung	Steuert die operativen Funktionen der Maschine, enthält die funktionale Applikation und ist der Feldbusmaster für den unterlagerten Feldbus
Steuerung	SPS
SW	Software
System-Fehlerreaktion	Reaktion des Sicherheitslaufzeitsystems auf einen Systemfehler
Task	Zeitliche Ablaufeinheit einer Software **-** IEC-Task: In der IEC-Programmierung bezeichnet "Task" eine zeitliche Ablaufeinheit einer IEC-Applikation. In einer **sicherheitsgerichteten** Applikation kann nur eine Task konfiguriert werden. **-** LZS-Task: Das Laufzeitsystem enthält Tasks zur Ausführung der konfigurierten IEC-Task(s) und zur Ausführung anderer Aufgaben. Alle im Laufzeitsystem enthaltenen Tasks werden vom OEM bereitgestellt, da das Laufzeitsystem selbst keine Taskverwaltung beinhaltet. Das Laufzeitsystem stellt die Einsprungspunkte für die benötigten Tasks bereit, führt die vorgesehenen Aufgaben im Kontext der jeweiligen Task aus und kehrt danach zurück.
Taskkonfiguration	In der Taskkonfiguration zu einer sicherheitsgerichteten Steuerung werden vom Anwender die Zykluszeit des Tasks und die Reihenfolge der im Task abzuarbeitenden Anwendungen festgelegt.
Treiberinstanz	Instanz des Protokollstacks eines sicheren Feldbusses (PROFIsafe, FSoE) für die Kommunikation mit einem Gerät dieses Feldbusses. Technisch gesehen handelt es sich um eine vom logischen Gerät implizit erzeugte Instanz des Funktionsbausteins, der den Protokollstack implementiert. **-** E/A-Stack-Variablen sind Instanzen von E/A-Stack-FBs und implementieren das Sicherheitsprotokoll und die API zu einem E/A-Modul. Die Ein-/Ausgangsvariablen der Instanz dienen dem Zugriff auf den Status des E/A-Moduls, nicht zum Zugriff auf seine E/A-Kanäle.

Typen

In der Sicherheitsprogrammierung werden folgende Unterarten unterschieden:

- FB-Typen: Namen von Funktionsbausteinen als Typen von Variablen (FB-Instanzen)

- Boolesche Typen: BOOL, SAFEBOOL (für Wahrheits- und Bitwerte)

- Numerische Typen: INT, DINT, SAFEINT, SAFEDINT (für Faktoren und Divisoren)

- Bitaccess Typen: WORD, BYTE, DWORD, SAFEWORD, SAFEBYTE, SAFEDWORD

- Arithmetische Typen: INT, DINT, SAFEINT, SAFEDINT, TIME, SAFETIME (für Operanden arithmetischer Operationen)

Typkonsistenz

Eigenschaft eines Konstrukts im IEC-Code, dass die Typen der in ihm vorkommenden Variablen oder Werte zusammen passen

1. Das Konstrukt VAR_EXTERNAL-Deklaration ist typkonsistent, wenn es der Variablen den gleichen gibt wie die VAR_GLOBAL-Deklaration dieser Variablen.

2. Das Konstrukt Initialisierung einer Variable mit einem Wert oder Zuweisung eines Wertes an eine Variable ist typkonsistent,

a. wenn Wert und Variable den gleichen Typ haben, oder

b. wenn der Wert einen Typ SAFE-X hat und die Variable den Typ X, oder

c. wenn der Wert den Typ INT hat und die Variable den Typ DINT, oder

d. wenn der Wert den Typ SAFEINT hat und die Variable den Typ SAFEDINT oder DINT.

3. Das Konstrukt FB-Aufruf (Box mit FB-Namen) ist typkonsistent, wenn der Name des FBs gleich dem Typ der darüberstehenden FB-Instanz ist und wenn die Zuweisungen der Eingangswerte auf Eingangsvariablen typkonsistent sind.

4. Die Typkonsistenz des Konstrukts Operatoraufruf (Box mit Operatornamen) hängt vom Operator ab: Aufrufe einer Konvertierung (X_TO_Y) sind typkonsistent, wenn der Eingangswert den Typen X oder SAFE-X hat.

Aufrufe boolescher Operatoren (AND, OR, XOR, NOT) sind typkonsistent, wenn die Eingangswerte die Typen BOOL oder SAFEBOOL haben.

Aufrufe mathematischer Operatoren (ADD, SUB, MUL, DIV, LT, LE, GT, GE) sind typkonsistent, wenn die Eingangswerte die Typen INT, DINT, SAFEINT oder SAFEDINT haben; darüber hinaus ist ein Aufruf von MUL und DIV auch dann typkonsistent, wenn der erste Eingangswert den Typ TIME oder SAFETIME hat und nur die anderen Eingangswerte den Typ INT, DINT, SAFEINT oder SAFEDINT haben.

Aufrufe der allgemeinen Vergleichsoperatoren (EQ, NE) sind typkonsistent, wenn die Eingangswerte untereinander typkonsistent sind (siehe unten). Aufrufe des Auswahloperators SEL sind typkonsistent, wenn der erste Eingangswerte den Typ BOOL oder SAFEBOOL hat, und die anderen Eingangswerte untereinander typkonsistent sind.

Aufrufe des Auswahloperators MUX sind typkonsistent, wenn der erste Eingangswerte den Typ INT, DINT, SAFEINT oder SAFEDINT hat und die anderen Eingangswerte untereinander typkonsistent sind. Mehrere Eingangswerte sind untereinander typkonsistent,

a. wenn sie den gleichen Typ haben, oder

b. wenn sie teils den Typ T, teils den Typ SAFE-T haben, oder

c. wenn sie die Typen INT, DINT, SAFEINT oder SAFEDINT haben.

5. Die Konstrukte bedingter Sprung und bedingtes Return sind typkonsistent, wenn der Eingangswert BOOL oder SAFEBOOL ist.

Übergeordneter Feldbus

Die CODESYS-Standardsteuerung mit der Sicherheitssteuerung stellen als ein PROFIsafe F-Device ein I/O dieses Feldbusses dar und kommunizieren über diesen übergeordneten Feldbus mit einer übergeordneten Steuerung.

Übergeordnete Steuerung

Eine Steuerung, die über einen übergeordneten Feldbus mit der CODESYS Standardsteuerung kommuniziert.

Validierung

Überprüfung, ob das fertig erzeugte, verifizierte Produkt den gestellten Anforderungen genügt

Verifikation

Überprüfung, ob die erzeugten Dokumente und Softwarefunktionen, -Bausteine und -Komponenten ihren Spezifikationen (bzw. ihren übergeordneten Spezifikationen) genügen

Zweiterzeugung einer Bootapplikation

Eine Applikation im Projekt, aus der schon einmal eine Bootapplikation auf einer SPS erzeugt wurde (insbesondere für Tests), wird unverändert auf einer weiteren SPS zur Bootapplikation gemacht.

Glossar

19 Index

1, 2, 3 ...

A

B

C

D

E

F

G

Index

T

U

V

W

Z

Index

Anhang

A Zertifikat 2017

Certificate

Functional Safety

TÜVRheinland CERTIFIED

www.tuv.com
ID 0600000000

No.: 968/EZ 568.10/17

Product tested	Run-time System Development-Kit, Programming System incl. function block libraries and fieldbus configuration	**Certificate holder**	3S-Smart Software Solutions GmbH Memminger Str. 151 87435 Kempten Germany
Type designation	CODESYS Safety See Revision List		
Codes and standards	IEC 61508 Parts 1-7:2010 EN ISO 13849-1:2015		EN 62061:2005 + AC:2010 + A1:2013 + A2:2015
Intended application	CODESYS Safety Run-time System complies with the applicable requirements of the relevant standards (SC3 acc. to IEC 61508, PL e acc. to EN ISO 13849-1) and can be used in applications up to SIL 3 acc. to IEC 61508, EN 62061 and PL e acc. to EN ISO 13849-1. PLCopen Function blocks are developed in accordance to the PLCopen Technical Specification Part 1, V1.0, IEC 61508, SC 3 and can be used in applications up to SIL 3 acc. to IEC 61508, PL e acc. to EN ISO 13849-1. The CODESYS Programming System complies with the applicable requirements for off-line support tools of class T3 according to IEC 61508-3.		
Specific requirements	For the use of CODESYS Safety the operating conditions and functional characteristics as specified in the user manual and accompanying implementation documents by the manufacturer needs to be observed. The current versions of software are specified in the currently valid version release list. The list is released by the manufacturer in cooperation with the Test Institute.		

Valid until 2022-08-28

The issue of this certificate is based upon an examination, whose results are documented in Report No. 968/EZ 568.10/17 dated 2017-08-28.
This certificate is valid only for products which are identical with the product tested. It becomes invalid at any change of the codes and standards forming the basis of testing for the intended application.

TÜV Rheinland Industrie Service GmbH
Bereich Automation
Funktionale Sicherheit
Am Grauen Stein, 51105 Köln

Köln, 2017-08-28

Certification Body Safety & Security for Automation & Grid

Dipl.-Ing. Thomas Steffens

www.fs-products.com
www.tuv.com

DAkkS Deutsche Akkreditierungsstelle D-ZE-11061-02-00

TÜVRheinland®
Precisely Right.

TÜV Rheinland Industrie Service GmbH, Am Grauen Stein, 51105 Köln / Germany
Tel.: +49 221 806-1790, Fax: +49 221 806-1539, E-Mail: industrie-service@de.tuv.com

B IEC 61131-3 Compliance

B.1 Compliance Liste: Unterstützte Features

Die IEC Compliance Statement muss auflisten, welche in den Tabellen von [N1.1.3] genannten Features unterstützt werden [N1.1.3-Kap. 5.1 a].

IEC 61131-3 3rd Edition "PLC Programming Languages"

Implementer: 3S-Smart Software Solutions GmbH
Memminger Str. 151 | 87439 Kempten
Product: CODESYS Safety 1.2.0
Date: 2016-11-28

This Product complies with the requirements of the standard for the following language features:

Feature No.	Table Number and Title / Feature Description			Compliantly implemented (✓)				Implementer's note
				LD	FBD	ST	IL	
	Table 1 – Character sets							
1	"ISO-10646				✓			comments: UCS identifiers: ASCII
2a	Lower case characters a:	a, b, c, ...			✓			
2b	Number sign:	#	See Table 5		✓			
2c	Dollar sign:	$	See Table 6		–			PLCopen Safety 1.0: no CHAR, no STRING
	Table 2 – Identifiers							
1	Upper case letters and numbers:		`IW215`		✓			
2	Upper and lower case letters, numbers, embedded underscores				✓			
3	Upper and lower case, numbers, leading or embedded underscores				✓			only in system programming
	Table 3 – Comments							
1	Single-line comment		//...		✓			network label
2a	Multi-line comment		(* ... *)		✓			network comment, out-commented network
2b	Multi-line comment		/* ... */					
3a	Nested comment		(* ..(* .. *) ..*)		–			
3b	Nested comment		/* .. /* .. */ .. */					
	Table 4 – Pragma							
1	Pragma with curly brackets		{ ... }		–			
	Table 5 – Numeric and bit string literals							
1	Integer literal:		`-12`		✓			
2	Real literal:		`-12.0`		–			
3	Real literals with exponent:		`-1.34E-12`		–			
4	Binary literal:		`2#1111_1111`		✓			
5	Octal literal:		`8#377`					
6	Hexadecimal literal:		`16#FF`		✓			
7	Boolean zero and one				–			
8	Boolean `FALSE` and `TRUE`				✓			
9	Typed literal:		`INT#-123`		✓			
	Table 6 – Character string literals							

	Single-byte characters or character strings with ‘ ‘					
1a	Empty string (length zero)		–			PLCopen Safety 1.0: no STRING
1b	String of length one or character CHAR containing a single character		–			PLCopen Safety 1.0: no CHAR
1c	String of length one or character CHAR containing the “space” character		–			PLCopen Safety 1.0: no CHAR
1d	String of length one or character CHAR containing the “single quote” character		–			PLCopen Safety 1.0: no CHAR
1e	String of length one or character CHAR containing the “double quote” character		–			PLCopen Safety 1.0: no CHAR
1f	Support of two character combinations of Table 7		–			PLCopen Safety 1.0: no CHAR
1g	Support of a character representation with ‘$’ and two hexadecimal characters		–			PLCopen Safety 1.0: no CHAR
	Double-byte characters or character strings with **""** (NOTE)					
2a	Empty string (length zero)		–			PLCopen Safety 1.0: no STRING
2b	String of length one or character WCHAR containing a single character		–			PLCopen Safety 1.0: no CHAR
2c	String of length one or character WCHAR containing the “space” character		–			PLCopen Safety 1.0: no CHAR
2d	String of length one or character WCHAR containing the “single quote” character		–			PLCopen Safety 1.0: no CHAR
2e	String of length one or character WCHAR containing the “double quote” character		–			PLCopen Safety 1.0: no CHAR
2f	Support of two character combinations of Table 7		–			PLCopen Safety 1.0: no CHAR
2g	Support of the character representation with ‘$’ and four hexadecimal characters		–			PLCopen Safety 1.0: no CHAR
	Single-byte typed characters or string literals with #					
3a	Typed string					
3b	Typed character					
3c	Typed character (using hexadecimal representation)					
	Double-byte typed string literals with # (NOTE)					
4a	Typed double-byte string (using “double quote” character)					
4b	Typed double-byte character (using “double quote” character)					
4c	Typed double-byte string (using “single quote” character)					
4d	Typed double-byte character (using “single quote” character)					
4e	Typed double-byte character (using hexadecimal representation)					
	Table 7 – Two-character combinations in character strings					
1	Dollar sign		–			PLCopen Safety 1.0: no STRING
2	Single quote		–			PLCopen Safety 1.0: no STRING
3	Line feed		–			PLCopen Safety 1.0: no STRING
4	Newline		–			PLCopen Safety 1.0: no STRING
5	Form feed (page)		–			PLCopen Safety 1.0: no STRING
6	Carriage return		–			PLCopen Safety 1.0: no STRING
7	Tabulator		–			PLCopen Safety 1.0: no STRING
8	Double quote		–			PLCopen Safety 1.0: no STRING
	Table 8 – Duration literals					

	Duration abbreviations		✓			
1a	d		✓			
1b	h		✓			
1c	m		✓			
1d	s		✓			
1e	ms		✓			
1f	us		–			PLCopen Safety 1.0: no LTIME
1g	ns		–			PLCopen Safety 1.0: no LTIME
	Duration literals without underscores					
2a	short prefix		✓			
2b	long prefix		–			
	Duration literals with underscores					
3a	short prefix		✓			
3b	long prefix		–			
	Table 9 – Date and Time of Day literals					
1a	Date literal (long prefix)		–			PLCopen Safety 1.0: no DATE
1b	Date literal (short prefix)		–			PLCopen Safety 1.0: no DATE
2a	Long date literal (long prefix)					PLCopen Safety 1.0: no DATE
2b	Long date literal (short prefix)					PLCopen Safety 1.0: no DATE
3a	Time of day literal (long prefix)		–			PLCopen Safety 1.0: no TOD
3b	Time of day literal (short prefix)		–			PLCopen Safety 1.0: no TOD
4a	Long time of day literal (short prefix)					PLCopen Safety 1.0: no TOD
4b	Long time of day literal (long prefix)					PLCopen Safety 1.0: no TOD
5a	Date and time literal (long prefix)		–			PLCopen Safety 1.0: no DT
5b	Date and time literal (short prefix)		–			PLCopen Safety 1.0: no DT
6a	Long date and time literal (long prefix)					PLCopen Safety 1.0: no DT
6b	Long date and time literal (short prefix)					PLCopen Safety 1.0: no DT
	Table 10 – Elementary data types					
1	Boolean		✓			
2	Short integer		–			PLCopen Safety 1.0: no SINT
3	Integer		✓			
4	Double integer		✓			
5	Long integer		–			PLCopen Safety 1.0: no LINT
6	Unsigned short integer		–			PLCopen Safety 1.0: no UxINT
7	Unsigned integer		–			PLCopen Safety 1.0: no UxINT
8	Unsigned double integer		–			PLCopen Safety 1.0: no UxINT
9	Unsigned long integer		–			PLCopen Safety 1.0: no UxINT
10	Real number		–			
11	Long real		–			
12a	Duration		✓			
12b	Duration		–			PLCopen Safety 1.0: no LTIME
13a	Date (only)		–			PLCopen Safety 1.0: no DATE
13b	Long Date (only)					PLCopen Safety 1.0: no DATE
14a	Time of day (only)		–			PLCopen Safety 1.0: no TOD
14b	Time of day (only)					PLCopen Safety 1.0: no TOD

15a	Date and time of Day		–			PLCopen Safety 1.0
15b	Date and time of Day					PLCopen Safety 1.0
16a	Variable-length single-byte character string		–			PLCopen Safety 1.0: no STRING
16b	Variable-length double-byte character string		–			PLCopen Safety 1.0: no STRING
17a	Single-byte character					PLCopen Safety 1.0: no CHAR
17b	Double-byte character					PLCopen Safety 1.0: no CHAR
18	Bit string of length 8		✓			PLCopen Safety 1.0: no BYTE, input / output types only
19	Bit string of length 16		✓			
20	Bit string of length 32		✓			PLCopen Safety 1.0: no DWORD, input / output types only
21	Bit string of length 64		–			PLCopen Safety 1.0: no LWORD
	Table 11 – Declaration of user-defined data types and initialization					
1a 1b	Enumerated data type		–			PLCopen Safety 1.0: no TYPE
2a 2b	Data type with named values		–			PLCopen Safety 1.0: no TYPE
3a 3b	Subrange data type		–			PLCopen Safety 1.0: no TYPE
4a 4b	Array data type		–			PLCopen Safety 1.0: no TYPE
5a 5b	Function block type and class as array element		–			PLCopen Safety 1.0: no TYPE
6a 6b	Structured data type		–			PLCopen Safety 1.0: no TYPE
7a 7b	Function block type and class as structure elements		–			PLCopen Safety 1.0: no TYPE
8a 8b	Structured data type with relative addressing AT					PLCopen Safety 1.0: no TYPE
9a	Structured data type with relative addressing AT and OVERLAP					PLCopen Safety 1.0: no TYPE
10a 10b	Directly represented elements of a structure - partly specified using “ * ”		–			PLCopen Safety 1.0: no TYPE
11a 11b	Directly derived data type					PLCopen Safety 1.0: no TYPE
12	Initialization using constant expression		–			PLCopen Safety 1.0: no TYPE
	Table 12 – Operation of references					
	Declaration					
1	Declaration of a reference type		–			PLCopen Safety 1.0: no REF_TO
	Assignment and comparison					
2a	Assignment reference to reference <reference> := <reference>		–			PLCopen Safety 1.0: no REF_TO
2b	Assignment reference to parameter of function, function block and method		–			PLCopen Safety 1.0: no REF_TO
2c	Comparison with NULL		–			PLCopen Safety 1.0: no REF_TO

	Referencing					
3a	`REF(<variable>)` Provides of the typed reference to the variable		–			PLCopen Safety 1.0: no REF_TO
3b	`REF(<function block instance>)` Provides the typed reference to the function block or class instance		–			PLCopen Safety 1.0: no REF_TO
	Dereferencing					
4	<reference> ^ Provides the content of the variable or content of the instance to which the reference variable contains the reference		–			PLCopen Safety 1.0: no REF_TO
	Table 13 – Declaration of variables					
1	Variable with elementary data type		✓			
2	Variable with user-defined data type		✓			function block type
3	Array		–			PLCopen Safety 1.0: no ARRAY
4	Reference		–			PLCopen Safety 1.0: no REF_TO
	Table 14 – Initialization of variables					
1	Initialization of a variable with elementary data type		✓			
2	Initialization of a variable with user-defined data type		–			
3	Array		–			PLCopen Safety 1.0: no ARRAY
4	Declaration and initialization of constants		✓			
5	Initialization using constant expressions		–			
6	Initialization of a reference		–			PLCopen Safety 1.0: no REF_TO
	Table 15 – Variable-length ARRAY variables					
1	Declaration `using *` `ARRAY [*, *, . . . ] OF data type`					PLCopen Safety 1.0: no ARRAY
	Standard functions LOWER_BOUND / UPPER_BOUND					
2a	Graphical representation					PLCopen Safety 1.0: no ARRAY
2b	Textual representation					PLCopen Safety 1.0: no ARRAY
	Table 16 – Directly represented variables					
	Location (NOTE 1)					
1	Input location	`I`	–			PLCopen Safety 1.0: no direct v.
2	Output location	`Q`	–			PLCopen Safety 1.0: no direct v.
3	Memory location	`M`	–			PLCopen Safety 1.0: no direct v.
	Size and data type					
4a	Single bit size	`X`	–			PLCopen Safety 1.0: no direct v.
4b	Single bit size	None				PLCopen Safety 1.0: no direct v.
5	Byte (8 bits) size	`B`	–			PLCopen Safety 1.0: no direct v.
6	Word (16 bits) size	`W`	–			PLCopen Safety 1.0: no direct v.
7	Double word (32 bits) size	`D`	–			PLCopen Safety 1.0: no direct v.
8	Long (quad) word (64 bits) size	`L`	–			PLCopen Safety 1.0: no direct v.
	Addressing					
9	Simple addressing	`%IX1`	–			PLCopen Safety 1.0: no direct v.
10	Hierarchical addressing using "**.**" (NOTE 3)	`%QX7.5`				PLCopen Safety 1.0: no direct v.
11	partly specified direct representation using asterisk "*****"		–			PLCopen Safety 1.0: no direct v.
	Table 17 – Partial access to ANY_BIT variables					

	Data Type - Access to					
1a	BYTE – bit VB2.%X0					PLCopen Safety 1.0: no bit acc.
1b	WORD – bit VW3.%X15					PLCopen Safety 1.0: no bit acc.
1c	DWORD - bit					PLCopen Safety 1.0: no bit acc.
1d	LWORD - bit					PLCopen Safety 1.0: no bit acc.
2a	WORD – byte VW4.%B0					PLCopen Safety 1.0: no bit acc.
2b	DWORD - byte					PLCopen Safety 1.0: no bit acc.
2c	LWORD - byte					PLCopen Safety 1.0: no bit acc.
3a	DWORD - word					PLCopen Safety 1.0: no bit acc.
3b	LWORD - word					PLCopen Safety 1.0: no bit acc.
4	LWORD – dword VL5.%D1					PLCopen Safety 1.0: no bit acc.
	Table 18 – Execution control using EN and ENO					
1	Usage **without** EN and ENO		–			PLCopen Safety 1.0: no EN/ENO
2	Usage of EN **only** (without ENO)		–			PLCopen Safety 1.0: no EN/ENO
3	Usage of ENO **only** (without EN)					
4	Usage of EN **and** ENO		–			PLCopen Safety 1.0: no EN/ENO
	Table 19 – Function declaration					
1a	Without result FUNCTION ... END_FUNCTION		–			PLCopen Safety 1.0: no FUNCTION
1b	With result FUNCTION <name> : <data type> END _FUNCTION		–			PLCopen Safety 1.0: no FUNCTION
2a	Inputs VAR_INPUT...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
2b	Outputs VAR_OUTPUT...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
2c	In-outs VAR_IN_OUT...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
2d	Temporary variables VAR_TEMP...END_VAR					PLCopen Safety 1.0: no FUNCTION
2e	**Temporary** variables VAR...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
2f	External variables VAR_EXTERNAL...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
2g	External constants VAR_EXTERNAL CONSTANT...END_VAR		–			PLCopen Safety 1.0: no FUNCTION
3a	Initialization of inputs					PLCopen Safety 1.0: no FUNCTION
3b	Initialization of outputs		–			PLCopen Safety 1.0: no FUNCTION
3c	Initialization of temporary variables		–			PLCopen Safety 1.0: no FUNCTION
--	EN/ENO inputs and outputs					
	Table 20 – Function call					

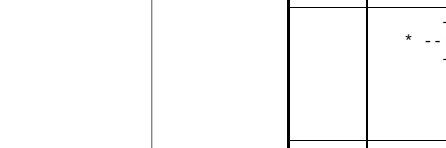
1a	Complete formal call (textual only) NOTE Shall be used if EN/ENO is necessary in the call.		–			FBD is not textual
1b	Incomplete formal call (textual only) NOTE May be used if EN/ENO is not necessary in the call.					
2	Non-formal call (textual only) (fix order and complete) NOTE Shall be used for call of standard functions without formal names.		–			FBD is not textual
3	Function without function result		–			PLCopen Safety 1.0: no FUNCTION
4	Graphical representation		✓			
5	Graphical usage of negated boolean input and output in graphical representation		–			
6	Graphical usage of `VAR_IN_OUT`					PLCopen Safety 1.0: no VAR_IN_OUT
	Table 21 – Typed and overloaded functions					
1a	**Overloaded function** **`ADD (ANY_Num to ANY_Num)`**		✓			All built-in Operators are overloaded
1b	**Conversion of inputs** **`ANY_ELEMENT TO_INT`**					
2a [a]	Typed function `ADD_INT`					
2b [a]	Conversion `WORD_TO_INT`		✓			
	Table 22 – Data type conversion function					
1a	Typed conversion input_TO_output		✓			
1b[a,b,e]	Overloaded conversion TO_output					
2a[c]	“Old” overloaded truncation TRUNC		–			
2b[c]	Typed truncation input_TRUNC_output					
2c[c]	Overloaded truncation TRUNC_output		–			
3a[d]	Typed input_BCD_TO_output					
3b[d]	Overloaded BCD_TO_output					
4a[d]	Typed input_TO_BCD_output					
4b[d]	Overloaded TO_BCD_output					
	Table 23 – Data type conversion of numeric data types					
1	LREAL TO REAL		–			PLCopen Safety 1.0: no REAL
2	LREAL TO LINT		–			PLCopen Safety 1.0: no REAL
3	LREAL TO DINT		–			PLCopen Safety 1.0: no REAL
4	LREAL TO INT		–			PLCopen Safety 1.0: no REAL
5	LREAL TO SINT		–			PLCopen Safety 1.0: no REAL
6	LREAL TO ULINT		–			PLCopen Safety 1.0: no REAL
7	LREAL TO UDINT		–			PLCopen Safety 1.0: no REAL
8	LREAL TO UINT		–			PLCopen Safety 1.0: no REAL

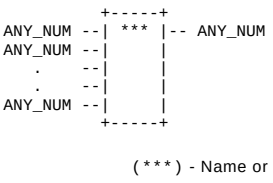
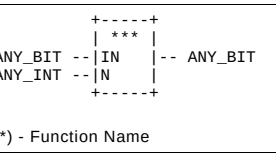
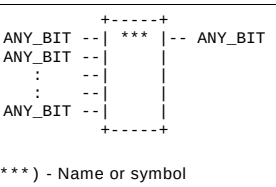
9	LREAL TO USINT		–			PLCopen Safety 1.0: no REAL
10	REAL TO LREAL		–			PLCopen Safety 1.0: no REAL
11	REAL TO LINT		–			PLCopen Safety 1.0: no REAL
12	REAL TO DINT		–			PLCopen Safety 1.0: no REAL
13	REAL TO INT		–			PLCopen Safety 1.0: no REAL
14	REAL TO SINT		–			PLCopen Safety 1.0: no REAL
15	REAL TO ULINT		–			PLCopen Safety 1.0: no REAL
16	REAL TO UDINT		–			PLCopen Safety 1.0: no REAL
17	REAL TO UINT		–			PLCopen Safety 1.0: no REAL
18	REAL TO USINT		–			PLCopen Safety 1.0: no REAL
19	LINT TO LREAL		–			PLCopen Safety 1.0: no LINT
20	LINT TO REAL		–			PLCopen Safety 1.0: no LINT
21	LINT TO DINT		–			PLCopen Safety 1.0: no LINT
22	LINT TO INT		–			PLCopen Safety 1.0: no LINT
23	LINT TO SINT		–			PLCopen Safety 1.0: no LINT
24	LINT TO ULINT		–			PLCopen Safety 1.0: no LINT
25	LINT TO UDINT		–			PLCopen Safety 1.0: no LINT
26	LINT TO UINT		–			PLCopen Safety 1.0: no LINT
27	LINT TO USINT		–			PLCopen Safety 1.0: no LINT
28	DINT TO LREAL		–			PLCopen Safety 1.0: no REAL
29	DINT TO REAL		–			PLCopen Safety 1.0: no REAL
30	DINT TO LINT		–			PLCopen Safety 1.0: no LINT
31	DINT TO INT		✓			
32	DINT TO SINT		–			PLCopen Safety 1.0: no SINT
33	DINT TO ULINT		–			PLCopen Safety 1.0: no UxINT
34	DINT TO UDINT		–			PLCopen Safety 1.0: no UxINT
35	DINT TO UINT		–			PLCopen Safety 1.0: no UxINT
36	DINT TO USINT		–			PLCopen Safety 1.0: no UxINT
37	INT TO LREAL		–			PLCopen Safety 1.0: no REAL
38	INT TO REAL		–			PLCopen Safety 1.0: no REAL
39	INT TO LINT		–			PLCopen Safety 1.0: no LINT
40	INT TO DINT		✓			
41	INT TO SINT		–			PLCopen Safety 1.0: no SINT
42	INT TO ULINT		–			PLCopen Safety 1.0: no UxINT
43	INT TO UDINT		–			PLCopen Safety 1.0: no UxINT
44	INT TO UINT		–			PLCopen Safety 1.0: no UxINT
45	INT TO USINT		–			PLCopen Safety 1.0: no UxINT
46	SINT TO LREAL		–			PLCopen Safety 1.0: no SINT
47	SINT TO REAL		–			PLCopen Safety 1.0: no SINT
48	SINT TO LINT		–			PLCopen Safety 1.0: no SINT
49	SINT TO DINT		–			PLCopen Safety 1.0: no SINT
50	SINT TO INT		–			PLCopen Safety 1.0: no SINT
51	SINT TO ULINT		–			PLCopen Safety 1.0: no SINT
52	SINT TO UDINT		–			PLCopen Safety 1.0: no SINT
53	SINT TO UINT		–			PLCopen Safety 1.0: no SINT
54	SINT TO USINT		–			PLCopen Safety 1.0: no SINT
55	ULINT TO LREAL		–			PLCopen Safety 1.0: no UxINT
56	ULINT TO REAL		–			PLCopen Safety 1.0: no UxINT
57	ULINT TO LINT		–			PLCopen Safety 1.0: no UxINT

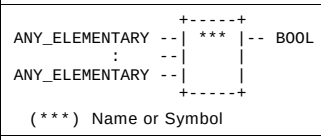
58	ULINT TO DINT		–			PLCopen Safety 1.0: no UxINT
59	ULINT TO INT		–			PLCopen Safety 1.0: no UxINT
60	ULINT TO SINT		–			PLCopen Safety 1.0: no UxINT
61	ULINT TO UDINT		–			PLCopen Safety 1.0: no UxINT
62	ULINT TO UINT		–			PLCopen Safety 1.0: no UxINT
63	ULINT TO USINT		–			PLCopen Safety 1.0: no UxINT
64	UDINT TO LREAL		–			PLCopen Safety 1.0: no UxINT
65	UDINT TO REAL		–			PLCopen Safety 1.0: no UxINT
66	UDINT TO LINT		–			PLCopen Safety 1.0: no UxINT
67	UDINT TO DINT		–			PLCopen Safety 1.0: no UxINT
68	UDINT TO INT		–			PLCopen Safety 1.0: no UxINT
69	UDINT TO SINT		–			PLCopen Safety 1.0: no UxINT
70	UDINT TO ULINT		–			PLCopen Safety 1.0: no UxINT
71	UDINT TO UINT		–			PLCopen Safety 1.0: no UxINT
72	UDINT TO USINT		–			PLCopen Safety 1.0: no UxINT
73	UINT TO LREAL		–			PLCopen Safety 1.0: no UxINT
74	UINT TO REAL		–			PLCopen Safety 1.0: no UxINT
75	UINT TO LINT		–			PLCopen Safety 1.0: no UxINT
76	UINT TO DINT		–			PLCopen Safety 1.0: no UxINT
77	UINT TO INT		–			PLCopen Safety 1.0: no UxINT
78	UINT TO SINT		–			PLCopen Safety 1.0: no UxINT
79	UINT TO ULINT		–			PLCopen Safety 1.0: no UxINT
80	UINT TO UDINT		–			PLCopen Safety 1.0: no UxINT
81	UINT TO USINT		–			PLCopen Safety 1.0: no UxINT
82	USINT TO LREAL		–			PLCopen Safety 1.0: no UxINT
83	USINT TO REAL		–			PLCopen Safety 1.0: no UxINT
84	USINT TO LINT		–			PLCopen Safety 1.0: no UxINT
85	USINT TO DINT		–			PLCopen Safety 1.0: no UxINT
86	USINT TO INT		–			PLCopen Safety 1.0: no UxINT
87	USINT TO SINT		–			PLCopen Safety 1.0: no UxINT
88	USINT TO ULINT		–			PLCopen Safety 1.0: no UxINT
89	USINT TO UDINT		–			PLCopen Safety 1.0: no UxINT
90	USINT TO UINT		–			PLCopen Safety 1.0: no UxINT
	Table 24 – Data type conversion of bit data types					
1	LWORD TO DWORD		–			PLCopen Safety 1.0: no LWORD
2	LWORD TO WORD		–			PLCopen Safety 1.0: no LWORD
3	LWORD TO BYTE		–			PLCopen Safety 1.0: no LWORD
4	LWORD TO BOOL		–			PLCopen Safety 1.0: no LWORD
5	DWORD TO LWORD		–			PLCopen Safety 1.0: no DWORD
6	DWORD TO WORD		✓			Conversion of inputs only
7	DWORD TO BYTE		–			PLCopen Safety 1.0: no DWORD
8	DWORD TO BOOL		–			PLCopen Safety 1.0: no DWORD
9	WORD TO LWORD		–			PLCopen Safety 1.0: no LWORD
10	WORD TO DWORD		✓			Conversion of outputs only
11	WORD TO BYTE		✓			Conversion of outputs only
12	WORD TO BOOL		✓			
13	BYTE TO LWORD		–			PLCopen Safety 1.0: no BYTE
14	BYTE TO DWORD		–			PLCopen Safety 1.0: no BYTE
15	BYTE TO WORD		✓			Conversion of inputs only

16	BYTE TO BOOL		–			PLCopen Safety 1.0: no BYTE
17	BYTE TO CHAR		–			PLCopen Safety 1.0: no BYTE
18	BOOL TO LWORD		–			PLCopen Safety 1.0: no LWORD
19	BOOL TO DWORD		–			PLCopen Safety 1.0: no DWORD
20	BOOL TO WORD		✓			
21	BOOL TO BYTE		–			PLCopen Safety 1.0: no BYTE
22	CHAR TO BYTE		–			PLCopen Safety 1.0: no CHAR
23	CHAR TO WORD		–			PLCopen Safety 1.0: no CHAR
24	CHAR TO DWORD		–			PLCopen Safety 1.0: no CHAR
25	CHAR TO LWORD		–			PLCopen Safety 1.0: no CHAR
26	WCHAR TO WORD		–			PLCopen Safety 1.0: no CHAR
27	WCHAR TO DWORD		–			PLCopen Safety 1.0: no CHAR
28	WCHAR TO LWORD		–			PLCopen Safety 1.0: no CHAR
	Table 25 – Data type conversion of bit types to numeric types					
1	LWORD TO LREAL		–			PLCopen Safety 1.0: no REAL
2	DWORD TO REAL		–			PLCopen Safety 1.0: no REAL
3	LWORD TO LINT		–			PLCopen Safety 1.0: no LWORD
4	LWORD TO DINT		–			PLCopen Safety 1.0: no LWORD
5	LWORD TO INT		–			PLCopen Safety 1.0: no LWORD
6	LWORD TO SINT		–			PLCopen Safety 1.0: no LWORD
7	LWORD TO ULINT		–			PLCopen Safety 1.0: no LWORD
8	LWORD TO UDINT		–			PLCopen Safety 1.0: no LWORD
9	LWORD TO UINT		–			PLCopen Safety 1.0: no LWORD
10	LWORD TO USINT		–			PLCopen Safety 1.0: no LWORD
11	DWORD TO LINT		–			PLCopen Safety 1.0: no LINT
12	DWORD TO DINT		✓			PLCopen Safety 1.0: no DWORD, Conversion of inputs only
13	DWORD TO INT		–			PLCopen Safety 1.0: no DWORD
14	DWORD TO SINT		–			PLCopen Safety 1.0: no SINT
15	DWORD TO ULINT		–			PLCopen Safety 1.0: no UxINT
16	DWORD TO UDINT		–			PLCopen Safety 1.0: no UxINT
17	DWORD TO UINT		–			PLCopen Safety 1.0: no UxINT
18	DWORD TO USINT		–			PLCopen Safety 1.0: no UxINT
19	WORD TO LINT		–			PLCopen Safety 1.0: no LINT
20	WORD TO DINT		✓			
21	WORD TO INT		✓			
22	WORD TO SINT		–			PLCopen Safety 1.0: no SINT
23	WORD TO ULINT		–			PLCopen Safety 1.0: no UxINT
24	WORD TO UDINT		–			PLCopen Safety 1.0: no UxINT
25	WORD TO UINT		–			PLCopen Safety 1.0: no UxINT
26	WORD TO USINT		–			PLCopen Safety 1.0: no UxINT
27	BYTE TO LINT		–			PLCopen Safety 1.0: no LINT
28	BYTE TO DINT		✓			
29	BYTE TO INT		✓			
30	BYTE TO SINT		–			PLCopen Safety 1.0: no SINT
31	BYTE TO ULINT		–			PLCopen Safety 1.0: no UxINT
32	BYTE TO UDINT		–			PLCopen Safety 1.0: no UxINT
33	BYTE TO UINT		–			PLCopen Safety 1.0: no UxINT
34	BYTE TO USINT		–			PLCopen Safety 1.0: no UxINT

35	BOOL TO LINT		–			PLCopen Safety 1.0: no LINT
36	BOOL TO DINT		✓			
37	BOOL TO INT		✓			
38	BOOL TO SINT		–			PLCopen Safety 1.0: no SINT
39	BOOL TO ULINT		–			PLCopen Safety 1.0: no UxINT
40	BOOL TO UDINT		–			PLCopen Safety 1.0: no UxINT
41	BOOL TO UINT		–			PLCopen Safety 1.0: no UxINT
42	BOOL TO USINT		–			PLCopen Safety 1.0: no UxINT
43	LREAL TO LWORD		–			PLCopen Safety 1.0: no REAL
44	REAL TO DWORD		–			PLCopen Safety 1.0: no REAL
45	LINT TO LWORD		–			PLCopen Safety 1.0: no LINT
46	LINT TO DWORD		–			PLCopen Safety 1.0: no LINT
47	LINT TO WORD		–			PLCopen Safety 1.0: no LINT
48	LINT TO BYTE		–			PLCopen Safety 1.0: no LINT
49	DINT TO LWORD		–			PLCopen Safety 1.0: no LWORD
50	DINT TO DWORD		✓			PLCopen Safety 1.0: no DWORD, Conversion of outputs only
51	DINT TO WORD		✓			
52	DINT TO BYTE		✓			PLCopen Safety 1.0: no BYTE, Conversion of outputs only
53	INT TO LWORD		–			PLCopen Safety 1.0: no LWORD
54	INT TO DWORD		✓			PLCopen Safety 1.0: no DWORD, Conversion of outputs only
55	INT TO WORD		✓			
56	INT TO BYTE		✓			PLCopen Safety 1.0: no BYTE, Conversion of outputs only
57	SINT TO LWORD		–			PLCopen Safety 1.0: no SINT
58	SINT TO DWORD		–			PLCopen Safety 1.0: no SINT
59	SINT TO WORD		–			PLCopen Safety 1.0: no SINT
60	SINT TO BYTE		–			PLCopen Safety 1.0: no SINT
61	ULINT TO LWORD		–			PLCopen Safety 1.0: no UxINT
62	ULINT TO DWORD		–			PLCopen Safety 1.0: no UxINT
63	ULINT TO WORD		–			PLCopen Safety 1.0: no UxINT
64	ULINT TO BYTE		–			PLCopen Safety 1.0: no UxINT
65	UDINT TO LWORD		–			PLCopen Safety 1.0: no UxINT
66	UDINT TO DWORD		–			PLCopen Safety 1.0: no UxINT
67	UDINT TO WORD		–			PLCopen Safety 1.0: no UxINT
68	UDINT TO BYTE		–			PLCopen Safety 1.0: no UxINT
69	UINT TO LWORD		–			PLCopen Safety 1.0: no UxINT
70	UINT TO DWORD		–			PLCopen Safety 1.0: no UxINT
71	UINT TO WORD		–			PLCopen Safety 1.0: no UxINT
72	UINT TO BYTE		–			PLCopen Safety 1.0: no UxINT
73	USINT TO LWORD		–			PLCopen Safety 1.0: no UxINT
74	USINT TO DWORD		–			PLCopen Safety 1.0: no UxINT
75	USINT TO WORD		–			PLCopen Safety 1.0: no UxINT
76	USINT TO BYTE		–			PLCopen Safety 1.0: no UxINT
	Table 26 – Data type conversion of date and time types					
1	LTIME TO TIME		–			PLCopen Safety 1.0: no LTIME
2	TIME TO LTIME		–			PLCopen Safety 1.0: no LTIME
3	LDT TO DT					

4	LDT TO DATE					
5	LDT TO LTOD					
6	LDT TO TOD					
7	DT TO LDT					
8	DT TO DATE		–			PLCopen Safety 1.0: noDT
9	DT TO LTOD					
10	DT TO TOD		–			PLCopen Safety 1.0: no DT
11	LTOD TO TOD					
12	TOD TO LTOD					
	Table 27 – Data type conversion of character types					
1	WSTRING TO STRING		–			PLCopen Safety 1.0: no STRING
2	WSTRING TO WCHAR					
3	STRING TO WSTRING		–			PLCopen Safety 1.0: no STRING
4	STRING TO CHAR					
5	WCHAR TO WSTRING					
6	WCHAR TO CHAR					
7	CHAR TO STRING					
8	CHAR TO WCHAR					
	Table 28 – Numerical Functions					
	Graphical form					
						
	General functions					
1	ABS(x)		–			
2	SQRT(x)		–			PLCopen Safety 1.0: no REAL
	Logarithmic functions					
3	LN(x)		–			PLCopen Safety 1.0: no REAL
4	LOG(x)		–			PLCopen Safety 1.0: no REAL
5	EXP(x)		–			PLCopen Safety 1.0: no REAL
	Trigonometric functions					
6	SIN(x)		–			PLCopen Safety 1.0: no REAL
7	COS(x)		–			PLCopen Safety 1.0: no REAL
8	TAN(x)		–			PLCopen Safety 1.0: no REAL
9	ASIN(x)		–			PLCopen Safety 1.0: no REAL
10	ACOS(x)		–			PLCopen Safety 1.0: no REAL
11	ATAN(x)		–			PLCopen Safety 1.0: no REAL
12	ATAN2(y, x) 					
	Table 29 – Arithmetic functions					

	Graphical form					
						
	Extensible arithmetic functions					
1 [c]	Addition		✓			
2	Multiplication		✓			
	Non-extensible arithmetic functions					
3 [c]	Subtraction		✓			
4 [d]	Division		✓			
5 [e]	Modulo		–			PLCopen Safety 1.0: no MOD
6 [f]	Exponentiation		–			PLCopen Safety 1.0: no EXPT
	Table 30 – Bit shift functions					
	Graphical form					
						
1	Shift left		–			PLCopen Safety 1.0: no shift ops
2	Shift right		–			PLCopen Safety 1.0: no shift ops
3	Rotation left		–			PLCopen Safety 1.0: no shift ops
4	Rotation right		–			PLCopen Safety 1.0: no shift ops
	Table 31 – Bitwise Boolean functions					
	Graphical form					
						
1	And		✓			
2	Or		✓			
3	Exclusive Or		✓			
4	Not		✓			
	Table 32 – Assignment and selection functions					

1	Move a,d (assignment)		–			PLCopen Safety 1.0: no MOVE
2	Binary selection		✓			
3	Extensible maximum function		–			
4	Extensible minimum function		–			
5	Limiter		–			
6	Extensible multiplexer		✓			
	Table 33 – Comparison functions					
	Graphical form					
	 (***) Name or Symbol					We only support comparison with two operands
1	Decreasing sequence		✓			not extensible
2	Monotonic sequence:		✓			not extensible
3	Equality		✓			not extensible
4	Monotonic sequence		✓			not extensible
5	Increasing sequence		✓			not extensible
6	Inequality		✓			not extensible
	Table 34 – Character string functions					
1	String length		–			PLCopen Safety 1.0: no STRING
2	Left		–			PLCopen Safety 1.0: no STRING
3	Right		–			PLCopen Safety 1.0: no STRING
4	Middle		–			PLCopen Safety 1.0: no STRING
5	Extensible concatenation		–			PLCopen Safety 1.0: no STRING
6	Insert		–			PLCopen Safety 1.0: no STRING
7	Delete		–			PLCopen Safety 1.0: no STRING
8	Replace		–			PLCopen Safety 1.0: no STRING
9	Find		–			PLCopen Safety 1.0: no STRING
	Table 35 – Numerical functions of date and duration					
1a	ADD		✓			
1b	ADD_TIME					
1c	ADD_LTIME					
2a	ADD		–			PLCopen Safety 1.0: no TOD
2b	ADD_TOD_TIME					
2c	ADD_LTOD_LTIME					
3a	ADD		–			PLCopen Safety 1.0: no DT
3b	ADD_DT_TIME					
3c	ADD_LDT_LTIME					
4a	SUB		✓			
4b	SUB_TIME					
4c	SUB_LTIME					
5a	SUB		–			PLCopen Safety 1.0: no DATE
5b	SUB_DATE_DATE					

5c	SUB_LDATE_LDATE					
6a	SUB		–			PLCopen Safety 1.0: no TOD
6b	SUB_TOD_TIME					
6c	SUB_LTOD_LTIME					
7a	SUB		–			PLCopen Safety 1.0: no TOD
7b	SUB_TOD_TOD					
7c	SUB_LTOD_LTOD_~~LTIME~~					
8a	SUB		–			PLCopen Safety 1.0: no DT
8b	SUB_DT_TIME					
8C	SUB_LDT_LTIME					
9a	SUB		–			PLCopen Safety 1.0: no DT
9b	SUB_DT_DT					
9b	SUB_LDT_LDT					
10a	MUL		✓			
10b	MUL_TIME					
10c	MUL_LTIME					
11a	DIV		✓			
11b	DIV_TIME					
11c	DIV_LTIME					
	Table 36 – Functions of time data types CONCAT and SPLIT					
	Concatenate time data types					
1a	CONCAT_DATE_TOD					
1b	CONCAT_LDATE_LTOD					
2	CONCAT_DATE					
3a	CONCAT_TOD					
3b	CONCAT_LTOD					
4a	CONCAT_DT					
4b	CONCAT_LDT					
	Split time data types					
5	SPLIT_DATE					
6a	SPLIT_TOD					
6b	SPLIT_LTOD					
7a	SPLIT_DT					
7b	SPLIT_LDT					
	Get day of the week					
8	DAY_OF_WEEK					
	Table 37 – Functions for Endianess Conversion					
1	TO_BIG_ENDIAN					
2	TO_LITTLE_ENDIAN					
3	BIG_ENDIAN_TO					
4	LITTLE_ENDIAN_TO					
	Table 38 – Functions of enumerated data types					
1	SEL		–			PLCopen Safety 1.0: no enum.
2	MUX		–			PLCopen Safety 1.0: no enum.
3[a]	EQ		–			PLCopen Safety 1.0: no enum.
4[a]	NE		–			PLCopen Safety 1.0: no enum.

	Table 39 – Validate functions					
1	IS_VALID					
2	IS_VALID_BCD					
	Table 40 – Function block type declaration					
1	Declaration of function block type FUNCTION_BLOCK ... END_FUNCTION_BLOCK		✓			
2a	Declaration of inputs VAR_INPUT ... END_VAR		✓			
2b	Declaration of outputs VAR_OUTPUT ... END_VAR		✓			
2c	Declaration of in-outs VAR_IN_OUT ... END_VAR		✓			PLCopen Safety 1.0: no VAR_IN_OUT, only system programming
2d	Declaration of temporary variables VAR_TEMP ... END_VAR		–			PLCopen Safety 1.0: no VAR_TEMP
2e	Declaration of **static** variables VAR ... END_VAR		✓			
2f	Declaration of external variables VAR_EXTERNAL ... END_VAR		✓			
2g	Declaration of external constants VAR_EXTERNAL CONSTANT ... END_VAR		✓			
3a	Initialization of inputs		✓			
3b	Initialization of outputs		✓			
3c	Initialization of static variables		✓			
3d	Initialization of temporary variables		–			PLCopen Safety 1.0: no VAR_TEMP
-	EN/ENO inputs and outputs					
4a	Declaration of retained behaviour on input variables (RETAIN)		–			PLCopen Safety 1.0: no RETAIN
4b	Declaration of retained behaviour on output variables (RETAIN)		–			PLCopen Safety 1.0: no RETAIN
4c	Declaration of non-retained behaviour on input variables (NON_RETAIN)					PLCopen Safety 1.0: no RETAIN
4d	Declaration of non-retained behaviour on output variables (NON_RETAIN)					PLCopen Safety 1.0: no RETAIN
4e	Declaration of retained behaviour on static variables (RETAIN)		–			PLCopen Safety 1.0: no RETAIN
4f	Declaration of retained behaviour on static variables (NON_RETAIN)					PLCopen Safety 1.0: no RETAIN
5a	Declaration of retained behaviour on static variables (RETAIN)		–			PLCopen Safety 1.0: no RETAIN
5b	Declaration of retained behaviour on local function block instances (RETAIN)		–			PLCopen Safety 1.0: no RETAIN
6a	Textual declaration of - rising edge inputs (R_EDGE)					PLCopen Safety 1.0: no x_EDGE
6b	Textual declaration of - falling edge inputs (F_EDGE)					PLCopen Safety 1.0: no x_EDGE
7a	Graphical declaration of - rising edge inputs (>)		–			PLCopen Safety 1.0: no x_EDGE
7b	Graphical declaration of - falling edge inputs (<)		–			PLCopen Safety 1.0: no x_EDGE
	Table 41 – Function block instance declaration					
1	Declaration of function block instance(s)		✓			
2	Declaration of function block instance with **initialization** of its variables		–			

	Table 42 – Function block call					
1	**Complete** formal call (textual only) NOTE - Shall be used if EN/ENO is necessary in calls.		–			FBD is not textual
2	**Incomplete** formal call (textual only)		–			FBD is not textual
3	Graphical call		✓			
4	Graphical call with **negated** boolean input and output		–			
5a	Graphical call with usage of `VAR_IN_OUT`		–			PLCopen Safety 1.0: no VAR_IN_OUT
5b	Graphical call with assignment of `VAR_IN_OUT` to a variable					
6a	Textual call with **separate** assignment of input `FB_Instance.Input := x;`		–			FBD is not textual
6b	Graphical call with **separate** assignment of input		–			
7	Textual output read **after** function block call `x:= FB_Instance.Output;`		–			FBD is not textual
8a	Textual output assignment **in** function block call		–			FBD is not textual
8b	Textual output assignment **in** function block call with negation		–			FBD is not textual
9a	Textual call with function block instance name as input		–			FBD is not textual
9b	Graphical call with function block instance name as input		–			
10a	Textual call with function block instance name as `VAR_IN_OUT`		–			PLCopen Safety 1.0: no VAR_IN_OUT
10b	Graphical call with function block instance name as `VAR_IN_OUT`		–			PLCopen Safety 1.0: no VAR_IN_OUT
11a	Textual call with function block instance name as external variable		–			FBD is not textual
11b	Graphical call with function block instance name as external variable		✓			in programs
	Table 43 – Standard bistable function blocks					

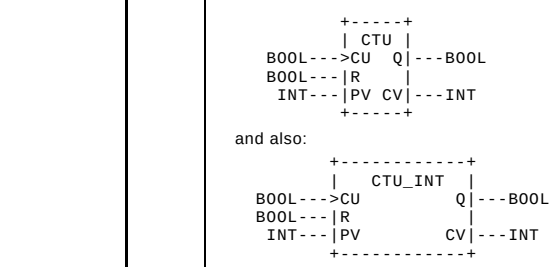
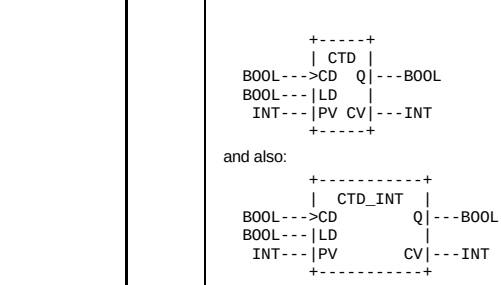
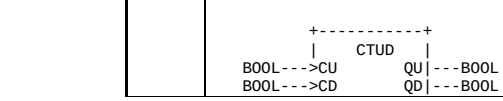
1a	Bistable function block (set dominant): `SR(S1,R,Q1)`
	`+-----+`
	`|  SR |`
	`BOOL---|S1 Q1|---BOOL`
	`BOOL---|R  |`
	`+-----+`
			–			

1b	Bistable function block (set dominant)
	with long input names: `SR(SET1, RESET, Q1)`
	`+--------+`
	`|  SR  |`
	`BOOL---|SET1 Q1|---BOOL`
	`BOOL---|RESET  |`
	`+--------+`
			✓			SF_SR with SAFE data types

2a	Bistable function block (reset dominant): `RS(S, R1, Q1)`
	`+-----+`
	`|  RS |`
	`BOOL---|S  Q1|---BOOL`
	`BOOL---|R1  |`
	`+-----+`
			–			

2b	Bistable function block (reset dominant)
	with long input names: `RS(SET,RESET1, Q1)`
	`+--------+`
	`|  RS  |`
	`BOOL---|SET  Q1|---BOOL`
	`BOOL---|R1  |`
	`+--------+`
			✓			SF_RS with SAFE data types

	Table 44 – Standard edge detection function blocks					

1	Rising edge detector: R_TRIG(CLK, Q) 		✓			SF_R_TRIG with SAFE data types
2	Falling edge detector: F_TRIG(CLK, Q) 		✓			SF_F_TRIG with SAFE data types
	Table 45 – Standard counter function blocks					
	Up-Counter					
1a	**CTU_INT**(CU, R, **PV**, Q, **CV**) or CTU(..)  		✓			only untyped: SF_CTU with SAFE data types
1b	CTU_DINT PV, CV: DINT					
1c	CTU_LINT PV, CV: LINT					
1d	CTU_UDINT PV, CV: UDINT					
1e	CTU_ULINT(CD, LD, PV, CV) PV, CV: ULINT					
	Down-counters					
2a	CTD_INT(CD, LD, PV, Q, CV) or CTD  		✓			only untyped: SF_CTD with SAFE data types
2b	CTD_DINT PV, CV: DINT					
2c	CTD_LINT PV, CV: LINT					
2d	CTD_UDINT PV, CV: UDINT					
2e	CTD_ULINT PV, CV: UDINT					
	Up-down counters					
3a	CTUD_INT(CD, LD, PV, Q, CV) or CTUD(..)  		✓			only untyped: SF_CTUD with SAFE data types

	BOOL---|R BOOL---|LD INT---|PV CV|---INT and also: CTUD_INT BOOL--->CU QU|---BOOL BOOL--->CD QD|---BOOL BOOL---|R BOOL---|LD INT---|PV CV|---INT					
3b	CTUD_DINT PV, CV: DINT					
3c	CTUD_LINT PV, CV: LINT					
3d	CTUD_UDINT PV, CV: UDINT					
3e	CTUD_ULINT PV, CV: ULINT					
	Table 46 – Standard timer function blocks					
1a	**Pulse**, overloaded **TP**		✓			SF_TP with SAFE data types
1b	Pulse using TIME		–			
1c	Pulse using LTIME		–			
2a	**On-delay**, overloaded **TON**		✓			SF_TON with SAFE data types
2b	On-delay using TIME		–			
2c	On-delay using LTIME		–			
2d [a]	On-delay, overloaded (Graphical)					
3a	**Off-delay**, overloaded **TOF**		✓			SF_TOF with SAFE data types
3b	Off-delay using TIME		–			
3c	Off-delay using LTIME		–			
3d [a]	Off-delay, overloaded (Graphical)					
	Table 47 – Program declaration					
1	Declaration of a program PROGRAM ... END_PROGRAM		✓			
2a	Declaration of inputs VAR_INPUT ... END_VAR		–			
2b	Declaration of outputs VAR_OUTPUT ... END_VAR		–			
2c	Declaration of in-outs VAR_IN_OUT ... END_VAR		–			PLCopen Safety 1.0: no VAR_IN_OUT
2d	Declaration of temporary variables VAR_TEMP ... END_VAR		–			PLCopen Safety 1.0: no VAR_TEMP
2e	Declaration of static variables VAR ... END_VAR		✓			
2f	Declaration of external variables VAR_EXTERNAL ... END_VAR		✓			
2g	Declaration of external constants VAR_EXTERNAL CONSTANT ... END_VAR		✓			
3a	Initialization of inputs		–			no program inputs
3b	Initialization of outputs		–			no program outputs
3c	Initialization of static variables		✓			
3d	Initialization of temporary variables		–			PLCopen Safety 1.0: no VAR_TEMP

4a	Declaration of `RETAIN` qualifier on input variables		–			PLCopen Safety 1.0: no RETAIN
4b	Declaration of `RETAIN` qualifier on output variables		–			PLCopen Safety 1.0: no RETAIN
4c	Declaration of `NON_RETAIN` qualifier on input variables					PLCopen Safety 1.0: no RETAIN
4d	Declaration of `NON_RETAIN` qualifier on output variables					PLCopen Safety 1.0: no RETAIN
4e	Declaration of `RETAIN` qualifier on static variables		–			PLCopen Safety 1.0: no RETAIN
4f	Declaration of `NON_RETAIN` qualifier on static variables					PLCopen Safety 1.0: no RETAIN
5a	Declaration of `RETAIN` qualifier on local Function block instances		–			PLCopen Safety 1.0: no RETAIN
5b	Declaration of `NON_RETAIN` qualifier on local FB instances					PLCopen Safety 1.0: no RETAIN
6a	Textual declaration of - rising edge inputs					PLCopen Safety 1.0: no x_EDGE
6b	- falling edge inputs (textual)					PLCopen Safety 1.0: no x_EDGE
7a	Graphical declaration of - rising edge inputs (>)		–			PLCopen Safety 1.0: no x_EDGE
7b	- falling edge inputs (<)		–			PLCopen Safety 1.0: no x_EDGE
8a	`VAR_GLOBAL...END_VAR` declaration within a `PROGRAM`					In own object
8b	`VAR_GLOBAL CONSTANT` declarations within `PROGRAM` type declarations					In own object
9	`VAR_ACCESS...END_VAR` declaration within a `PROGRAM`					PLCopen Safety 1.0: no VAR_ACCESS
	Table 48 - Class					

1	CLASS ... END_CLASS					New in ed.3: OOP
1a	FINAL specifier					New in ed.3: OOP
	Adapted from function block					
2a	Declaration of variables VAR ... END_VAR					New in ed.3: OOP
2b	Initialization of variables					New in ed.3: OOP
3a	RETAIN qualifier on internal variables					New in ed.3: OOP
3b	NON_RETAIN qualifier on internal variables					New in ed.3: OOP
4a	VAR_EXTERNAL declarations within class declarations					New in ed.3: OOP
4b	VAR_EXTERNAL CONSTANT declarations within class declarations					New in ed.3: OOP
	Methods and specifiers					
5	METHOD...END_METHOD					New in ed.3: OOP
5a	PUBLIC specifier					New in ed.3: OOP
5b	PRIVATE specifier					New in ed.3: OOP
5c	INTERNAL specifier					New in ed.3: OOP
5d	PROTECTED specifier					New in ed.3: OOP
5e	FINAL specifier					New in ed.3: OOP
	Inheritance					
6	EXTENDS					New in ed.3: OOP
7	OVERRIDE					New in ed.3: OOP
8	ABSTRACT					New in ed.3: OOP
	Access reference					
9a	THIS					New in ed.3: OOP
9b	SUPER					New in ed.3: OOP
	Variable access specifiers					
10a	PUBLIC specifier					New in ed.3: OOP
10b	PRIVATE specifier					New in ed.3: OOP
10c	INTERNAL specifier					New in ed.3: OOP
10d	PROTECTED specifier					New in ed.3: OOP
	Polymorphism					
11a	with VAR_IN_OUT					New in ed.3: OOP
11b	with reference					New in ed.3: OOP
	Table 49 – Class instance declaration					
1	Declaration of class instance(s) with default initialization					New in ed.3: OOP
2	Declaration of class instance with **initialization** of its public variables					New in ed.3: OOP
	Table 50 – Textual call of methods – Formal and non-formal parameter list					

1a	Complete formal call (textual only) NOTE Shall be used if EN/ENO is necessary in calls.		–			New in ed.3: OOP
1b	Incomplete formal call (textual only) NOTE Shall be used if EN/ENO is not necessary in calls.		–			New in ed.3: OOP
2	Non-formal call (textual only) (fix order and complete)		–			New in ed.3: OOP
	Table 51 - Interface					
1	`INTERFACE ... END_INTERFACE`		–			New in ed.3: OOP
	Methods and specifiers					
2	`METHOD...END_METHOD`		–			New in ed.3: OOP
	Inheritance					
3	`EXTENDS`		–			New in ed.3: OOP
	Usage of interface					
4a	`IMPLEMENTS` interface		–			New in ed.3: OOP
4b	`IMPLEMENTS` multi-interfaces		–			New in ed.3: OOP
4c	Interface as type of a variable		–			New in ed.3: OOP
	Table 52 – Assignment Attempt					
1	Assignment attempt with interfaces using ?=					New in ed.3: OOP
2	Assignment attempt with references using ?=					New in ed.3: OOP
	Table 53 – Object oriented function block					
1	Object oriented Function block		–			New in ed.3: OOP
1a	`FINAL` specifier		–			New in ed.3: OOP
	Methods and specifiers					
5	`METHOD...END_METHOD`		–			New in ed.3: OOP
5a	`PUBLIC` specifier		–			New in ed.3: OOP
5b	`PRIVATE` specifier		–			New in ed.3: OOP
5c	`INTERNAL` specifier		–			New in ed.3: OOP
5d	`PROTECTED` specifier		–			New in ed.3: OOP
5e	`FINAL` specifier		–			New in ed.3: OOP
	Usage of interface					
6a	`IMPLEMENTS` interface		–			New in ed.3: OOP
6b	`IMPLEMENTS` multi-interfaces		–			New in ed.3: OOP
6c	Interface as type of a variable		–			New in ed.3: OOP
	Inheritance					
7a	`EXTENDS`		–			New in ed.3: OOP
7b	`EXTENDS`		–			New in ed.3: OOP
8	`OVERRIDE`					New in ed.3: OOP
9	`ABSTRACT`					New in ed.3: OOP
	Access reference					
10a	`THIS`		–			New in ed.3: OOP
10b	`SUPER`		–			New in ed.3: OOP
10c	`SUPER()`		–			New in ed.3: OOP

	Variable access specifiers					
11a	PUBLIC specifier					New in ed.3: OOP
11b	PRIVATE specifier					New in ed.3: OOP
11c	INTERNAL specifier					New in ed.3: OOP
11d	PROTECTED specifier					New in ed.3: OOP
	Polymorphism					
12a	with VAR_IN_OUT with equal signature		–			New in ed.3: OOP
12b	With VAR_IN_OUT with compatible signature		–			New in ed.3: OOP
12c	with reference with equal signature		–			New in ed.3: OOP
12d	with reference with compatible signature		–			New in ed.3: OOP
	Table 54 – SFC step					
1a	Step - graphical form with directed links		–			no SFC
1b	Initial step - graphical form with directed link		–			no SFC
2a	Step - textual form without directed links		–			no SFC
2b	Initial step - textual form without directed links		–			no SFC
3a [a]	Step flag - general form ***.X = BOOL#1 when *** is active, BOOL#0 otherwise		–			no SFC
3b [a]	Step flag - direct connection of Boolean variable ***.X to right side of step		–			no SFC
4 [a]	Step elapsed time - general form ***.T = a variable of type TIME		–			no SFC
	Table 55 – SFC transition and transition condition					
1[a]	Transition condition physically or logically adjacent to the transition using **ST** language		–			no SFC
2[a]	Transition condition physically or logically adjacent to the transition using **LD** language					no SFC
3[a]	Transition condition physically or logically adjacent to the transition using **FBD** language					no SFC
4[a]	Use of connector					no SFC
5[a]	Transition condition: Using **LD** language		–			no SFC
6[b]	Transition condition: Using **FBD** language		–			no SFC
7[b]	Textual equivalent of feature 1 using **ST** language		–			no SFC
8[b]	Textual equivalent of feature 1 using **IL** language		–			no SFC
9[a]	Use of transition name		–			no SFC
10[a]	Transition condition using **LD** language		–			no SFC
11[b]	Transition condition using **FBD** language		–			no SFC
12[c]	Transition condition using **IL** language		–			no SFC
13[d]	Transition condition using **ST** language		–			no SFC
	Table 56 – SFC declaration of actions					
1	Any Boolean variable declared in a VAR or VAR_OUTPUT block, or their graphical equivalents, can be an action.		–			no SFC
2l	Graphical declaration in **LD** language					no SFC
2s	Inclusion of SFC elements in **action**					no SFC
2f	Graphical declaration in **FBD** language					no SFC

3s	Textual declaration in **ST** language					no SFC
3i	Textual declaration in **IL** language					no SFC
	Table 57 – Step/action association					
1	Action block physically or logically adjacent to the step		–			no SFC
2	Concatenated action blocks physically or logically adjacent to the step		–			no SFC
3	Textual step body		–			no SFC
4 [a]	Action block "d" field					no SFC
	Table 58 – Action block					
1 [a]	"a" : Qualifier as per 6.6.4.5		–			no SFC
2	"b" : Action name		–			no SFC
3 [b]	"c" : Boolean "indicator" variables (**deprecated**)					no SFC
	"d" : Action using:					
4i	IL language					no SFC
4s	ST language					no SFC
4l	LD language					no SFC
4f	FBD language					no SFC
5l	Use of action blocks LD					no SFC
5f	Use of action blocks in FBD					no SFC
	Table 59 – Action qualifiers					
1	Non-stored (null qualifier)		–			no SFC
2	**N**on-stored		–			no SFC
3	overriding **R**eset		–			no SFC
4	**S**et (**S**tored)		–			no SFC
5	time **L**imited		–			no SFC
6	time **D**elayed		–			no SFC
7	**P**ulse		–			no SFC
8	**S**tored and time **D**elayed		–			no SFC
9	**D**elayed and **S**tored		–			no SFC
10	**S**tored and time **L**imited		–			no SFC
11	**P**ulse (rising edge)		–			no SFC
12	**P**ulse (falling edge)		–			no SFC
	Table 60 – Action control features					
1	With final scan		–			no SFC
2	Without final scan					no SFC
	Table 61 – Sequence evolution – graphical					
1	Single sequence		–			no SFC
2a	Divergence of sequence with left to right priority		–			no SFC
2b	Divergence of sequence with numbered branches					no SFC
2c	Divergence of sequence with mutual exclusion					no SFC
3	Convergence of sequence		–			no SFC
4a	Simultaneous divergence after a single transition		–			no SFC

4b	Simultaneous convergence before one transition					no SFC
4c	Simultaneous divergence after conversion					no SFC
4d	Simultaneous convergence before a sequence selection					no SFC
5a,b,c	Sequence skip		–			no SFC
6a,b,c	Sequence loop					no SFC
7	Directional arrows		–			no SFC
	Table 62 – Configuration and resource declaration					Resource configuration and Application configuration is done with specialized editors
1	CONFIGURATION...END_CONFIGURATION					
2	VAR_GLOBAL...END_VAR within CONFIGURATION					
3	RESOURCE...ON ...END_RESOURCE					
4	VAR_GLOBAL...END_VAR within RESOURCE					
5a	Periodic TASK (see NOTE 1)					
5b	Non-periodic TASK (see NOTE 1)					
6a	WITH for PROGRAM to TASK association (see NOTE 1)					
6b	WITH for FUNCTION_BLOCK to TASK association (see NOTE 1)					
6c	PROGRAM with no TASK association (see NOTE 1)					
7	Directly represented variables in VAR_GLOBAL					
8a	Connection of directly represented variables to PROGRAM inputs					
8b	Connection of GLOBAL variables to PROGRAM inputs					
9a	Connection of PROGRAM outputs to directly represented variables					
9b	Connection of PROGRAM outputs to GLOBAL variables					
10a	VAR_ACCESS...END_VAR					
10b	Access paths to directly represented variables					
10c	Access paths to PROGRAM inputs					
10d	Access paths to GLOBAL variables in RESOURCEs					
10e	Access paths to GLOBAL variables in CONFIGURATIONs					
10f	Access paths to PROGRAM outputs					
10g	Access paths to PROGRAM internal variables					
10h	Access paths to function block inputs					
10i	Access paths to function block outputs					
11a	VAR_CONFIG...END_VAR to variables [a]		–			PLCopen Safety 1.0: no VAR_CONFIG
11b	VAR_CONFIG...END_VAR to components of structures		–			PLCopen Safety 1.0: no VAR_CONFIG
12a	VAR_GLOBAL CONSTANT in RESOURCE					
12b	VAR_GLOBAL CONSTANT in CONFIGURATION					
13a	VAR_EXTERNAL in RESOURCE					
13b	VAR_EXTERNAL CONSTANT in RESOURCE					
	Table 63 - Task					Task declaration is done with a specialized editor.

1a	Textual declaration of periodic TASK					
1b	Textual declaration of non-periodic TASK					
	Graphical representation of TASKs (general form)					
2a	Graphical representation of periodic TASKs (witrh INTERVAL)					
2b	Graphical representation of non-periodic TASK (with SINGLE)					
3a	Textual association with PROGRAMs					
3b	Textual association with function blocks					
4a	Graphical association with PROGRAMs					
4b	Graphical association with function blocks within PROGRAMs					
5a	Non-preemptive scheduling					
5b	Preemptive scheduling					
	Table 64 – Namespace (Features)					Namespaces are a feature of our library concept, when properly used (qualified-only)
1a	Public namespace (without access specifier)		–			New in ed.3: NAMESPACE
1b	Internal namespace (with INTERNAL specifier)		–			New in ed.3: NAMESPACE
2	Nested namespaces		–			New in ed.3: NAMESPACE
3	Variable access specifier INTERNAL Equivalent to feature in Table 48		–			New in ed.3: NAMESPACE
4	Method access specifier INTERNAL Equivalent to feature in Table 48		–			New in ed.3: NAMESPACE
5	Language element with access specifier INTERNAL: • User-defined data types - using keyword TYPE • Functions • Function block types • Classes • Interfaces		–			New in ed.3: NAMESPACE
	Table 65 – Nested namespace declaration options (Feature)					
1	Lexically nested namespace declaration Equivalent to feature 2 of Table 64		–			New in ed.3: NAMESPACE
2	Nested namespace declaration by fully qualified name		–			New in ed.3: NAMESPACE
3	Mixed lexically nested namespace and namespace nested by fully qualified name		–			New in ed.3: NAMESPACE
	Table 66 – Namespace directive USING					
1	USING in global namespace					New in ed.3: NAMESPACE
2	USING in other namespace					New in ed.3: NAMESPACE
3	USING in POUs ▪ Functions ▪ Function block types ▪ Classes ▪ Methods ▪ Interfaces					New in ed.3: NAMESPACE

	Table 67 – Parenthesized expression for IL language					
1	Parenthesized expression beginning with explicit operator:					no IL
2	Parenthesized expression (short form)					no IL
	Table 68 – Instruction list operators					
1	LD					no IL
2	ST					no IL
3	S[e], R[e]					no IL
4	AND					no IL
5	&					no IL
6	OR					no IL
7	XOR					no IL
8	NOT[d]					no IL
9	ADD					no IL
10	SUB					no IL
11	MUL					no IL
12	DIV					no IL
13	MOD					no IL
14	GT					no IL
15	GE					no IL
16	EQ					no IL
17	NE					no IL
18	LE					no IL
9	LT					no IL
20	JMP[b]					no IL
21	CAL[c]					no IL
22	RET[f]					no IL
23	)					no IL
24	ST?					no IL
	Table 69 – Calls for IL language					
1a	Function block call with **non-formal** parameter list					no IL
1b	Function block call with **formal** parameter list					no IL
2	Function block call with **load/stor**e of standard input parameters					no IL
3a	Function call with **formal** parameter list					no IL
3b	Function call with **non-formal** parameter list					no IL
4a	Method call with **formal** parameter list					no IL
4b	Method call with **non-formal** parameter list					no IL
	Table 70 – Standard function block operators for IL language					
1	SR					no IL
2	RS					no IL
3	F/R_TRIG					no IL
4	CTU					no IL
5	CTD					no IL

6	CTUD					no IL
7	TP					no IL
8	TON					no IL
9	TOF					no IL
	Table 71 – Operators of the ST language					
1	Parentheses					no ST
2	Evaluation of result of function and method – if a result is declared					no ST
3	Dereference					no ST
4	Negation					no ST
5	Unary Plus					no ST
6	Complement					no ST
7	Exponentiation[b]					no ST
8	Multiply					no ST
9	Divide					no ST
10	Modulo					no ST
11	Add					no ST
12	Subtract					no ST
13	Comparison					no ST
14	Equality					no ST
15	Inequality					no ST
16a	Boolean AND					no ST
16b	Boolean AND					no ST
17	Boolean Exclusive OR					no ST
18	Boolean OR					no ST
	Table 72 – ST language statements					
1	**Assignment** Variable := expression;					
1a	Variable and expression of elementary data type					no ST
1b	Variables and expression of different elementary data types with implicit type conversion according Figure 11					no ST
1c	Variable and expression of user-defined type					no ST
1d	Instances of function block type					no ST
	Call					
2a [b]	Function call					no ST
2b [b]	Function block call and function block output usage					no ST
2c [b]	Method call					no ST
3	RETURN					no ST
	Selection					
4	IF ... THEN ... ELSIF ... THEN ... ELSE ...END_IF					no ST
5	CASE ... OF ... ELSE ...					no ST

	END_CASE					
	Iteration					
6	FOR ... TO ... BY ... DO ... END_FOR					no ST
7	WHILE ... DO ... END_WHILE					no ST
8	REPEAT ... UNTIL ... END_REPEAT					no ST
9 [a]	CONTINUE					no ST
10 [a]	EXIT an iteration					no ST
11	Empty Statement					no ST
	Table 73 – Graphic execution control elements					
	Unconditional jump					
1a	FBD language		✓			TRUE---->>LABELA
1b	LD language					noLD
	Conditional jump					
2a	FBD language		✓			
2b	LD language					no LD
	Conditional return					
3a	LD language					no LD
3b	FBD language		✓			
	Unconditional return					
4	LD language					no LD
	Table 74 – Power rails and link elements					
1	Left power rail (with attached horizontal link)					no LD
2	Right power rail (with attached horizontal link)					no LD
3	Horizontal link					no LD
4	Vertical link (with attached horizontal links)					no LD
	Table 75 - Contacts					
	Static contacts					
1	Normally open contact					no LD
2	Normally closed contact					no LD
	Transition-sensing contacts					
3	Positive transition-sensing contact					no LD
4	Negative transition-sensing contact					no LD
5a	Compare contact (typed)					no LD
5b	Compare contact (overloaded)					no LD
	Table 76 (73) - Coils					
	Momentary coils					
1	Coil					no LD
2	Negated coil					no LD

	Latched Coils					
3	Set (latch) coil					no LD
4	Reset (unlatch) coil .					no LD
	Transition-sensing coils					
8	Positive transition-sensing coil					no LD
9	Negative transition-sensing coil					no LD

B.2 Compliance Liste: Implementierungsspezifische Erweiterungen

Es gibt nur die folgenden Erweiterungen über die IEC hinaus [N1.1.3-Kap.5.1.c].

Bereich	Erweiterungen der IEC	Begründung
Datentypen	sicherheitsgerichtete Datentypen SAFEBOOL, SAFEINT, etc. für alle unterstützten IEC-Datentypen	Für PLCopen: Datentyp zur Unterscheidung sicherer Signale
Generische Datentypen / Funktionen	Jeder Datentyp SAFEX wird neben X in die Hierarchie der generischen Datentypen [N1.1.3-Figure 5] eingereiht. Das heisst, die generischen Funktionen (AND, ADD, SEL, EQ, etc.), die unter anderem auf Typ X definiert sind, sind analog auch auf Typ SAFEX definiert.	Für PLCopen: Generische Funktionen auf Typ X machen genauso Sinn auf Typ SAFEX. Es müssen keine neuen Funktionen "erfunden" werden.
	Zusätzliche Varianten der Funktion "AND", die nicht dem generischen Typschema entspricht: AND: BOOL x SAFEBOOL -> SAFEBOOL AND: SAFEBOOL x BOOL -> SAFEBOOL	Für PLCopen: "confirmation functionality" bzw. "enabeling function"
	Konvertierungsfunktionen A_TO_B sind generisch bzgl. der SAFE-Qualifikation: A_TO_B: A->B A_TO_B: SAFEA->SAFEB	Analogie zu generischen Funktionen: ADD auf INT und auf TIME wird als **eine** Funktion verstanden, weil das gleiche passiert; erspart dem Anwender, unterschiedliche Funktionsnamen hinzuschreiben (ADD_INT, ADD_TIME). Auch bei INT_TO_BOOL auf INT/BOOL wie auf SAFEINT/SAFEBOOL passiert das gleiche; es bringt keinen Vorteil, zwischen INT_TO_BOOL und SAFEINT_TO_SAFEBOOL zu differenzieren.

Bereich	Erweiterungen der IEC	Begründung
Implizite Konvertierungen	Daten vom Typ SAFEX können auf Variablen oder Eingänge vom Typ X zugewiesen werden ("SAFE-Polymorphie"). Unter anderem ist auf diese Weise der Aufruf der generischen Funktion ADD mit einem INT- und einem SAFEINT-Wert möglich. SAFEBOOL -> BOOL SAFEINT -> INT SAFEDINT -> DINT SAFETIME -> TIME SAFEBYTE -> BYTE SAFEWORD -> WORD SAFEDWORD -> DWORD	Für PLCopen; SAFEX-Daten und X-Daten sind nicht andersartige Daten, sondern Daten mit unterschiedlicher Integrität. Daten höherer Integrität können verwendet werden, wo niedrigere Integrität ausreicht.
	Daten vom Typ INT bzw. SAFEINT können auf Variablen oder Eingänge vom Typ DINT bzw. SAFEDINT zugewiesen werden ("INT-Polymorphie") INT -> DINT SAFEINT -> SAFEDINT	Der Wertebereich von INT bzw. SAFEINT ist enthalten im Wertebereich von DINT bzw. SAFEDINT.
FB-Typen	Statt der Standardfunktionsbausteine: **SF_XXX** Varianten mit SAFE-Varianten der Inputs/Outputs (bis auf Inputs mit Resetsemantik: RESET, LOAD bzw. IN)	Notwendig für Verarbeitung sicherer Signale mit Standard-FBs
POUs	POUs können qualifiziert werden, um ihren Sprachumfang oder ihre Verwendung einzuschränken: ■ Level: Basic, Extended, External ■ Singlecall ■ IOAPI-only	Level: zur expliziten Unterscheidung der PLCopen Programmierlevel. Singlecall: Kennzeichnet FBs, für die die PLCOpen-Regel "einmaliger Aufruf" gilt. IOAPI-only: reserviert FBs für Systemzwecke (E/A-Anbindung)
Variablendeklarationen	In externen FBs: neuer Modifizierer **SYSONLY**. In implizitem Code: neue Modifizierer **IOIN**, **IOOUT**, **IOAPI**, **SYSONLY**.	Für Safety-spezifische Variablenarten

Listen zusätzlich reservierter Schlüsselwörter

Tab. 31: Zusätzliche und vorsorglich reservierte Worte von CODESYS Safety

Deklarationen, Typkonstrukte	Datentypen	
IOAPI	SF_CDT	SAFEBIT
IOIN	SF_CTD_DINT	SAFEBOOL

Deklarationen, Typkonstrukte	Datentypen	
SAFBYTE	SF_CTD_LINT	SAFEBYTE
SYSONLY	SF_CTD_UDINT	SAFECHAR
	SF_CTD_ULINT	SAFEDATE
	SF_CTU	SAFEDATE_AND_TIME
	SF_CTU_DINT	SAFEDINT
	SF_CTU_LINT	SAFEDT
	SF_CTU_UDINT	SAFEDWORD
	SF_CTU_ULINT	SAFELDT
	SF_CTUD	SAFELINT
	SF_CTUD_DINT	SAFELREAL
	SF_CTUD_LINT	SAFELTIME
	SF_CTUD_UDINT	SAFELTOD
	SF_CTUD_ULINT	SAFELWORD
	SF_SR	SAFEREAL
	SF_F_TRIG	SAFESINT
	SF_R_TRIG	SAFESTRING
	SF_RS	
	SF_TOF	
	SF_TON	
	SF_TP	

Tab. 32: Standard IEC-Operatoren, die spezifisch für die Sprache IL sind, und von CODESYS über alle Sprachen hinweg reserviert werden

Implementierungsteil	
ANDN	R
CAL	RET
CALC	RETC
JMP	RETCN
JMPC	S
JMPCN	ST
LD	STN

Implementierungsteil	
LDN	XORN
ORN	

Tab. 33: Weitere zusätzliche reservierte Schlüsselwörter

Deklarationen, Typkonstrukte	Implementierungsteil	
PERSISTENT	ADR	__GETLTICK
POINTER	BITADR	__QUERYINTERFACE
PROPERTY	INDEXOF	__QUERYPOINTER
UNION END_UNION	INI	__NEW
VAR_STAT	SIZEOF	__DELETE
__COPY	TEST_AND_SET	__WAIT
__LAZY	BIT_TO_XXX	__TRY
__CRC	XXX_TO_BIT	__ENDTRY
__MAXOFFSET	ADD_BIT	__CATCH
__LOCALOFFSET	__XWORD	__FINALLY
__TYPEOF	__UXINT	__THROW
__VARINFO	__XINT	__BITOFFSET
__SYSTEM	__XSTRING	__CURRENTTASK
__POOL	__ISVALIREF	__CHECKLICENSE
__INIT	__FCALL	__CHECKLICENSEBIT
__CAST	__ADRINST	__CALLINITFUNCTION
	__REFADR	__LATECOMPILEDEXPR
	__MEMORYSET	

B.3 Compliance Liste: Implementierungsabhängige Parameter

Dieser Abschnitt listet die Werte der implementierungsabhängigen Parameter auf [N1.1.3-Kap.1.5 d].

"**n.a.**" kennzeichnet Fälle, bei denen das Sprachkonstrukt, auf das sich der Parameter bezieht, nicht zum Safety-Sprachumfang gehört.

Parameter	Implementierung
Maximum length of identifiers	1015 Bytes (UTF-8)
Maximum comment length	1015 Bytes (UTF-8)
Syntax and semantics of pragmas	gibt es nicht
Syntax and semantics for the use of the double-quote character when a particular implementation supports feature 4 but not feature 2 of table 6.	gibt es nicht
Range of values and precision of representation for variables of type TIME, DATE, TIME_OF_DAY and DATE_AND_TIME Precision of representation of seconds in types TIME, TIME_OF_DAY and DATE_AND_TIME	TIME-Werte gehen von 0 Sekunden bis 2^{32}-1 Millisekunden mit einer Präzision von 1 Millisekunde. Sekunden werden millisekundengenau repräsentiert.
Maximum number of enumerated values Maximum number of array subscripts Maximum array size Maximum number of structure elements Maximum structure size Maximum range of subscript values Maximum number of levels of nested structures	n.a. (keine Aufzählungstypen, keine Arrays, keine Strukturen, keine Bereichstypen)
Default maximum length of STRING and WSTRING variables Maximum allowed length of STRING and WSTRING variables	n.a. (keine Strings)
Initialization of system inputs	systemspezifisch (OEM)
Maximum number of variables per declaration	Keine Beschränkung
Effect of using AT qualifier in declaration of function block instances	n.a. (kein AT)
Warm start behavior if variable is declared as neither RETAIN nor NON_RETAIN	Alle Variablen sind NON_RETAIN
Information to determine execution times of program organization units	systemspezifisch (OEM)
Values of outputs when ENO is FALSE	n.a. (kein ENO)
Maximum number of function specifications	n.a. (keine anwenderdefinierten Funktionen)

Parameter	Implementierung
Maximum number of inputs of extensible functions	Keine Beschränkung
Effects of type conversions on accuracy Error conditions during type conversions	Error conditions: Fehler zur Laufzeit bei verschiedenen Konvertierungsfunktionen, wenn Ausgangswert ist nicht im Wertebereich des Zieltyps (siehe ↪ *„Zusammenfassung Laufzeitfehler“ auf Seite 371*) Effects: Konvertierung nach BOOL ergibt TRUE gdw. Ausgangswert ungleich 0. Nummerische Werte bleiben unverändert. Bitmuster werden um führende 0-Bits erweitert bzw. gekürzt.
Accuracy of numerical functions	Berechnungen erfolgen mit 32bit interner Genauigkeit für Zwischenwerte. - Bei Overflow (nach oben wie nach unten) erfolgt ein Vorzeichenwechsel. - Bei Underflow (im Fall von Division) wird zur nächst-größeren oder -kleinerern Ganzzahl gerundet: systemspezifisch (OEM)
Effects of type conversions between time data types and other data types not defined in table 35	Bei Konvertierungen wird ein TIME-Wert als die Anzahl seiner Millisekunden behandelt. Beispiele: `TIME_TO_INT(t#1s) = 1000` `INT_TO_TIME(1) = t#1ms`
Maximum number of function blocks	Keine Beschränkung
Specifications and instantiations	
Function block input variable assignment when EN is FALSE	n.a. (kein EN)
Pvmin, Pvmax of counters	Pvmin = 0, Pvmax = 16#7fff
Effect of a change in the value of a PT input during a timing operation	Erhöhung verlängert die Zeit bis zum Ablauf des Timer entsprechend. Erniedrigung verkürzt die Zeit. Wenn das verkürzte PT bereits verstrichen ist, läuft der Timer sofort ab.
Program size limitations	systemspezifisch (OEM)
Precision of step elapsed time Maximum number of steps per SFC	n.a. (kein SFC)
Maximum number of transitions per SFC and per step	n.a. (kein SFC)
Maximum number of action blocks per step	n.a. (kein SFC)
Access to the functional equivalent of the Q or A outputs	n.a. (kein SFC)
Transition clearing time Maximum width of diverge/ converge constructs	n.a. (kein SFC)
Contents of RESOURCE libraries	n.a.
Effect of using READ_WRITE access to function block outputs	n.a.

Parameter	Implementierung
Maximum number of tasks	1
Task interval resolution	systemspezifisch (OEM)
Maximum length of expressions	n.a. (kein ST)
Maximum length of statements	n.a. (kein ST)
Maximum number of CASE selections	n.a. (kein ST)
Value of control variable upon termination of FOR loop	n.a. (kein ST)
Restrictions on network topology	Die Grundstruktur eines FBD-Netzwerks ist ein Baum: Die Wurzel liegt rechts und fächert sich nach links auf. Nur Mehrfachzuweisungen fächern sich nach rechts auf. Einschränkung für den Anwender: ■ Keine expliziten Feedback-Schleifen. - Von einem FB-Aufruf f kann er auf grafische Weise nur einen Ausgang o1 (beliebig wählbar) auf den Eingang eines anderen Auftrufs weiterverschalten. Weitere Ausgänge o2 kann er nur textuell weiterverschalten, indem er „f.o2“ auf den Eingang legt. ■ Den ausgewählten Ausgang o1 eines FB-Aufrufs f kann er auf grafische Weise nur einmal auf den Eingang eines anderen Aufrufs schalten. o1 kann er nur textuell auf einen weiteren Eingang (des gleichen oder eines anderen Auftrufs) schalten, indem er „f.o1“ auf den Eingang legt ■ Den Ausgang eines Operator-Aufrufs kann er nur einmal (grafisch) auf den Eingang eines anderen Aufrufs weiterverschalten. Um das Operator-Ergebnis mehrfach zu verwenden, müsste der Anwender eine Zwischenvariable x einführen und den Ausgang des Operator-Aufrufs von der grafischen Weiterverschaltung abzweigen, um ihn auf x zuzuweisen, und dann „x“ auf den weiteren Eingang (des gleichen oder eines anderen Auftrufs) legen.
Evaluation order of feedback loops	Explizite Feedback-Loops nicht möglich. Implizite Feedback-Loops sind im Basic-Level verboten.

B.4 Compliance Liste: Error Conditions

Die Fehlerbedingungen aus [N1.1.3-Kap.5.1 e] sind auf verschiedene Arten abgedeckt:

- Die Fehlerbedingung wird beim Übersetzen oder zur Laufzeit geprüft.
- Die Fehlerbedingung bezieht sich auf Konstrukte, die gar nicht im Safety-Sprachumfang enthalten sind („n.a.“), was beim Editieren oder beim Übersetzen geprüft wird.
- Nicht diagnostiziert wird der Fehler, dass das Ergebnis numerischer Operationen [N1.1.3- Kap.6.6.2.5.8] und Zeit-Operationen [N1.1.3-Kap.6.6.2.5.12] ausserhalb des Datenbereichs liegt. Wie gefordert, wird darauf in einem eigenen Abschnitt ↯ *„Compliance Liste: Nicht erkannte Fehler “ auf Seite 370* hingewiesen und es werden die Fälle aufgelistet.

Fehlerbedingungen	Lösung
Nested comments	n.a (Kommentarfelder statt Kommentarzeichen
Ambiguous enumerated value	n.a. (kein Enum)
Value of a variable exceeds the specified subrange	n.a. (kein Subrange)
Missing configuration of an incomplete address specification ("*" notation)	n.a. (kein *)
Attempt by a program organization unit to modify a variable which has been declared CONSTANT	Buildfehler
Declaration of a variable as VAR_GLOBAL CONSTANT in a containing element having a contained element in which the same variable is declared VAR_EXTERNAL without the CONSTANT qualifier.	Buildfehler
Improper use of directly represented or external variables in functions	n.a. (keine Funktionen)
A VAR_IN_OUT variable is not “properly mapped”	n.a. (nur in implizitem Code)
Ambiguous value caused by a VAR_IN_OUT connection	n.a. (nur in implizitem Code)
Type conversion errors	Laufzeitfehler
Numerical result exceeds range for data type	nicht erkannt
Division by zero	Laufzeitfehler
N input is less than zero in a bit-shift function	n.a. (kein Shift)
Mixed input data types to a selection function	Buildfehler
Selector (K) out of range for MUX function	Laufzeitfehler
Invalid character position specified Result exceeds maximum string length ANY_INT input is less than zero in a string function	n.a. (keine Strings)

Fehlerbedingungen	Lösung
bei TIME-Funktionen: Result exceeds range for data type	nicht erkannt
No value specified for a function block instance used as input variable	n.a. (kein FB als Input)
No value specified for an in-out variable	n.a. (nur impliziter Code)
Zero or more than one initial steps in SFC network User program attempts to modify step state or time	n.a. (kein SFC)
Side effects in evaluation of transition condition	n.a. (kein SFC)
Action control contention error	n.a. (kein SFC)
Simultaneously true, non-prioritized transitions in a selection divergence Unsafe or unreachable SFC	n.a. (kein SFC)
Data type conflict in VAR_ACCESS	n.a. (kein VAR_ACCESS)
A task fails to be scheduled or to meet its execution deadline	Laufzeitfehler
Bei Instruction List: Numerical result exceeds range for data type Current result and operand not of same data type	n.a. (kein IL)
Bei ST: Division by zero	n.a. (kein ST)
Bei ST: Numerical result exceeds range for data type	n.a. (kein ST)
Bei ST: Invalid data type for operation	n.a. (kein ST)
Return from function without value assigned	n.a. (keine Funktionen)
Bei While und Repeat: Iteration fails to terminate	n.a. (kein ST)
Same identifier used as connector label and element name	n.a. (kein Connector) Uninitialized feedback
Uninitialized feedback variable	Buildfehler

Compliance Liste: Nicht erkannte Fehler

Dieser Abschnitt listet in einem eigenen Abschnitt die Fehler auf, die nicht erkannt oder nicht gemeldet werden.

Level	Fehlerbedingung
Extended	Numerical result exceeds range for data type
Extended	bei TIME-Funktionen: Result exceeds range for data type

Zusammenfassung Laufzeitfehler

Level	Sprachfeature	Laufzeitfehler bei
Extended	DIV	Division durch 0
Extended	MUX	Aufruf mit erstem Input mit negativem Wert oder mit Wert N größer als die Anzahl der Eingänge minus 1. Zum Beispiel MUX(2, 16#8000, 16#8001)
Extended	DINT_TO_INT, TIME_TO_DINT, TIME_TO_INT, DINT_TO_TIME, INT_TO_TIME, DINT_TO_WORD, TIME_TO_WORD, DINT_TO_BYTE, INT_TO_BYTE, TIME_TO_BYTE, WORD_TO_BYTE	Ausgangswert ist nicht im Wertebereich des Zieltyps: Bei Konvertierung zwischen zwei ANY_MAGNITUDE Typen muss der numerische Ausgangswert im Wertebereich des Zieltyps liegen (wobei TIME-Werte als Anzahl von Millisekunden gerechnet werden). Bei Konvertierung von/zu Bitstring-Typen muss das Bitmuster des Ausgangswerts ein Bitmuster der Zieltyps sein. Beispiele: DINT_TO_INT(16#0000FFFF), weil 2^{16}-1 kein INT-Wert ist, ebenso DINT_TO_TIME(-1), weil es keine negative TIME-Werte gibt TIME_TO_DINT(t#365d), weil 365 Tage = 3,153,600,000ms = 16#BBF81E00 ist, und damit größer als die größte DINT-Zahl 2^{31}-1 = 16#7FFFFFFF INT_TO_BYTE(-1), da BYTE nur 0 bis 255 umfasst, WORD_TO_BYTE(0xFFFF), da BYTE nur bis 0xFF geht.