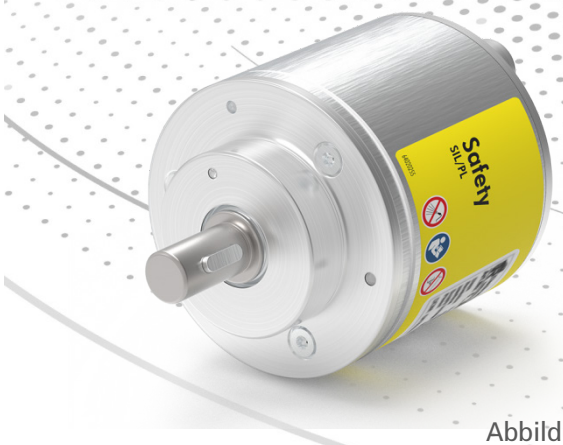
Absolut Encoder CD_582_-EPN
PROFINET IO / PROFIsafe

Parametrierung mit
ABB AC500-S PM583/SM560
Steuerungssystemen

CDV582



CDS582 / CDH582

Abbildungen ähnlich

_**Sicherheitsprogramm erstellen**
- Konfigurationsbeispiel
_**Zugriff auf den sicherheitsgerichteten Datenkanal**
_**Festlegen der Parameter / CRC-Berechnung**

Technische Information

TR - ECE - TI - DGB - 0366 - 01 04/24/2020

TR-Electronic GmbH
D-78647 Trossingen
Eglishalde 6
Tel.: (0049) 07425/228-0
Fax: (0049) 07425/228-33
E-mail: info@tr-electronic.de
http://www.tr-electronic.de

Urheberrechtsschutz

Änderungsvorbehalt

Jegliche Änderungen, die dem technischen Fortschritt dienen, vorbehalten.

Dokumenteninformation

Ausgabe-/Rev.-Datum:	04/24/2020
Dokument-/Rev.-Nr.:	TR - ECE - TI - DGB - 0366 - 01
Dateiname:	TR-ECE-TI-DGB-0366-01.docx
Verfasser:	KUC

Schreibweisen

Kursive oder **fette** Schreibweise steht für den Titel eines Dokuments oder wird zur Hervorhebung benutzt.

`Courier`-Schrift zeigt Text an, der auf dem Display bzw. Bildschirm sichtbar ist und Menüauswahlen von Software.

″ < > ″ weist auf Tasten der Tastatur Ihres Computers hin (wie etwa <RETURN>).

Marken

PROFIBUS™, PROFINET™ und PROFIsafe™, sowie die zugehörigen Logos, sind eingetragene Warenzeichen der PROFIBUS Nutzerorganisation e.V. (PNO).

Inhaltsverzeichnis

Änderungs-Index

Änderung	Datum	Index
Erstausgabe	16.04.20	00
Allgemeine Anpassungen nach Rückmeldung von ABB	24.04.20	01

1 Allgemeines

Die vorliegende „Technische Information“ beinhaltet folgende Themen:

- Festlegen der Parameter / CRC-Berechnung
- Sicherheitsprogramm erstellen
- Zugriff auf den sicherheitsgerichteten Datenkanal

Die „Technische Information“ kann separat angefordert werden.

1.1 Geltungsbereich

Diese „Technische Information“ gilt ausschließlich für folgende Mess-System-Baureihen mit ***PROFINET IO*** Schnittstelle und ***PROFIsafe*** Profil in Verbindung mit einer ABB AC500-S Steuerung der Serie PM583/SM560:

- CDV-582
- CDS-582
- CDH-582

Die Produkte sind durch aufgeklebte Typenschilder gekennzeichnet und sind Bestandteil einer Anlage.

Es gelten somit zusammen folgende Dokumentationen:

- ABB Handbuch *AC500-S Sicherheitshandbuch V1.1.0* (Dokumentbestellnummer: *3ADR025091M0107*),
- ABB Handbuch *AC500 PLC - System Assembly and Device Specifications for AC500 V2 Products* (Dokumentbestellnummer: 3ADR010121),
- ABB-Online-Hilfe zu *Automation Builder V2.2.4*
- siehe Kapitel „Mitgeltende Dokumente“ im Sicherheitshandbuch www.tr-electronic.de/f/TR-ECE-BA-D-0142
- und diese optionale „Technische Information“

2 Sicherheitshinweise

2.1 Symbol- und Hinweis-Definition

⚠ GEFAHR	bedeutet, dass Tod oder schwere Körperverletzung eintreten wird, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
⚠WARNUNG	bedeutet, dass Tod oder schwere Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
⚠VORSICHT	bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
ACHTUNG	bedeutet, dass ein Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.
	bezeichnet wichtige Informationen bzw. Merkmale und Anwendungstipps des verwendeten Produkts.

2.2 Organisatorische Maßnahmen

Das mit Tätigkeiten am Mess-System beauftragte Personal muss vor Arbeitsbeginn das Sicherheitshandbuch TR-ECE-BA-D-0142, insbesondere das Kapitel "Grundlegende Sicherheitshinweise", gelesen und verstanden haben.

2.3 Personalqualifikation

Die Konfiguration des Mess-Systems darf nur von qualifiziertem Fachpersonal durchgeführt werden, siehe ABB Sicherheitshandbuch.

2.4 Nutzungsbedingungen der Softwarebeispiele

⚠WARNUNG

Für die fehlerfreie Funktion des Sicherheitsprogrammes und der Anwendungsbeispiele übernimmt die Firma TR-Electronic GmbH keine Haftung und keine Gewährleistung.

ACHTUNG

Die zum Download angebotenen Softwarebeispiele dienen ausschließlich zu Demonstrationszwecken, der Einsatz durch den Anwender erfolgt auf eigene Gefahr.

3 GSDML

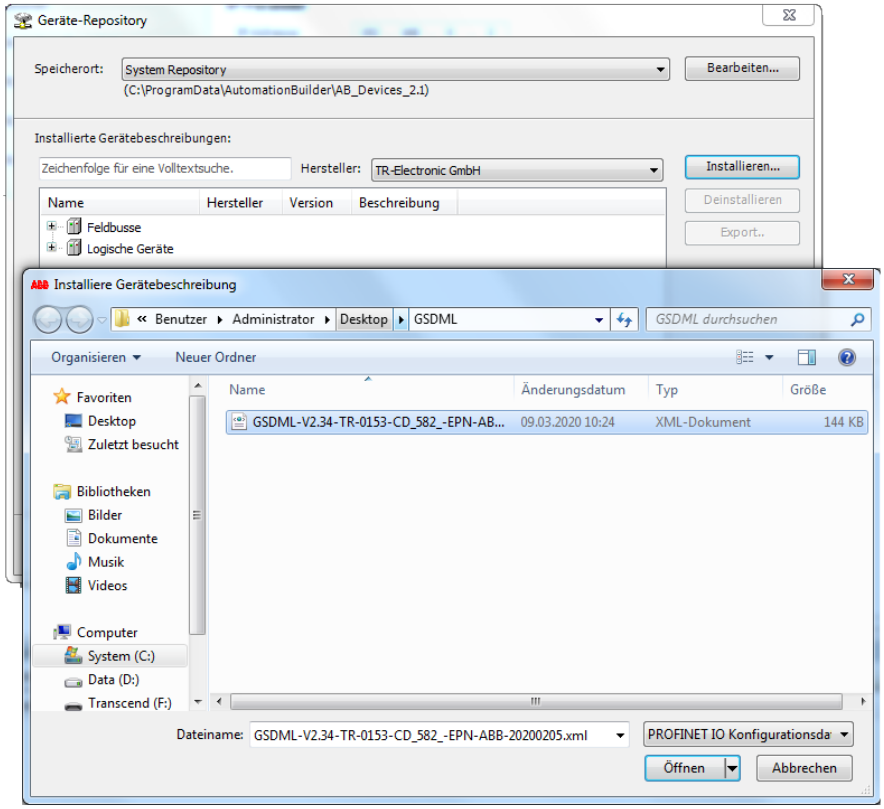
Für die Konfiguration des CD_582_-EPN mit der ABB Projektierungssoftware Automation Builder V2.1 muss eine auf die Projektierungssoftware angepasste Gerätebeschreibungsdatei (GSDML) verwendet werden.

Die GSDML trägt die Kennung „ABB“ im Dateinamen, z.B. „GSDML-V2.34-TR-0153-CD_582_-EPN-ABB-xxxxxxxx.xml“.

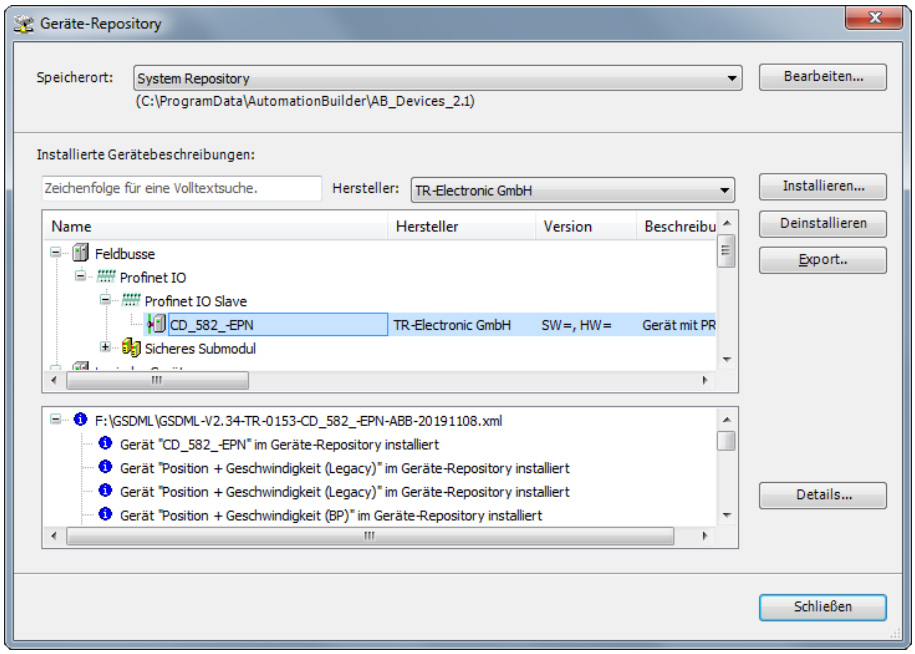
Download unter www.tr-electronic.de/f/ZIP/TR-ECE-ID-MUL-0058

Die Projektierung ist nur mit der angepassten GSDML möglich.

Wenn der CD_582_-EPN in einem Automation-Builder-Projekt projektiert werden soll, muss die GSDML des Geräts im Geräte-Repository installiert werden.
Der Editor des Geräte-Repositorys kann über die Menüleiste des Automation-Builder unter dem Menüpunkt Tools -> Geräte-Repository geöffnet werden. Die GSDML des CD_582_-EPN muss in das System-Repository installiert werden, damit Geräte dieses Typs in Projekten konfiguriert werden können.



Nach der Installation ist der CD_582_-EPN als Gerät unter den „PROFINET IO Slaves“ verfügbar.



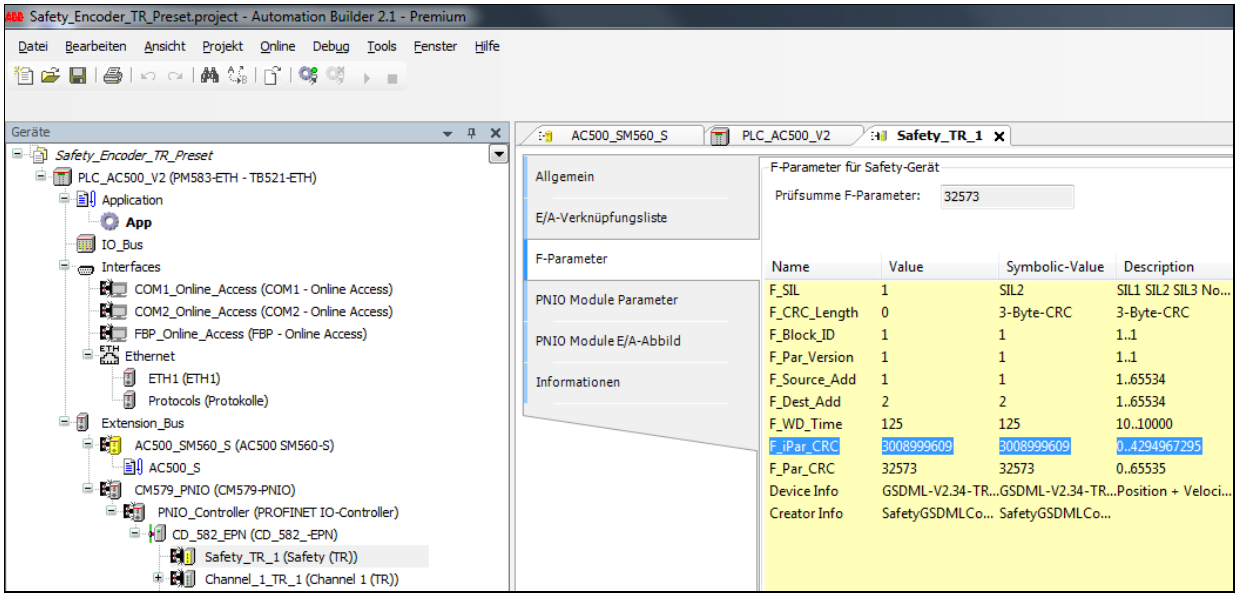
4 Festlegen der Parameter / CRC-Berechnung

Es ist zweckmäßig, die bekannten Parameter schon vor der Projektierung im F-Host festzulegen, damit diese bei der Projektierung bereits berücksichtigt werden können.

Nachfolgend wird die Vorgehensweise in Verbindung mit der ABB Projektierungssoftware `Automation Builder V2.1` und dem Optionspaket `Safety DM220-FSE` beschrieben.
Die zur CRC-Berechnung erforderliche Software `TR TCI Device Tool` kann in Kap.: 7 „Software-, Beispiel- und Bibliotheken-Download“ auf Seite 56 herunter geladen werden.
Für die Installation und Benutzung der Software `TR TCI Device Tool` ist die Anleitung TR-ECE-TI-DGB-0327 zu beachten.

4.1 iParameter

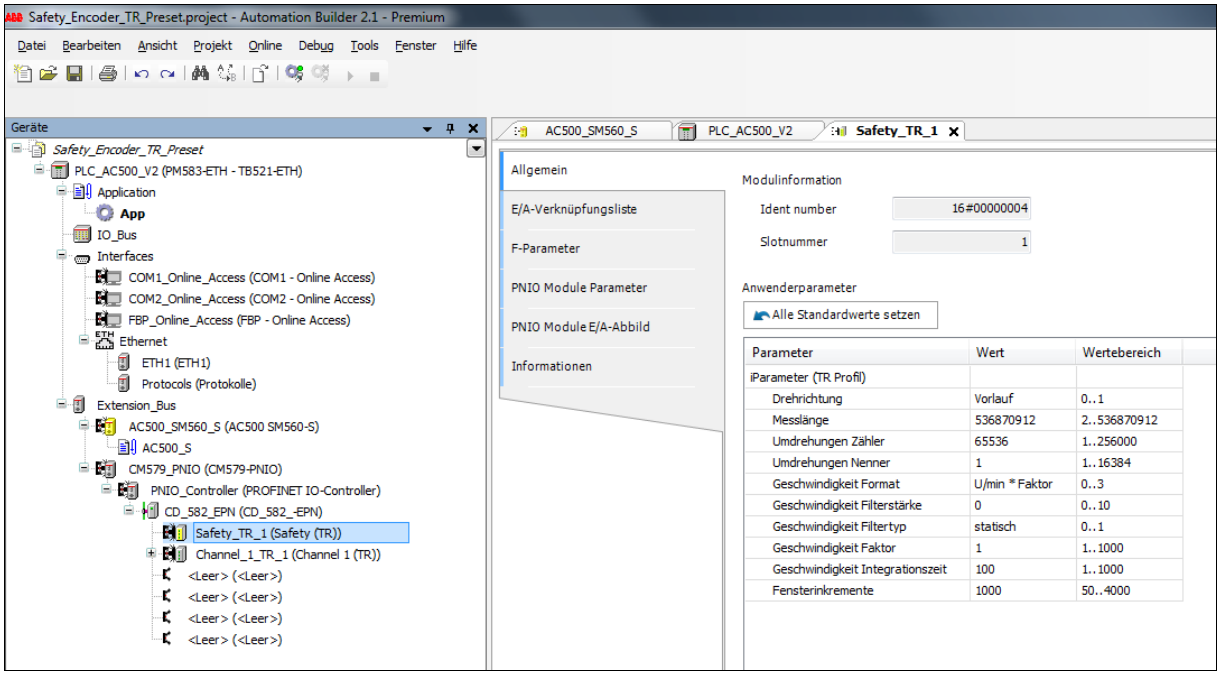
Die iParameter sind in der Standardeinstellung bereits mit sinnvollen Werten voreingestellt und sollten nur dann verändert werden, wenn die Automatisierungsaufgabe dies ausdrücklich erfordert. Zur sicheren Übertragung der individuell eingestellten iParameter ist eine CRC-Berechnung erforderlich. Diese muss bei Änderung der voreingestellten iParameter über das TR-Programm „TR TCI Device Tool“ durchgeführt werden. Die so berechnete Checksumme entspricht dem F-Parameter `F_iPar_CRC`. Dieser muss bei der Projektierung des Mess-Systems in das Feld `F_iPar_CRC` eingetragen werden. Das Feld `F_iPar_CRC` ist in der Bearbeitungsansicht des konfigurierten Safety-Moduls unter dem Reiter `F-Parameter` zu finden, siehe auch Kapitel „Einstellen der iParameter“ auf Seite 30.



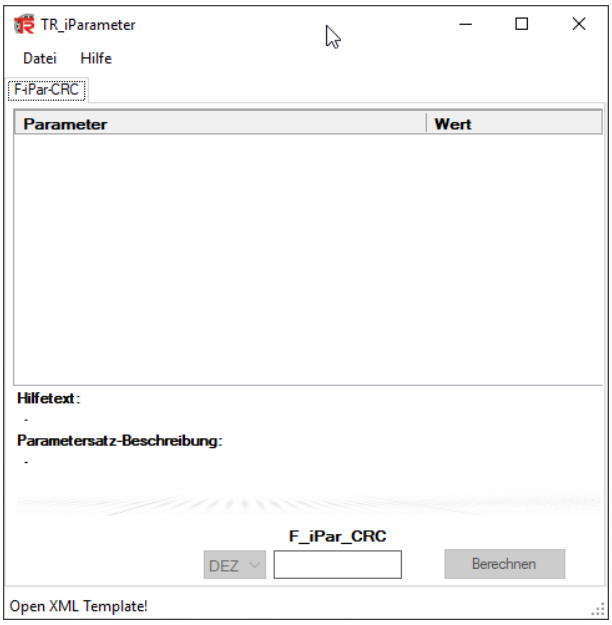
4.1.1 CRC-Berechnung über die iParameter

Für das nachfolgende Beispiel einer CRC-Berechnung werden die voreingestellten Standardwerte verwendet.

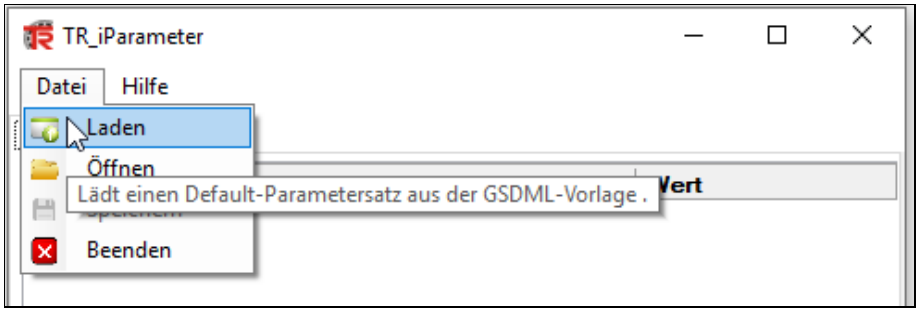
- Um die iParameter zu ändern, wird die `Bearbeitungsansicht` des Safety-Moduls in der IO-Konfiguration des Mess-Systems geöffnet. Danach editiert der Anwender die iParameter des Moduls in der Tabelle des Reiters `Allgemein`.



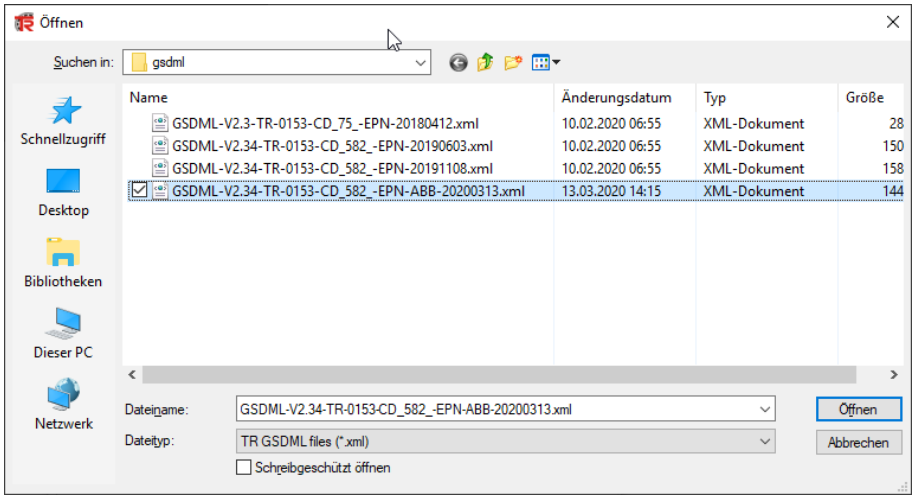
- Für die CRC-Berechnung wird das Programm `TR TCI Device Tool` gestartet. Das Programm `TR TCI Device Tool` ist eine Standalone-Windows-Anwendung, die außerhalb und unabhängig des `ABB Automation Builder` gestartet werden muss.



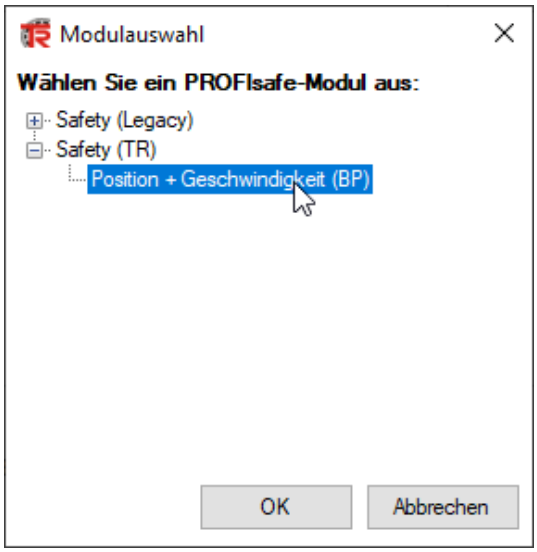
➢ Nach dem Start des `TR TCI Device Tool` muss der Anwender die GSDML des `CD_582_-EPN` laden. Hierzu wählt der Benutzer den Menüpunkt `Datei -> Laden` aus.



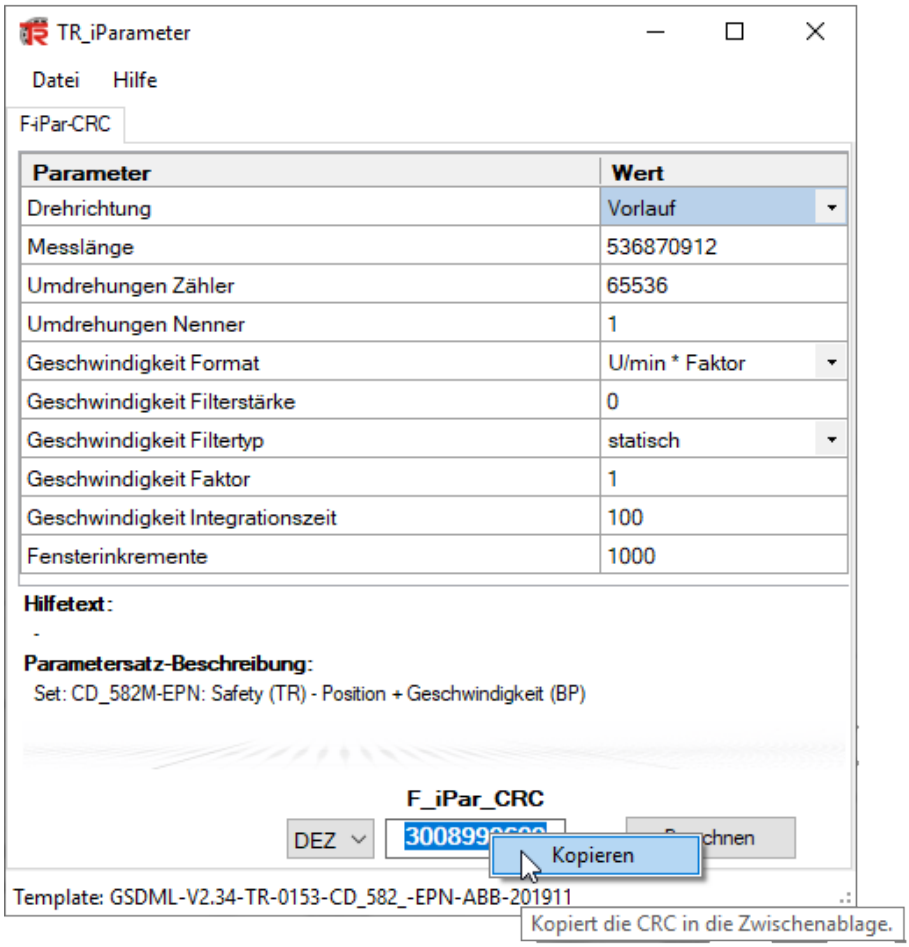
➢ Im Dateiauswahldialog wählt der Anwender die angepasste GSDML als Vorlage aus und öffnet diese.



➢ Die Zusammensetzung des Parametersatzes wird durch das zu konfigurierende PROFIsafe-Modul bestimmt. Der Anwender wählt gemäß des gewünschten IO-Profils das Modul entsprechend aus. Innerhalb dieser Anleitung wird immer das TR-Profil projektiert.

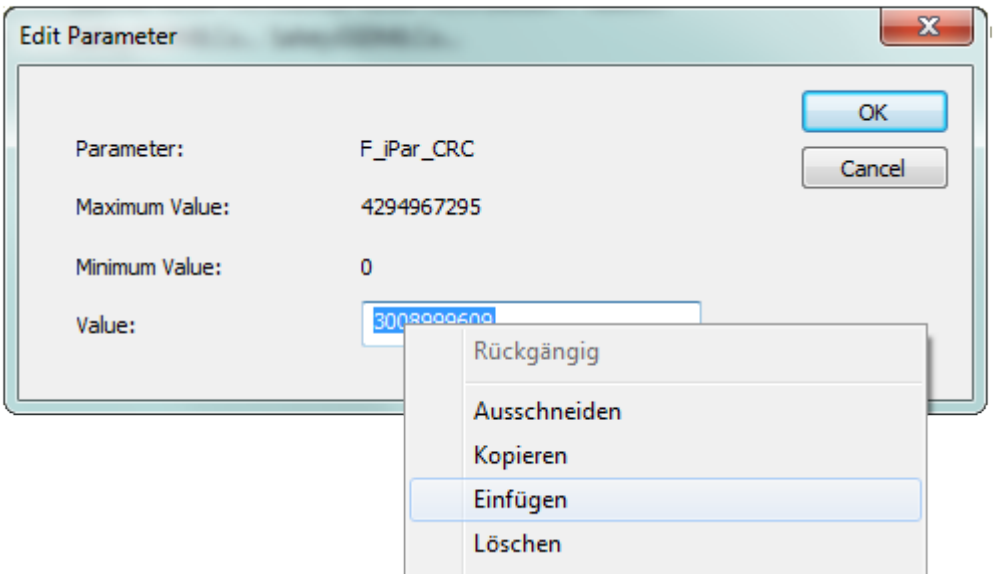


➢ Nach dem Öffnen des Parametersatzes die Auswahlschaltfläche für das CRC-Zahlenformat auf den Wert `DEZ` stellen und die Schaltfläche `Berechnen` betätigen.



Die Einstellungen innerhalb der Tabelle müssen mit den Einstellungen im Projektierungs-Tool übereinstimmen.

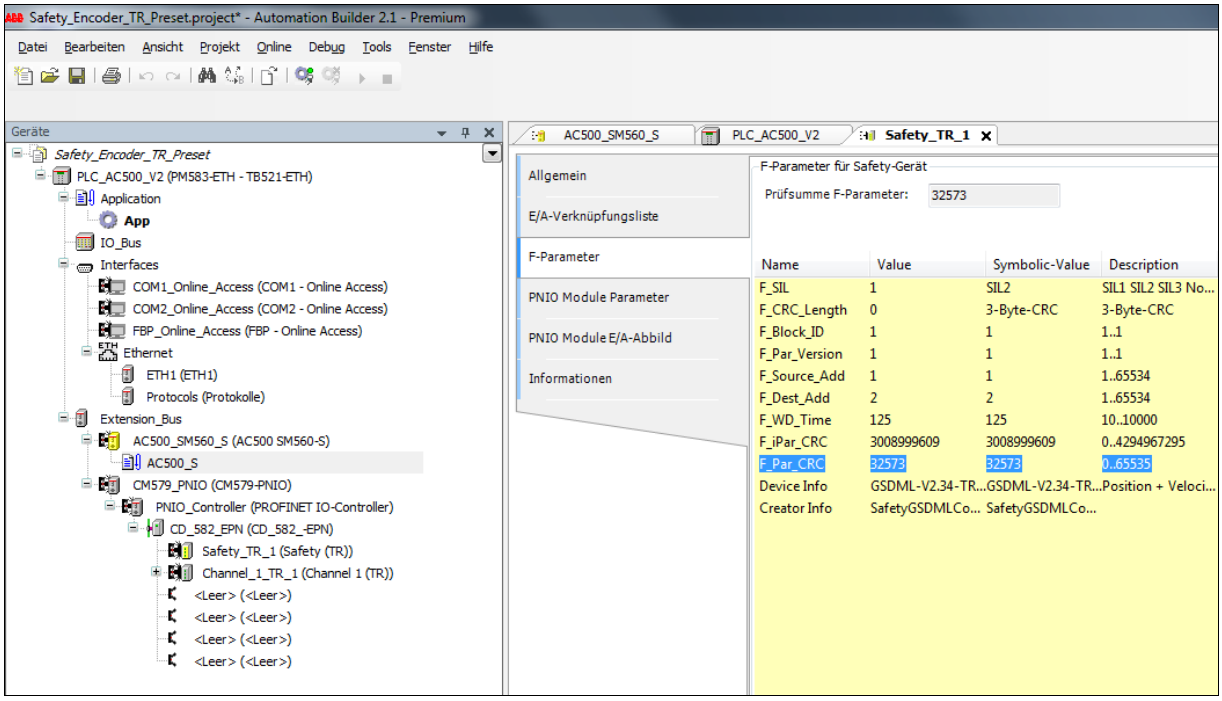
➢ Den berechneten Wert mit der rechten Maustaste anwählen und in die Zwischenablage kopieren. Überwechseln in den `Automation Builder`. Dort in der `Bearbeitungsansicht` des Safety-Moduls unter dem Reiter `F-Parameter` das Feld `F_iPar_CRC` per Doppel-Click anwählen und im darauffolgenden Bearbeitungsdialog den berechneten Wert einfügen.



Jede Parameteränderung erfordern eine erneute `F_iPar_CRC`-Berechnung im `TR TCI Device Tool` und ein erneuter Übertrag in das Projektierungs-Tool. Sind bereits ein Sicherheitsprogramm und eine entsprechende Konfiguration vorhanden, müssen die Konfigurationsdaten neu generiert werden. Der neue `F_iPar_CRC`-Wert und die geänderten Parameter sind bei der Projektierung im `Automation Builder` einzutragen. Siehe Kap.: 5.3.1 „Einstellen der iParameter“ auf Seite 30 und Kap.: 5.3.2 „Einstellen der F-Parameter“ auf Seite 31.

4.2 F-Parameter

Bis auf die `F_Dest_Add` sind die F-Parameter in der Standardeinstellung bereits mit sinnvollen Werten voreingestellt und sollten nur dann verändert werden, wenn die Automatisierungsaufgabe dies ausdrücklich erfordert. Zur sicheren Übertragung der individuell eingestellten F-Parameter ist eine CRC erforderlich, welche vom `Automation Builder` automatisch berechnet wird. Diese Checksumme entspricht dem F-Parameter `F_Par_CRC`, welcher bei der Projektierung des Mess-Systems in der `Bearbeitungsansicht` des Safety-Moduls im Reiter `F-Parameter` angezeigt wird. Siehe auch Kapitel „Einstellen der F-Parameter“ auf Seite 31.



4.2.1 Nicht einstellbare F-Parameter

Die nachfolgend aufgeführten F-Parameter werden vom Mess-System bzw. vom F-Host verwaltet und können deshalb nicht manuell verändert werden:

- F_CRC_Length: 3-Byte-CRC
- F_Block_ID: 1
- F_Par_Version: 1 (V2-mode)

4.2.2 Einstellbare F-Parameter

Bei den folgenden Parametern wird davon ausgegangen, dass diese mit ihren Standardwerten belegt sind:

- F_SIL: SIL2
- F_Source_Add: 1 (Adresse F-Host)
- F_Dest_Add: 1 (Adress-Schalter)
- F_WD_Time: 125
- F_iPar_CRC: 3008999609 (Berechnung mittels TR TCI Device Tool)

Nach jeder Parameteränderung berechnet der `Automation Builder` einen neuen `F_Par_CRC`-Wert, welcher wie oben dargestellt, eingetragen und angezeigt wird. Sind bereits ein Sicherheitsprogramm und eine entsprechende Konfiguration vorhanden, müssen die Konfigurationsdaten neu generiert werden.

5 Steuerungsprogramm erstellen - Konfigurationsbeispiel

Dieses Kapitel beschreibt die Vorgehensweise bei der Erstellung des Sicherheitsprogramms mit Verwendung der ABB Projektierungssoftware `Automation Builder V2.1` und dem Safety-Optionspaket `DM220-FSE`.

Das Sicherheitsprogramm wird mit dem `CoDeSys-Programmeditor (gelber Hintergrund)` im `Automation Builder` erstellt. Hierfür muss ein AC500 SM560-S-Knoten im Konfigurationsbaum des Projekts enthalten sein. Die Programmierung der fehlersicheren PRGs, FBs und FUNs erfolgt in den IEC61131-Programmiersprachen `ST`, `FUP` oder `KOP`. In dem von ABB mitgelieferten Safety-Optionspaket `DM220-FSE` stehen dem Anwender fehlersichere Applikationsbausteine zur Verfügung, welche im Sicherheitsprogramm verwendet werden können.

Bei der Generierung des Sicherheitsprogramms werden automatisch Sicherheitsprüfungen durchgeführt und zusätzliche fehlersichere Bausteine zur Fehlererkennung und Verarbeitung des PROFIsafe-Telegramms eingebaut. Damit wird sichergestellt, dass Ausfälle und Fehler erkannt und entsprechende Reaktionen ausgelöst werden, die das F-System im sicheren Zustand halten oder es in einen sicheren Zustand überführen.

In der F-CPU SM560-S läuft ausschließlich das Sicherheitsprogramm (Safety-Applikation) ab; das Standard-Anwenderprogramm (Non-Safety-Applikation) läuft hingegen auf der Zentralbaugruppe AC500 PM583-ETH. Die Programmierung der Non-Safety-Applikation erfolgt über einen separaten `CoDeSys-Programmeditor (weißer Hintergrund)`. Hierzu muss der Anwender einen separaten Applikationsknoten in das Projekt einfügen. Die Konfigurationsdaten beider Programme müssen konsistent gehalten werden, weil alle IO-Daten über die Zentralbaugruppe laufen und von dieser an die F-CPU weitergeleitet werden.

Ein Zugriff auf das fehlersichere Prozessabbild der PROFIsafe-Eingänge und PROFIsafe-Ausgänge aus dem Non-Safety-Programm heraus ist durch entsprechende E/A-Verknüpfungen möglich.

Zugriffschutz

Der Zugang zum F-System des `Automation Builder` ist durch eine Passwortabfrage gesichert, das sowohl für die IO-Konfiguration als auch für das Sicherheitsprogramm gilt.

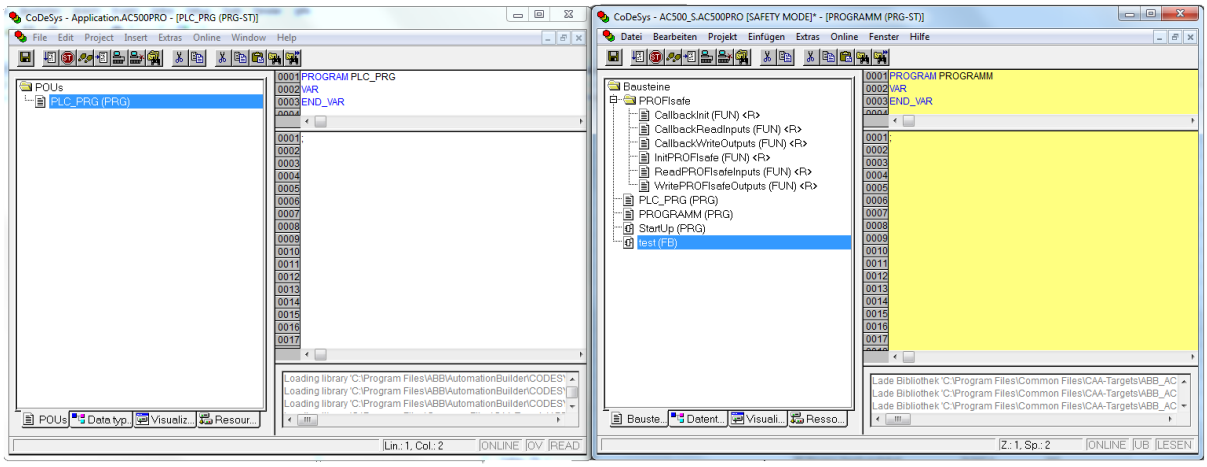


Abbildung 1: CoDeSys-Programmeditor (weiß) und Safety-CoDeSys-Programmeditor (gelb)

5.1 Voraussetzungen

⚠WARNUNG ***Gefahr der Außerkraftsetzung der fehlersicheren Funktion durch unsachgemäße Projektierung des Sicherheitsprogramms!***

- Die Erstellung des Sicherheitsprogramms darf nur in Verbindung mit der von ABB zur Software bzw. Hardware mitgelieferten Systemdokumentation erfolgen.
- Eine umfassende Dokumentation zum Thema „Projektieren und Programmieren“ einer sicheren Steuerung liefert die Firma ABB in ihrem Handbuch ***ACC500-S Sicherheitshandbuch V1.1.0***, Dokumentbestellnummer: ***3ADR025091M0107***. Diese Dokumentation ist Teil der Produktdokumentation und steht online zum Download zur Verfügung.
- Nachfolgende Beschreibungen beziehen sich auf den reinen Ablauf, ohne dabei die Hinweise aus dem ABB-Handbuch mit zu berücksichtigen.
  Die im ABB-Handbuch gegebenen Informationen, Hinweise, insbesondere die Sicherheitshinweise und Warnungen, sind daher zwingend zu beachten und einzuhalten.
- Die aufgezeigte Projektierung ist als Beispiel aufzufassen. Der Anwender ist daher verpflichtet, die Verwendbarkeit der Projektierung für seine Applikation zu überprüfen und anzupassen. Dazu gehören auch die Auswahl der geeigneten sicherheitsgerichteten Hardwarekomponenten, sowie die notwendigen Softwarevoraussetzungen.

Für das AC500-S-Konfigurationsbeispiel benutzte Software-Komponenten:

- Automation Builder V2.1
- Safety-Add-on DM220-FSE

Für das AC500-S-Konfigurationsbeispiel benutzte Hardware-Komponenten der AC500-S-Serie:

- Profilschiene TB521-ETH A2 (1SAP112100R0270)
- CPU PM583-ETH A9 (1SAP140300R0271)
- F-CPU SM560-S A5 (1SAP280000R0001)
- Profinet-IO-Controller CM579-PNIO (1SAP170901R0101)

Für das AC500-S-Konfigurationsbeispiel benutzte Hardware-Komponente der CD_582M-Serie:

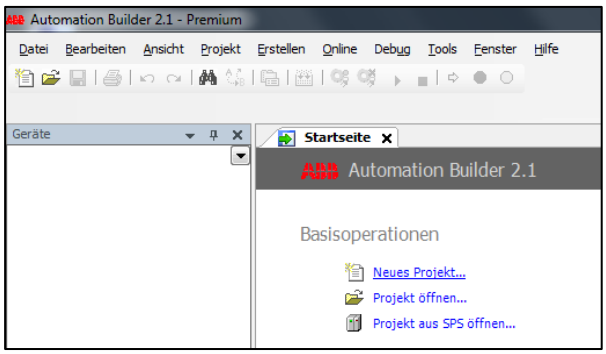
- CDH582M-00002 (CDH582M*8192/65536 EPN NTS 12H7 + FS2)

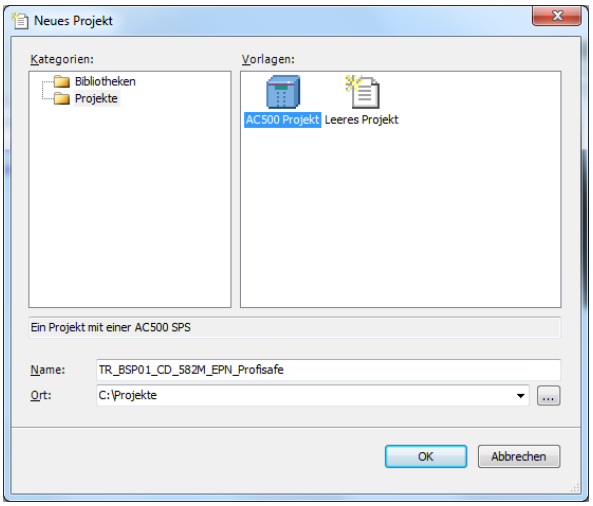
Abbildung 2: Hardware-Komponenten der AC500-S-Serie von ABB

Abbildung 3: Hardware-Komponente der CD_582M-Serie von TR-Electronic

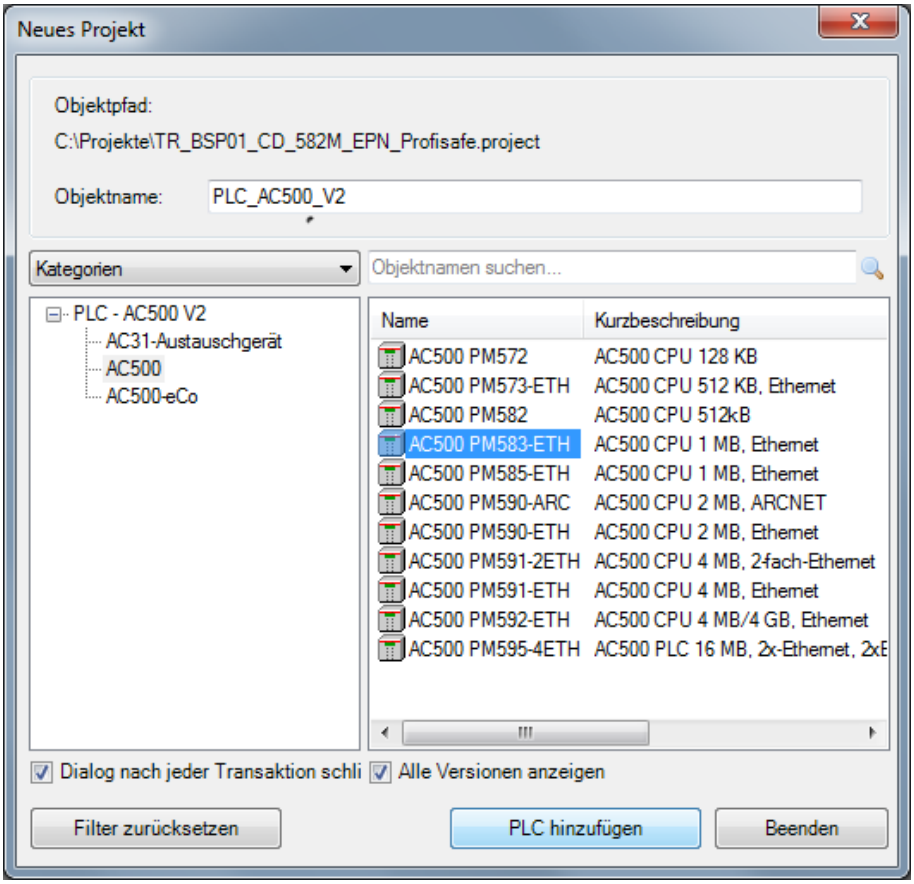
5.2 Hardware-Konfiguration

➢ Den ABB `Automation Builder` V2.1 starten und ein neues `AC500 Projekt` anlegen.

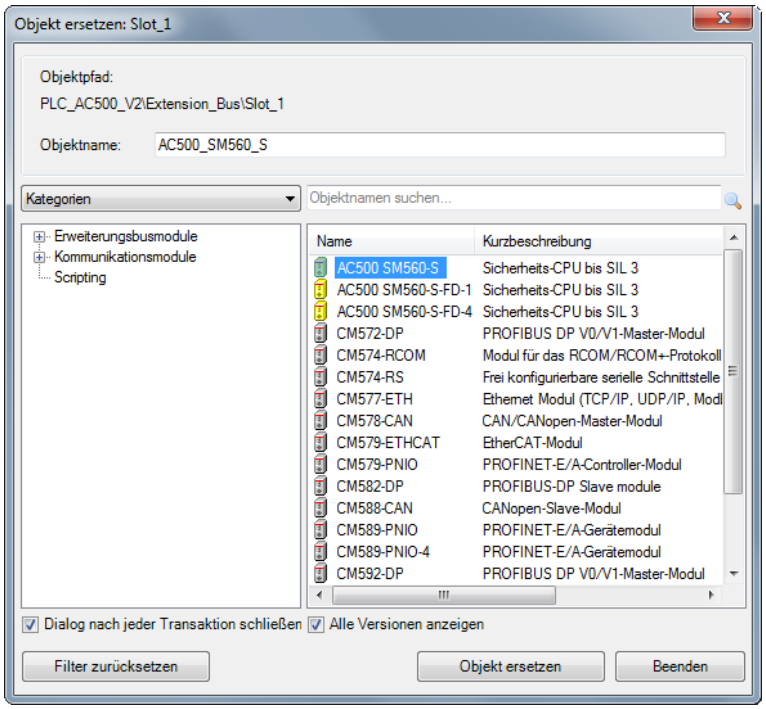




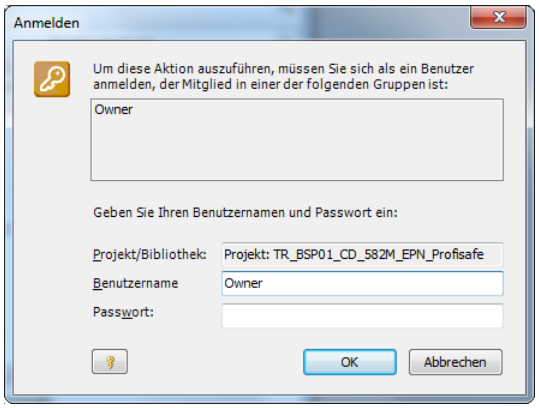
➢ Als PLC-Kopfstation die `AC500 CPU PM583-ETH` wählen und dem Projekt hinzufügen.



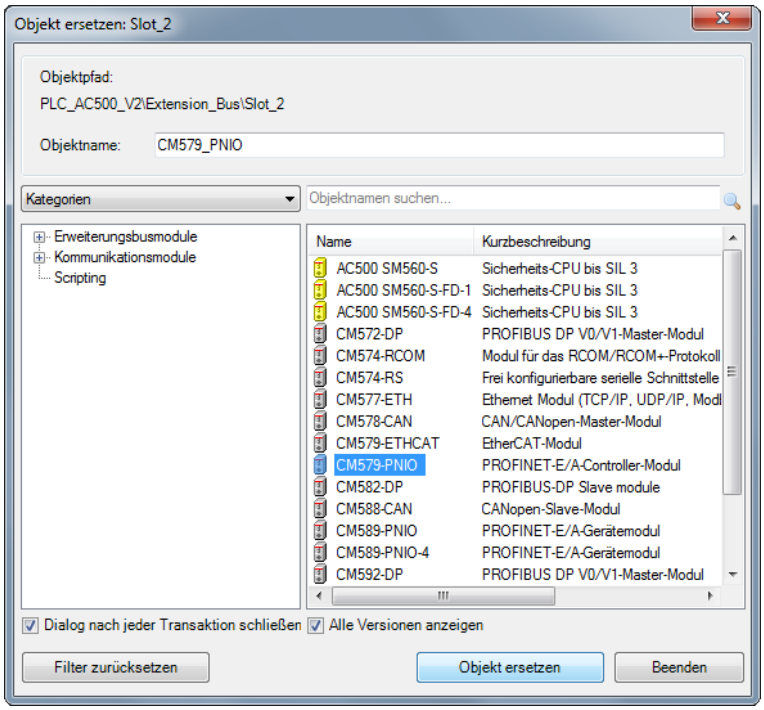
- Nach der Auswahl der PLC-Kopfstation baut der `Automation Builder` einen leeren Gerätekonfigurationsbaum in der Geräteansicht auf der linken Seite des Hauptfensters auf.
- Unter dem Zweig `Extension_Bus` müssen die beiden fehlenden ABB-Hardware-Komponenten eingefügt werden, die auf der Profilschiene montiert sind. Zum Einfügen der beiden Objekte muss aus dem Kontextmenü die Option `Objekt hinzufügen` gewählt werden.
- Im Extension – Slot 1 soll die F-CPU `SM560-S` hinzugefügt werden.



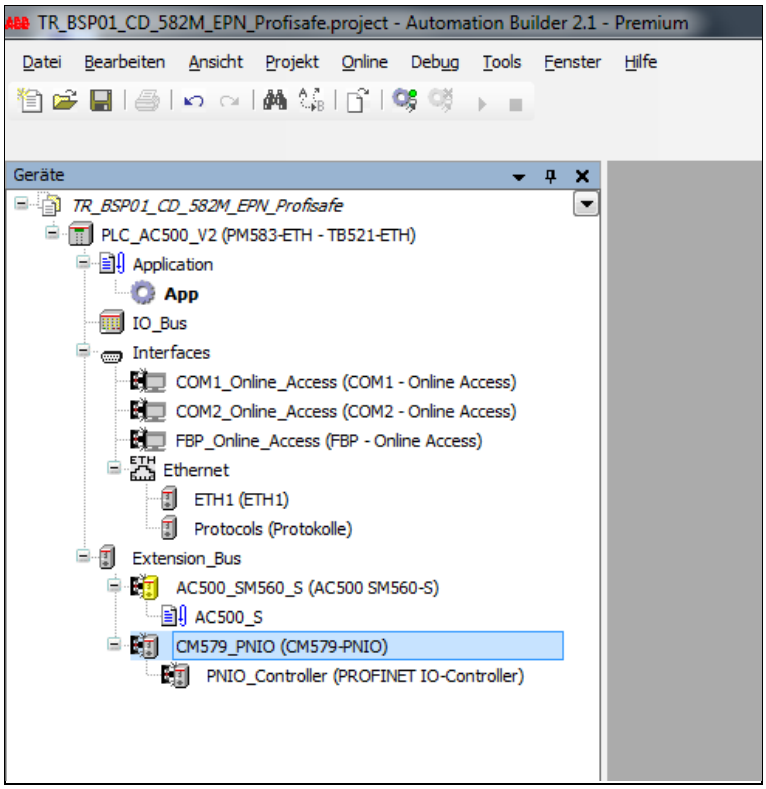
- Wenn die F-CPU in den Konfigurationsbaum eingefügt wird, fragt der `Automation Builder` nach einem Benutzernamen und dem entsprechenden Passwort. Im Kontext dieses Beispiels wird immer der Benutzer „Owner“ und ein Leerstring als Passwort verwendet.



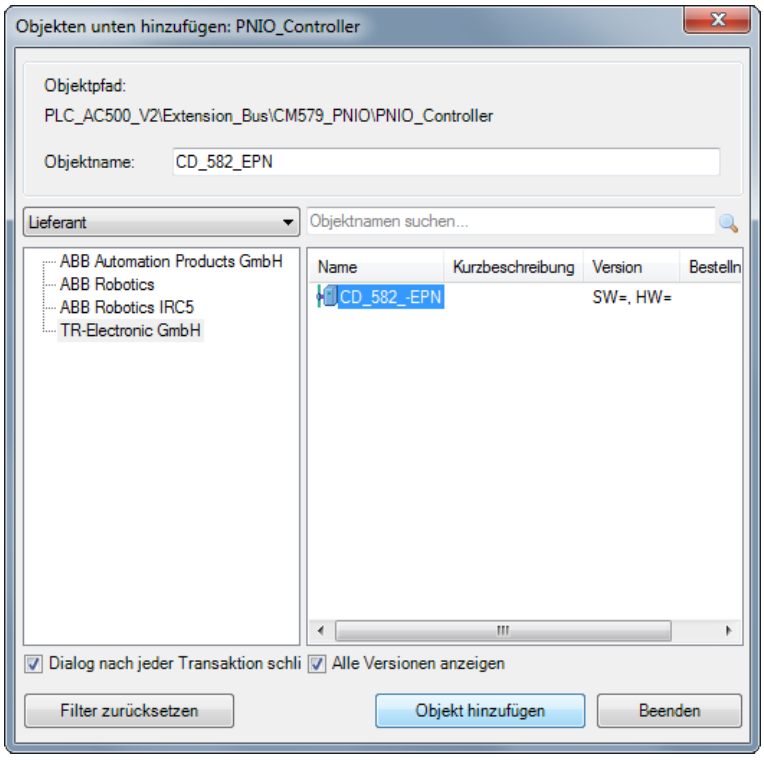
➢ Im Extension – Slot 2 soll das Kommunikationsmodul `CM579-PNIO` eingefügt werden.



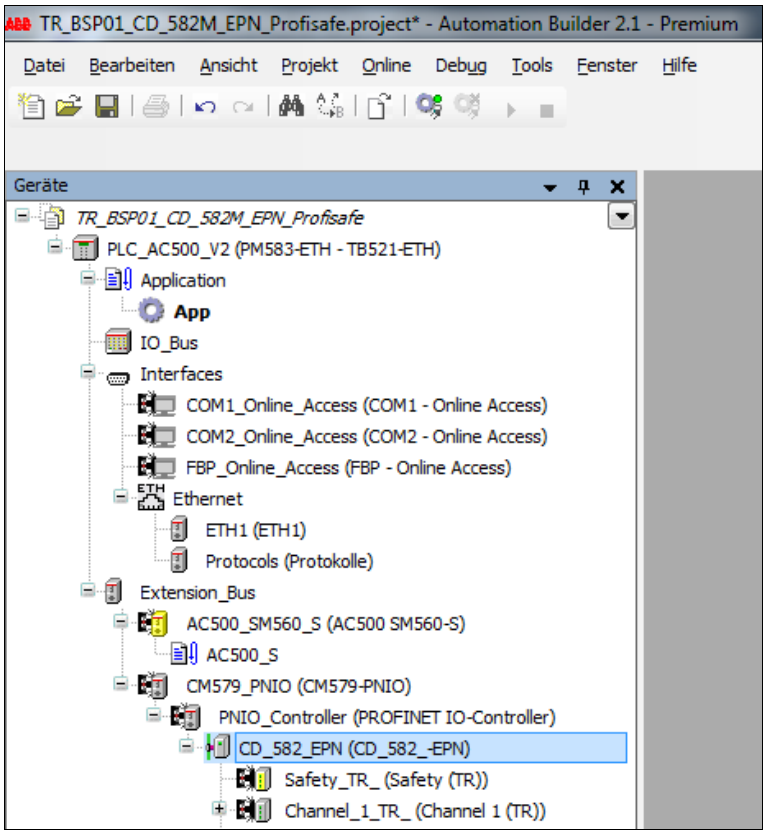
➢ Der Gerätekonfigurationsbaum enthält nun die F-CPU mit einem Programmknoten für das Safety-Programm und das Profinet-IO-Kommunikationsmodul `CM579-PNIO` mit dem entsprechenden PNIO-Controller-Knoten.



➢ Zur Vervollständigung der Hardware-Konfiguration muss das Mess-System - Objekt des `CD_582_-EPN` unter den PNIO-Controller hinzugefügt werden. Hierfür muss zuvor die GSDML im `Geräte-Repository` installiert worden sein (s. Kap. 3).

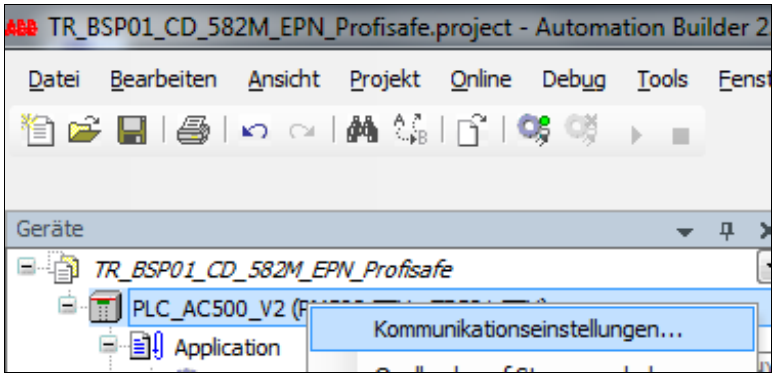


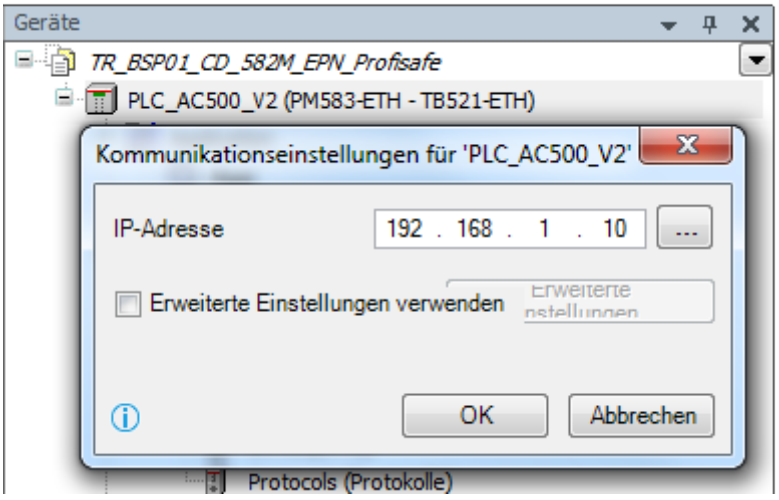
➢ Wenn das Objekt hinzugefügt worden ist, hängt unterhalb des PNIO-Controllers ein `CD_582_-EPN`-Objekt mit der Default-IO-Konfiguration.



5.2.1 Kommunikationseinstellungen zur Steuerung

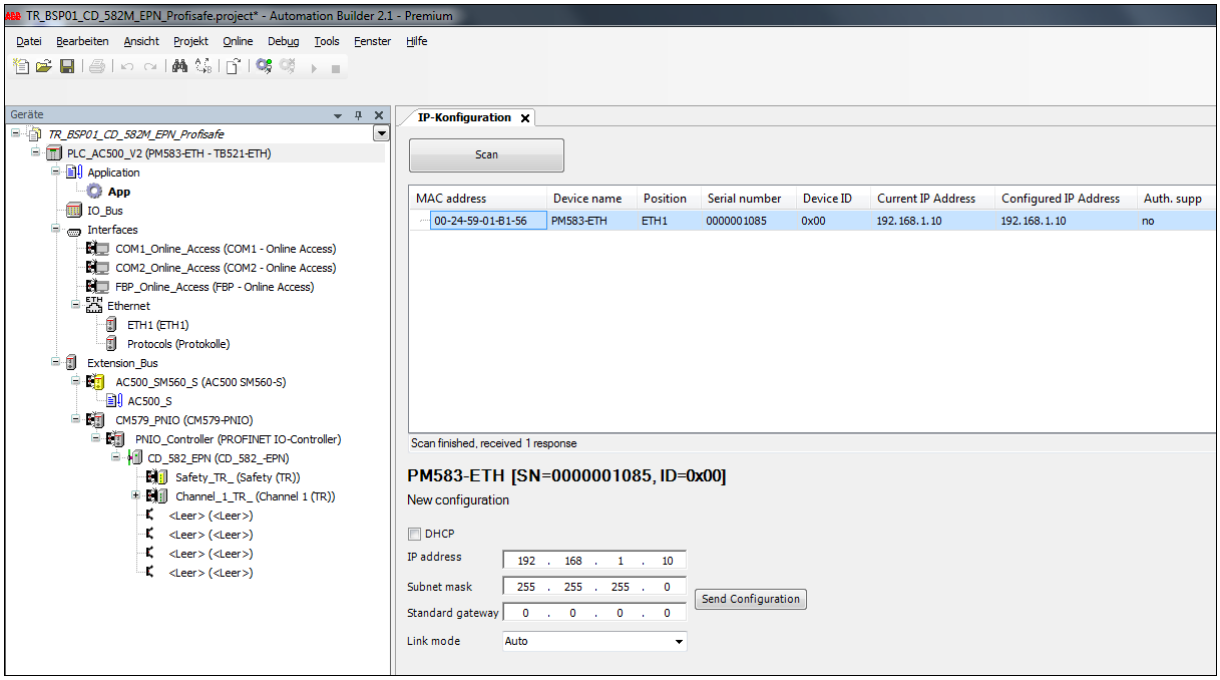
- Das Programm und die IO-Konfiguration wird vom `Automation Builder` zur Zentralbaugruppe `PM583-ETH` per LAN übertragen. Hierfür müssen die Kommunikationseinstellungen im Projekt zur IP-Konfiguration der `PM583-ETH` passen. Der Anwender muss ggfs. die Einstellungen der Steuerung an das Subnetz des Programmiergeräts anpassen.
- Der Anwender gibt in den Kommunikationseinstellungen die Ziel-IP-Adresse der Zentralbaugruppe an, die konfiguriert werden soll. Die Kommunikations-einstellungen können im Kontextmenü des Objektknotens der PM583-ETH `PLC_AC500_V2` editiert werden.





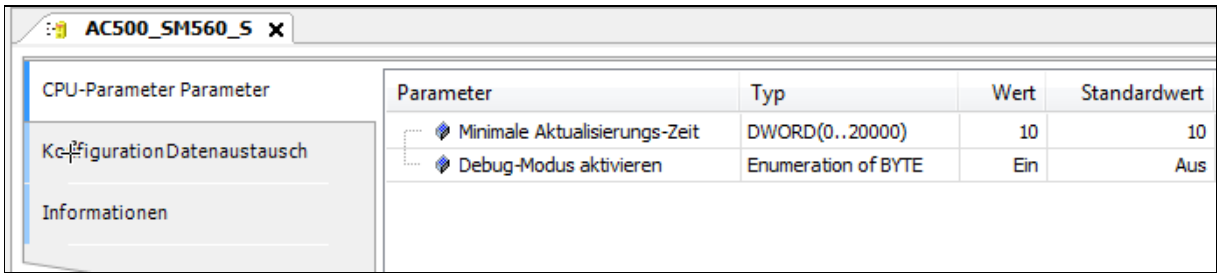
Die LAN-Verbindung zur Programmierung der Steuerung und zur Diagnose des Steuerungsprogramms erfolgt immer über die LAN-Buchse der Profilschiene `TB521-ETH`, die mit der Zentralbaugruppe `PM582-ETH` verbunden ist.

- Der Anwender kann die Kommunikationseinstellungen zur Steuerung verifizieren, indem das Werkzeug `IP-Konfiguration` unter dem Menüpunkt `Tools -> IP-Konfiguration` aufgerufen wird.
- Nach Betätigen der Schaltfläche `Scan` werden alle erreichbaren Netzwerkteilnehmer mit der jeweils gültigen IP-Konfiguration aufgelistet.



- Die Programmierung der `F-CPU SM560-S` kann nur im Debug-Modus erfolgen. Deshalb muss in der Bearbeitungsansicht des `SM560-S` unter dem Reiter `CPU-Parameter` der Parameter `Debug-Modus aktivieren` auf `Ein` gestellt werden.

Der Debug-Modus muss vor dem ersten Download zur F-CPU aktiviert werden. Außerdem muss vorher ein Bootprojekt für die PM583-ETH geladen und gespeichert worden sein.

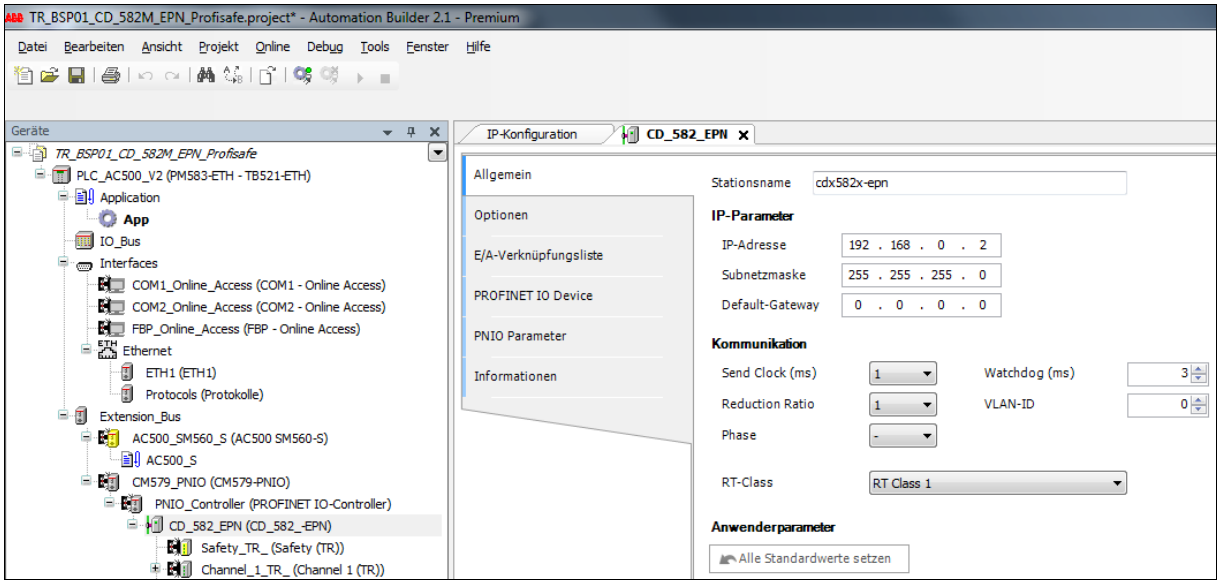


5.2.2 Kommunikationseinstellungen zum Mess-System

Damit das Mess-System am Steuerungsnetzwerk betrieben werden kann, muss diesem ein gültiger `Stationsname` (Profinet-IO-Gerätename) und eine `F-Dest-Adresse` (F-Parameter) zugeordnet werden.

Im Auslieferungszustand, sowie nach einer Rücksetzung, hat das Mess-System keinen Stationsnamen gespeichert.

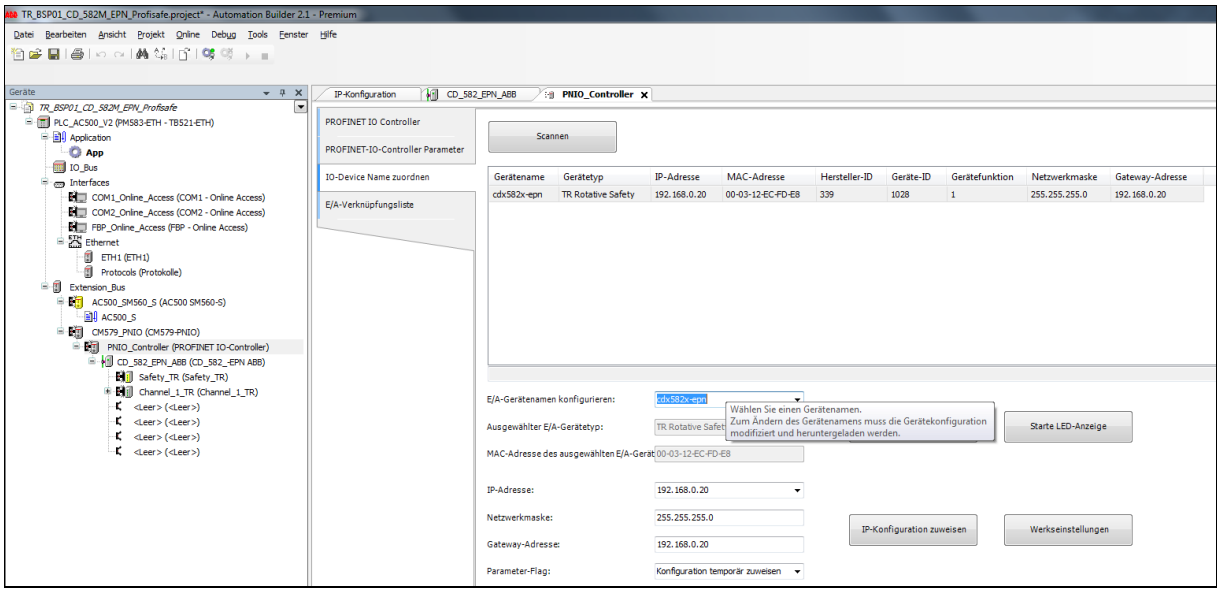
- Für die Einstellung des Stationsnamens muss der Anwender die Bearbeitungsansicht des Objektknotens des `CD_582_-EPN` öffnen. Unter dem Reiter `Allgemein` können die Kommunikationsparameter des Mess-Systems – u.a. der `Stationsname` festgelegt werden.



- Per Default ist für den `CD_582_-EPN` im Projekt immer der Stationsname `cdx582x-epn` voreingestellt. Der Stationsname muss für alle Teilnehmer eindeutig sein.
- Die Einstellung im Projekt muss mit der Einstellung im Gerät übereinstimmen, d.h. der Anwender muss der angeschlossenen Hardware-Komponente den eingestellten Stationsnamen zuweisen.

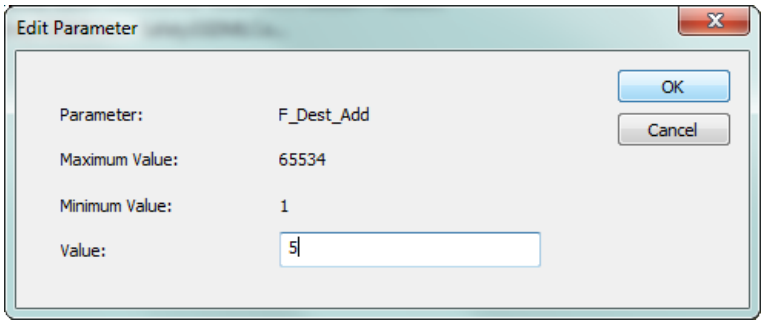
Die Zuweisung des Stationsnamens erfolgt über das DCP-Protokoll.

- Die Zuweisung des Stationsnamens erfolgt über die Bearbeitungsansicht des Objektknotens des PNIO-Controllers. Diese Funktionen stehen im dem Reiter `IO-Device Name zuordnen` zur Verfügung. Nach einem `Scan` des Netzwerks kann jedem angeschlossenen Gerät der voreingestellte Stationsname zugeordnet werden.



- Als zweites muss die Einstellung der F-Dest-Adresse mit der Einstellung des Drehschalters am `CD_582_-EPN` übereinstimmen.

- In diesem Projekt wurde der Drehschalter am Gerät auf `5` eingestellt. Also wird der F-Parameter `F_Dest_Add` ebenfalls auf `5` eingestellt
(s. auch Kap. 5.3.2 „Einstellen der F-Parameter“ auf Seite 31).

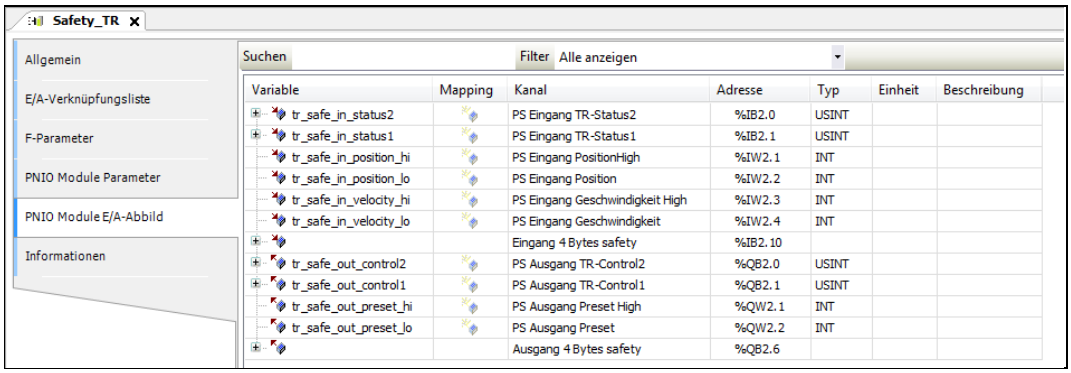


5.2.3 E/A-Verknüpfungen festlegen

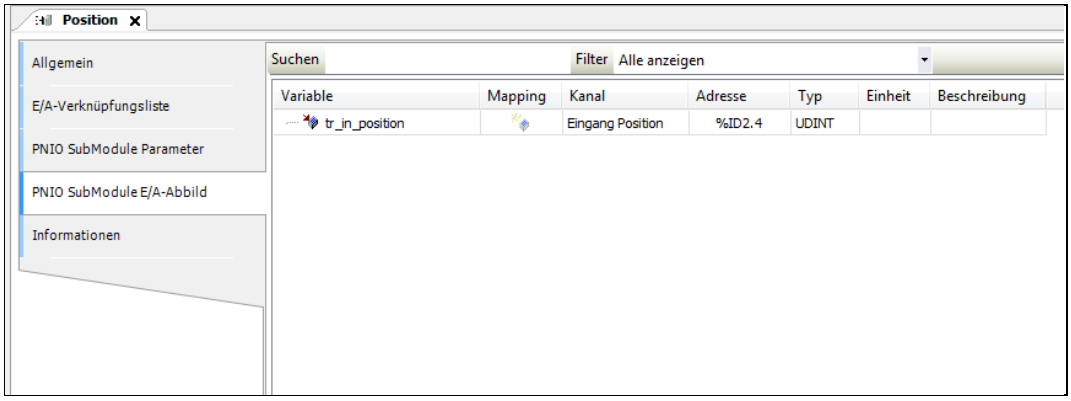
Zur einfacheren Verwendung der Prozessdaten – Ein- und Ausgänge in einem Steuerungsprojekt können diese mit symbolischen Variablennamen belegt werden. Zur Definition der symbolischen Namen muss in der Bearbeitungsansicht der IO-Module bzw. der IO-Submodule der Reiter `PNIO Module E/A-Abbild` bzw. `PNIO Submodule E/A-Abbild` geöffnet werden.

- Für die Definition der symbolischen Variablennamen können in der Spalte `Variable` sprechende Namen vergeben werden.

- Für die Definition der symbolischen Variablennamen im Safety-Bereich muss die Bearbeitungssicht des Safety-Moduls geöffnet werden. Alle symbolischen Variablennamen im Safety-Bereich werden auch direkt im Non-Safety Programm als „Read-Only“ verfügbar.



- Für die Definition der symbolischen Variablennamen im grauen Non-Safety-Bereich muss die Bearbeitungssicht der Profinet-Submodule geöffnet werden.



Nach Änderungen an der Hardware- und IO-Konfiguration sollten sowohl die `Konfigurationsdaten` als auch die `Safety-Konfigurationsdaten` neu erzeugt werden (s. Kap. 5.4 „Erstellen der Konfigurationsdaten“ auf Seite 32).

5.3 Parametrierung

5.3.1 Einstellen der iParameter

- Um die iParameter des `CD_582_-EPN` einstellen zu können, muss in der `Bearbeitungsansicht` der IO-Module bzw. der IO-SubModule unterhalb des Knotens `CD_582_-EPN` der Reiter `Allgemein` geöffnet werden. Zur Einstellung der iParameter muss die Spalte `Wert` editiert werden.

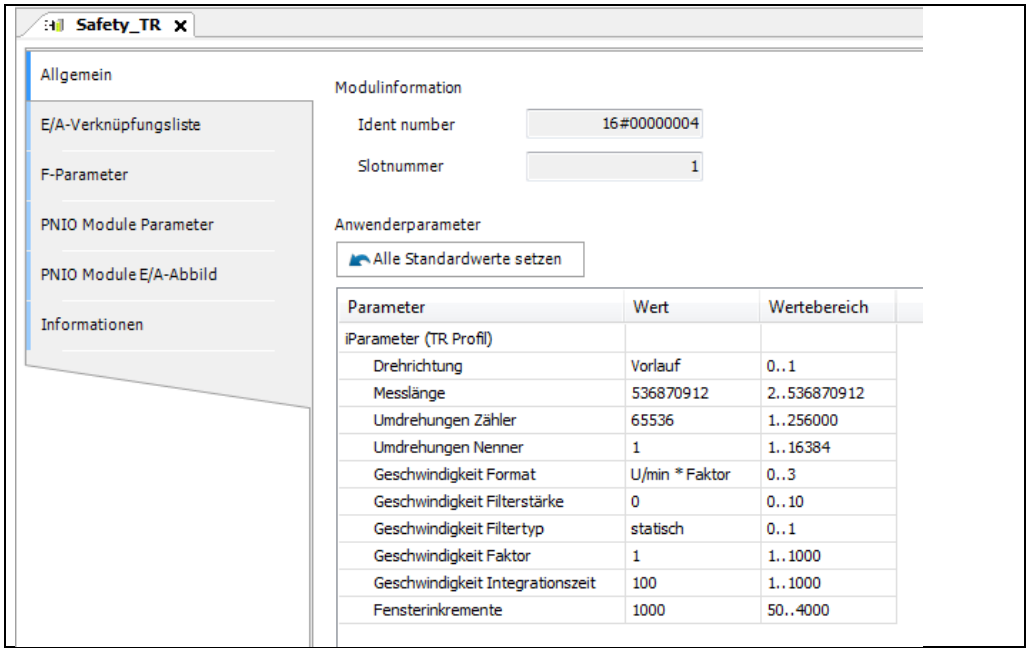


Abbildung 4: Bearbeitungsansicht des IO-Moduls Safety_TR mit iParameter-Tabelle

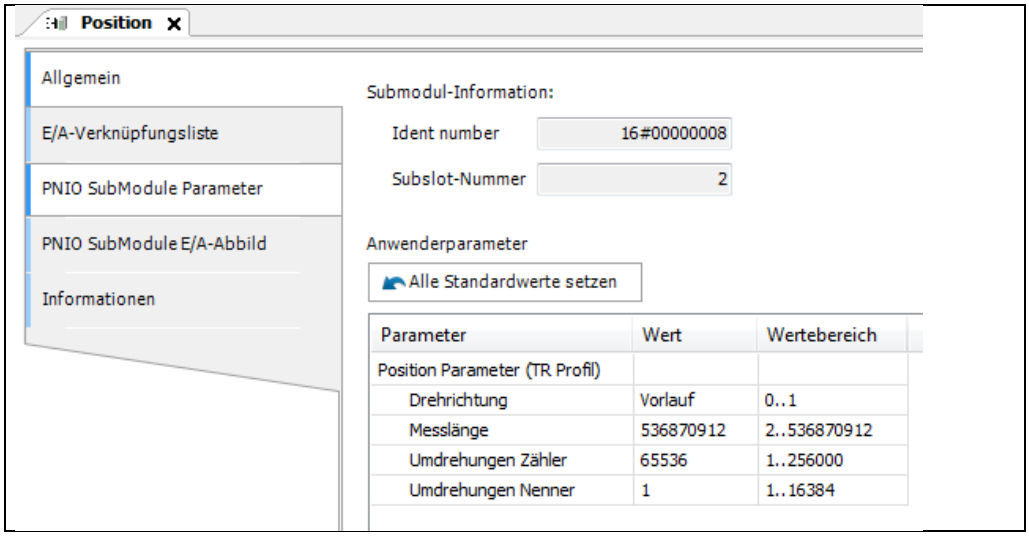
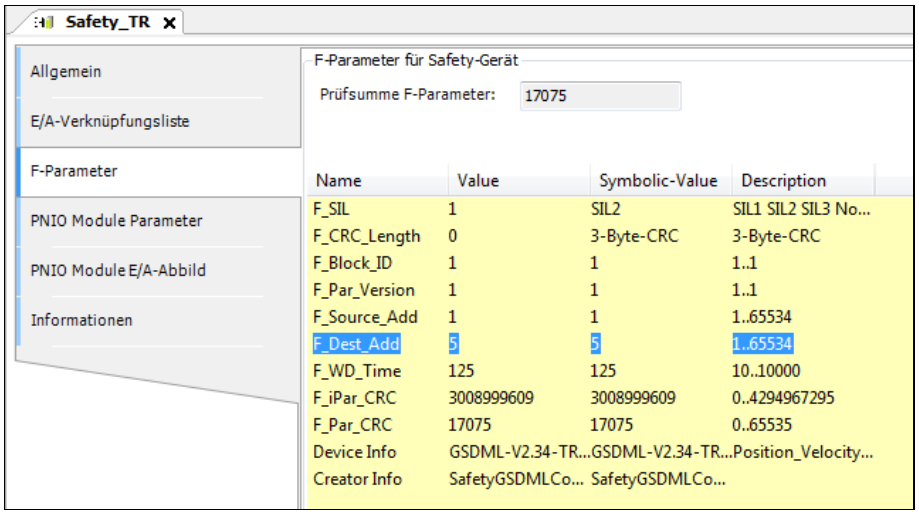


Abbildung 5: Bearbeitungsansicht des IO-SubModuls Position mit iParameter-Tabelle

- Oben sind die Default-Werte der iParameter eingetragen. Werden davon abweichende Parameterwerte benötigt, muss für diesen neuen Parameterdatensatz eine F_iPar_CRC-Berechnung erfolgen. Siehe Kap.: 4.1 „iParameter“ auf Seite 11.
- Der dort errechnete Wert ist dann im Parameterdatensatz der F-Parameter unter F_iPar_CRC einzutragen. Siehe Kap.: 5.3.2 „Einstellen der F-Parameter“ auf Seite 31.

5.3.2 Einstellen der F-Parameter

- Für das Safety-IO-Modul des `CD_582_-EPN` müssen die F-Parameter eingestellt werden.
- Um die F-Parameter einstellen zu können, muss in der `Bearbeitungsansicht` des IO-Moduls `Safety_TR` unterhalb des Knotens `CD_582_-EPN` der Reiter `F-Parameter` geöffnet werden.



Der `F_Dest_Add`-Eintrag und die Einstellung der Adressschalter des Mess-Systems müssen übereinstimmen!

Der Parameterwert für den Parameter `F_iPar_CRC` ergibt sich aus dem eingestellten Parameterdatensatz der iParameter und dem daraus berechneten CRC-Wert. Siehe Kap.: 5.3.1 „Einstellen der iParameter“ auf Seite 30.

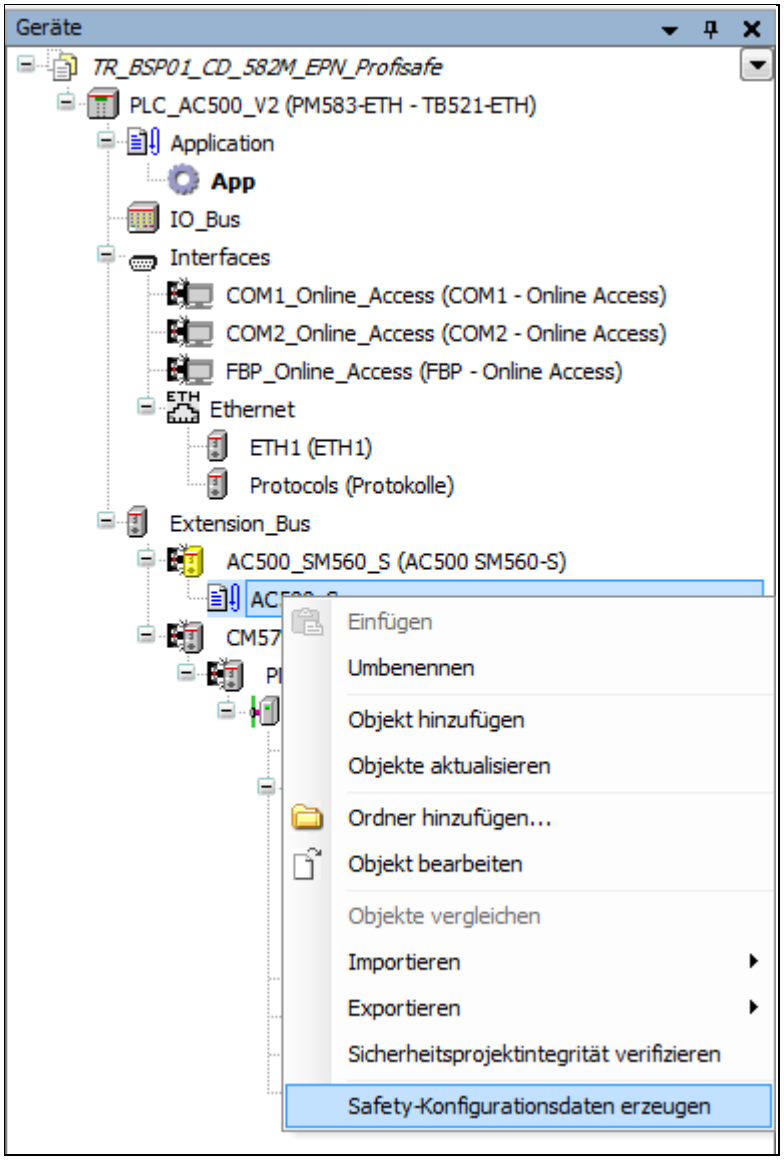
- Der Wert der `F_Par_CRC` aktualisiert sich automatisch nach Einstellung der F-Dest-Adress bzw. der F-iPar-CRC.

5.4 Erstellen der Konfigurationsdaten

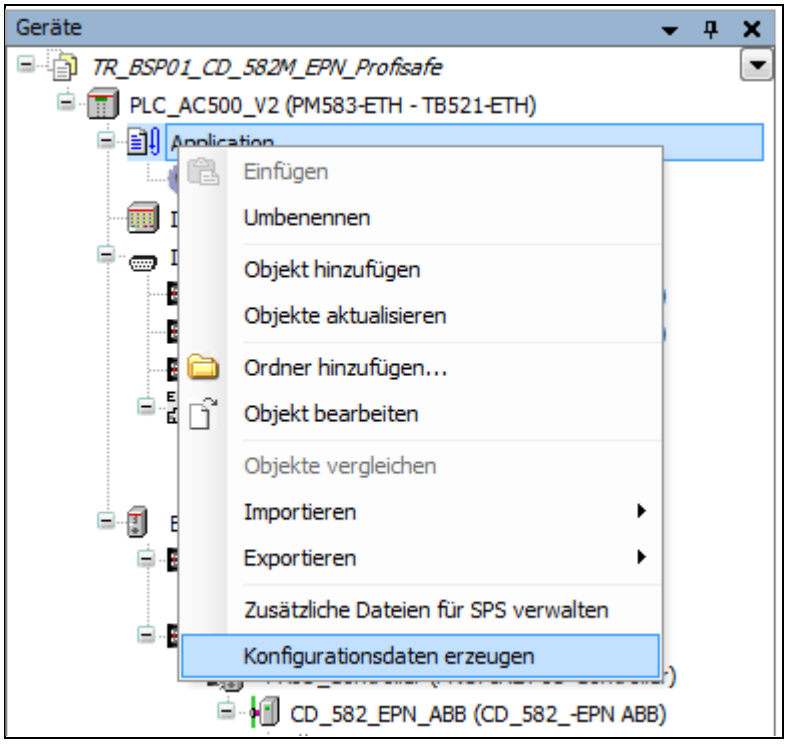
Wenn die Hardware-Konfiguration komplett ist, müssen vor der Erstellung der SPS-Programme die Konfigurationsdaten für beide Zentralbaugruppen erstellt werden.

`Automation Builder` erkennt die Notwendigkeit für die Erstellung der Konfigurationsdaten automatisch und wird den Anwender entsprechend auffordern.

➢ Zuerst sollen die Konfigurationsdaten für die Safety-Applikation erstellt werden. Der Anwender ruft hierfür die Funktion `Safety-Konfigurationsdaten erzeugen` aus dem Kontextmenü der Safety-Applikation unterhalb des SM560-S-Knotens auf.



➢ Danach sollen die Konfigurationsdaten für die Non-Safety-Applikation erstellt werden. Der Anwender ruft hierfür die Funktion `Konfigurationsdaten erzeugen` aus dem Kontextmenü der Non-Safety-Applikation unterhalb des PLC_AC500_V2-Knotens auf.



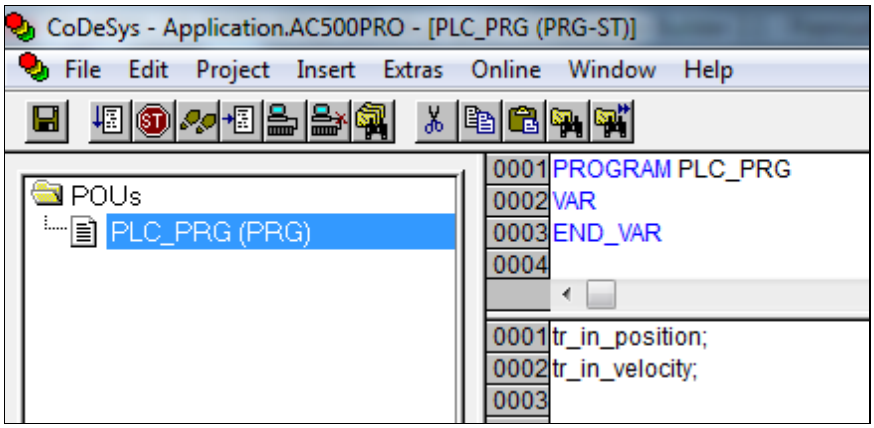
➢ Nachdem alle Konfigurationsdaten erzeugt worden sind, kann mit dem Laden der Non-Safety-Applikation begonnen werden.

5.5 Non-Safety-Applikation laden

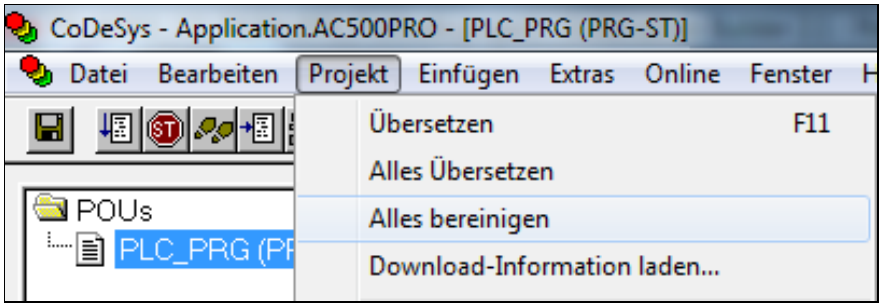
Bei der Programmierung der AC500-S wird zwischen der Non-Safety-Applikation und der Safety-Applikation unterschieden. Die Non-Safety-Applikation wird in einem CoDeSys-Editor mit weißem Hintergrund erstellt, geladen und getestet.

- Der Non-Safety-CoDeSys-Editor öffnet nach einem Doppel-Click auf die Non-Safety-Applikation unterhalb des PLC_AC500_V2-Knotens im Gerätekonfigurationsbaum.

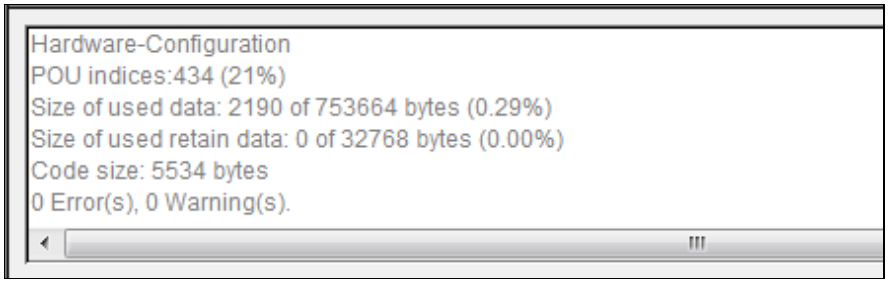
- Im Hauptprogramm der Non-Safety-Applikation PLC_PRG wird das SPS-Programm in der IEC61131-Sprache ST erstellt.



- In diesem Beispiel werden die verknüpften Variablen der Prozessdateneingänge des CD_582_-EPN aufgerufen. Damit können die Werte im Programm verwendet werden (vgl. Kap. 5.2.3 „E/A-Verknüpfungen festlegen“ auf Seite 29).

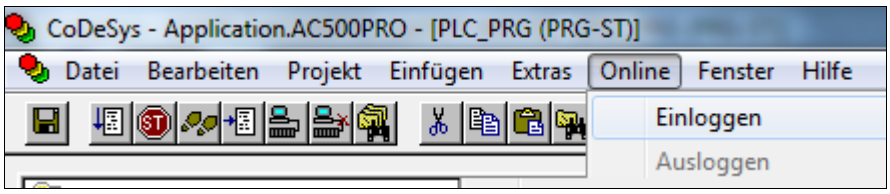


- Vor dem Laden der Non-Safety-Applikation muss das Programm übersetzt werden. Es ist zweckmäßig immer die Funktionen Projekt -> Alles bereinigen und danach Projekt -> Alles Übersetzen aufzurufen, um den Übersetzungsvorgang anzustoßen.



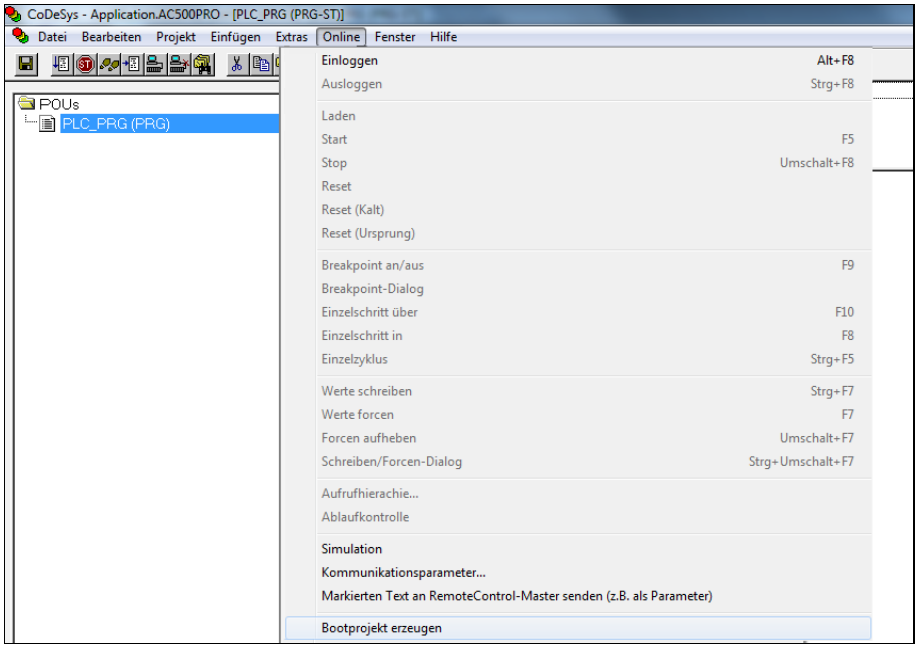
- Die Applikation soll mit 0 Fehler, 0 Warnungen übersetzen.

- Nach der fehlerfreien Übersetzung kann das Laden des Programms auf die Steuerung erfolgen. Hierfür muss eine Verbindung zwischen `Automation Builder` und der Zentralbaugruppe `PM583-ETH` aufgebaut werden, indem der Befehl `Online -> Einloggen` ausgeführt wird.



- Wenn kein Programm vorhanden ist, oder das Programm sich geändert hat, wird der Anwender aufgefordert, das Laden des Programms mit `Ja` zu bestätigen.

- Nach dem Laden muss ein Bootprojekt erzeugt werden, damit das Steuerungsprogramm remanent auf der Zentralbaugruppe gespeichert wird. Das Bootprojekt wird nach dem Aufruf der Funktion `Online -> Bootprojekt erzeugen` angelegt.



Das Erzeugen des Bootprojekts auf der Steuerung dauert ca. 30s. Dies wird durch ein Blinken der RUN-LED an der PM583-ETH angezeigt. Der über CoDeSys ausgelöste Befehl ist dabei schon beendet.

Solange die Steuerung das Bootprojekt speichert, sollte die Versorgung des Steuerungssystems nicht unterbrochen werden.

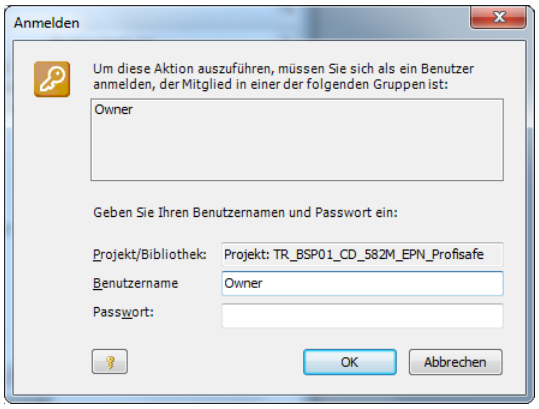
5.6 Safety-Applikation laden

Bei der Programmierung der AC500-S wird zwischen der Non-Safety-Applikation und der Safety-Applikation unterschieden. Die Safety-Applikation wird in einem CoDeSys-Editor mit gelbem Hintergrund erstellt, geladen und getestet.

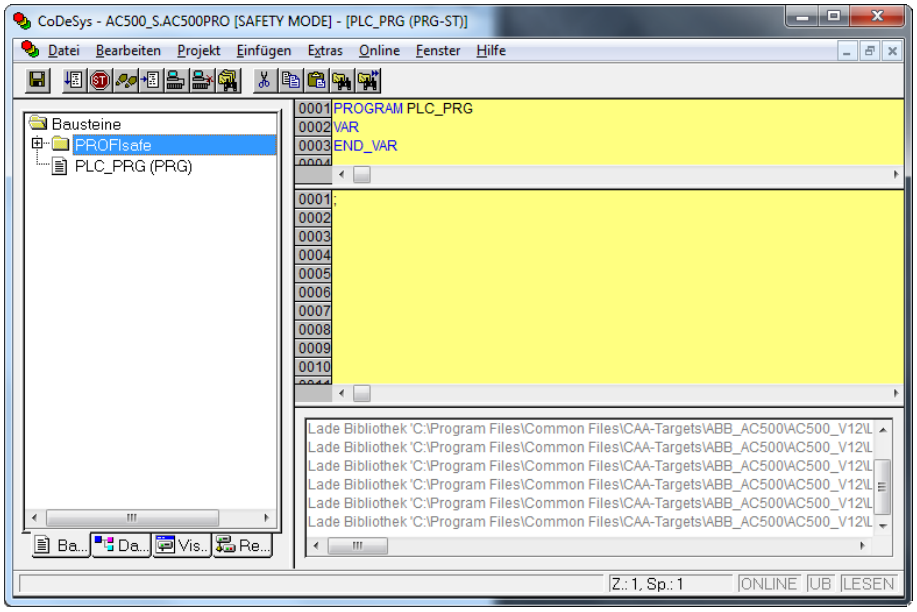
➢ Der `Safety-CoDeSys-Editor` öffnet nach einem Doppel-Click auf die Safety-Applikation `AC500_S` unterhalb des `AC500_SM560_S`-Knotens im Gerätekonfigurationsbaum.

Beim ersten Öffnen des `Safety-CoDeSys-Editors` wird der Anwender aufgefordert, das Einfügen mehrerer System-Bibliotheken in die Safety-Applikation zu bestätigen. Der Anwender muss alle Informationsdialoge mit `OK` bestätigen.

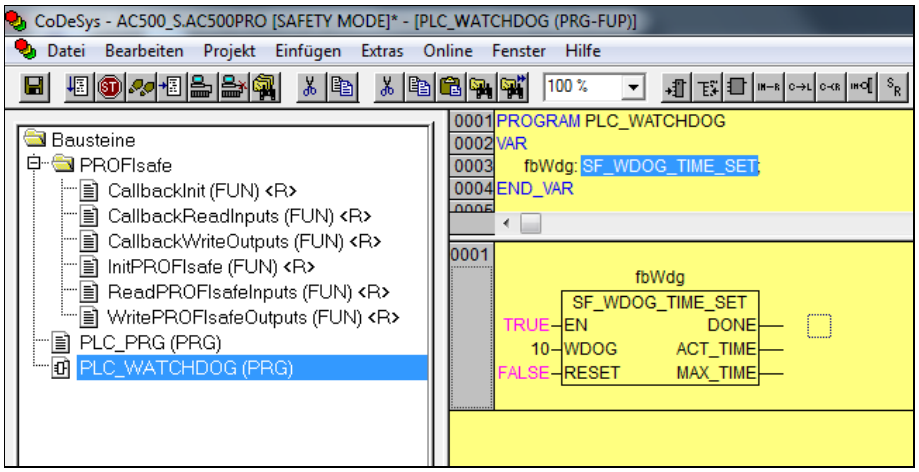
➢ Der Anwender wird zur Authentifizierung zur Eingabe eines Benutzers und eines Passworts aufgefordert. Im Kontext dieses Beispiels wird immer der Benutzer „Owner“ und ein Leerstring als Passwort verwendet.



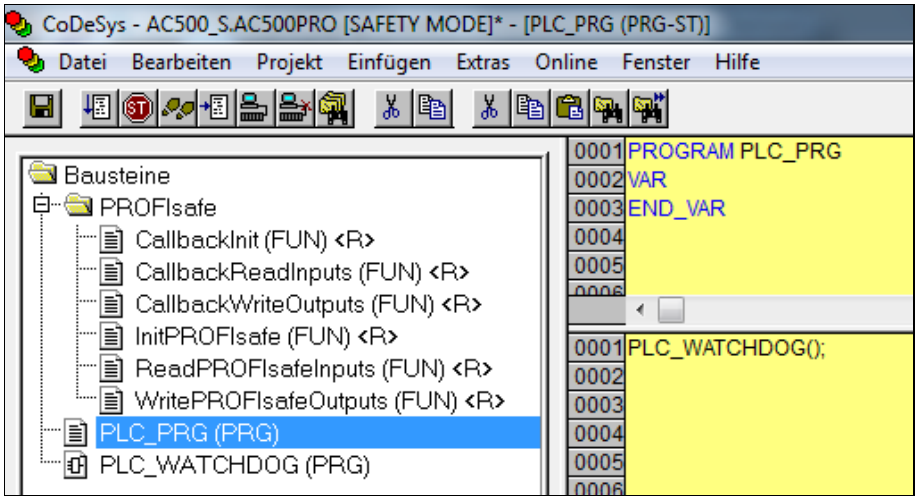
➢ Im Hauptprogramm der Safety-Applikation `PLC_PRG` wird das Safety-SPS-Programm in der IEC61131-Sprache `ST` erstellt. Zusätzlich zum Hauptprogramm lädt CoDeSys eine Reihe von Bibliotheken zur Bearbeitung der PROFIsafe-Telegramme.



- Im Hintergrund der Safety-Steuerung `SM560-S` läuft ein Watchdog-Timer, der zyklisch getriggert werden muss. Als erstes wird also das Unterprogramm `PLC_WATCHDOG` erstellt, das den Trigger ansteuern soll. Zur Programmierung des Watchdog-Triggers steht der System-Funktionsbaustein `SF_WDOG_TIME_SET` zur Verfügung.



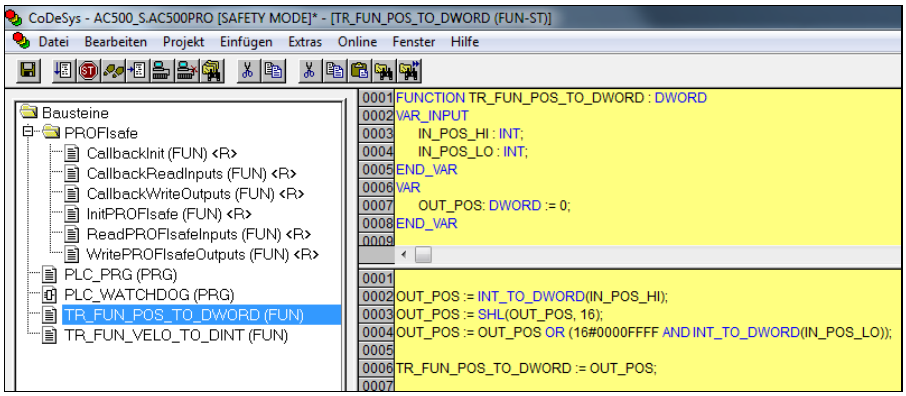
- Im Kontext dieses Beispiels wird der Watchdog-Timer auf 10 ms eingestellt. Dieser Wert muss in Abhängigkeit der Zykluszeit des Hauptprogramms angepasst werden.



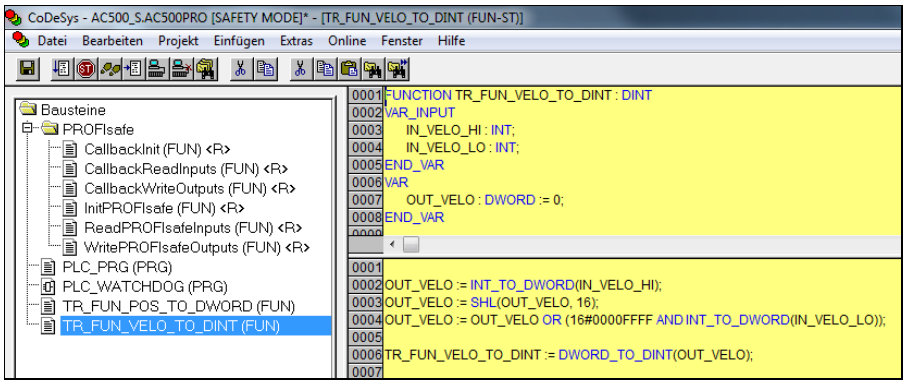
- Der Aufruf für den Watchdog-Trigger sollte am Anfang des Hauptprogramms `PLC_PRG` stehen.

- In diesem Beispiel sollen die verknüpften Variablen der fehlersicheren Prozessdateneingänge des CD_582_-EPN aufgerufen werden. In den PROFIsafe-Prozessdaten werden wortorientierte Informationen als INT (16bit, signed) verarbeitet. Allerdings übermittelt der CD_582_-EPN seine Position als DWORD (32bit, unsigned) und seine Geschwindigkeit als DINT (32bit, signed).

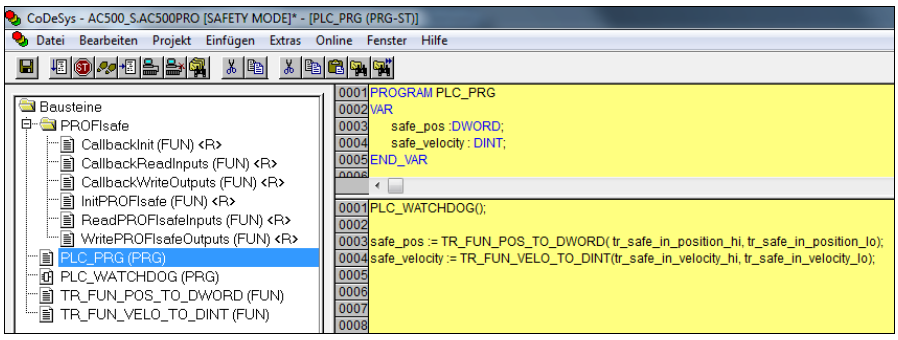
- Zur weiteren Verwendung sollen hier also die Prozesseingangsdaten zusammengebaut und gewandelt werden. Hierfür werden die verknüpften Variablen für Position und Geschwindigkeit in Unterfunktionen vorverarbeitet.



- In der Unterfunktion TR_FUN_POS_TO_DWORD werden HI- und LO-Wort der Position aus den Prozessdateneingängen zu einem DWORD zusammengebaut.

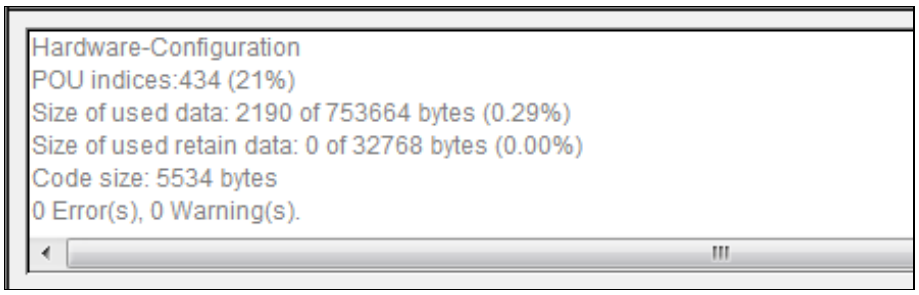


- In der Unterfunktion TR_FUN_VELO_TO_DINT werden HI- und LO-Wort der Geschwindigkeit aus den Prozessdateneingängen zu einem DINT zusammengebaut.
- Im Hauptprogramm PLC_PRG werden diese Unterfunktionen genutzt, um die verknüpften Prozessdateneingänge den beiden lokalen Variablen safe_pos und safe_velocity zuzuweisen.



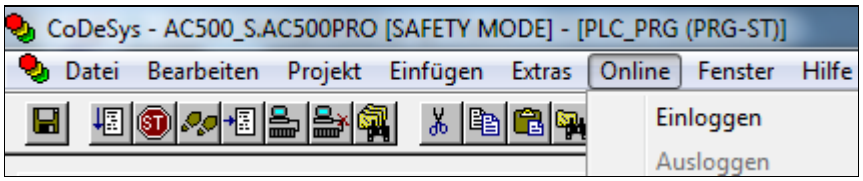
- Der CD_582_-EPN wird die Wiedereingliederung in das PROFIsafe-Netzwerk einfordern, wenn die Kommunikation zum PROFIsafe-F-Host synchronisiert werden muss. Dies ist zum Beispiel nach einer Netzwerkunterbrechung notwendig. Die Wiedereingliederung wird über das OA-Bit der PROFIsafe-Instanz gesteuert. Ohne Wiedereingliederung können keine PROFIsafe-Prozessdaten in der Safety-Applikation verarbeitet werden.
  Nähere Informationen hierzu, siehe ABB AC500-S Sicherheitshandbuch.

- Vor dem Laden der Safety-Applikation muss das Programm übersetzt werden. Es ist zweckmäßig immer die Funktionen Projekt -> Alles bereinigen und danach Projekt -> Alles Übersetzen aufzurufen, um den Übersetzungsvorgang durchzuführen.



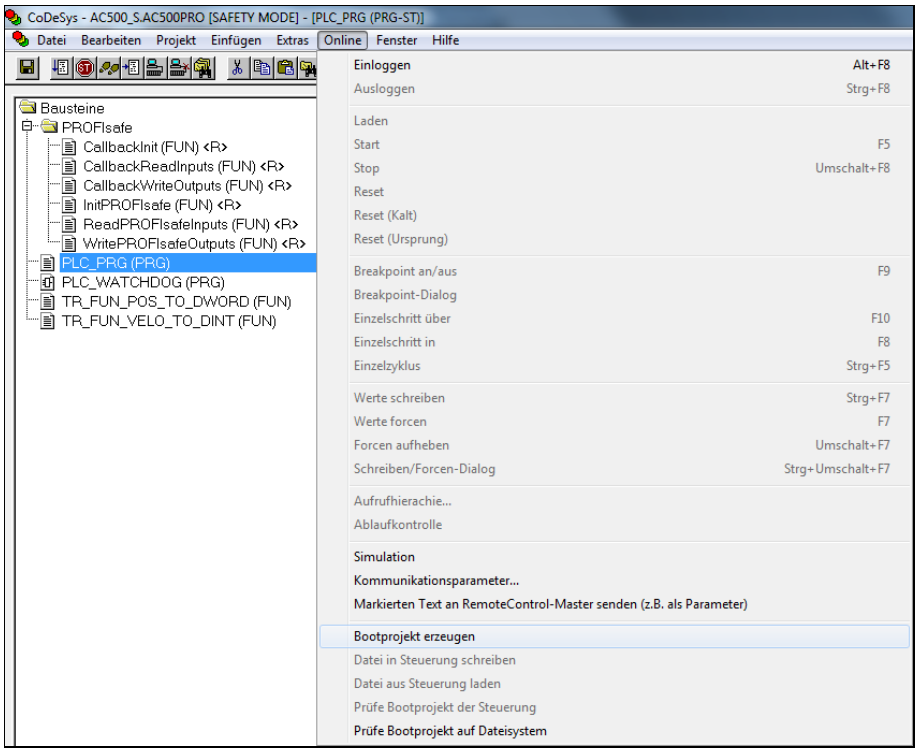
- Die Applikation soll mit 0 Fehler, 0 Warnungen übersetzen.

- Nach der fehlerfreien Übersetzung kann das Laden des Programms auf die Steuerung erfolgen. Hierfür muss eine Verbindung zwischen Automation Builder und der F-CPU SM560-S aufgebaut werden, indem der Befehl Online -> Einloggen ausgeführt wird.



- Wenn kein Programm vorhanden ist, oder das Programm sich geändert hat, wird der Anwender aufgefordert, das Laden des Programms mit Ja zu bestätigen.

- Nach dem Laden muss ein Bootprojekt erzeugt werden, damit das Steuerungsprogramm remanent auf der Zentralbaugruppe gespeichert wird. Das Bootprojekt wird nach dem Aufruf der Funktion Online -> Bootprojekt erzeugen angelegt.



Das Erzeugen des Bootprojekts auf der Steuerung dauert ca. 2s. Dies wird durch ein Blinken der RUN-LED an der SM560-S angezeigt. Der über CoDeSys ausgelöste Befehl ist dabei schon beendet.
Solange die F-CPU das Bootprojekt speichert, sollte die Versorgung des Steuerungssystems nicht unterbrochen werden.

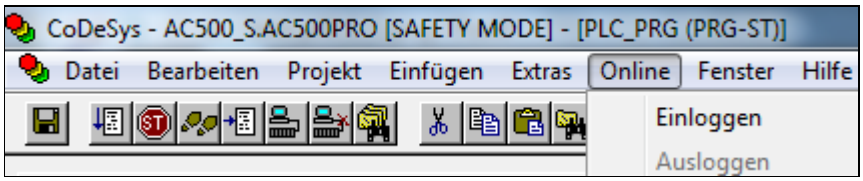
- Das neu programmierte Steuerungsprogramm läuft nach einem Power-Cycle (Systemneustart) an.

Die F-CPU SM560-S überwacht die Versorgung der Steuerungseinheit auf Spannungsausfälle.
Wenn der Anwender einen Power-Cylce an der Steuerungseinheit durchführt, sollte die Steuerungseinheit für mindestens 2 s ohne Versorgung bleiben.

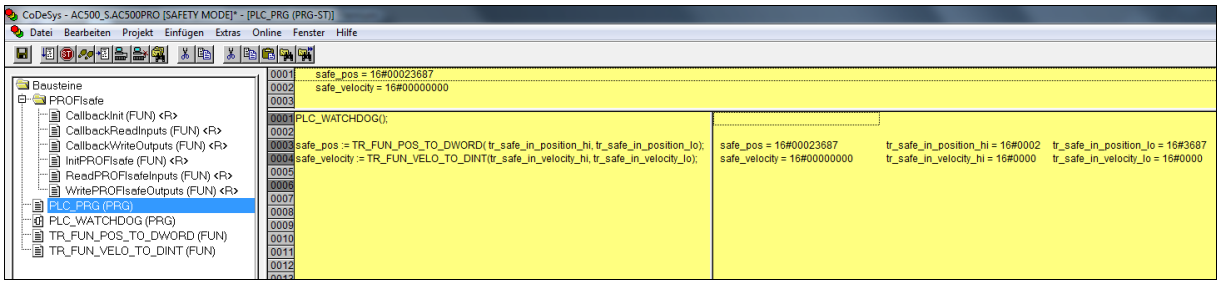
5.7 Safety-Applikation testen

Nach Erstellung der Safety-Applikation muss ein vollständiger Funktionstest entsprechend der Automatisierungsaufgabe durchgeführt werden.

- Wenn zuvor eine gültige Safety-Applikation in die F-CPU geladen und ein Boot-Projekt erzeugt worden ist, kann der Anwender das Programm testen, indem er sich über den Safety-CoDeSys-Editor in die Steuerung einloggt.



- Anstelle eines Ladevorgangs startet der Editor nun im Debug-Modus und zeigt den Zustand der lokalen Variablen, der verknüpften Prozessdateneingänge und des Programmablaufs an.



- Die lokalen Variablen `safe_pos` und `safe_velocity` zeigen die aktuellen Prozessdaten des `CD_582_-EPN` an.

6 Steuerungsprogramm erweitern – Anwendungsbeispiele

Das unter Kapitel 5 erstellte Sicherheitsprogramm wird in den nachfolgenden Abschnitten um Anwendungsbeispiele für die Preset-Durchführung und die Fehlerauswertung erweitert.

Die Beispiele stellen jedoch keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bei unterschiedlichen Automatisierungsaufgaben leisten.

Mit Hilfe der vorgestellten Funktionsbausteine soll die Integration des Mess-Systems in eine Applikation vereinfacht werden.

Bei dem nachfolgenden Anwendungsbeispiel

- Preset-Durchführung

werden die Fehlerzustände von dem hier vorgestellten Funktionsbaustein ausgegeben. Die zugehörige Fehlerbehandlung ist nicht Teil der Beispiele und muss vom Anwender umgesetzt werden.

Nutzungsbedingungen der Softwarebeispiele in Kapitel 2.4 beachten!

6.1 Preset-Durchführung

Der Preset-Baustein, der für die Preset-Justage-Funktion erstellt wird, setzt die aktuelle sichere Position des Mess-Systems auf einen beliebigen neuen Wert innerhalb seines Messbereichs. Der Preset-Baustein zeigt über die Bits `OUT_ERROR` und `OUT_VALID` an, ob die Preset-Justage-Funktion durchgeführt werden konnte. Die Preset-Justage-Funktion kann nur ausgeführt werden, solange keine Passivierung des Mess-Systems vorliegt oder bei keiner anderen sicheren Position gerade ein Preset-Justage-Funktion durchgeführt wird. Die Bereitschaft eine Preset-Justage-Funktion ausführen zu können, wird durch das Bit `OUT_READY` angezeigt. Siehe auch Kap.: 5.6 „Safety-Applikation laden“ auf Seite 36.

Der Preset-Baustein führt keine Überprüfung der neuen Position durch. Dies muss durch den Anwender umgesetzt werden!

6.1.1 Parameter Beschreibung

Eingangsparameter	Datentyp	Beschreibung
IN_REQ	BOOL	Startet die Preset-Justage-Funktion.
IN_PRESET	DWORD	Neuer Position-Wert der eingestellt werden soll.
IN_STATUS1	BYTE	Kennzeichnet den aktuellen Status des Mess-Systems. In den Eingangsdaten des Mess-Systems aus Register `TR-Status1` einlesen.
IN_STATUS2	BYTE	Kennzeichnet den aktuellen Status des Mess-Systems. In den Eingangsdaten des Mess-Systems aus Register `TR-Status2` einlesen.

Ausgangsparameter	Datentyp	Beschreibung
OUT_READY	BOOL	Gibt an, ob mit dem Baustein eine Preset-Justage-Funktion ausführt werden kann.
OUT_BUSY	BOOL	Gibt an, ob der Baustein gerade die Preset-Justage-Funktion ausführt.
OUT_VALID	BOOL	Gibt an, ob die Ausführung der Preset-Justage-Funktion erfolgreich beendet wurde.
OUT_ERROR	BOOL	Gibt an, ob die Ausführung der Preset-Justage-Funktion mit einem Fehler beendet wurde.
OUT_PRESET	DINT	Preset Wert für das Mess-System. In den Ausgangsdaten des Mess-Systems an Register `Preset` ausgeben.
OUT_CTRL	BYTE	Vorgabewert für das Steuerwort des Mess-Systems in den Prozessausgangsdaten. In den Ausgangsdaten des Mess-Systems an Register `TR-Control1` ausgeben.

6.1.2 Funktionsbeschreibung

Eine vollständige Beschreibung der Preset-Justage-Funktion und der zugehörigen Status- und Kontrollbits ist in der Schnittstellenbeschreibung TR-ECE-BA-D-0139 enthalten.

- Der Eingang `tr_safe_in_status1` muss 0x10 sein, damit die Preset-Justage-Funktion ausgeführt werden kann. Der Ausgang `OUT_READY` zeigt den aktuellen Zustand des Eingangs `tr_safe_in_status1` an.

- Der Eingang `IN_PRESET` wird immer gelesen und an dem Ausgang `OUT_PRESET` gelatcht ausgegeben, wenn das Signal `IN_REQ` kommt. Nachdem der Preset-Baustein über den Eingang `IN_REQ` gestartet wurde, läuft die Preset-Sequenz in einer Zustandsmaschine ab.

- Mit steigender Flanke des Eingangs `IN_REQ` wird der Preset-Baustein ausgeführt. Die Ausgänge `OUT_VALID` und `OUT_ERROR` werden auf 0 zurückgesetzt. Das Control-Byte wird dem Status der Zustandsmaschine entsprechend gesetzt und über den Ausgang `tr_safe_out_control1` zum Gerät übertragen.

- Das Mess-System führt danach die Preset-Justage-Funktion aus. Der Zeitpunkt für das Rücksetzen des Eingangs `IN_REQ` auf 0 hat keinen Einfluss auf die weitere Ausführung der Preset-Justage-Funktion.

- Nachdem die Preset-Justage-Funktion ausgeführt wurde, setzt das Mess-System am Eingang das `TR_PresOk` bzw. das `TR_PresError` Bit im Eingangswort `tr_safe_in_status1`. Mit diesen Bits werden die entsprechenden Ausgänge `OUT_VALID` bzw. `OUT_ERROR` gesetzt.

- Nachdem einer der Ausgänge `OUT_VALID` bzw. `OUT_ERROR` gesetzt wurde, werden die Ausgänge `TR_PresPrep`, `TR_PresReq` im Control-Byte `tr_safe_out_control1` und `OUT_BUSY` wieder auf 0 zurückgesetzt. Die Ausführung des Preset-Bausteins ist damit beendet.

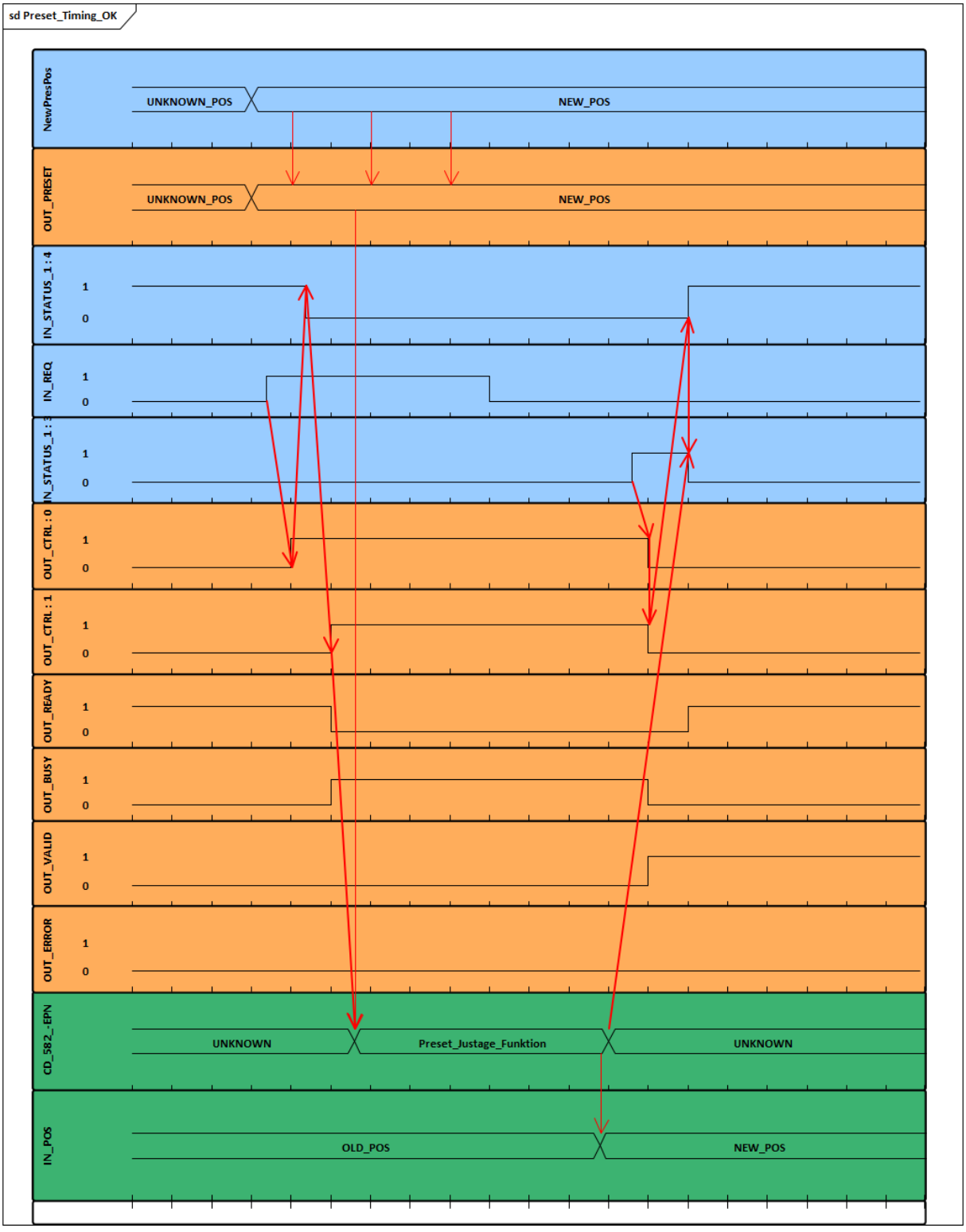


Abbildung 6: Timing-Diagramm der Preset-Justage-Funktion mit fehlerfreiem Ablauf

blauer Bereich: Eingangssignale Preset-Baustein
oranger Bereich: Ausgangssignale Preset-Baustein
grüner Bereich: CD_582_-EPN Mess-System-Funktion bzw. Mess-System-Werte

- Solange der Eingang IN_STATUS_1 : 4 den Wert 0 hat, wird die Preset-Justage-Funktion nicht ausgeführt. Mit steigender Flanke des Eingangs IN_REQ werden die Ausgänge OUT_VALID und OUT_ERROR auf 0 zurückgesetzt. Die Ausgänge OUT_CTRL : 0, OUT_CTRL : 1, OUT_READY und OUT_BUSY ändern ihren Wert nicht.

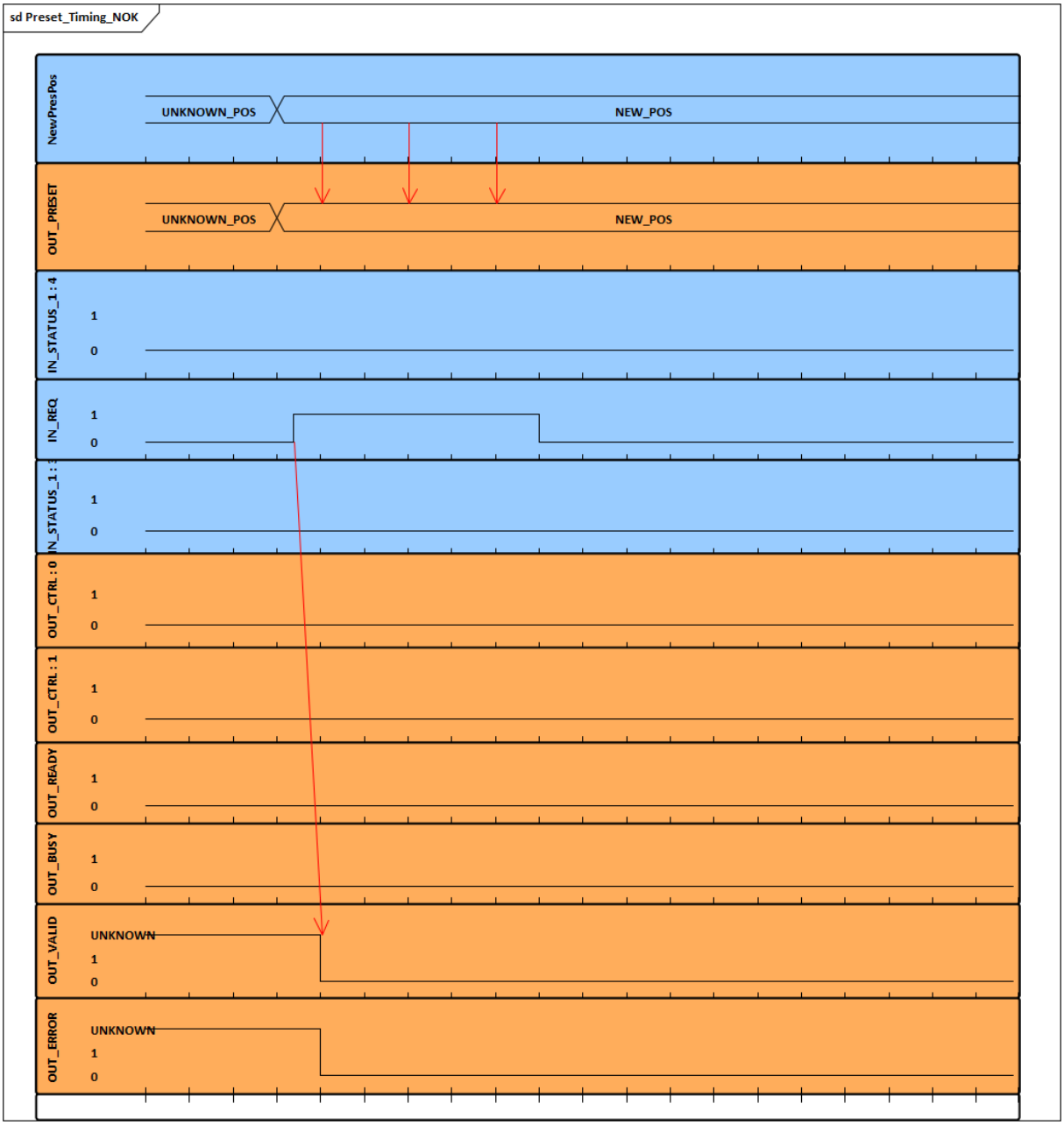
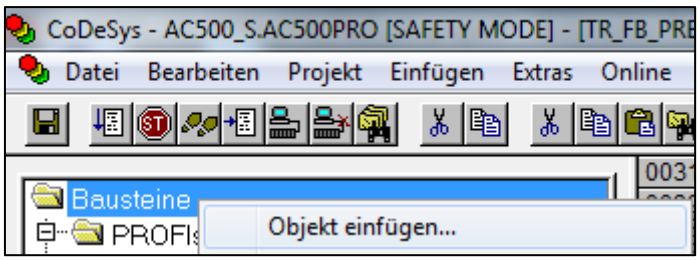


Abbildung 7: Timing-Diagramm der Preset-Justage-Funktion, wenn IN_STATUS_1 : 4 den Wert 0 hat

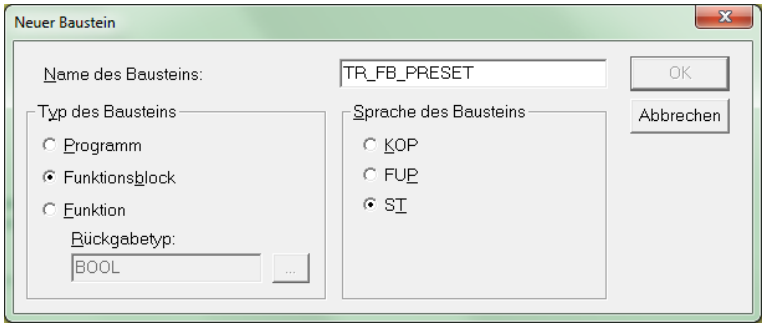
blauer Bereich: Eingangssignale Preset-Baustein
oranger Bereich: Ausgangssignale Preset-Baustein

6.1.3 Baustein Erstellung

- Um den Preset-Baustein zu erstellen, muss zuerst ein neuer Safe Funktionsbaustein mit dem Name `TR_FB_PRESET` angelegt werden. Dazu muss im Kontextmenü des CoDeSys-Bausteinverzeichnisses die Funktion `Objekt einfügen…` aufgerufen werden.



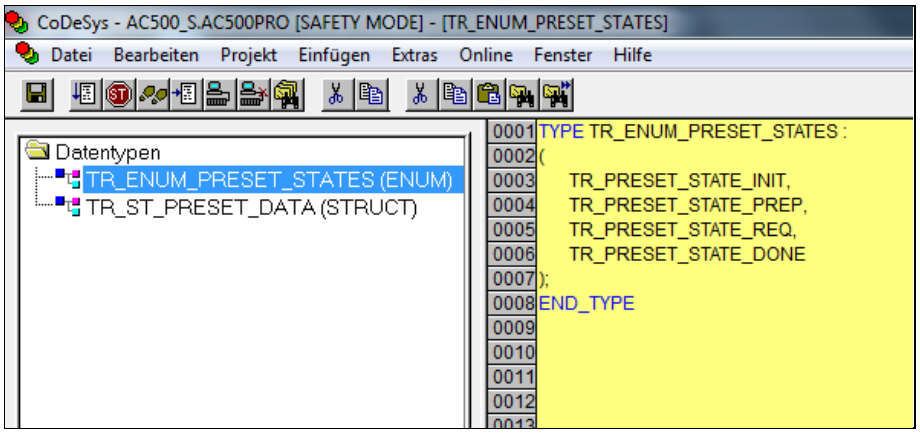
- Im folgenden Dialog wird der Name des Funktionsbausteins `TR_FB_PRESET` eingetragen. Als Typ wird die Option `Funktionsblock` gewählt. Im Kontext dieses Beispiels wird die Sprache `ST` gewählt.



- Im Preset-Funktionsbaustein müssen die folgenden Variablen angelegt werden.

```
0021 FUNCTION_BLOCK TR_FB_PRESET
0022 VAR_INPUT
0023     IN_REQ : BOOL;
0024     IN_PRESET : DWORD;
0025     IN_STATUS_1 : BYTE;
0026     IN_STATUS_2 : BYTE;
0027 END_VAR
0028 VAR_OUTPUT
0029     OUT_READY : BOOL;
0030     OUT_BUSY : BOOL;
0031     OUT_VALID : BOOL;
0032     OUT_ERROR : BOOL;
0033     OUT_PRESET : DWORD;
0034     OUT_CTRL : BYTE;
0035 END_VAR
0036 VAR
0037     fb_preset_state : TR_ENUM_PRESET_STATES := TR_PRESET_STATE_INIT;
0038     fb_preset_ready : BOOL := FALSE;
0039     fb_preset_control : BYTE := 0;
0040     fb_preset_error : BOOL := FALSE;
0041     fb_preset_timeout : DWORD := 0;
0042     fb_preset_value : DWORD := 0;
0043 END_VAR
```

- Für die Zustandsvariable `fb_preset_state` wird im Projekt der enumerierte Datentyp `TR_ENUM_PRESET_STATES` angelegt.



- Um die Funktionalität der Preset-Justage-Funktion zu realisieren, müssen folgende Befehle im Programm des Funktionsbausteins erstellt werden.

```
0001 (* process inputs *)
0002 IF ((IN_STATUS_1 AND 16#30) = 16#10) AND
0003    ((IN_STATUS_2 AND 16#FE) = 16#00)
0004 THEN
0005     fb_preset_ready := TRUE;
0006 ELSE
0007     fb_preset_ready := FALSE;
0008 END_IF;
0009
0010 (* process instance data  *)
0011 fb_preset_timeout := fb_preset_timeout + 1;
0012 fb_preset_control := 0;
```

- Die Status-Bytes `IN_STATUS1` und `IN_STATUS2` werden ausgewertet, um die Verfügbarkeit der Preset-Justage-Funktion zu ermitteln. Außerdem werden die Instanzdaten des Funktionsbausteins aktualisiert.

- Die Preset-Justage-Funktion wird in der Form einer Zustandsmaschine realisiert.

```
0014 (* state machine *)
0015 CASE fb_preset_state OF
0016
0017 TR_PRESET_STATE_INIT:
0018     fb_preset_error := FALSE;
0019     fb_preset_timeout := 0;
0020     IF  (IN_REQ = TRUE) THEN
0021         IF(fb_preset_ready = TRUE) THEN
0022             fb_preset_value := IN_PRESET;
0023             fb_preset_state := TR_PRESET_STATE_PREP;
0024         ELSE
0025             fb_preset_error := TRUE;
0026             fb_preset_state := TR_PRESET_STATE_DONE;
0027         END_IF;
0028     END_IF;
0029
0030 TR_PRESET_STATE_PREP:
0031     fb_preset_control := 16#01; (* preset preparation *)
0032     IF((IN_STATUS_1 AND 16#3C) = 16#00) THEN
0033         fb_preset_timeout := 0;
0034         fb_preset_state := TR_PRESET_STATE_REQ;
0035     END_IF;
0036
0037     (* check timeout *)
0038     IF (fb_preset_timeout > 10000) THEN
0039         fb_preset_error := TRUE;
0040         fb_preset_state := TR_PRESET_STATE_DONE;
0041     END_IF;
```

- Der Ausgangszustand ist `TR_PRESET_STATE_INIT`. Wenn ein `IN_REQ` kommt und die Status-Bytes die Bereitschaft des `CD_582_-EPN` anzeigen, wird der Ablauf gestartet.

- Im Zustand `TR_PRESET_STATE_PREP` wird das Kommando `PresetPreparation` per Kontrollbyte an den `CD_582_-EPN` übertragen. Nach Quittierung durch das Mess-System findet ein Zustandswechsel statt. Der Vorgang kann durch einen Timeout abgebrochen werden.

```
0043 TR_PRESET_STATE_REQ:
0044     fb_preset_control := 16#03;  (* preset request & preparation *)
0045     IF((IN_STATUS_1 AND 16#3C) = 16#04) THEN (* success *)
0046         fb_preset_timeout := 0;
0047         fb_preset_state := TR_PRESET_STATE_DONE;
0048     ELSIF ((IN_STATUS_1 AND 16#3C) = 16#08) THEN (* error *)
0049         fb_preset_timeout := 0;
0050         fb_preset_error := TRUE;
0051         fb_preset_state := TR_PRESET_STATE_DONE;
0052     END_IF;
0053
0054     (* check timeout *)
0055     IF (fb_preset_timeout > 10000) THEN
0056         fb_preset_error := TRUE;
0057         fb_preset_state := TR_PRESET_STATE_DONE;
0058     END_IF;
0059
0060 TR_PRESET_STATE_DONE:
0061     IF  (IN_REQ = FALSE) THEN
0062         fb_preset_state := TR_PRESET_STATE_INIT;
0063     END_IF;
0064
0065 ELSE
0066     fb_preset_error := TRUE;
0067     fb_preset_state := TR_PRESET_STATE_DONE;
0068 END_CASE;
```

- Im Zustand `TR_PRESET_STATE_REQ` wird das eigentliche Preset-Kommando `PresetRequest` per Kontrollbyte an den `CD_582_-EPN` übertragen. Nach Quittierung durch das Mess-System findet ein Zustandswechsel statt. Der Vorgang kann durch einen Timeout abgebrochen werden.

- Im Zustand `TR_PRESET_STATE_DONE` wird auf den Abschluss und das Rücksetzen der Preset-Justage-Funktion gewartet. Der Funktionsbaustein wird durch ein Rücksetzen des `IN_REQ` in den Ausgangszustand TR_PRESET_STATE_INIT überführt.

```
0070 (* process outputs *)
0071 OUT_READY := fb_preset_ready;
0072
0073 IF (fb_preset_state = TR_PRESET_STATE_DONE) OR
0074    (fb_preset_state = TR_PRESET_STATE_INIT)
0075 THEN
0076     OUT_BUSY := FALSE;
0077 ELSE
0078     OUT_BUSY := TRUE;
0079 END_IF;
0080
0081 IF (fb_preset_state = TR_PRESET_STATE_DONE) AND
0082    NOT fb_preset_error
0083 THEN
0084     OUT_VALID := TRUE;
0085 ELSE
0086     OUT_VALID := FALSE;
0087 END_IF;
0088
0089 OUT_ERROR := fb_preset_error;
0090 OUT_PRESET := fb_preset_value;
0091 OUT_CTRL := fb_preset_control;
```

➢ Am Ende des Funktionsbausteins werden die Zustände der Ausgangsvariablen je nach Ablauf gesetzt.

Im Hauptprogramm müssen die Ein- und Ausgänge des Funktionsbausteins mit den entsprechenden PROFIsafe-Ein- bzw. Ausgangsdaten verbunden werden.

```
0010 (* preset *)
0011 stTrPresetData.bReq;
0012 stTrPresetData.dwPreset;
0013
0014 fbTrPreset(IN_REQ := stTrPresetData.bReq,
0015            IN_PRESET := stTrPresetData.dwPreset,
0016            IN_STATUS_1 := tr_safe_in_status1,
0017            IN_STATUS_2 := tr_safe_in_status2);
0018
0019 stTrPresetData.bReady := fbTrPreset.OUT_READY;
0020 stTrPresetData.bValid := fbTrPreset.OUT_VALID;
0021 stTrPresetData.bBusy := fbTrPreset.OUT_BUSY;
0022 stTrPresetData.bError := fbTrPreset.OUT_ERROR;
0023
0024 (* outputs *)
0025 tr_safe_out_preset_hi :=TR_FUN_DWORD_TO_PRESET_HI(fbTrPreset.OUT_PRESET);
0026 tr_safe_out_preset_lo := TR_FUN_DWORD_TO_PRESET_LO(fbTrPreset.OUT_PRESET);
0027 tr_safe_out_control1 := fbTrPreset.OUT_CTRL;
0028
```

- Im Hauptprogramm werden die Status-Bytes 1 und 2 von den PROFIsafe-Prozesseingängen an den Funktionsbaustein übergeben. Außerdem werden der Preset-Vorgabewert und das Kontrollbyte den entsprechenden PROFIsafe-Prozessausgangsdaten übergeben.

- Das Doppelwort des Preset-Justage-Vorgabewerts OUT_PRESET wird mittels zweier Hilfsfunktionen in zwei Worte aufgeschlüsselt und den Prozessdatenausgängen `tr_safe_out_preset_hi` bzw. `tr_safe_out_preset_lo` zugewiesen.

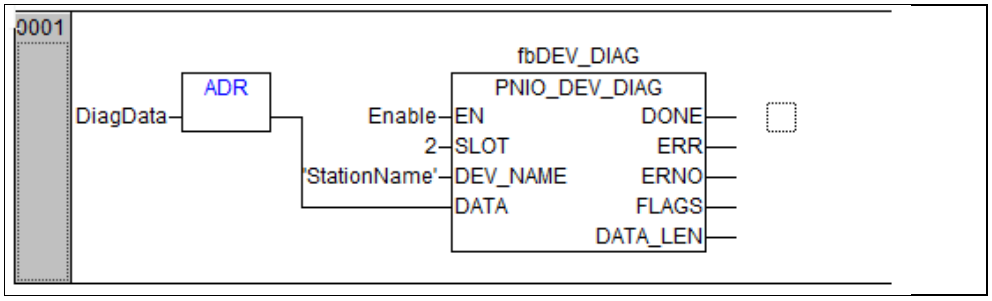
- Das Kontrollbyte `OUT_CTRL` des Preset-Funktionsbausteins wird dem Prozessdatenausgang `tr_safe_out_control1` zugewiesen.

6.2 Fehlerauswertung

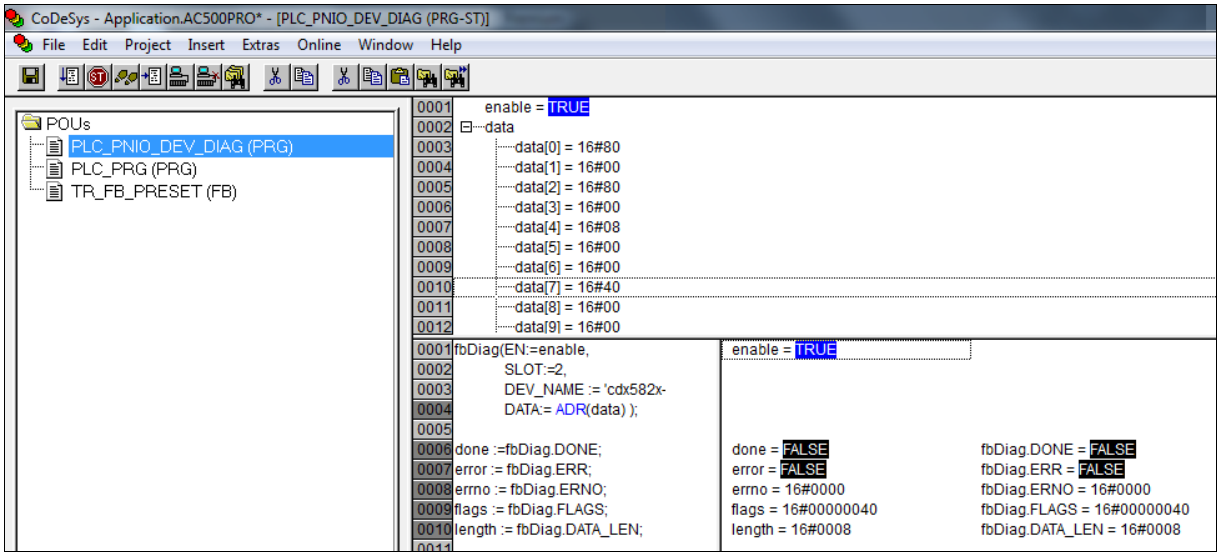
Das Mess-System liefert im Fehlerfall eine Diagnosenachricht. Diese kann im `Non-Safety-Programm` aus dem Kommunikationsmodul `CM579-PNIO` ausgelesen und ausgewertet werden.

6.2.1 Anzeige der Diagnosenachrichten

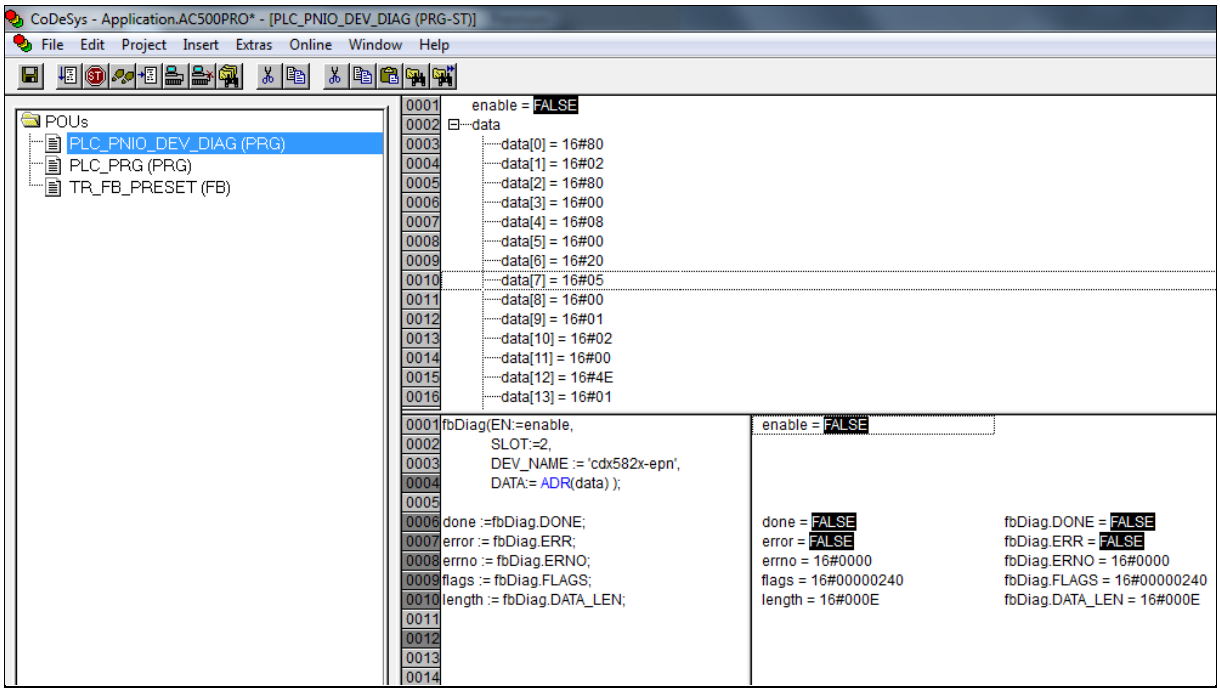
- Um die Diagnosemeldungen via PROFINET auszulesen, benötigen Sie den Funktionsbaustein `PNIO_DEV_DIAG`. Dieser liest die Standard-Diagnosemeldung eines Devices aus. Der Funktionsbaustein für Non-Safety-Programme befindet sich in der System-Library `Profinet_AC500_V13.lib`.
- Zur Auswertung der Diagnosemeldungen muss eine Instanz des Bausteins angelegt und im Hauptprogramm aufgerufen werden. Wenn der Enable-Eingang `EN` des Funktionsbausteins gesetzt wird, werden in den bereitgestellten Datenpuffer Diagnosemeldungen kopiert – falls vorhanden.



- Der Eingangsparameter `SLOT` spezifiziert den Steckplatz des `CM579-PNIO` in der Hardware-Konfiguration. Im Kontext dieses Beispiels steckt das Kommunikationsmodul im Slot 2.
- Der Eingangsparameter `DEV_NAME` erhält den Stationsnamen des auszulesenden Geräts – hier `cdx582x-epn`.



- Einfache Channel-Diagnosen beginnen mit dem HEX-Code `0x8000`; unter anderem steht in den Bytes [6] und [7], der gesuchte Fehler-Code; hier: `0x0040`, d.h. „Fehlerhafte Safety-Zieladresse (F_Dest_Add)“



- Erweiterte Channel-Diagnosen beginnen mit dem HEX-Code 0x8002; unter anderem steht in den Bytes [6] und [7], der gesuchte Fehler-Code; hier: 0x2005, d.h. „Fehler im MT-Abtastung - Kanal 1“. Der erweiterte detaillierte Fehler-Code steht dann in den Bytes [12] und [13]. Der erweiterte Fehler-Code dient der detaillierten Fehleranalyse im Werk.

6.3 Mess-System - Passivierung und Operator Acknowledge

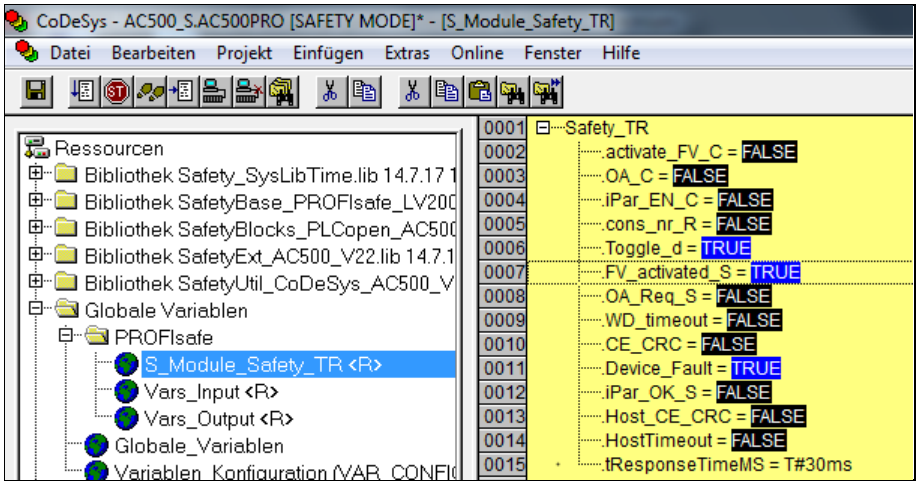
6.3.1 Nach Anlauf des F-Systems

Nach einem Anlauf des F-Systems muss die Kommunikation zwischen F-CPU und Mess-System über das PROFIsafe-Protokoll erst aufgebaut werden. In dieser Zeit erfolgt eine Passivierung des Mess-Systems mit der Verwendung der Ersatzwerte (0).

Dauert der Aufbau der Kommunikation zwischen F-CPU und Mess-System länger als die im `Automation Builder` für die F-Peripherie eingestellte Überwachungszeit, so erfolgt keine automatische Anwenderquittierung (Operator Acknowledge).
In diesem Fall ist eine Anwenderquittierung mit positiver Flanke an der Variable `OA_C` der globalen Variablen der F-Peripherie erforderlich, welche im Beispielprojekt mit Unterprogramm `TR_FUN_ACK_REI` angesteuert wird.

6.3.2 Nach Kommunikationsfehlern

Wird vom F-System ein Fehler in der sicherheitsgerichteten Kommunikation zwischen der F-CPU und Mess-System über das PROFIsafe-Protokoll erkannt, erfolgt eine Passivierung des Mess-Systems.
Während der Verwendung der Ersatzwerte (0) ist die Variable `FV_activated_S` gesetzt.



Die Anwenderquittierung (Operator Acknowledge) des Mess-Systems, d.h. die Ausgabe von zyklischen Daten zu den fehlersicheren Ausgängen erfolgt erst dann, wenn:

- kein Kommunikationsfehler mehr vorhanden ist
- eine Anwenderquittierung mit positiver Flanke an der Variable `OA_C` der globalen Variablen der F-Peripherie erfolgt ist.

Im Beispielprojekt wird dies mit dem Unterprogramm `TR_FUN_ACK_REI` realisiert.

7 Software-, Beispiel- und Bibliotheken-Download

- **GSDML des CD_582_-EPN für ABB AC500-S Steuerung:**

  www.tr-electronic.de/f/zip/TR-ECE-ID-MUL-0058

- **Software TR TCI Device Tool zur CRC-Berechnung:**

  www.tr-electronic.de/f/zip/TR-ECE-SW-MUL-0008

- **Beispiel Projekt für ABB AC500-S Steuerungssystem:**

  www.tr-electronic.de/f/zip/TR-ECE-SW-MUL-0017

Absolute Encoder CD_582_-EPN
PROFINET IO / PROFIsafe

Parameterization with ABB AC500-S PM583/SM560 control systems

CDV582

CDS582 / CDH582

Pictures are similar

_**Creating the safety program**
– Sample configuration
_**Accessing the safety-oriented data channel**
_**Setting the parameters / CRC calculation**

Technical Information

TR-Electronic GmbH
D-78647 Trossingen
Eglishalde 6
Tel.: (0049) 07425/228-0
Fax: (0049) 07425/228-33
E-mail: info@tr-electronic.de
http://www.tr-electronic.de

Copyright protection

Subject to modifications

The right to make any changes in the interest of technical progress is reserved.

Document information

Release date / Rev. date: 04/24/2020
Document / Rev. no.: TR - ECE - TI - DGB - 0366 - 01
File name: TR-ECE-TI-DGB-0366-01.docx
Author: KUC

Font styles

Italic or **bold** font styles are used for the title of a document or are used for highlighting.

`Courier` font displays text, which is visible on the display or screen and software menu selections.

″ < > ″ indicates keys on your computer keyboard (such as <RETURN>).

Brand names

PROFIBUS™, PROFINET™ and PROFIsafe™, as well as the associated logos, are registered trademarks of PROFIBUS Nutzerorganisation e.V. (PNO).

Contents

Revision index

Modification	Datum	Index
First release	04/16/20	00
General adjustments after ABB feedback	04/24/20	01

1 General information

This “Technical Information” addresses the following topics:

- Setting the parameters / CRC calculation
- Creating the safety program
- Accessing the safety-oriented data channel

The “Technical Information” is available as a separate document.

1.1 Scope

This “Technical Information” only applies to the measuring system model ranges featuring a ***PROFINET IO*** interface and a ***PROFIsafe*** Profile in connection with an ABB AC500-S controller of the PM583/SM560 series:

- CDV-582
- CDS-582
- CDH-582

The products are labeled with affixed nameplates and are components of a system.

The following documentations therefore apply as well:

- ABB Manual *AC500-S Safety User Manual V1.1.0*
  (Document order number: *3ADR025091M0207*),
- ABB Manual *AC500 PLC - System Assembly and Device Specifications for AC500 V2 Products*
  (Document order number: 3ADR010121),
- ABB online help for *Automation Builder V2.2.4*
- see Chapter “Other Applicable Documents” in the Safety Manual
  www.tr-electronic.de/f/TR-ECE-BA-GB-0142
- and this optional “Technical Information”

2 Safety instructions

2.1 Definition of symbols and notes

⚠ DANGER	means that death or serious injury will occur if the user fails to take the respective precautionary measures.
⚠ WARNING	means that death or serious injury can occur if the required precautions are not met.
⚠ CAUTION	means that minor injuries can occur if the required precautions are not met.
NOTICE	means that damage to property can occur if the required precautions are not met.
	indicates important information or features and application tips for the product used.

2.2 Organizational measures

Prior to commencing work, personnel handling the measuring system must have read and understood the Safety Manual (TR-ECE-BA-GB-0142), in particular Chapter “Basic safety instructions”.

2.3 Personnel qualification

The measuring system may only be configured by qualified specialist personnel; see ABB Safety Manual.

2.4 Terms of use for the software examples

⚠ WARNING	TR-Electronic GmbH assumes no liability or warranty for the error-free operation of the safety program and the application examples.
NOTICE	The software examples offered for download are exclusively for demonstration purposes and users use such at their own risk.

3 GSDML

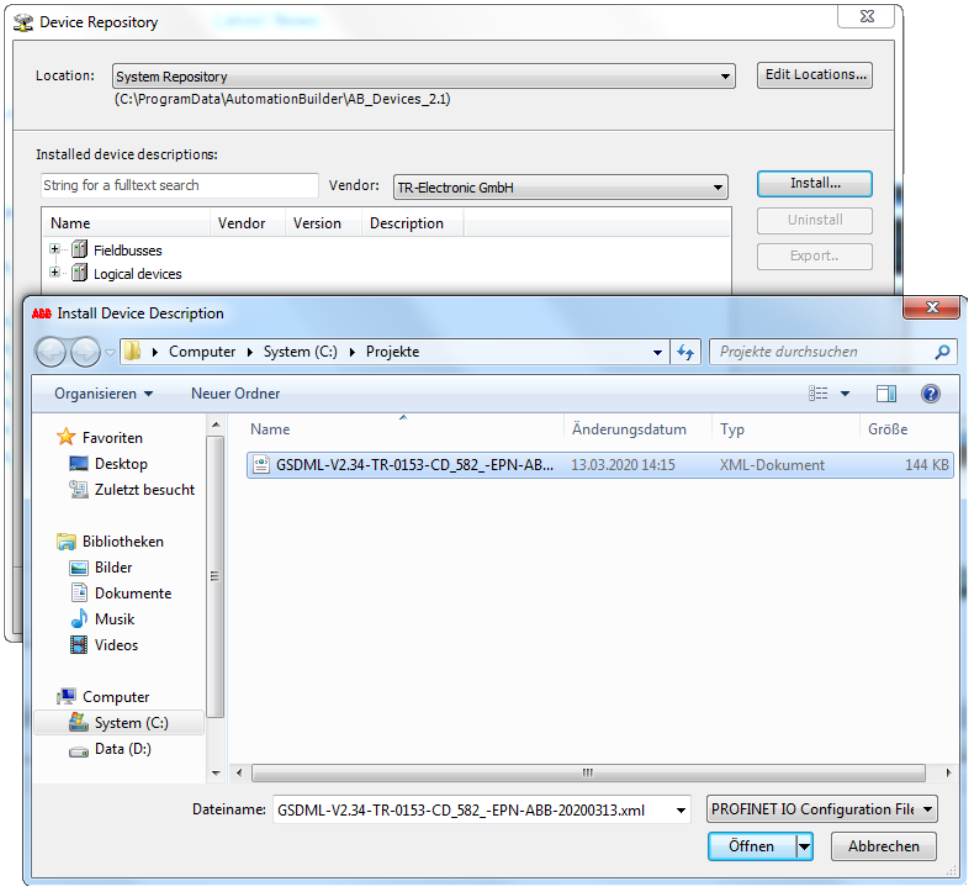
A device description file (GSDML) adapted to the project planning software must be used for configuring the CD_582_-EPN with the ABB project planning software Automation Builder V2.1.

The GSDML has the identifier “ABB” in the file name, e.g. “GSDML-V2.34-TR-0153-CD_582_-EPN-ABB-xxxxxxxx.xml”.

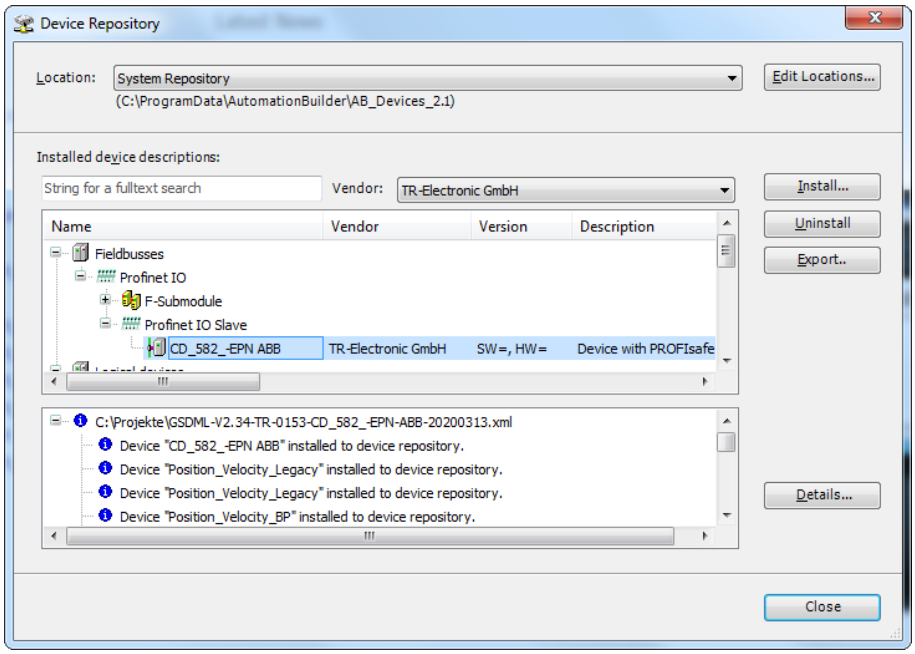
Download from www.tr-electronic.de/f/ZIP/TR-ECE-ID-MUL-0058

The configuration is only possible with an adapted GSDML file.

Configuring the CD_582_-EPN in an Automation builder project requires the device’s GSDML file to be installed in the Device Repository.
The Device Repository editor is opened via the Automation builder menu bar under the menu item Tools -> Device Repository. The CD_582_-EPN GSDML must be installed in the System Repository so that devices of this type can be configured in projects.



After the installation, CD_582_-EPN is available as a device under “PROFINET IO Slaves”.



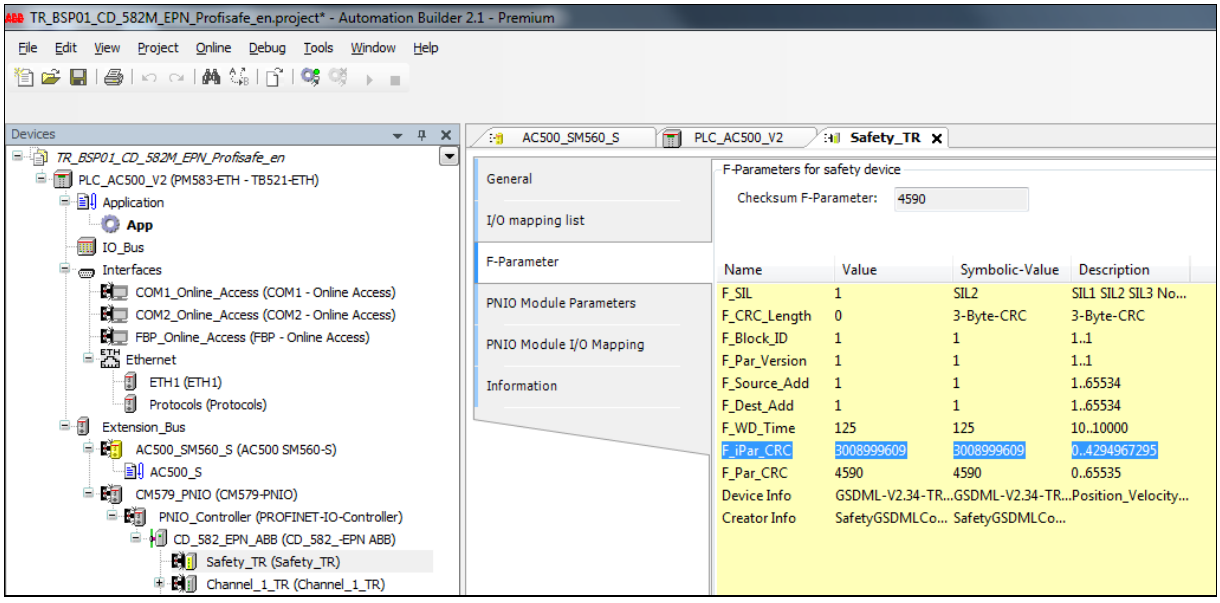
4 Setting the parameters / CRC calculation

It is advisable to define known parameters in the F-host before the actual project planning so that such parameters will be considered during project planning.

Below is a description of the procedure used for the ABB project planning software `Automation Builder V2.1` and the option package `Safety DM220-FSE`.
Download the `TR TCI Device Tool` software required for CRC calculations as indicated in Chapter: 7 “Software, Sample and Library download” on page 112.
Install and use the software `TR TCI Device Tool` as described in the TR-ECE-TI-DGB-0327 manual.

4.1 iParameter

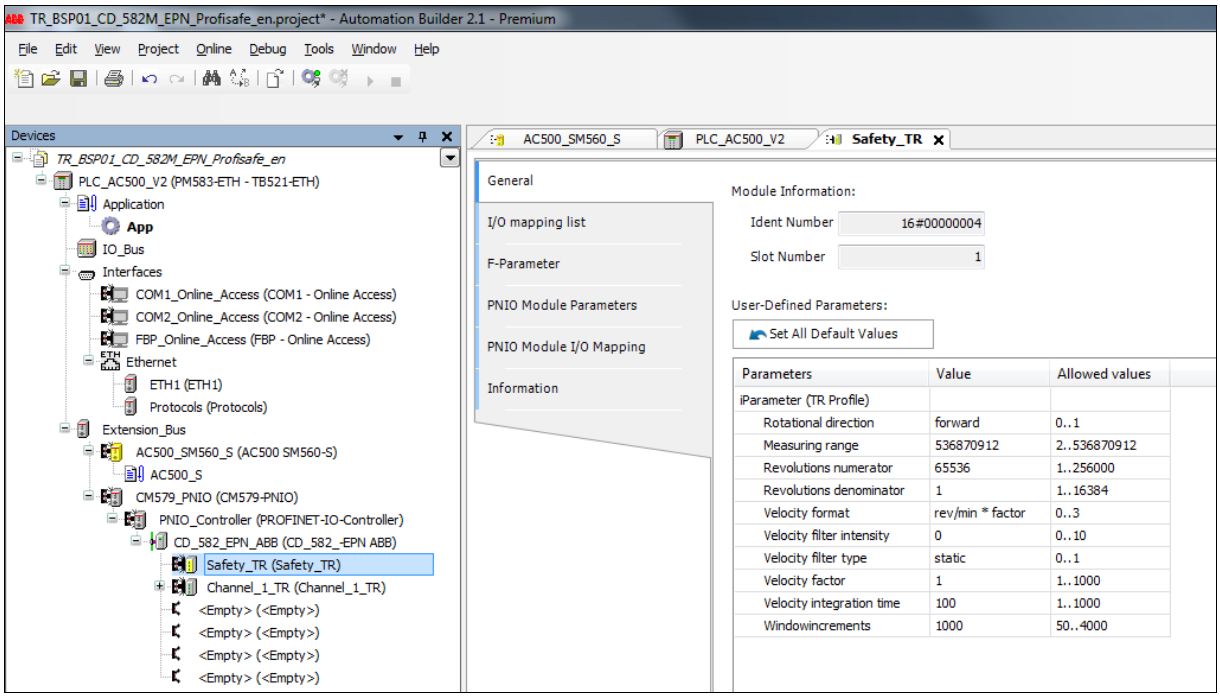
The iParameters of the default setting are preset with meaningful values and should only be changed if the automation task expressly requires this. The secure transmission of individually set iParameters requires a CRC calculation. It must be performed using the TR-program “TR TCI Device Tool” when the preset iParameters are changed. The checksum calculated in this way corresponds to the F-parameter `F_iPar_CRC`. It must be entered in the field `F_iPar_CRC` when configuring the measuring system. In Configuration View, the field `F_iPar_CRC` of the configured safety module is available under the tab `F-parameter`;
see also Chapter “Setting the iParameters” on page 86.



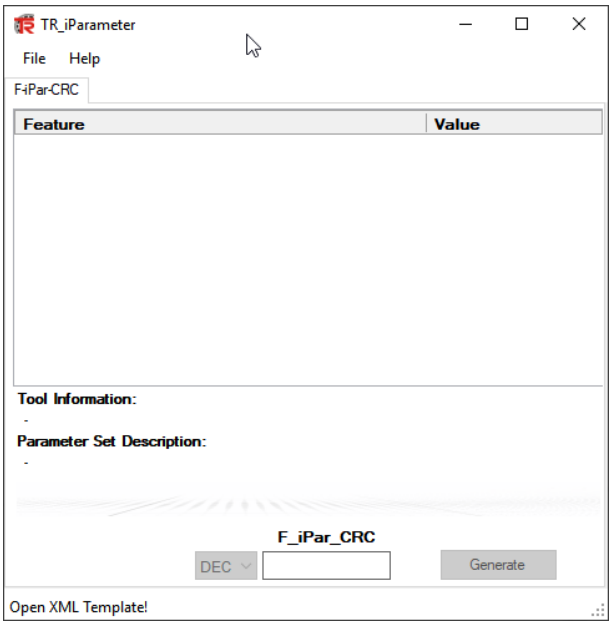
4.1.1 CRC calculation using iParameters

The preset standard values are used for the following sample CRC calculation.

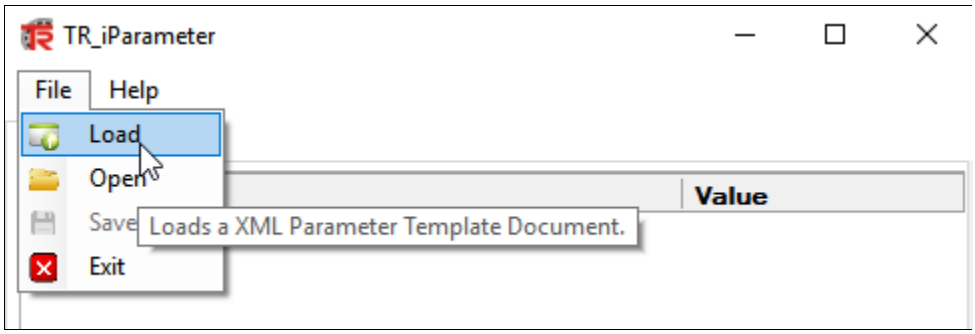
➢ Open the safety module `Configuration View` in the IO configuration of the measuring system to change the iParameters. Then edit the module's iParameters in the table under the `General` tab.



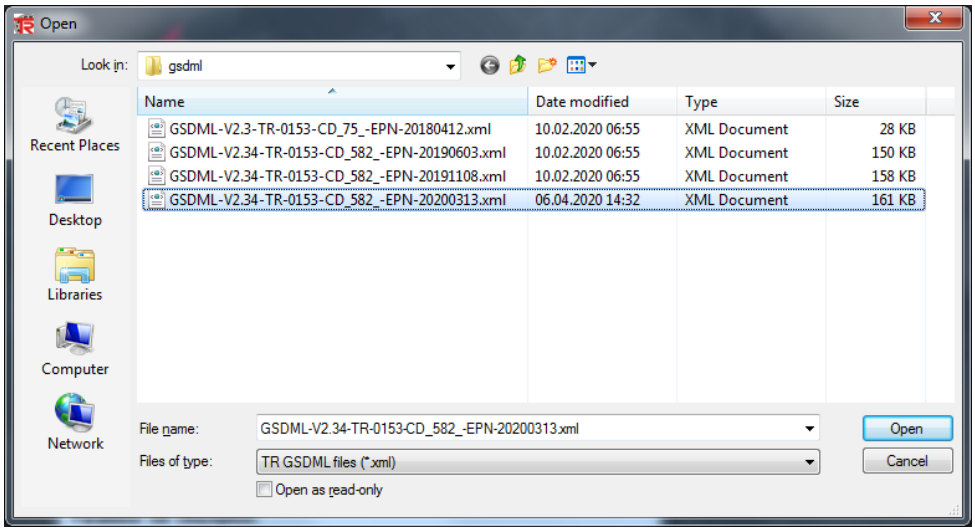
➢ Start the program `TR TCI Device Tool` for the CRC calculation. The `TR TCI Device Tool` is a standalone Windows application that must be started outside and independent of the `ABB Automation Builder`.



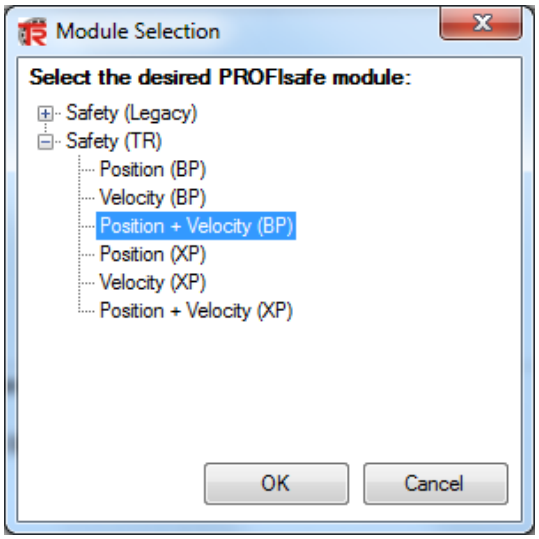
➢ Load the `CD_582_-EPN` GSDML after starting the `TR TCI Device Tool`. To do so, select the menu item `File -> Load`.



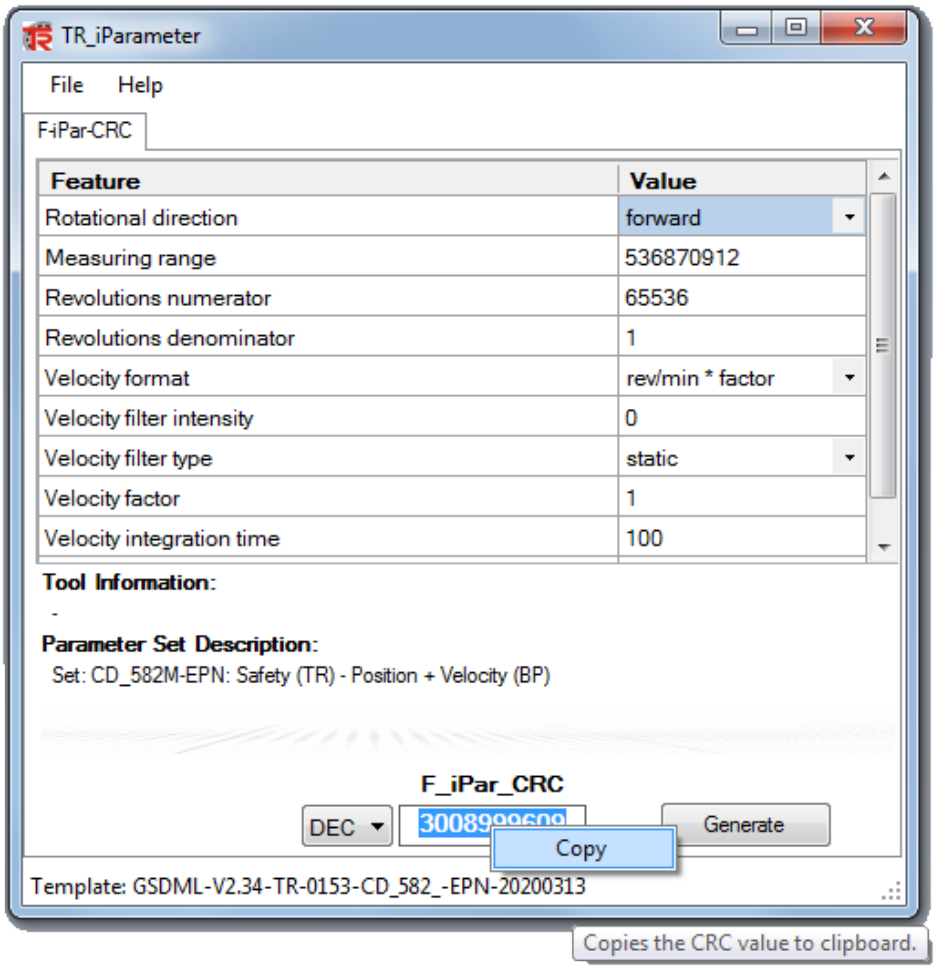
➢ Selects the customized GSDML as a template and open it in the file selection dialog.



➢ The PROFIsafe module determines the composition of the parameter set to be configured. Select the module according to the desired IO profile. This manual always refers configuring the TR profile.

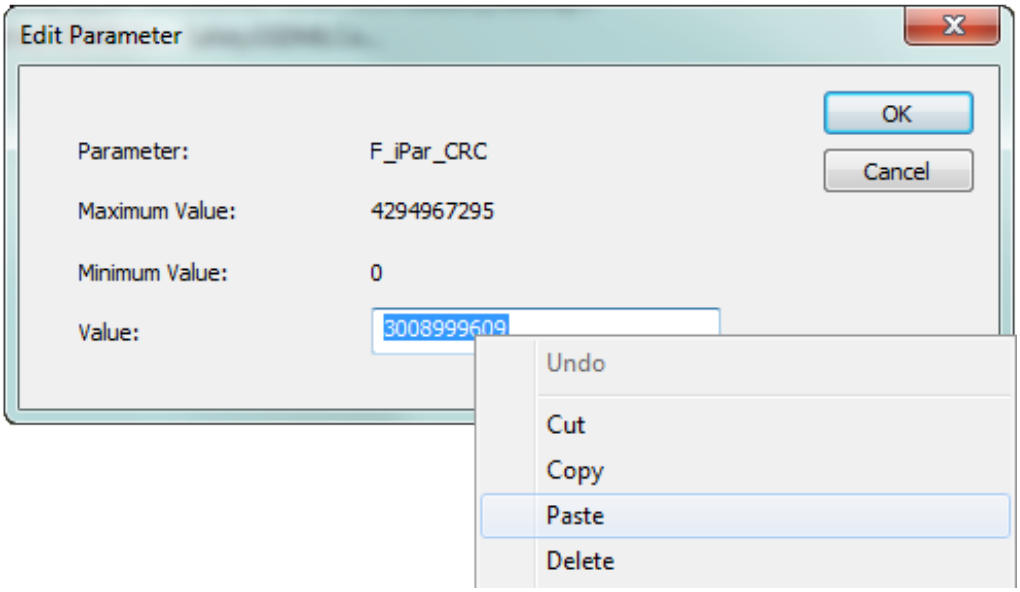


➢ After opening the parameter set, set the CRC number format selection button to the value `DEC` and confirm it with the `Generate` button.



The table settings must match the configuration tool settings.

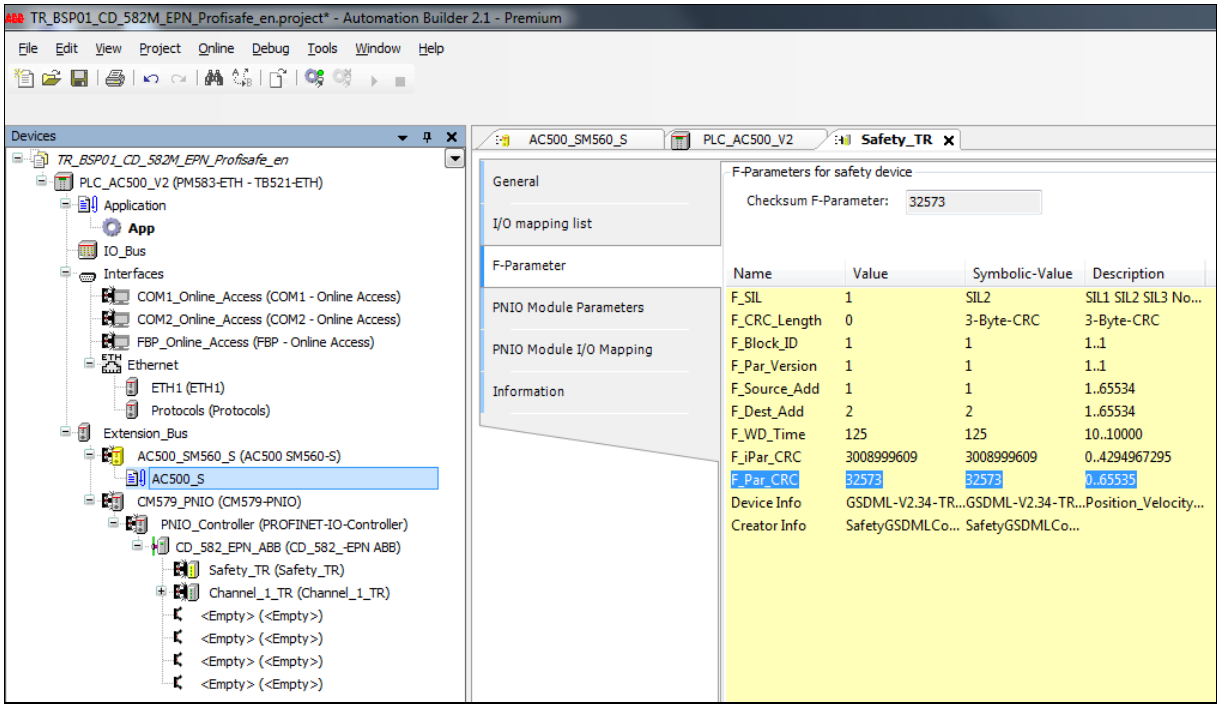
➢ Select the calculated value with the right mouse button and copy it to the clipboard. Switching to the `Automation Builder`. In the safety module `Configuration View`, double click the field `F_iPar_CRC` under the `F-parameter` tab to select it and insert the calculated value in the next editing dialog.



Every parameter change requires `F_iPar_CRC` to be newly calculated in the `TR TCI Device Tool`, which must be newly transferred to the configuration tool. The configuration data must be regenerated if a safety program and the corresponding configuration already exist. The new `F_iPar_CRC` value and the changed parameters must be entered into the `Automation Builder` during configuration. See Chapter: 5.3.1 “Setting the iParameters” on page 86 and Chapter: 5.3.2 “Setting the F-parameters” on page 87.

4.2 F-Parameter

The F-parameters (except `F_Dest_Add`) of the default setting are preset with meaningful values and should only be changed if the automation task expressly requires this. A CRC is required for the secure transmission of the individually set F-parameters, which is automatically calculated by `Automation Builder`. This checksum corresponds to the F-parameter `F_Par_CRC`, which is displayed in the safety module `Configuration View` under the `F Parameter` tab when configuring the measuring system. See also Chapter “Setting the F-parameters” on page 87.



4.2.1 Non-settable F-parameters

The F-parameters listed below are managed by the measuring system or by the F-host and can therefore not be changed manually:

- F_CRC_Length: 3-Byte-CRC
- F_Block_ID: 1
- F_Par_Version: 1 (V2-mode)

4.2.2 Settable F-parameters

It is assumed that the following parameters have been assigned their default values:

- F_SIL: SIL2
- F_Source_Add: 1 (F-host address)
- F_Dest_Add: 1 (address switch)
- F_WD_Time: 125
- F_iPar_CRC: 3008999609 (calculation using the TR TCI Device Tool)

The `Automation Builder` calculates a new `F_Par_CRC` value after each parameter change, which is entered and displayed as shown above. The configuration data must be regenerated if a safety program and the corresponding configuration already exist.

5 Creating a control program – sample configuration

This chapter describes the procedure for creating the safety program using the ABB configuration software Automation Builder V2.1 and the Safety option package DM220-FSE.

The safety program is created with the CoDeSys program editor (yellow background) in the Automation Builder. This requires an AC500 SM560-S node to be included in the project configuration tree. The fail-safe PRGs, FBs and FUNs are programmed in the IEC61131 programming languages ST, FUP or KOP. The ABB Safety option package DM220-FSE includes fail-safe application blocks that can be used in the safety program.

Automatic safety checks are performed when the safety program is generated, and additional fail-safe blocks are installed for error detection and PROFIsafe frame processing. This ensures that failures and errors are recognized, and appropriate reactions are triggered that keep or bring the F-system in a safe state.

Only the safety program (safety application) runs in the F-CPU SM560-S; on the other hand, the standard user program (non-safety application) runs on the central processing unit AC500 PM583-ETH. The non-safety application is programmed using a separate CoDeSys program editor (white background). To do this, the user must insert a separate application node into the project. The configuration data of both programs must be kept consistent because all IO data run via the central processing unit and are forwarded from there to the F-CPU.

The fail-safe process map of the PROFIsafe inputs and PROFIsafe outputs can be accessed from the non-safety program via appropriate I/O mappings.

Access protection
Access to the Automation Builder F-system is protected by a password that applies to both the IO configuration and the safety program.

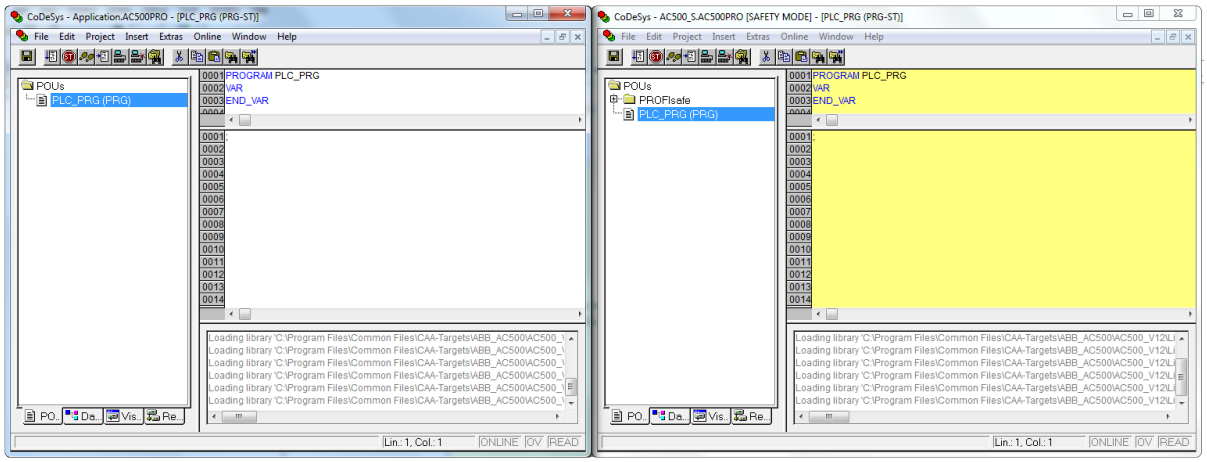


Figure 1: CoDeSys program editor (white) and Safety CoDeSys program editor (yellow)

5.1 Requirements

⚠ WARNING ***Risk of fail-safe function deactivation if the safety program is configured improperly!***

- The safety program may only be created based on the system documentation ABB supplies with its software and hardware.
- ABB's ***ACC500-S Safety User Manual V1.1.0*** (document order number: ***3ADR025091M0207***) provides comprehensive documentation on "Configuring and Programming" a safe controller. This documentation is part of the product documentation and is available for download.
- The descriptions below only refer to the steps required, without taking all instructions from the ABB manual into account. It is essential to observe and comply with the information and instructions provided in the ABB manual, particularly the safety instructions and warnings.
- The configuration shown is only a sample. Users must therefore verify whether the configuration is appropriate for their application and adjust it as needed. This also includes selecting the appropriate safety-oriented hardware components and defining the software requirements.

Software components used for the `AC500-S` sample configuration:

- Automation Builder V2.1
- Safety Add-on DM220-FSE

AC500-S series hardware components used for the `AC500-S` sample configuration:

- DIN rail TB521-ETH A2 (1SAP112100R0270)
- CPU PM583-ETH A9 (1SAP140300R0271)
- F-CPU SM560-S A5 (1SAP280000R0001)
- Profinet IO Controller CM579-PNIO (1SAP170901R0101)

CD_582M series hardware components used for the `AC500-S` sample configuration:

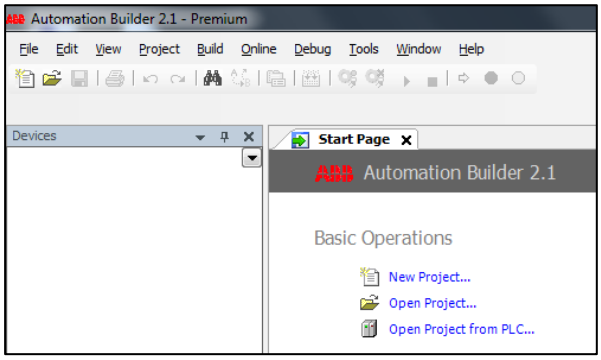
- CDH582M-00002 (CDH582M*8192/65536 EPN NTS 12H7 + FS2)

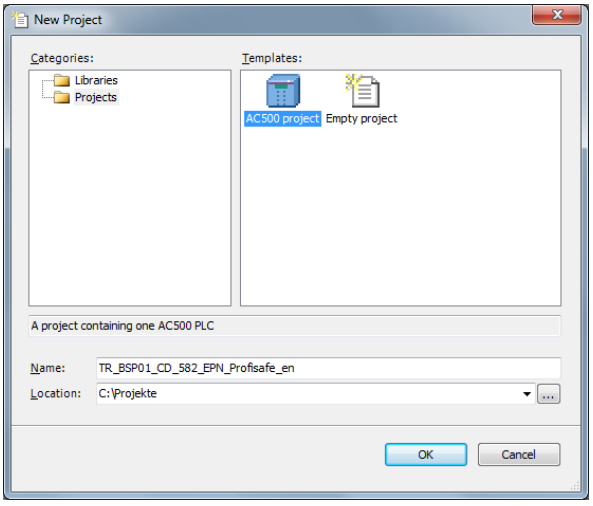
Figure 2: ABB AC500-S series hardware components

Figure 3: TR-Electronic CD_582M series hardware component

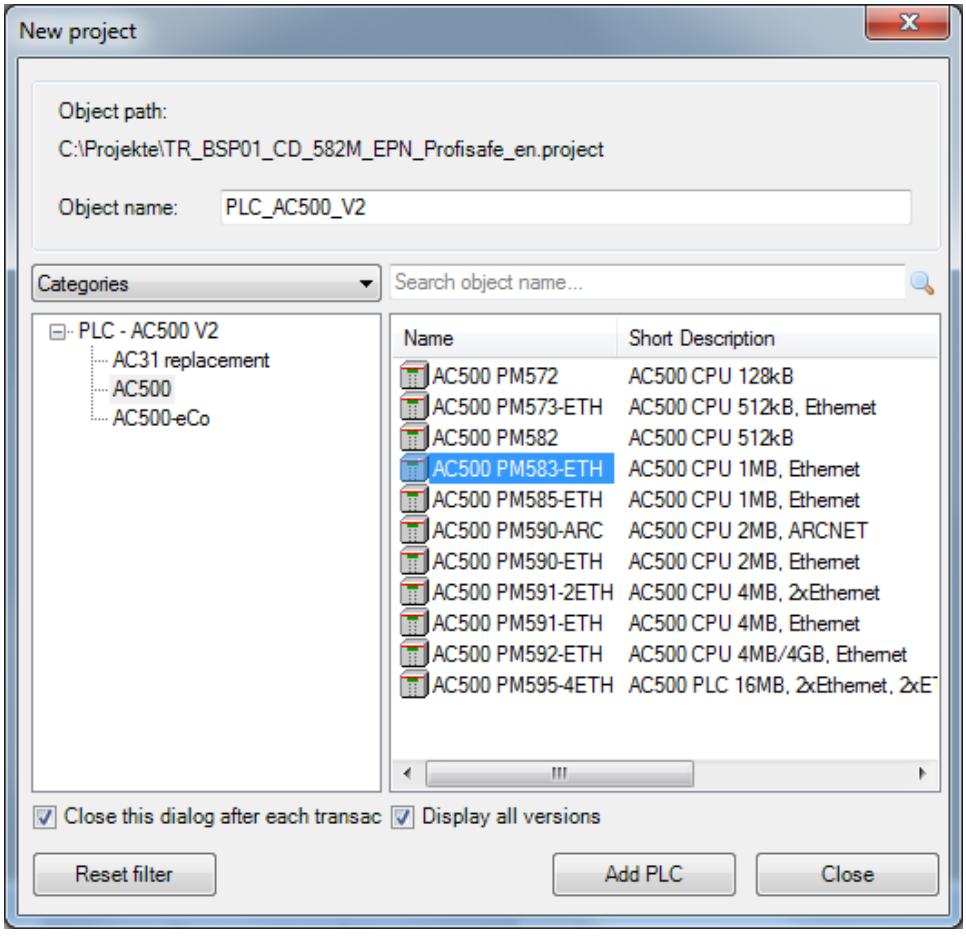
5.2 Hardware Configuration

➢ Start the ABB `Automation Builder` V2.1 and create a new `AC500 project`.

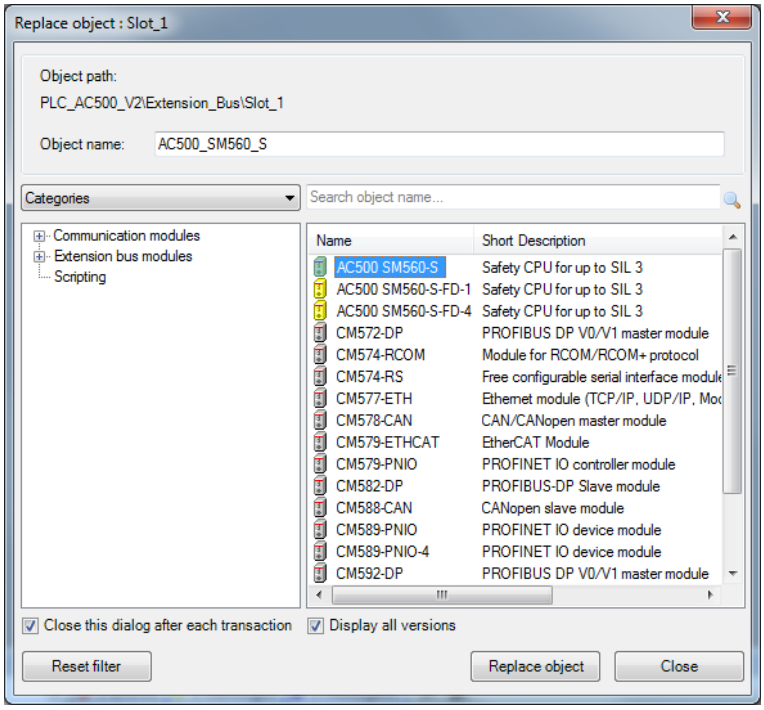




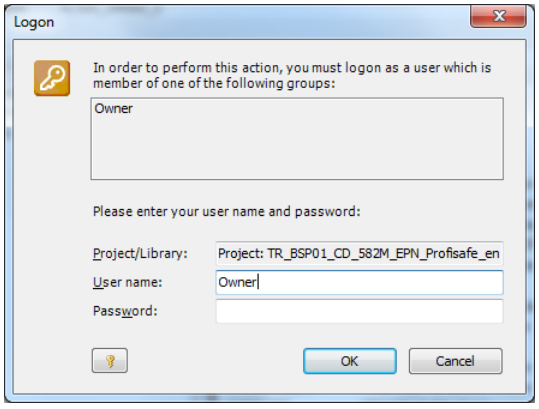
➢ Select the `AC500 CPU PM583-ETH` as PLC head-end station and add it to the project.



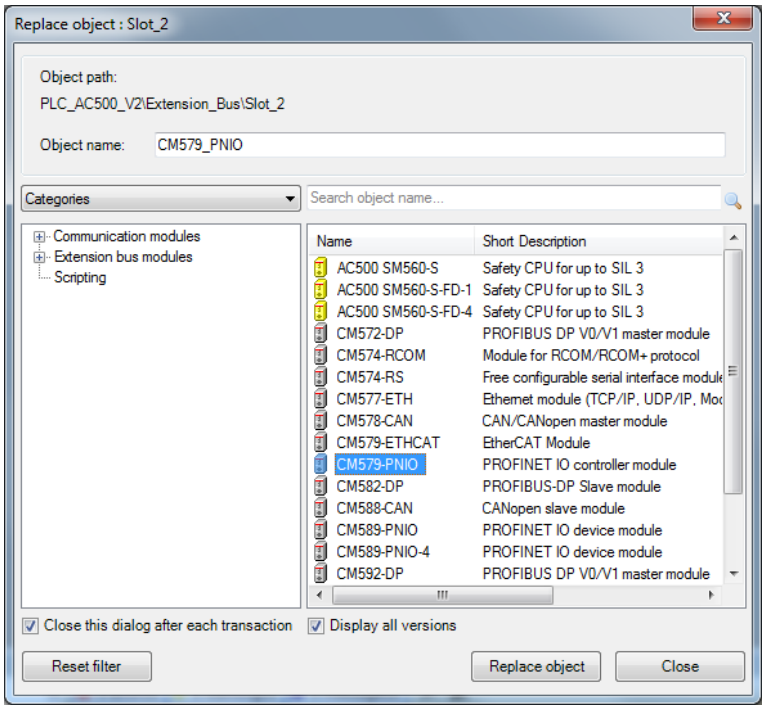
- After the PLC head-end station is selected, `Automation Builder` creates an empty device configuration tree on the left side of the main window in the device view.
- The two missing ABB hardware components mounted on the DIN rail must be inserted under the `Extension_Bus` branch. Select the option `Add object` from the context menu to insert the two objects.
- Add the F-CPU `SM560-S` in the Extension Slot 1.



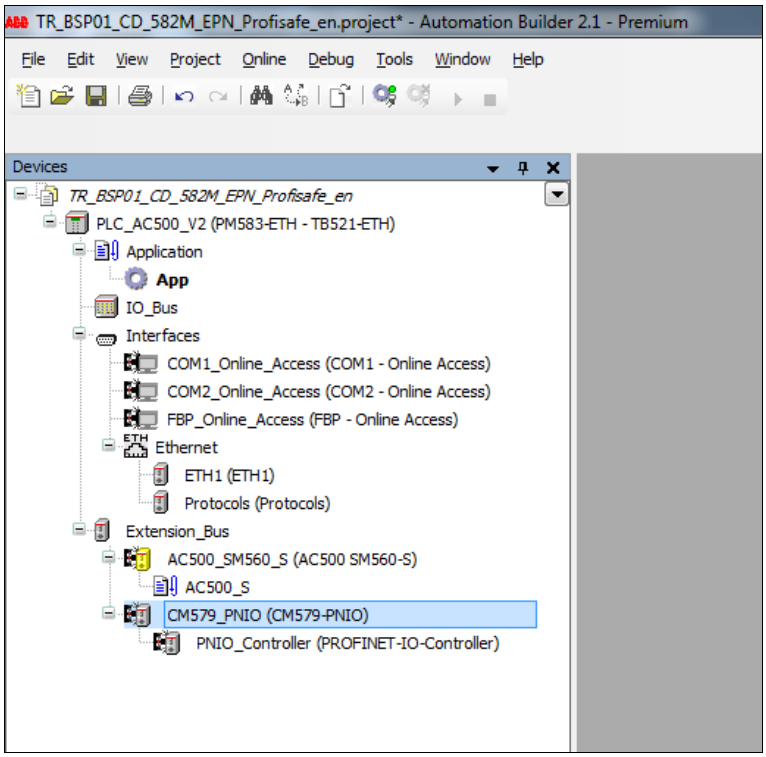
- `Automation Builder` requests a user name and a corresponding password when the F-CPU is added to the configuration tree. In this example, the user is always “Owner” and the password is always an empty string.



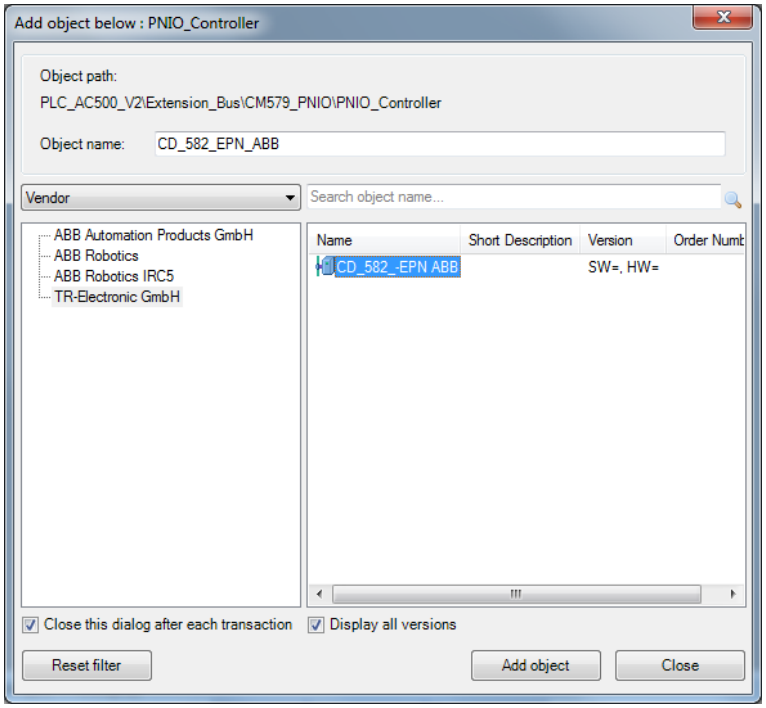
➢ Add the communication module CM579-PNIO in the Extension Slot 2.



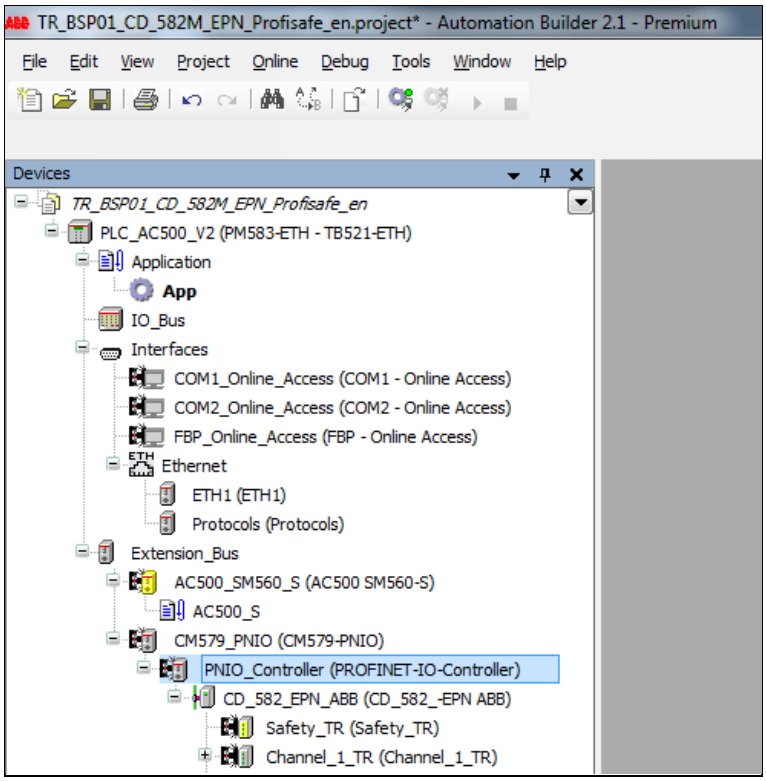
➢ Now the device configuration tree contains the F-CPU with a program node for the safety program and the Profinet IO communication module CM579-PNIO with the corresponding PNIO controller node.



➢ To complete the hardware configuration, add the `CD_582_-EPN` measuring system object under the PNIO controller. But first, install the GSDML in the `Device Repository` (see Chapter 3).

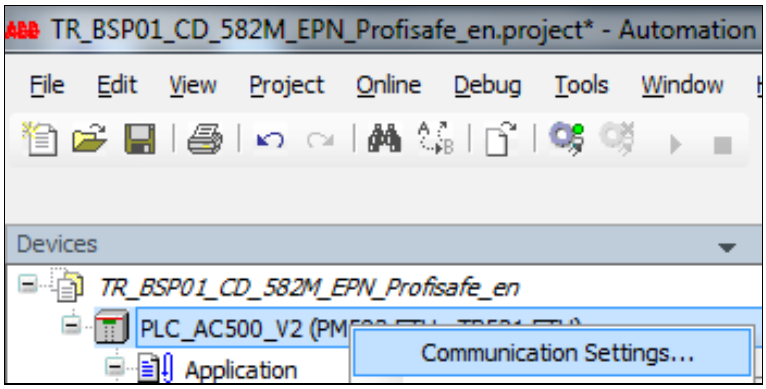


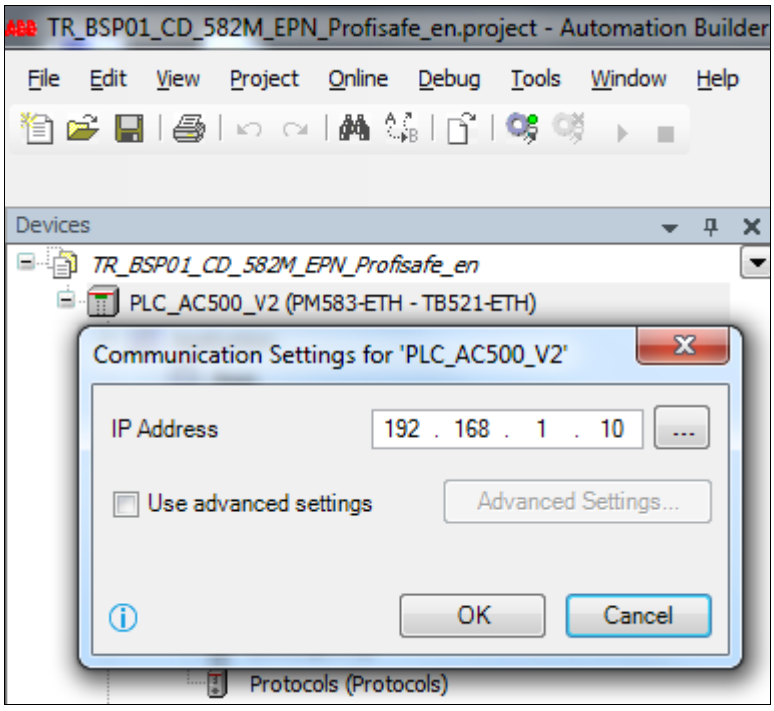
➢ Once the object has been added, an `CD_582_-EPN` object with the default IO configuration hangs below the PNIO controller.



5.2.1 Controller communication settings

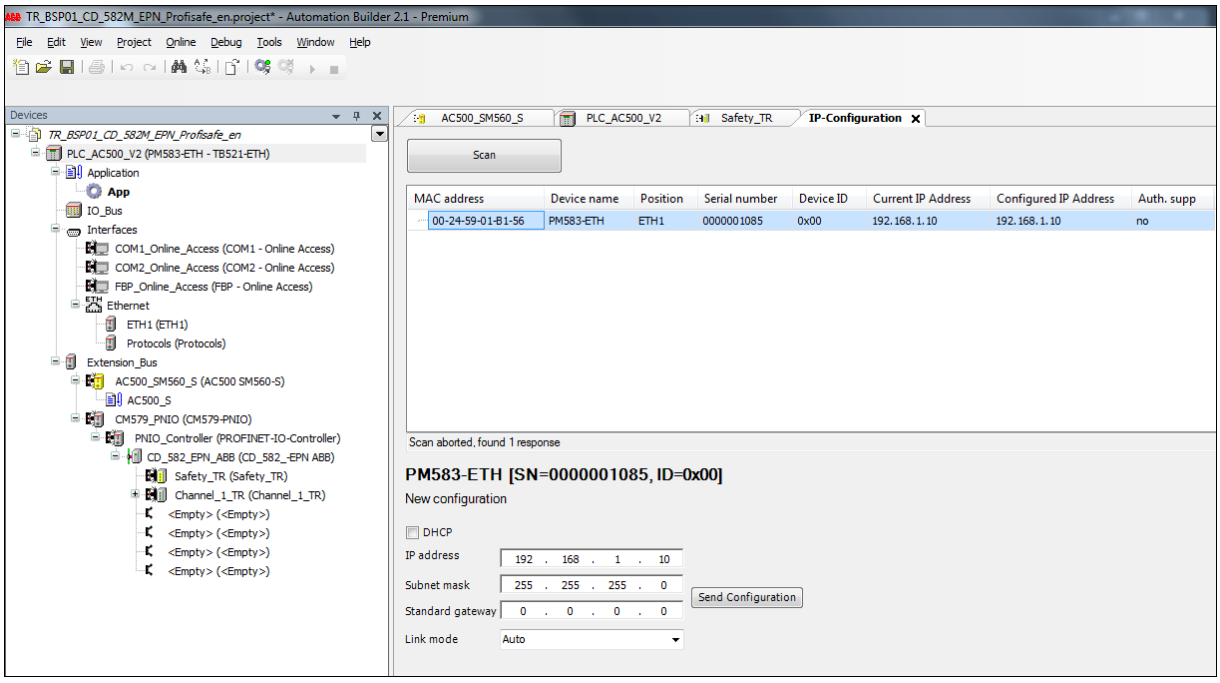
- The program and the IO configuration are transmitted from the `Automation Builder` to the central processing unit`PM583-ETH` via LAN. The project communication settings must match the `PM583-ETH` IP configuration. The user may have to adjust the controller settings to the programming device settings.
- The user specifies the target IP address of the central processing unit to be configured in the communication settings. The communication settings can be edited in the context menu of the PM583-ETH `PLC_AC500_V2` object node.





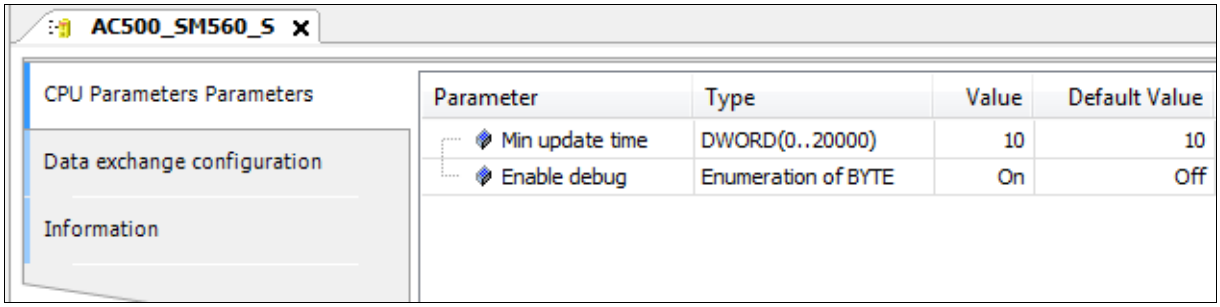
The LAN connection for programming the controller and for diagnosing the control program is always established via the LAN socket of the `TB521-ETH` DIN rail, which is connected to the `PM582-ETH` CPU.

- The user can verify the controller communication settings with the `IP configuration` tool available under the menu item `Tools -> IP configuration`.
- After pressing the `Scan` button, every accessible network user is listed with the currently valid IP configuration.



- The `F-CPU SM560-S` can only be programmed in Debug mode. The `Enable debug` parameter must thus be set to `On` in the `SM560-S` Configuration view under the `CPU parameters` tab.

The Debug mode must be activated before the first download to the F-CPU. A PM583-ETH boot project must have been loaded and saved beforehand as well.

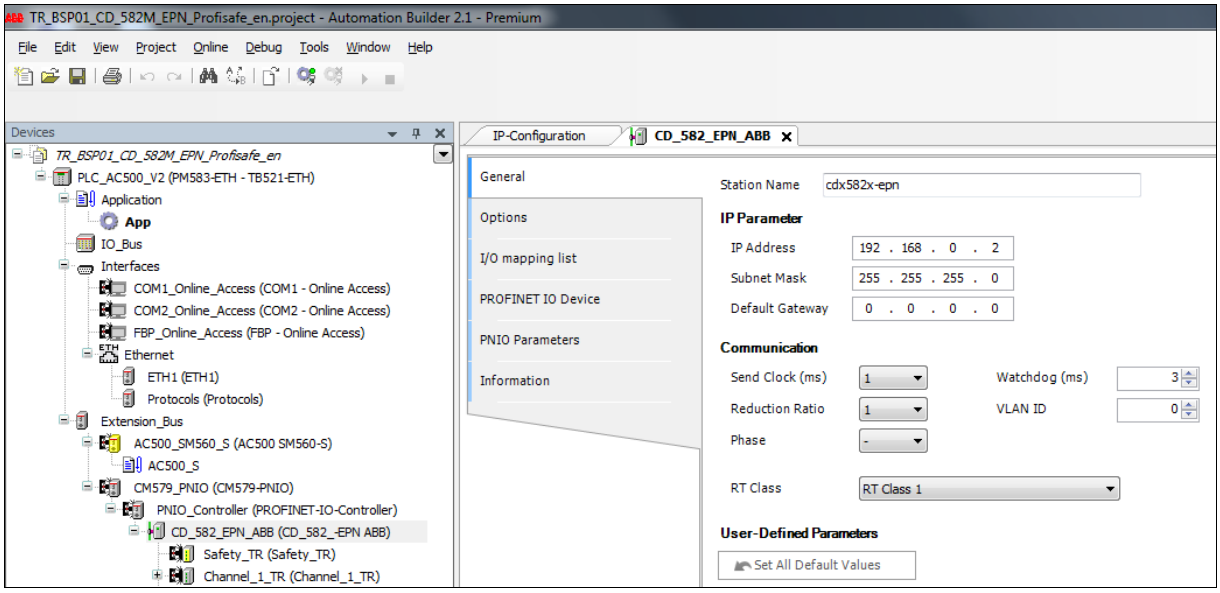


5.2.2 Measuring system communication settings

The measuring system must have been assigned a valid `Station Name`(Profinet IO device name) and `F-Dest-Address` (F parameter) for it to be operated on the control network.

The measuring system has no Station Name stored when delivered and after a reset.

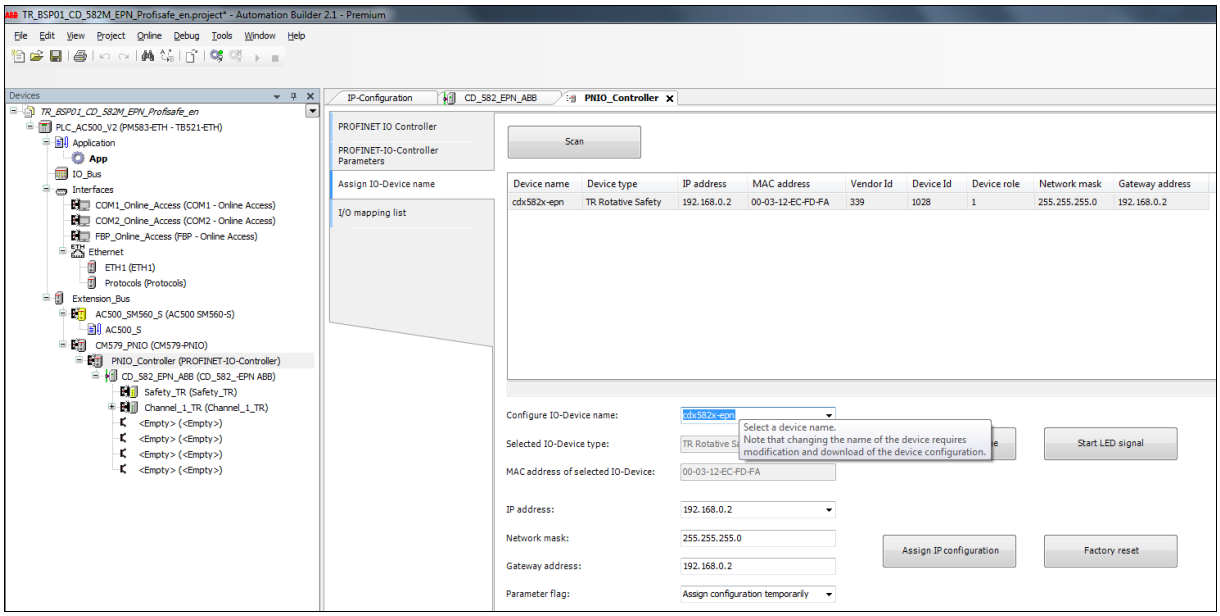
- Open the `CD_582_-EPN` object node in Configuration view to set the station name. The measuring system communication parameters – including the `Station Name` – can be set under the `General` tab.



- The preset `CD_582_-EPN` default project station name is always `cdx582x-epn`. Each device must have a unique station name.
- Project settings and device settings must agree, i.e. the user must assign the set station name to the connected hardware component.

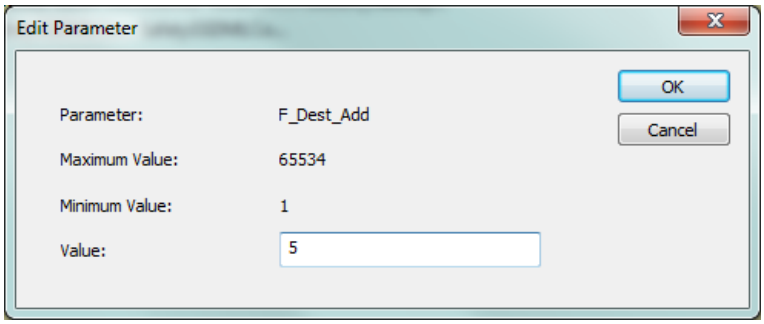
Use the DCP protocol to assign the station name.

- Use the Configuration view of the PNIO controller's object node to assign the station name. These functions are available in the `Assign IO device name` tab. The preset station name can be assigned to each connected device after a network `Scan`.



- Second, the F-Dest-Address setting must match the `CD_582_-EPN` rotary switch setting.

- In this project, the device rotary switch was set to `5`. Thus, the F-parameter `F_Dest_Add` is also set to `5`
(see also Chapter 5.3.2 “Setting the F-parameters” on page 87).

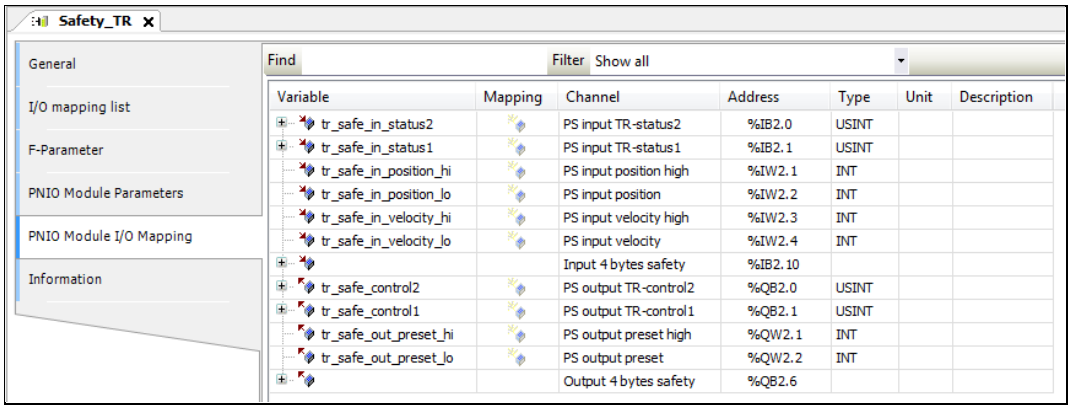


5.2.3 Defining the I/O mapping

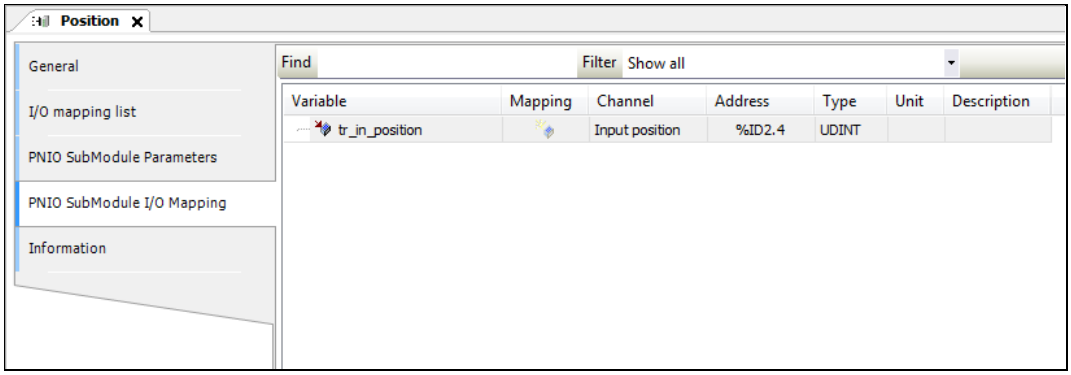
The control project process data inputs and outputs can be assigned symbolic variable names to make it easier to use them. Open the `PNIO module I/O Mapping` or the `PNIO SubModule I/O Mapping` tab in the Configuration view of the IO modules or IO submodules to define the symbolic names.

- The appropriate names can be assigned in the `Variable` column when defining the symbolic variable names.

- Open the safety module Configuration view to define the symbolic variable names in the safety area. All symbolic safety area variable names are also available as “read-only” in the non-safety program.



- Open the Configuration view of the Profinet submodules to define the symbolic variable names in the “gray” non-safety area.



Re-create the `Configuration data` and the `Safety configuration data` when modifying the hardware and IO configuration.
(see Chapter 5.4 “Setting configuration data” on page 88).

5.3 Parameterization

5.3.1 Setting the iParameters

➢ Setting the CD_582_-EPN iParameter requires opening the General tab in the Configuration view of the IO modules / IO sub-modules below the CD_582_-EPN node. Edit the Value column to set the iParameters.

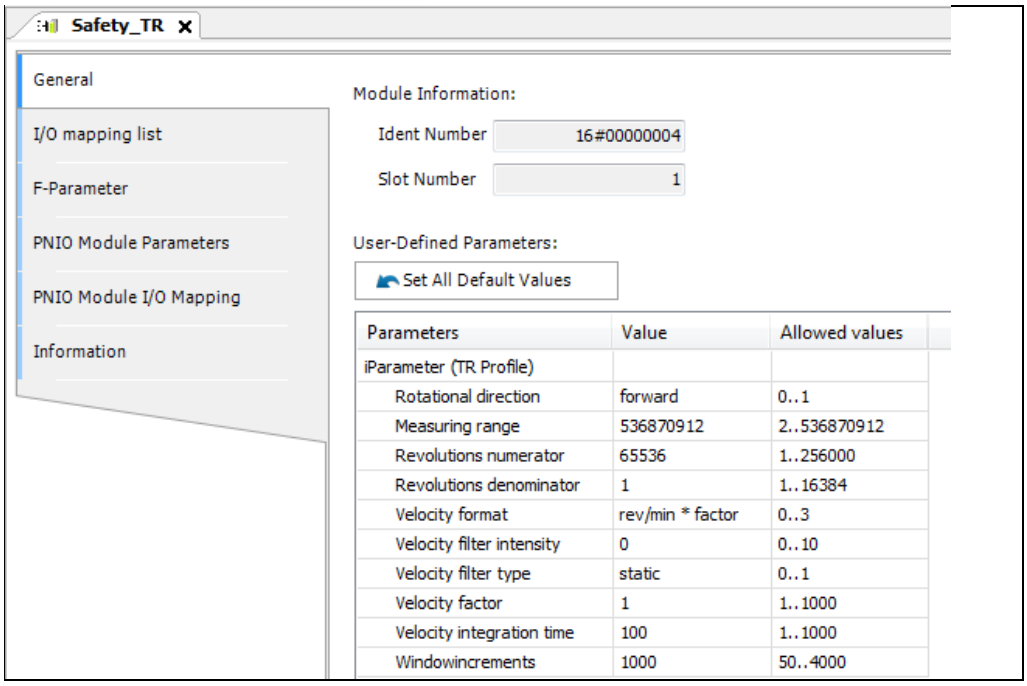


Figure 4: Configuration view of the Safety_TR IO module with iParameter table

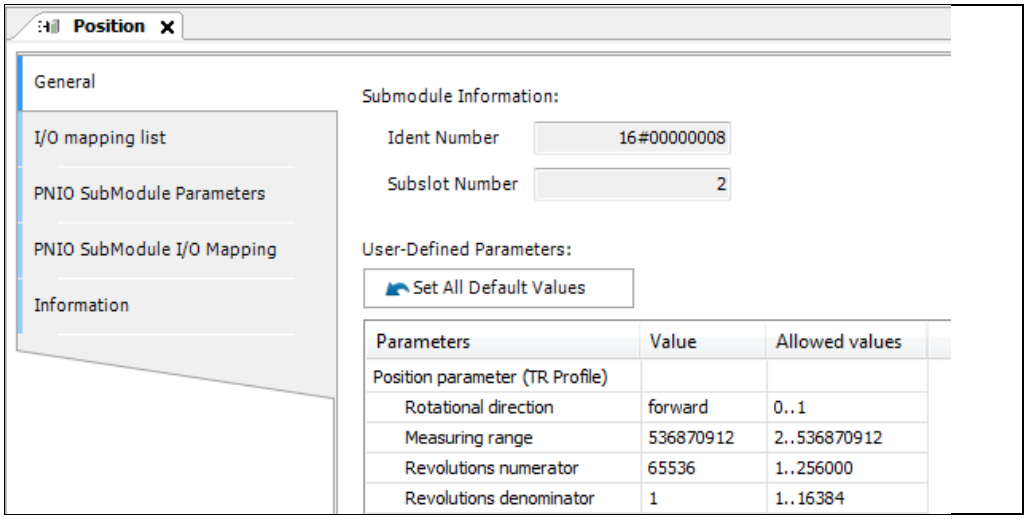


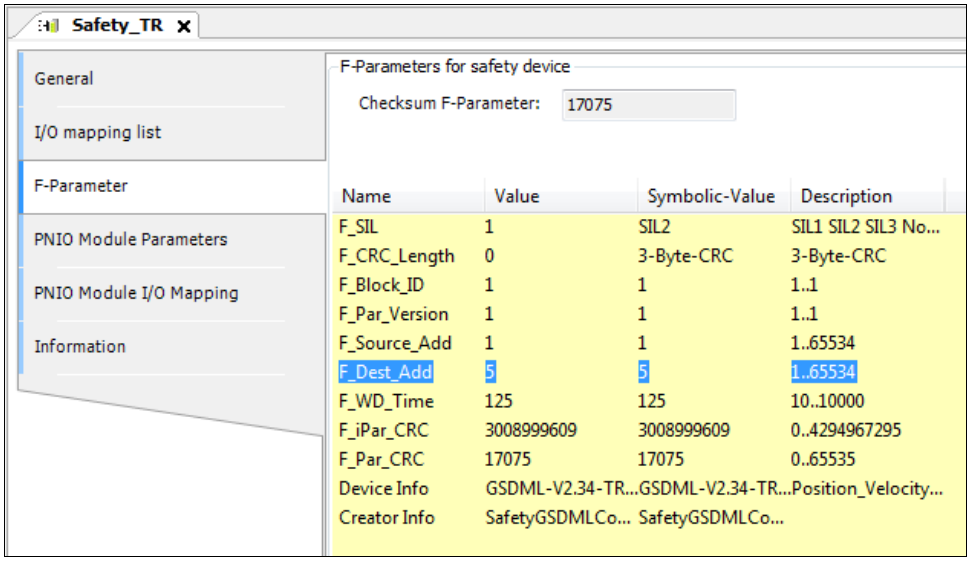
Figure 5: Configuration view of the Position IO sub-module with iParameter table

➢ The iParameter default values are listed in the above tables. An F_iPar_CRC calculation is required for a new parameter data set if different parameter values are required. See Chapter: 4.1 “iParameter” on page 67.

➢ The calculated value is then entered in the F-parameters’ parameter data set under F_iPar_CRC. See Chapter: 5.3.2 “Setting the F-parameters” on page 87.

5.3.2 Setting the F-parameters

- The CD_582_-EPN Safety IO module's F-parameters must be set.
- Setting the F-Parameter requires opening the F-parameter tab in the Configuration view of the Saftey_TR IO module below the CD_582_-EPN node.



The measuring system's F_Dest_Add entry and address switch setting must agree!

The F_iPar_CRC parameter value is derived from the parameter data set assigned to the iParameters and the CRC value calculated from it.
See Chapter: 5.3.1 "Setting the iParameters" on page 86.

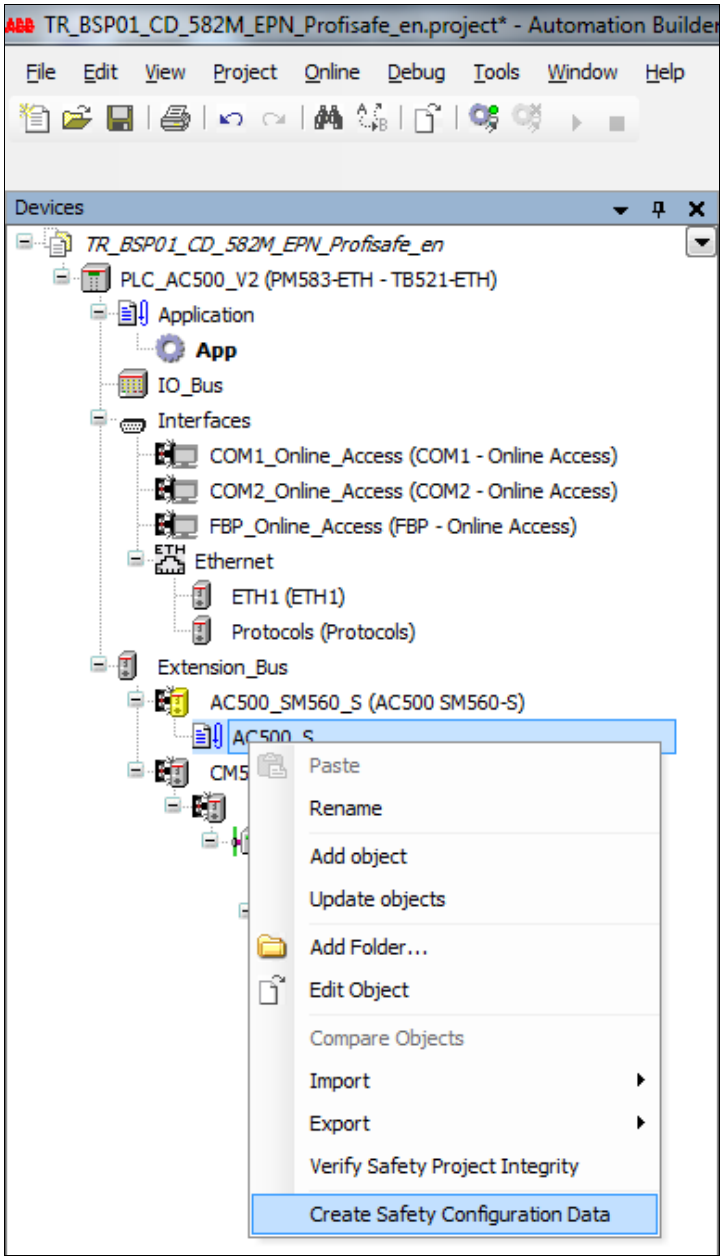
- The F_Par_CRC value is automatically updated after setting the F-Dest address or after setting F-iPar-CRC.

5.4 Setting configuration data

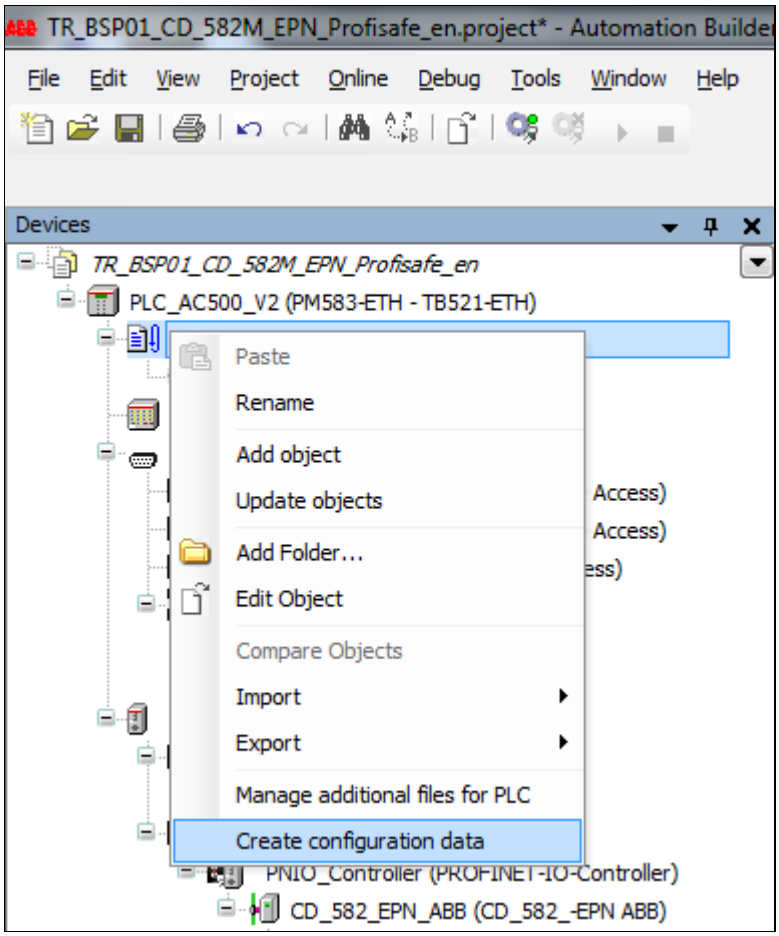
The configuration data of both central processing units must be created before creating the PLC programs once the hardware configuration is complete.

Automation Builder automatically recognizes the need to create the configuration data and will prompt the user accordingly.

➢ Create the configuration data for the safety application first. To do so, select the function `Create Safety Configuration Data` from the safety application context menu below the SM560-S node.



➢ Then create the configuration data for the non-safety application. To do so, select the function `Create Configuration Data` from the non-safety application context menu below the PLC_AC500_V2 node.



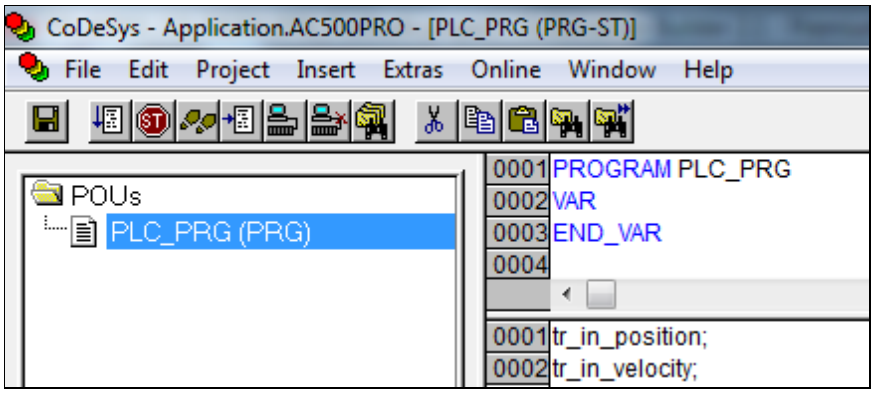
➢ Begin loading the non-safety application once all configuration data has been generated.

5.5 Loading the non-safety application

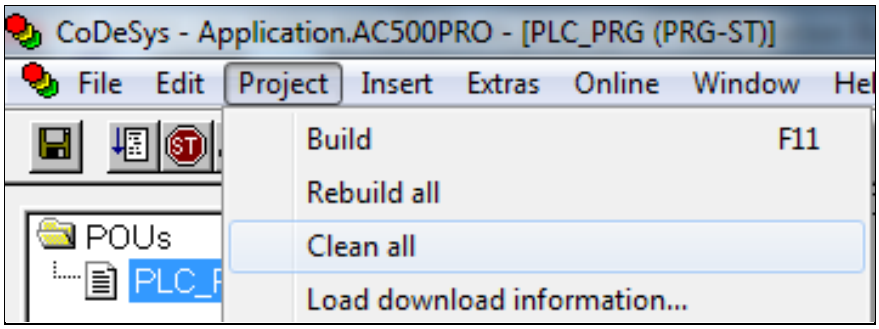
A distinction is made between the non-safety application and the safety application when programming the AC500-S. The non-safety application is created, loaded, and tested in a CoDeSys editor with a white background.

- The non-safety CoDeSys editor opens in the device configuration tree after double-clicking the non-safety application below the PLC_AC500_V2 node.

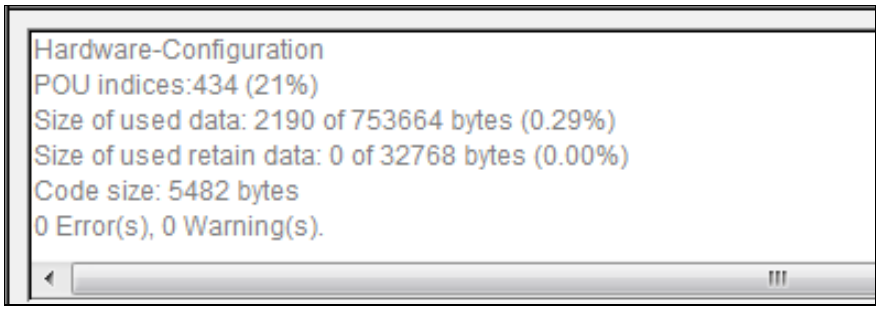
- The PLC program is created in the main program of the non-safety application PLC_PRG using the IEC61131 language ST.



- This example calls the linked CD_582_-EPN process data input variables. Thus, the values can be used in the program (see Chapter 5.2.3 “Defining the I/O mapping” on page 85).

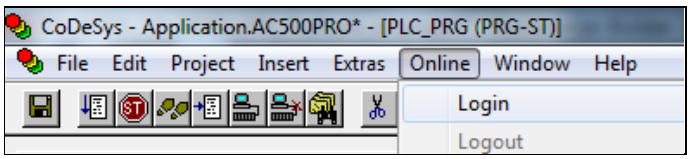


- Compile the program before loading the non-safety application. Always call the functions Project -> Clean all and then Project -> Rebuild all to start the compilation process.

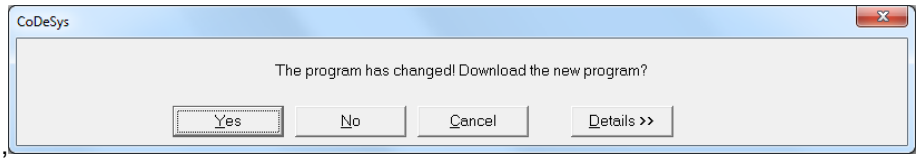


- The application must compile with 0 Error(s), 0 Warning(s).

- The program can be loaded onto the controller once it has been compiled without any errors. Execute the command `Online > Log in` to establish a connection between `Automation Builder` and the central processing unit `PM583-ETH`.

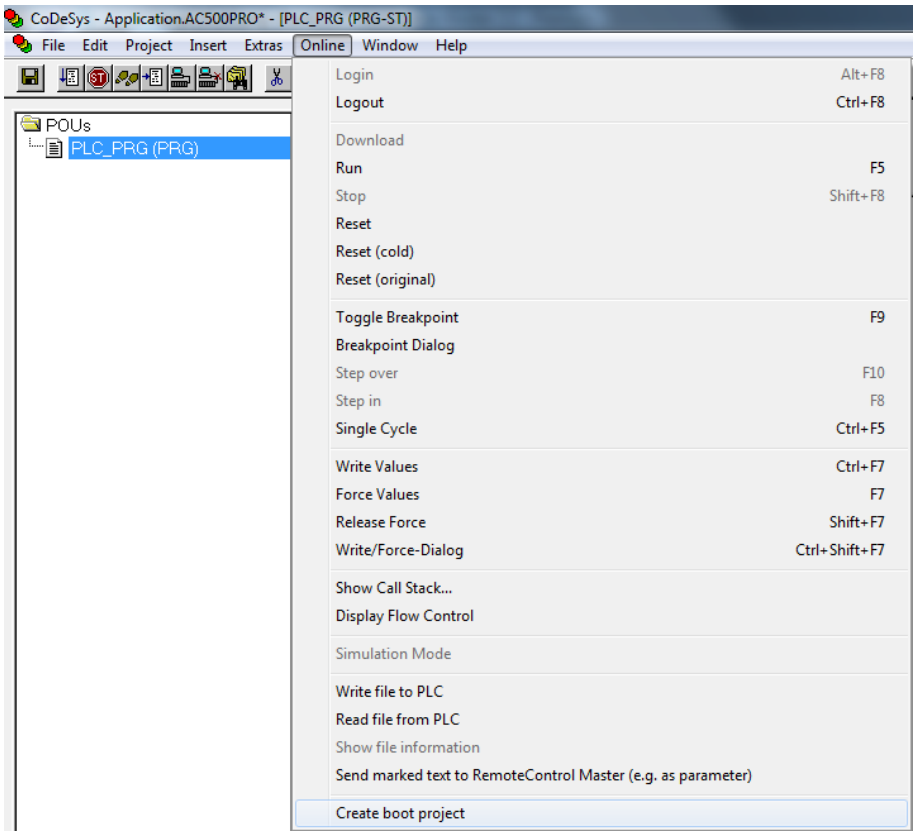


- The user is prompted to confirm loading the program with `Yes` if there is no program, or the program has changed.



,

- A boot project must be created after loading the program to permanently save the control program onto the central processing unit. The boot project is created after calling the function `Online > Create boot project`.



Creating the boot project on the controller takes approx. 30 seconds. This is indicated on the PM583-ETH by the flashing RUN LED. The command triggered via CoDeSys has then already been executed.

Do not interrupt the power supply to the control system while the control system saves the boot project.

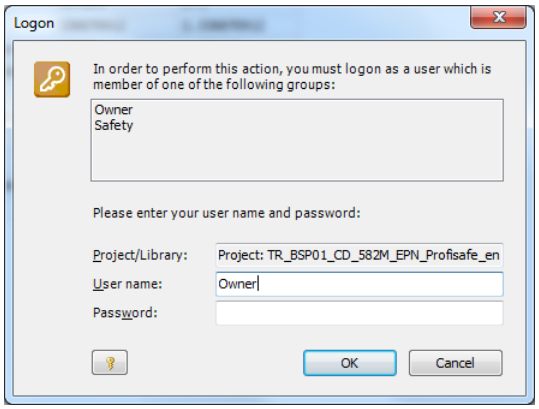
5.6 Loading the safety application

A distinction is made between the non-safety application and the safety application when programming the AC500-S. The safety application is created, loaded, and tested in a CoDeSys editor with a yellow background.

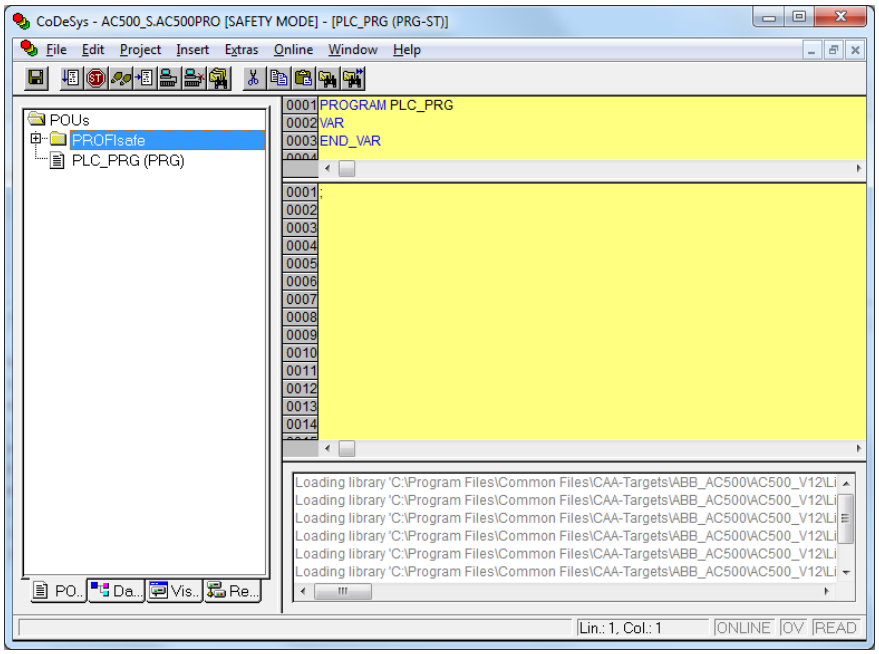
➢ The `Safety CoDeSys editor` opens in the device configuration tree after double-clicking the `AC500_S` safety application below the `AC500_SM560_S` node.

When opening the `Safety CoDeSys editor` for the first time, the user is prompted to confirm adding several system libraries to the safety application. The user must confirm all information dialogs with `OK`.

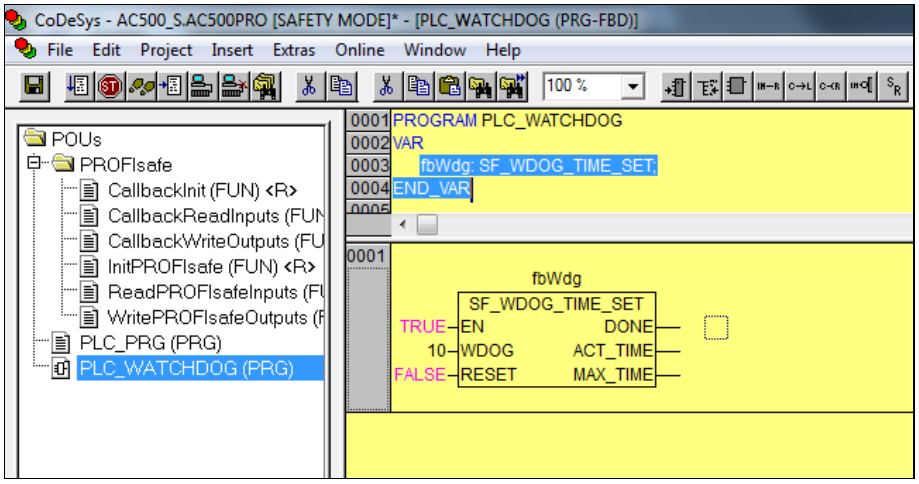
➢ The user is prompted to enter a user name and a password. In this example, the user is always “Owner” and the password is always an empty string.



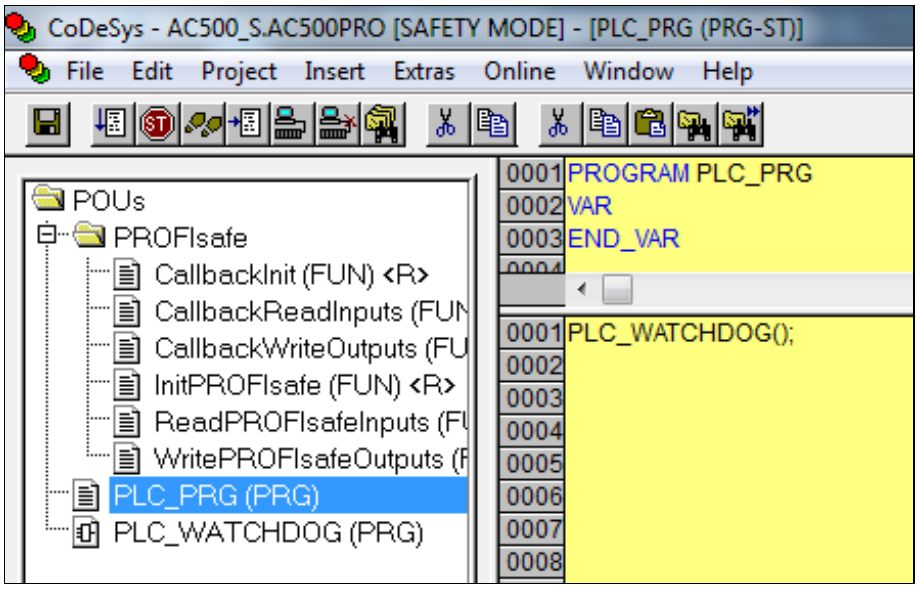
➢ The Safety PLC program is created in the main program of the safety application `PLC_PRG` using the IEC61131 language `ST`. In addition to the main program, CoDeSys loads several libraries for processing the PROFIsafe frames.



- A cyclically triggered watchdog timer runs in the background of the safety controller SM560-S. The PLC_WATCHDOG subroutine controlling the trigger is thus created first. The system function block SF_WDOG_TIME_SET is used to program the watchdog trigger.



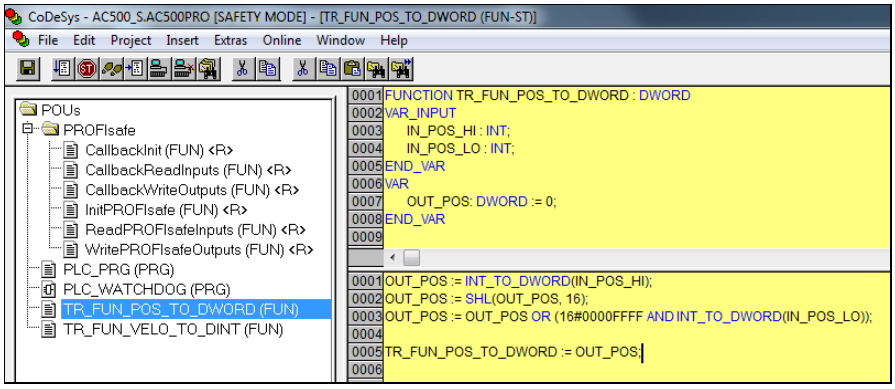
- The watchdog timer is set to 10 ms in this example. This value must be adjusted according to the cycle time of the main program.



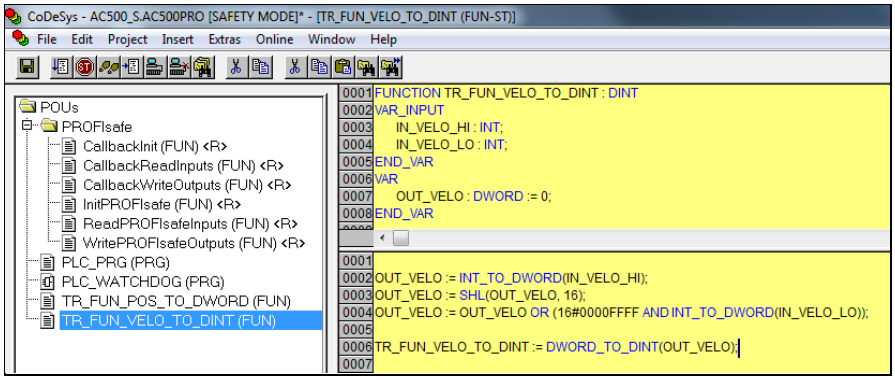
- The watchdog trigger should be called at the beginning of the main program PLC_PRG.

- This example calls the linked variables of the fail-safe CD_582_-EPN process data inputs. Word-oriented information is processed in the PROFIsafe process data as INT (16 bit, signed). However, CD_582_-EPN transmits its position as DWORD (32 bit, unsigned) and its speed as DINT (32 bit, signed).

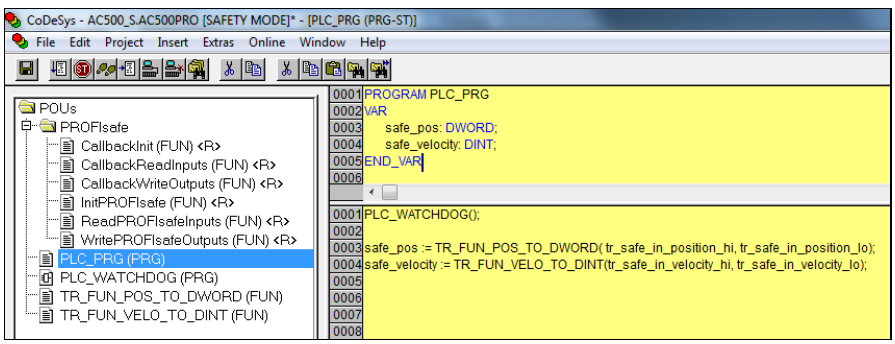
- This is where the process input data are assembled and converted for further use. To this end, the linked position and speed variables are preprocessed in sub-functions.



- The TR_FUN_POS_TO_DWORD sub-function assembles a HI and LO position word from the process data inputs into a DWORD.

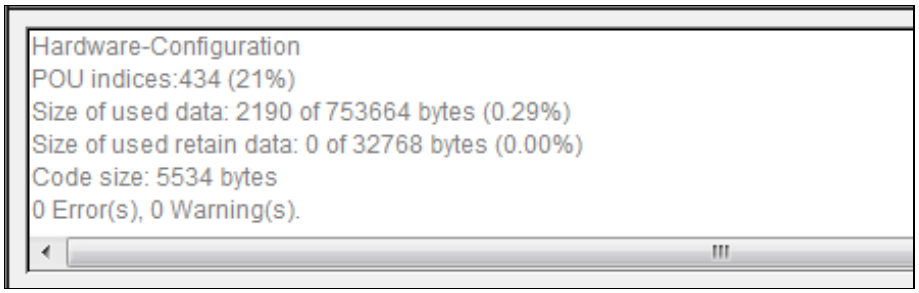


- The TR_FUN_VELO_TO_DINT sub-function assembles a HI and LO speed word from the process data inputs into a DINT.
- The main program PLC_PRG uses these sub-functions to assign the linked process data inputs to the two local variables safe_pos and safe_velocity.



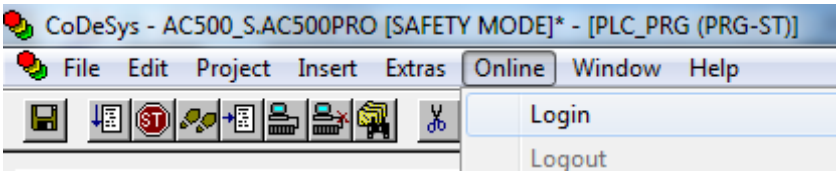
- `CD_582_-EPN` will request reintegration into the PROFIsafe network if the communication to the PROFIsafe F-host needs to be synchronized. This is necessary, for example, after a network outage. The reintegration is controlled via the `OA` bit of the PROFIsafe instance. PROFIsafe process data cannot be processed in the safety application without reintegration.
  For more information, see the ABB `AC500-S Safety User Manual`.

- Compile the program before loading the safety application. Always call the functions `Project -> Clean all` and then `Project -> Rebuild all` to execute the compilation process.

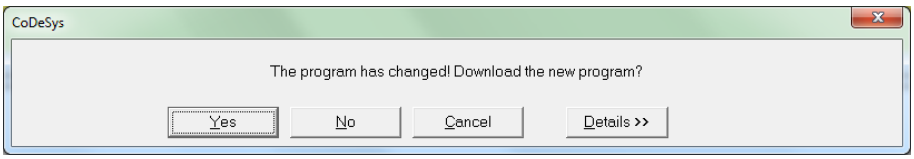


- The application must compile with `0 Error(s), 0 Warning(s).`

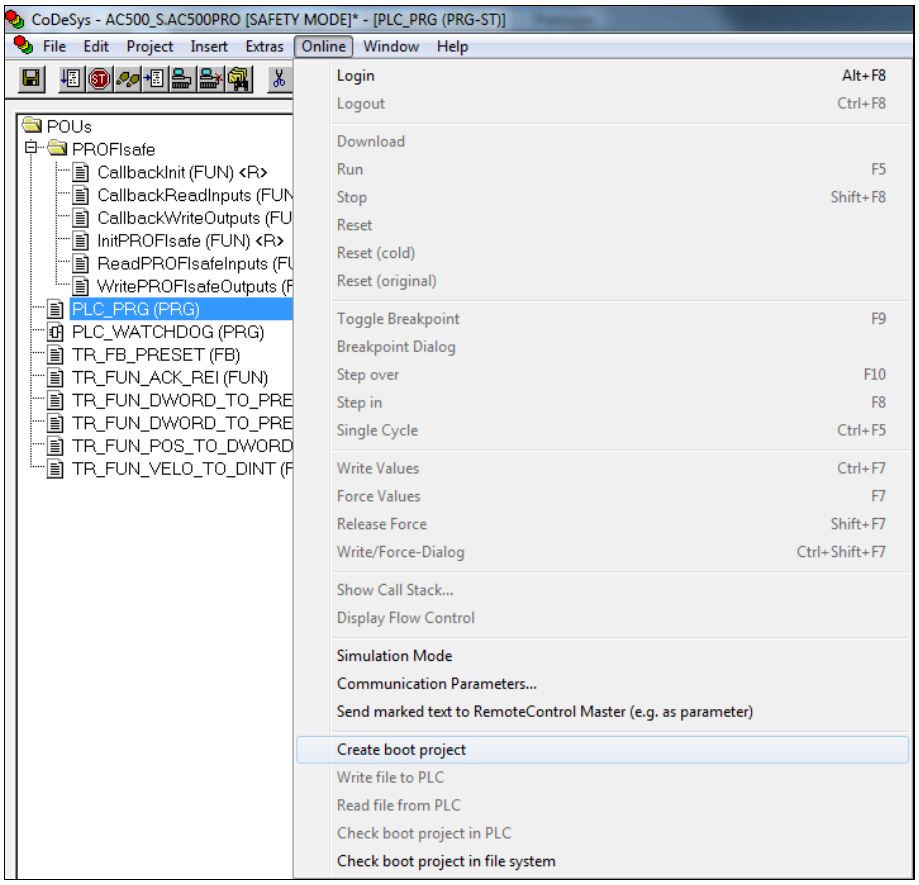
- The program can be loaded onto the controller once it has been compiled without any errors. Execute the command `Online > Log in` to establish a connection between `Automation Builder` and the F-CPU `SM560-S`.



- The user is prompted to confirm loading the program with `Yes` if there is no program, or the program has changed.



- A boot project must be created after loading the program to permanently save the control program onto the central processing unit. The boot project is created after calling the function `Online > Create boot project`.



Creating the boot project on the controller takes approx. 2 seconds. This is indicated on the SM560-S by the flashing RUN LED. The command triggered via CoDeSys has then already been executed.
Do not interrupt the power supply to the F-CPU while the F-CPU saves the boot project.

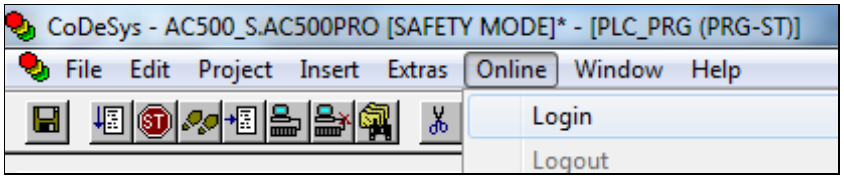
➢ The newly programmed control program starts after a power cycle (system restart).

The SM560-S F-CPU monitors the supply of the control unit for power failures.
The control must remain without power for at least 2 seconds if the user restarts the control unit.

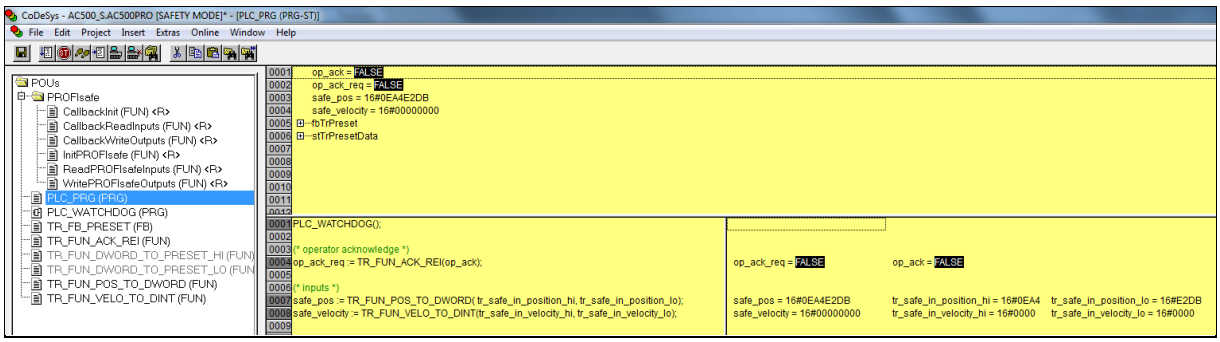
5.7 Testing the safety application

Fully functional test the automation task after creating the safety application.

- The user can test the program by logging into the controller via the Safety CoDeSys editor if a valid safety application has been loaded into the F-CPU and a boot project has been created before.



- Instead of performing a loading operation, the editor will now start in debug mode and show the status of the local variables, the linked process data inputs, and the program sequence.



- The local variables `safe_pos` and `safe_velocity` show the current process data of the `CD_582_-EPN`.

6 Extending the control program – application examples

The Safety program described in Chapter 5 is expanded in the following sections to include application examples for preset execution and error analysis.

These examples are merely intended to assist with various automation tasks and are not customer-specific solutions.

The provided function blocks are intended to simplify integrating the measurement system into an application.

In the application example

- Preset Execution

the error states are output by the provided function block. The corresponding error handling is not part of the example and must be implemented by the user.

Follow the “Terms of use for the software examples” in Chapter 2.4!

6.1 Preset Execution

The preset module created for the preset adjustment function sets the measuring system’s current safe position to any new value within its measuring range. Using the bits `OUT_ERROR` and `OUT_VALID`, the preset module indicates whether the preset adjustment function was executed. The preset adjustment function only works as long as the measuring system is not passivated or as long as no other preset adjustment function is being carried out at another safe position. The `OUT_READY` bit indicates the readiness to carry out a preset adjustment function.
See also Chapter: 5.6 “Loading the safety application” on page 92.

The preset module does not verify the new position. The user must do this!

6.1.1 Parameter description

Input parameters	Data type	Description
IN_REQ	BOOL	Starts the preset adjustment function.
IN_PRESET	DWORD	New position value of be set.
IN_STATUS1	BYTE	Identifies the status of the measuring system. Import the measuring system input data from the `TR-Status1` register.
IN_STATUS2	BYTE	Identifies the status of the measuring system. Import the measuring system input data from the `TR-Status2` register.

Output parameters	Data type	Description
OUT_READY	BOOL	Indicates whether the module can be used to perform a preset adjustment function.
OUT_BUSY	BOOL	Indicates whether the module is currently executing the preset adjustment function.
OUT_VALID	BOOL	Indicates whether the preset adjustment function was successfully executed.
OUT_ERROR	BOOL	Indicates whether the preset adjustment function was executed with an error.
OUT_PRESET	DINT	Preset value for the measuring system. Export the measuring system output data to the `Preset` register.
OUT_CTRL	BYTE	Control word default value of the measuring system in the process output data. Export the measuring system output data to the `TR-Control1` register.

6.1.2 Functional description

The interface description TR-ECE-BA-GB-0139 contains a complete description of the preset adjustment function and the associated status and control bits.

- The input `tr_safe_in_status1` must be 0x10 for the preset adjustment function to be executed. The output `OUT_READY` shows the current state of the input `tr_safe_in_status1`.

- The input `IN_PRESET` is always read and latched output to the output `OUT_PRESET` when the signal `IN_REQ` is received. The preset sequence runs in a finite state machine once the preset module has been started via the input `IN_REQ`.

- The preset module is executed as the edge of the input `IN_REQ` rises. The outputs `OUT_VALID` and `OUT_ERROR` are reset to 0. The control byte is set according to the status of the finite state machine and transferred to the device via the output `tr_safe_out_control1`.

- The measuring system will then execute the preset adjustment function. When the input `IN_REQ` is reset to 0 has no influence on the further execution of the preset adjustment function.

- The measuring system sets the input `TR_PresOk` bit or `TR_PresError` bit in the input word `tr_safe_in_status1` after the preset adjustment function has been executed. These bits are used to set the corresponding outputs `OUT_VALID` respectively `OUT_ERROR`.

- After the output `OUT_VALID` or the output `OUT_ERROR` has been set, the outputs `TR_PresPrep` and `TR_PresReq` in the control bytes `tr_safe_out_control1` and `OUT_BUSY` are reset to 0. This completes the execution of the preset module.

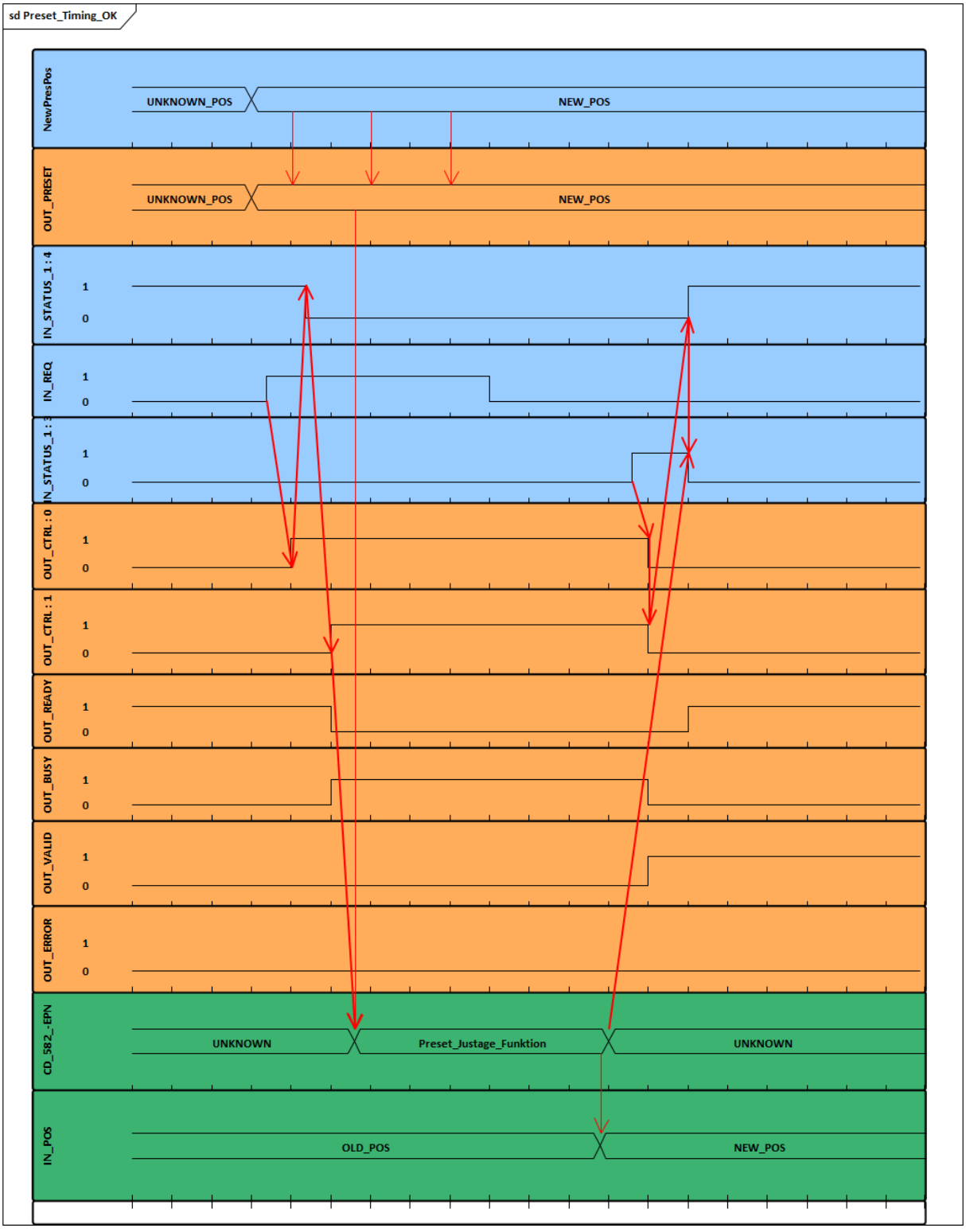


Figure 6: Timing diagram of an error-free preset adjustment function execution

blue area: Preset module input signals
orange area: Preset module output signals
green area: CD_582_-EPN measuring system function or measuring system values

➢ The preset adjustment function is not executed as long as the input IN_STATUS_1: 4 is 0. The outputs OUT_VALID and OUT_ERROR are reset to 0 as the edge of the input IN_REQ rises. The outputs OUT_CTRL : 0, OUT_CTRL : 1, OUT_READY, and OUT_BUSY do not change their values.

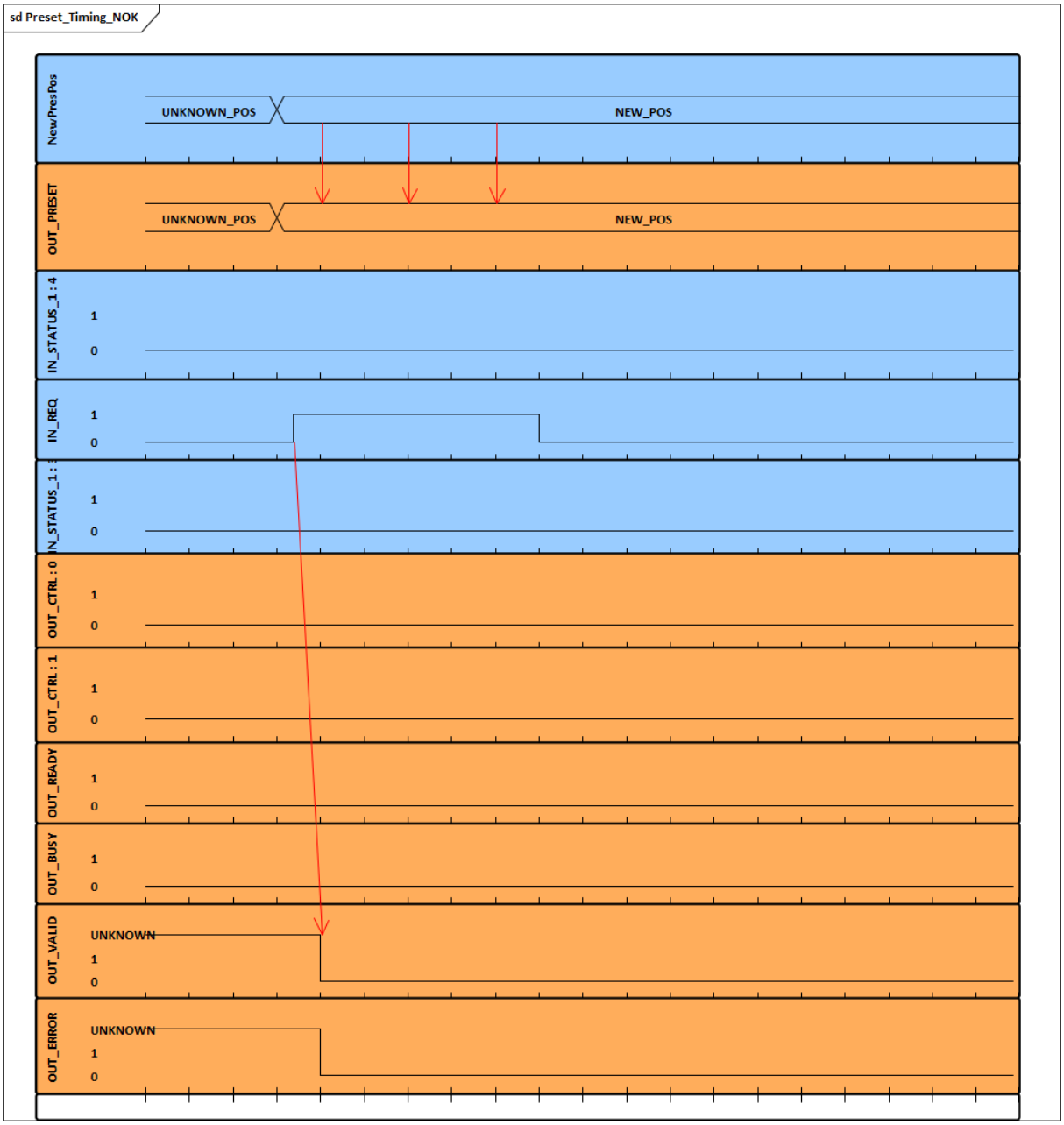
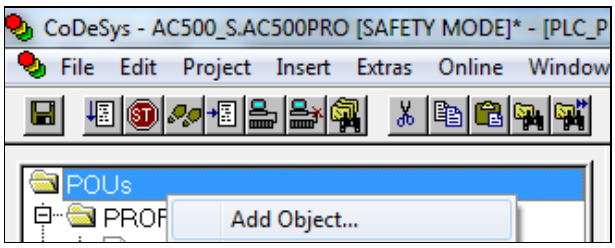


Figure 7: Timing diagram of the preset adjustment function if IN_STATUS_1: 4 is 0

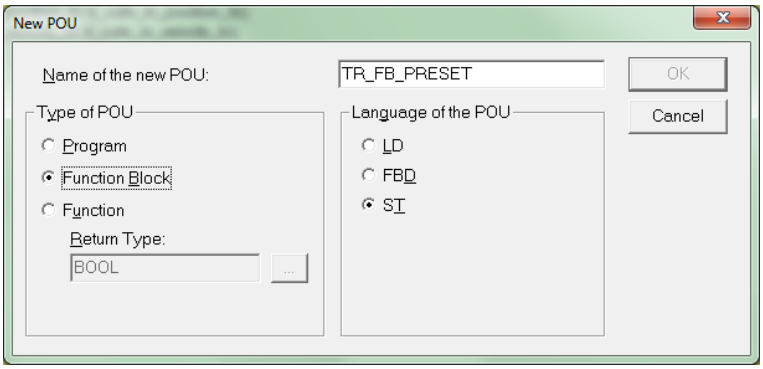
blue area: Preset module input signals
orange area: Preset module output signals

6.1.3 Creating a module

- A preset module is created by first creating a new Safe function block named `TR_FB_PRESET`. First, call the `Add Object...` function in the context menu of the CoDeSys block directory.



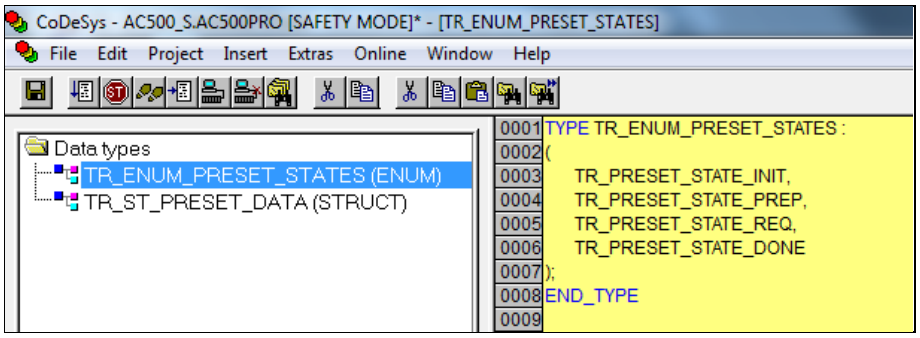
- In the next dialog, enter the name of the function block `TR_FB_PRESET`. Then chose the type option `Function Block`. Choose the language `ST` for this example.



- The following variables must be created in the preset function block.

```
0021 FUNCTION_BLOCK TR_FB_PRESET
0022 VAR_INPUT
0023     IN_REQ : BOOL;
0024     IN_PRESET : DWORD;
0025     IN_STATUS_1 : BYTE;
0026     IN_STATUS_2 : BYTE;
0027 END_VAR
0028 VAR_OUTPUT
0029     OUT_READY : BOOL;
0030     OUT_BUSY : BOOL;
0031     OUT_VALID : BOOL;
0032     OUT_ERROR : BOOL;
0033     OUT_PRESET : DWORD;
0034     OUT_CTRL : BYTE;
0035 END_VAR
0036 VAR
0037     fb_preset_state : TR_ENUM_PRESET_STATES := TR_PRESET_STATE_INIT;
0038     fb_preset_ready : BOOL := FALSE;
0039     fb_preset_control : BYTE := 0;
0040     fb_preset_error : BOOL := FALSE;
0041     fb_preset_timeout : DWORD := 0;
0042     fb_preset_value : DWORD := 0;
0043 END_VAR
```

- The enumerated data type `TR_ENUM_PRESET_STATES` is created in this project for the state variable `fb_preset_state`.



- The following commands must be created in the function block program to implement the functionality of the preset adjustment function.

```
0001 (* process inputs *)
0002 IF ((IN_STATUS_1 AND 16#30) = 16#10) AND
0003    ((IN_STATUS_2 AND 16#FE) = 16#00)
0004 THEN
0005     fb_preset_ready := TRUE;
0006 ELSE
0007     fb_preset_ready := FALSE;
0008 END_IF;
0009
0010 (* process instance data  *)
0011 fb_preset_timeout := fb_preset_timeout + 1;
0012 fb_preset_control := 0;
```

- The status bytes `IN_STATUS1` and `IN_STATUS2` are evaluated to determine the availability of the preset adjustment function. The instance data of the function block is also updated.

- The preset adjustment function is implemented as a finite state machine.

```
0014 (* state machine *)
0015 CASE fb_preset_state OF
0016
0017 TR_PRESET_STATE_INIT:
0018     fb_preset_error := FALSE;
0019     fb_preset_timeout := 0;
0020     IF  (IN_REQ = TRUE) THEN
0021         IF(fb_preset_ready = TRUE) THEN
0022             fb_preset_value := IN_PRESET;
0023             fb_preset_state := TR_PRESET_STATE_PREP;
0024         ELSE
0025             fb_preset_error := TRUE;
0026             fb_preset_state := TR_PRESET_STATE_DONE;
0027         END_IF;
0028     END_IF;
0029
0030 TR_PRESET_STATE_PREP:
0031     fb_preset_control := 16#01; (* preset preparation *)
0032     IF((IN_STATUS_1 AND 16#3C) = 16#00) THEN
0033         fb_preset_timeout := 0;
0034         fb_preset_state := TR_PRESET_STATE_REQ;
0035     END_IF;
0036
0037     (* check timeout *)
0038     IF (fb_preset_timeout > 10000) THEN
0039         fb_preset_error := TRUE;
0040         fb_preset_state := TR_PRESET_STATE_DONE;
0041     END_IF;
```

- The initial state is `TR_PRESET_STATE_INIT`. The process is started when an `IN_REQ` is received and the status bytes indicate the readiness of `CD_582_-EPN`.

- A control byte transfers the `Preset preparation` command to the `CD_582_-EPN` when in the state `TR_PRESET_STATE_PREP`. The status is changed upon acknowledgement by the measuring system. This process can be canceled by a timeout.

```
0043 TR_PRESET_STATE_REQ:
0044     fb_preset_control := 16#03;  (* preset request & preparation *)
0045     IF((IN_STATUS_1 AND 16#3C) = 16#04) THEN (* success *)
0046         fb_preset_timeout := 0;
0047         fb_preset_state := TR_PRESET_STATE_DONE;
0048     ELSIF ((IN_STATUS_1 AND 16#3C) = 16#08) THEN (* error *)
0049         fb_preset_timeout := 0;
0050         fb_preset_error := TRUE;
0051         fb_preset_state := TR_PRESET_STATE_DONE;
0052     END_IF;
0053
0054     (* check timeout *)
0055     IF (fb_preset_timeout > 10000) THEN
0056         fb_preset_error := TRUE;
0057         fb_preset_state := TR_PRESET_STATE_DONE;
0058     END_IF;
0059
0060 TR_PRESET_STATE_DONE:
0061     IF  (IN_REQ = FALSE) THEN
0062         fb_preset_state := TR_PRESET_STATE_INIT;
0063     END_IF;
0064
0065 ELSE
0066     fb_preset_error := TRUE;
0067     fb_preset_state := TR_PRESET_STATE_DONE;
0068 END_CASE;
```

- A control byte transfers the actual `PresetRequest` preset command to the `CD_582_-EPN` when in the state `TR_PRESET_STATE_REQ`. The status is changed upon acknowledgement by the measuring system. This process can be canceled by a timeout.

- The program waits for the completion and resetting of the preset adjustment function when om the state `TR_PRESET_STATE_DONE`. The function block is reset by resetting `IN_REQ` to the initial state `TR_PRESET_STATE_INIT`.

```
0070 (* process outputs *)
0071 OUT_READY := fb_preset_ready;
0072
0073 IF (fb_preset_state = TR_PRESET_STATE_DONE) OR
0074    (fb_preset_state = TR_PRESET_STATE_INIT)
0075 THEN
0076     OUT_BUSY := FALSE;
0077 ELSE
0078     OUT_BUSY := TRUE;
0079 END_IF;
0080
0081 IF (fb_preset_state = TR_PRESET_STATE_DONE) AND
0082    NOT fb_preset_error
0083 THEN
0084     OUT_VALID := TRUE;
0085 ELSE
0086     OUT_VALID := FALSE;
0087 END_IF;
0088
0089 OUT_ERROR := fb_preset_error;
0090 OUT_PRESET := fb_preset_value;
0091 OUT_CTRL := fb_preset_control;
```

➢ The states of the output variables are set at the end of the function block according to the program flow.

In the main program, function block inputs and outputs must be connected to the corresponding PROFIsafe input and output data.

```
0010 (* preset *)
0011 stTrPresetData.bReq;
0012 stTrPresetData.dwPreset;
0013
0014 fbTrPreset(IN_REQ := stTrPresetData.bReq,
0015            IN_PRESET := stTrPresetData.dwPreset,
0016            IN_STATUS_1 := tr_safe_in_status1,
0017            IN_STATUS_2 := tr_safe_in_status2);
0018
0019 stTrPresetData.bReady := fbTrPreset.OUT_READY;
0020 stTrPresetData.bValid := fbTrPreset.OUT_VALID;
0021 stTrPresetData.bBusy := fbTrPreset.OUT_BUSY;
0022 stTrPresetData.bError := fbTrPreset.OUT_ERROR;
0023
0024 (* outputs *)
0025 tr_safe_out_preset_hi :=TR_FUN_DWORD_TO_PRESET_HI(fbTrPreset.OUT_PRESET);
0026 tr_safe_out_preset_lo := TR_FUN_DWORD_TO_PRESET_LO(fbTrPreset.OUT_PRESET);
0027 tr_safe_out_control1 := fbTrPreset.OUT_CTRL;
0028
```

- In the main program, Status Bytes 1 and 2 are transferred to the function block by the PROFIsafe process inputs. The preset default value and the control byte are also transferred to the corresponding PROFIsafe process output data.

- The double word of the preset adjustment default value OUT_PRESET is broken down into two words by two auxiliary functions and assigned to the process data outputs `tr_safe_out_preset_hi` or `tr_safe_out_preset_lo`.

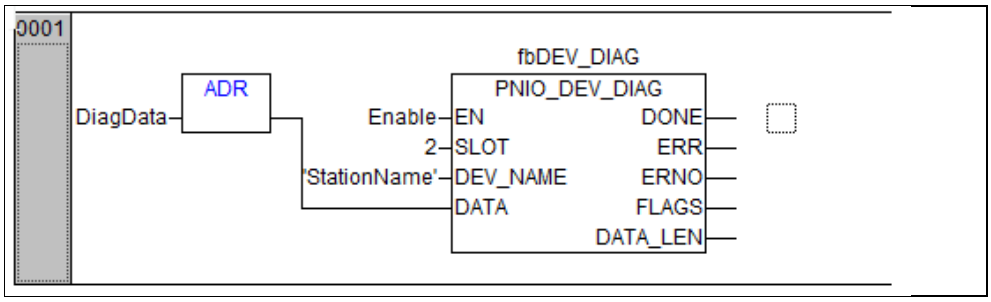
- The control byte `OUT_CTRL` of the preset function block is assigned to the process data output `tr_safe_out_control1`.

6.2 Error analysis

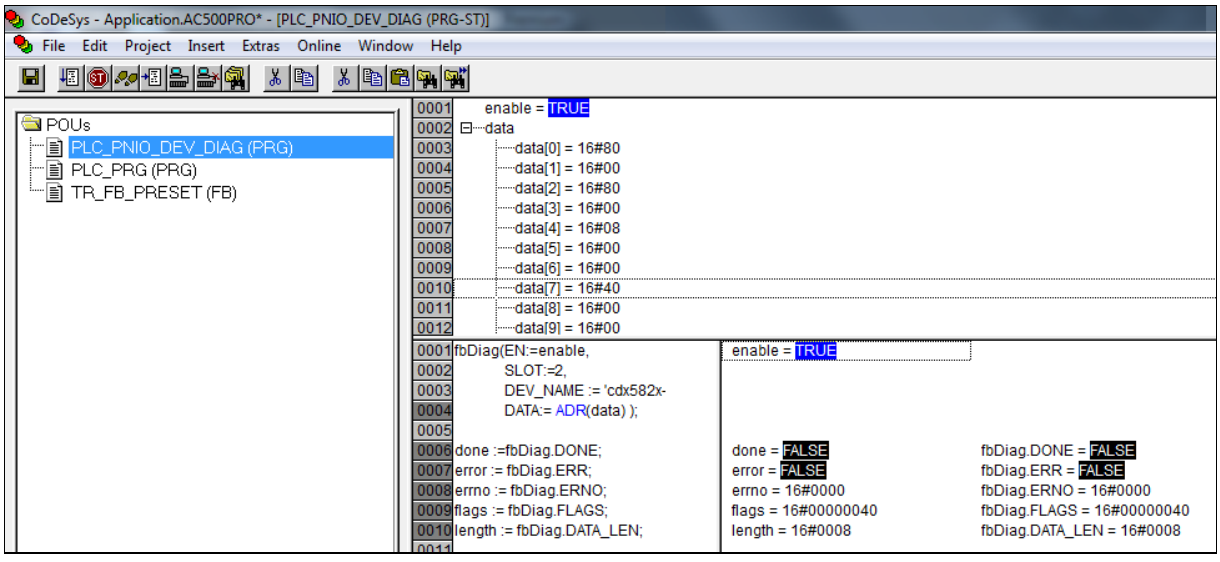
In the event of an error, the measuring system provides a diagnostic message. In the `Non-safety program`, it can be read out and evaluated from the communication module `CM579-PNIO`.

6.2.1 Display of diagnostic messages

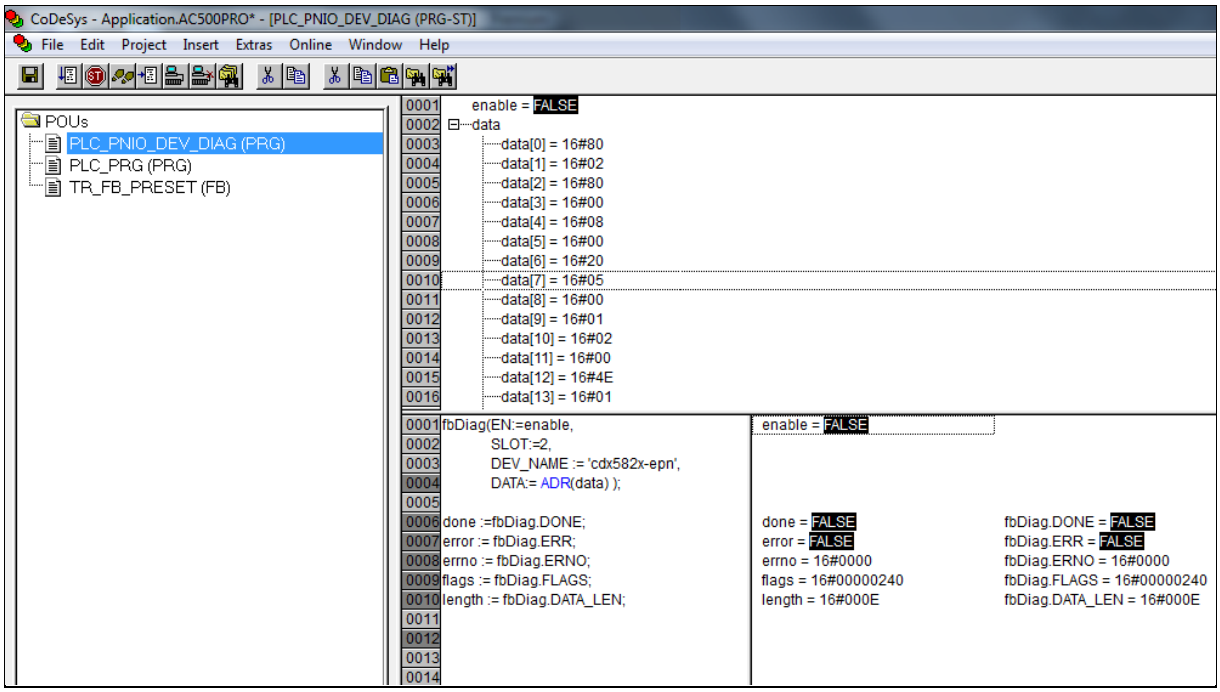
- You need the function block `PNIO_DEV_DIAG` to read the diagnostic messages via PROFINET. It reads the standard diagnostic messages of a device. The non-safety program function block is located in the system library `Profinet_AC500_V13.lib`.
- Create a block instance and call it in the main program to evaluate diagnostic messages. Diagnostic messages – if available – are copied into the data buffer if the function block enable input `EN` has been set.



- The input parameter `SLOT` specifies the `CM579-PNIO` slot in the hardware configuration. In this example, the communication module is in Slot 2.
- The input parameter `DEV_NAME` is assigned the station name of the device to be read – in this example `cdx582x-epn`.



- Simple channel diagnostics start with the HEX code `0x8000`; the error code looked for is in bytes [6] and [7]; in this case: `0x0040`, i.e. “Faulty safety target address (F_Dest_Add)”.



- Extended channel diagnostics start with the HEX code `0x8002`; the error code looked for is in bytes [6] and [7]; in this case: `0x2005`, i.e. “Error in MT scanning - Channel 1”. The extended detailed error code is in bytes [12] and [13]. The extended error code is used for a detailed error analysis in the factory.

6.3 Measuring system – Passivation and Operator Acknowledge

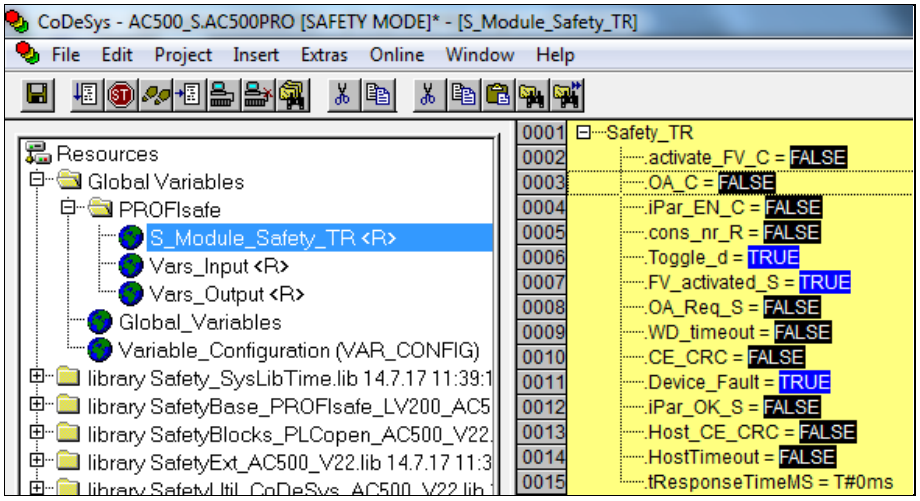
6.3.1 After starting the F-System

The communication between the F-CPU and the measuring system must be established via the PROFIsafe protocol when the F-System has started up. Using the default values (0), the measuring system is passivated during this time.

There is no automatic user acknowledgment (Operator Acknowledge) if the communication between the F-CPU and the measuring system exceeds the F-periphery monitoring time set in `Automation Builder`.
This requires a user acknowledgment with a positive edge on the variable `OA_C`, the global variable of the F-periphery, which is controlled – in the sample project – by the subroutine `TR_FUN_ACK_REI`.

6.3.2 After communication errors

The measuring system is passivated if the F-System detects an error in the safety-related PROFIsafe-protocol-based communication between the F-CPU and the measuring system.
The variable `FV_activated_S` is set while the substitute values (0) are used.



User acknowledgments (Operator Acknowledge) of the measuring system (i.e. output of cyclic data to the fail-safe outputs) only occur if:

- there is no longer a communication error
- a user acknowledgment with a positive edge on the variable `OA_C`, the global variable of the F-periphery, has occurred.

The sample project implements this with the subroutine `TR_FUN_ACK_REI`.

7 Software, Sample and Library download

- **CD_582_-EPN GSDML for ABB AC500-S controller:**

  www.tr-electronic.de/f/zip/TR-ECE-ID-MUL-0058

- **TR TCI Device Tool software for CRC calculation:**

  www.tr-electronic.de/f/zip/TR-ECE-SW-MUL-0008

- **Sample project for ABB AC500-S control system:**

  www.tr-electronic.de/f/zip/TR-ECE-SW-MUL-0017