SIEMENS

SCE Lehrunterlagen

Siemens Automation Cooperates with Education (SCE) | 09/2015

PA Modul P01-04

SIMATIC PCS 7 – Einzelsteuerfunktion

Cooperates with Education

Automation

SIEMENS

Passende SCE Trainer Pakete zu diesen Lehrunterlagen

- **SIMATIC PCS 7 Software 3er Paket**
  Bestellnr.: 6ES7650-0XX18-0YS5
- **SIMATIC PCS 7 Software 6er Paket**
  Bestellnr.: 6ES7650-0XX18-2YS5
- **SIMATIC PCS 7 Software Upgrade Pakete 3er**
  Bestellnr.: 6ES7650-0XX18-0YE5 (V8.0 → V8.1) bzw. 6ES7650-0XX08-0YE5 (V7.1 → V8.0)
- **SIMATIC PCS 7 Hardware Set inkl. RTX-Box**
  Bestellnr.: 6ES7654-0UE13-0XS0

Bitte beachten Sie, dass diese Trainer Pakete ggf. durch Nachfolge-Pakete ersetzt werden.
Eine Übersicht über die aktuell verfügbaren SCE Pakete finden Sie unter: siemens.de/sce/tp

Fortbildungen

Für regionale Siemens SCE Fortbildungen kontaktieren Sie Ihren regionalen SCE Kontaktpartner
siemens.de/sce/contact

Weiterführende Informationen zu SIMATIC PCS 7 und SIMIT

Insbesondere Getting started, Videos, Tutorials, Handbücher und Programmierleitfaden.
siemens.de/sce/pcs7

Weitere Informationen rund um SCE

siemens.de/sce

Verwendungshinweis

Die SCE Lehrunterlage für die durchgängige Automatisierungslösung Totally Integrated Automation (TIA) wurde für das Programm „Siemens Automation Cooperates with Education (SCE)“ speziell zu Ausbildungszwecken für öffentliche Bildungs- und F&E-Einrichtungen erstellt. Die Siemens AG übernimmt bezüglich des Inhalts keine Gewähr.

Diese Unterlage darf nur für die Erstausbildung an Siemens Produkten/Systemen verwendet werden. D.h. sie kann ganz oder teilweise kopiert und an die Auszubildenden zur Nutzung im Rahmen deren Ausbildung ausgehändigt werden. Weitergabe sowie Vervielfältigung dieser Unterlage und Mitteilung ihres Inhalts ist innerhalb öffentlicher Aus- und Weiterbildungsstätten für Zwecke der Ausbildung gestattet.

Ausnahmen bedürfen der schriftlichen Genehmigung durch die Siemens AG. Ansprechpartner: Herr Roland Scheuerer roland.scheuerer@siemens.com.

Zuwiderhandlungen verpflichten zu Schadensersatz. Alle Rechte auch der Übersetzung sind vorbehalten, insbesondere für den Fall der Patentierung oder GM-Eintragung.

Der Einsatz für Industriekunden-Kurse ist explizit nicht erlaubt. Einer kommerziellen Nutzung der Unterlagen stimmen wir nicht zu.

Wir danken der TU Dresden, besonders Prof. Dr.-Ing. Leon Urbas und Dipl.-Ing. Annett Krause, der Fa. Michael Dziallas Engineering und allen weiteren Beteiligten für die Unterstützung bei der Erstellung dieser SCE Lehrunterlage.

EINZELSTEUERFUNKTIONEN

LERNZIEL

Die Studierenden können nach der Bearbeitung dieses Moduls den Begriff der Einzelsteuerfunktion im Rahmen der objektorientierten Softwarestrukturierung definieren und einordnen. Sie verstehen das Konzept, den Aufbau sowie die Funktionsweise von Einzelsteuerfunktionen und kennen typische Einzelsteuerfunktionen sowie deren Umsetzung in ***PCS 7***.

THEORIE IN KÜRZE

Das Ziel der objektorientierten Softwarestrukturierung besteht darin, die Struktur der realen Anlage durch eine entsprechende Modularisierung der Anwendersoftware möglichst eindeutig nachzubilden. Dazu wird für jeden Feldgerätetyp mindestens ein Funktionsbaustein bereitgestellt, der die gesamte Ansteuerungslogik, notwendige Schutz- und Überwachungsfunktionen sowie geeignete Bedien- und Visualisierungsmöglichkeiten bereitstellt. Das Anwenderprogramm nutzt diesen Baustein um das gewünschte Betriebsverhalten einer Maschine oder eines Prozesses zu realisieren.

Motoren und Ventile sind steuerungstechnische Einrichtungen, die im Sinne einer objektorientierten Automatisierung ebenfalls nicht direkt angesteuert, sondern zunächst als Funktionsbaustein-Typen modelliert werden. Derartige Funktionsbaustein-Typen werden als ***Einzelsteuerfunktionen (ESF)*** oder ***Individual Drive Functions (IDF)*** bezeichnet. Sie ermöglichen die Steuerung, Überwachung und Bedienung der steuerungstechnischen Einrichtung durch die Bereitstellung von entsprechenden Anschlüssen für Stell- und Steuersignale sowie für Parametrier- und Überwachungsfunktionen. Die technische Umsetzung der Steuerung wird durch eine Instanz des Funktionsbaustein-Typs realisiert und bleibt dem Nutzer verborgen. Abbildung 1 zeigt den Übergang vom realen Motor, in diesem Fall eine Pumpe, hin zu einem Baustein der entsprechenden Einzelsteuerfunktion.

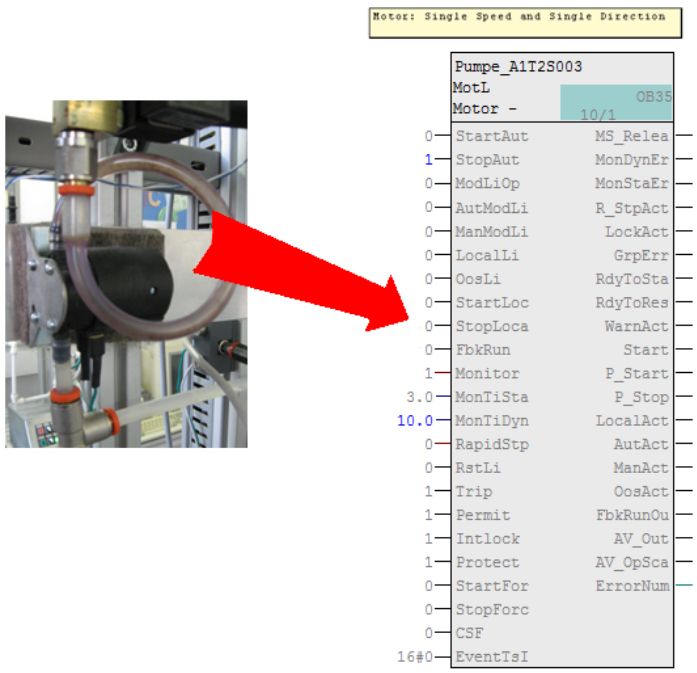


Abbildung 1: Der Übergang vom realen Motor zum Bausteine der Einzelsteuerfunktion

Steuerungstechnische Einrichtungen können grundsätzlich in vier verschiedenen Betriebsarten betrieben werden. Das Gerät kann folgende Betriebsarten aufweisen:

– ***Außer Betrieb***
– im ***Handbetrieb***
– im ***Automatikbetrieb*** oder
– im ***Vor-Ort-Betrieb***

Eine Einzelsteuerfunktion darf dabei immer nur eine Betriebsart einnehmen. Die genannten Betriebsarten können gleichwertig oder über Prioritäten hierarchisch gegliedert sein. Einzelsteuerfunktionen bieten zudem Funktionen zum Schutz vor Geräte- und Prozessfehlern an. Dazu werden verschiedene Verriegelungen sowie eine Laufzeitüberwachung für das Gerät und den gesteuerten Prozess implementiert.

Funktionsbaustein-Typen, in ***PCS 7*** als Bausteintypen bezeichnet, stellen vorgefertigte Programmteile für die Bearbeitung wiederkehrender Funktionen dar. Sie können in CFC-Pläne eingefügt und dort als Instanzen parametriert, verschaltet und an die Projekterfordernisse angepasst werden. Dabei legt der Baustein-Typ die Charakteristik für sämtliche Instanzen dieses Typs fest. ***PCS 7*** bietet bereits eine Vielzahl von leistungsfähigen und getesteten Einzelsteuerfunktionen als Bausteintypen in den leittechnischen Bibliotheken an. Diese modellieren jeweils eine steuerungstechnische Einrichtung und stellen die gesamte Ansteuerlogik bereit. Zusätzlich werden Funktionen angeboten

– zum ***Bedienen*** und ***Beobachten*** der Einzelsteuerfunktion
– zum ***Steuern*** von Signalen
– zur ***Überwachung*** und ***Alarmierung***
– zur ***Betriebszustandsauswahl***
– für ***Verriegelungen***.

Bildbausteine mit verschiedenen ***Sichten*** ermöglichen die nahtlose Integration in ein entsprechendes Prozessleitsystem.

Einzelsteuerfunktionen ermöglichen eine effiziente Entwicklung leistungsfähiger, qualitativ hochwertiger Lösungen. Sie modularisieren und typisieren wiederkehrende Funktionalitäten. Damit werden diese wiederverwendbar und zentral änderbar, was den Entwicklungsprozess erheblich beschleunigt.

THEORIE

OBJEKTORIENTIERTE SOFTWARESTRUKTURIERUNG

Das Ziel der objektorientierten Softwarestrukturierung besteht darin, die Struktur der realen Anlage durch eine entsprechende Modularisierung der Anwendersoftware möglichst eindeutig nachzubilden. Dazu wird für jedes Feldgerät in der Anlage ein eigenes Programm erstellt. Für jeden Feldgerätetyp wird mindestens ein Funktionsbaustein bereitgestellt.

Dieser Baustein realisiert die gesamte Ansteuerungslogik für diesen Feldgerätetyp. Darüber hinaus stellt er notwendige Schutz- und Überwachungsfunktionen sowie geeignete Bedien- und Visualisierungsmöglichkeiten bereit. Er kapselt damit die gesamte Funktionalität, die im Zusammenhang mit dem entsprechenden Feldgerätetyp notwendig ist. Das Anwenderprogramm nutzt diesen Baustein, um eine gewünschte Steuerung für eine Maschine oder einen Prozess zu realisieren, ohne dabei auf Kenntnisse der internen Daten und Operationen des Funktionsbausteins zurückgreifen zu müssen.

KANALFUNKTIONEN (DRIVER)

In Ergänzung zur Behandlung der Feldgeräte durch eigene wiederverwendbare Bausteine ist es häufig sinnvoll, die Peripherieanbindung ebenfalls durch ***Kanalbausteine*** (engl. Driver) zu abstrahieren. Es ist zwar stets möglich, über Symbolnamen oder Adresse direkt auf das Prozessabbild zuzugreifen, allerdings müssen hierbei die möglicherweise vielfältigen Parameter zur Konfiguration des Kanals an anderer Stelle gesetzt werden. Dies führt schnell zu unübersichtlichen Programmen. ***PCS 7*** bietet eine Reihe von Kanalbausteinen, die zum einen die Statussignale der Baugruppen auswerten und zum anderen das Testen und die Inbetriebnahme von Automatisierungsprogrammen durch Simulationsmodi unterstützen. In den analogen Kanalbausteinen entsprechend Tabelle 1 erfolgt zudem durch die Parameter VLRANGE und VHRANGE eine Abbildung von den internen digitalen Größen auf die physikalischen Rechen- und Anzeigegrößen. ***PCS 7*** kann bei Verwendung von Kanalbausteinen die notwendigen Treiber automatisch erzeugen. Die Kanalbausteine werden deshalb in den Vorlagen der ***PCS 7*** Bibliotheken vielfach eingesetzt.

Tabelle 1: Auflistung verschiedener Kanalbausteine zur Abstraktion der Peripherieanbindung

Kanalbaustein	Baustein	Verschaltung, Parameter
Digitaler Ausgang	PCS7DiOu	PV_Out
Digitaler Eingang	PCS7DiIn	PV_In
Analoger Ausgang	PCS7AnOu	PV_Out, Scale
Analoger Eingang	PCS7AnIn	PV_In, Scale

EINZELSTEUERFUNKTIONEN

Motoren und Ventile sind als steuerungstechnische Einrichtungen in der Fabrik- und Prozessautomatisierung von zentraler Bedeutung. Es existiert eine Vielzahl gängiger Typen mit spezifischem Bedien- und Meldeverhalten. Im Sinne einer objektorientierten Automatisierung werden derartige Einrichtungen nicht direkt angesteuert, sondern zunächst als Funktionsbaustein-Typen modelliert. Die Ansteuerung erfolgt stets indirekt über eine Instanz des entsprechenden Funktionsbaustein-Typs. Funktionsbausteine für Motoren und Ventile werden als ***Einzelsteuerfunktionen (ESF)*** oder ***Individual Drive Functions (IDF)*** bezeichnet. Einzelsteuerfunktionen ermöglichen die Steuerung, Überwachung und Bedienung von steuerungstechnischen Einrichtungen durch die Bereitstellung von entsprechenden Anschlüssen für Stell- und Steuersignale sowie für Parametrier- und Überwachungsfunktionen. Die technische Umsetzung der Steuerung, wie zum Beispiel das Anlaufverhalten, die Ansteuerung des Antriebs oder die Geräteüberwachung wird durch die Funktionsbaustein-Instanz realisiert und bleibt dem Nutzer verborgen. ***PCS 7*** bietet bereits eine Vielzahl von leistungsfähigen und getesteten Einzelsteuerfunktionen als Bausteintypen in den leittechnischen Bibliotheken an. Tabelle 2 fasst die Einzelsteuerfunktionen der ***PCS 7 Advanced Process Library*** [2] zusammen.

Tabelle 2: Die Einzelsteuerfunktionen der ***PCS 7 Advanced Prozess Library***

Einzelsteuer-funktion	Anwendungsbereich	Objektname
MotL	Ansteuerung von Motoren mit einem Steuersignal (ein/aus) und einer Laufrückmeldung	FB 1850
MotRevL	Ansteuerung von Motoren mit zwei Drehrichtungen (Linkslauf/Rechtslauf) und maximal zwei Rückmeldungen	FB 1851
MotSpedL	Ansteuerung von Motoren mit zwei Geschwindigkeiten (langsam/schnell) und maximal zwei Rückmeldungen	FB 1856
VlvL	Ansteuerung von Steuerventilen mit einem Steuersignal (öffnen/schließen) und Stellungs-rückmeldesignalen (offen/geschlossen)	FB 1899
VlvMotL	Ansteuerung von Motorventilen mit zwei Steuersignalen und Stellungsrückmeldesignalen (offen/geschlossen).	FB 1900

SCHUTZMAßNAHMEN

Bei der Ansteuerung von steuerungstechnischen Einrichtungen sind verschiedene Schutzmaßnahmen zu treffen. Zum einen sind die Einrichtungen selbst vor Fehlern zu schützen. Zum anderen muss der gesteuerte Prozess im Fehlerfall in einen sicheren Zustand überführt und dort solange gehalten werden, bis der Fehlerfall beseitigt ist.

Gerätefehler (zum Beispiel Kabelbruch, Achsbruch) können steuerungstechnisch nicht verhindert werden. Die Auswirkungen können jedoch durch Redundanzkonzepte minimiert werden. ***Prozessfehler*** (zum Beispiel Behälterüberlauf, Trockenlauf einer Pumpe) sollen hingegen direkt durch die Steuerung verhindert werden. Dazu werden entsprechende ***Verriegelungen*** implementiert. Erkennt die Einzelsteuerfunktion aufgrund der aktuellen Eingangswerte einen gefährlichen Prozesszustand, so wird das gesteuerte Gerät in einen sicheren Zustand überführt (siehe Kapitel ‚Anlagensicherung'). Das Gerät wird solange in diesem Zustand gehalten, wie der gefährliche Prozesszustand andauert. Üblicherweise werden Verriegelungen mithilfe einer ***Verriegelungsmatrix*** spezifiziert.

Um einen eingetretenen Gerätefehler zu erkennen, führt eine Einzelsteuerfunktion häufig eine ***Laufzeitüberwachung*** durch. Mithilfe bestimmter Sensorinformationen, zum Beispiel von Endlagensensoren in Ventilen, überprüft die Einzelsteuerfunktion, ob die ausgegebenen Stellsignale auch die geforderte Wirkung erzielen. Stehen die gemessenen Werte über einen bestimmten Zeitraum hinweg in Widerspruch zu den ausgegebenen Stellsignalen, so liegt eine Störung vor. Wird ein solcher Laufzeitfehler erkannt, so wird das übergeordnete Leitsystem alarmiert und das gesteuerte Gerät wird deaktiviert. Das Gerät bleibt solange inaktiv, bis der Laufzeitfehler beseitigt und der Alarm quittiert worden ist. Häufig werden zur Erkennung von Gerätefehlern einfache binäre Schutzschalter verwendet.

BETRIEBSARTEN

Steuerungstechnische Einrichtungen werden im Allgemeinen nicht ausschließlich automatisch betrieben. Zeitweise ist es notwendig die Steuerung in der Leitwarte manuell durchzuführen oder das Gerät direkt vor Ort anzusteuern, zum Beispiel für Reparaturarbeiten. Es wird daher zwischen vier grundlegenden Betriebsarten unterschieden:

- ***Außer Betrieb***: Das Gerät ist nicht aktiv.
- ***Automatikbetrieb***: Die Einzelsteuerfunktion wird von einem übergeordneten Programm automatisch angesteuert.
- ***Handbetrieb***: Die Einzelsteuerfunktion wird über eine Bediengraphik des Leitsystems direkt durch den Bediener angesteuert.
- ***Vor-Ort-Betrieb***: Der Bediener bedient das Gerät direkt vor Ort, zum Beispiel über ein Bedientableau.

Eine Einzelsteuerfunktion darf sich immer nur in genau einer Betriebsart befinden. Es existieren verschiedene Konzepte, wie die damit verbundene Betriebsartenumschaltung sicher und eindeutig realisiert werden kann. Grundsätzlich lassen sich diese Konzepte unterscheiden zwischen einer Gleichberechtigung der Betriebsarten und einer Betriebsartenhierarchie. Im letztgenannten Fall werden die möglichen Betriebsarten zusätzlich eindeutig priorisiert. Eine angewählte Betriebsart wird in diesem Fall genau dann geändert, wenn das Gerät entweder nicht aktiv ist (Betriebsart: ***Außer Betrieb***) oder wenn die gewünschte neue Betriebsart eine höhere Priorität hat als die bereits angewählte.

FUNKTIONSBAUSTEIN-TYPEN IN PCS 7

Funktionsbaustein-Typen werden in ***PCS 7*** als Bausteintypen bezeichnet und stellen vorgefertigte Programmteile für die Bearbeitung wiederkehrender Funktionen dar. Sie können in CFC-Pläne eingefügt werden und dort als Instanzen parametriert, verschaltet und an die Projekterfordernisse angepasst werden.

Dabei legt der Baustein-Typ die Charakteristik für sämtliche Instanzen dieses Typs fest. Die verwendeten Bausteintypen eines Projektes werden dazu in der Stammdatenbibliothek abgelegt. Wird der dort abgelegte Bausteintyp geändert, so werden die Änderungen unmittelbar von sämtlichen Instanzen übernommen. Dieses Konzept der Typisierung unterstützt das rationelle Engineering durch die Wiederverwendbarkeit und die zentrale Änderbarkeit häufig wiederkehrender Funktionen.

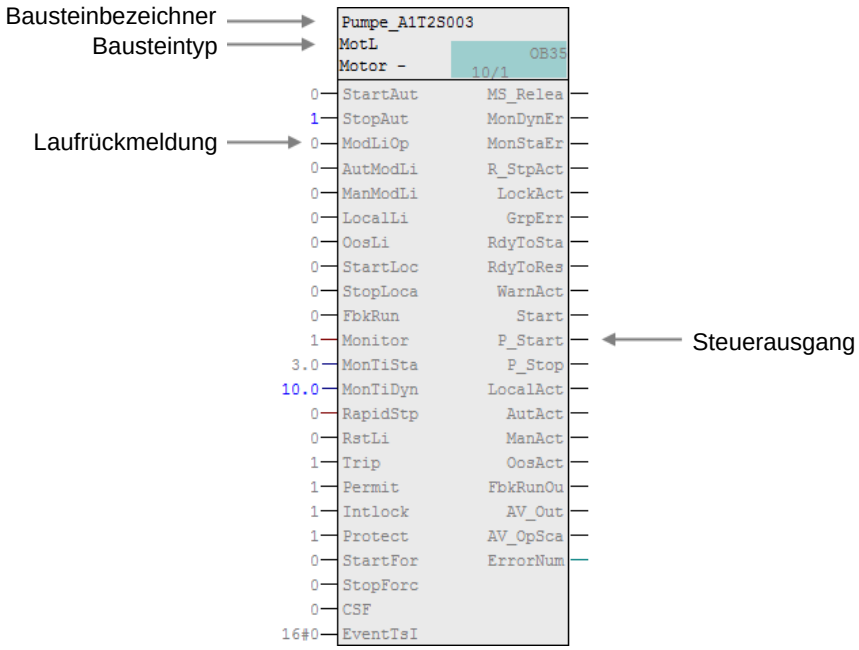


Abbildung 2: Baustein der Einzelsteuerfunktion Motor_Lean

Eine Einzelsteuerfunktion in ***PCS 7*** modelliert eine steuerungstechnische Einrichtung und stellt die gesamte Ansteuerlogik bereit. Abbildung 2 beschreibt am Beispiel der Einzelsteuerfunktion ***Motor_Lean*** den grundsätzlichen Aufbau des entsprechenden Motorbausteins.

Der Baustein bietet darüber hinaus die folgenden Funktionen:

Bedienen, Beobachten, Melden

Über einen Anzeige- und Bedienbereich können Prozess- und Sollwerte bedient und beobachtet werden. Bedienberechtigungen und Wartungsfreigaben können gesteuert werden. Allgemeine und instanzspezifische Meldungen geben Aufschluss über den Geräte- und Prozessstatus.

Steuerung von Signalen

Steuersignale können statisch oder gepulst ausgegeben werden. Der Signalstatus, also die Qualität der Stellsignale, wird überwacht. Es können interne und externe Sollwerte sowie simulierte Werte vorgegeben werden. Außerdem können Rampen oder Totzonen eingestellt werden.

Überwachung

Der Baustein kann Grenzwerte überwachen und bei Grenzwertverletzungen entsprechende Warnungen oder Alarme generieren. Außerdem können Rückmeldungen von Stellsignalen überwacht werden.

Verriegelungsfunktionen

Der Baustein ermöglicht eine einfache Einschaltfreigabe, eine Verriegelung ohne Rücksetzen sowie eine Verriegelung mit Rücksetzen. Er implementiert eine Motorschutzfunktion, die den Motor bei thermischer Überlast abschalten kann. Zusätzlich ist für Motoren ein Schnellstopp verfügbar, der in allen Betriebsarten und -zuständen höchste Priorität hat. Im Fall einer Verriegelung wird das Gerät automatisch in einen energielosen Zustand und damit in eine definierte Sicherheitsstellung überführt.

Betriebszustandsauswahl

Die eingangs vorgestellten Betriebsarten Vor-Ort-Betrieb, Automatikbetrieb, Handbetrieb und Außer Betrieb stehen für sämtliche Einzelsteuerfunktionen in ***PCS 7*** zur Verfügung. Sie sind in der angegebenen Reihenfolge absteigend priorisiert, Automatik- und Handbetrieb besitzen die gleiche Priorität. Darüber hinaus ist es möglich, den Baustein über verschaltbare Eingangsparameter unabhängig von der aktuell anstehenden Ansteuerung in einen anderen Betriebszustand zu versetzen (***Erzwingen*** bzw. ***Forcen*** von Betriebszuständen).

Bildbausteine mit verschiedenen Sichten

Bildbausteine bieten für jeden Bausteintyp ein entsprechendes Bausteinsymbol und je nach Anwendungsfall entsprechende Sichten an. Typische Bildbausteine sind zum Beispiel das Bausteinsymbol selbst, die Parametersicht von Motoren und Ventilen oder die Grenzwertsicht von Motoren.

Diese Aufzählung zeigt deutlich die Komplexität und den Funktionsumfang einer üblichen Einzelsteuerfunktion. Die Anzahl der verfügbaren Ein- und Ausgänge bei diesen Bausteintypen ist dementsprechend groß. So besitzt die Einzelsteuerfunktion ***Motor_Lean*** insgesamt 53 Anschlüsse. Um die Komplexität des Programmentwurfs dennoch gering zu halten, ist es möglich, nicht benötigte Ein- oder Ausgänge zu verbergen. Die Einzelsteuerfunktionen in ***PCS 7*** verwenden zudem ein einheitliches und durchgängiges Schema für die Bezeichnung der Ein- und Ausgänge.

Die Einzelsteuerfunktionen in ***PCS 7*** bieten einen großen Funktionsumfang und garantieren eine konstant hohe Qualität und Zuverlässigkeit der verwendeten Algorithmen. Sämtliche Bausteintypen sind umfangreich getestet und haben sich bereits industriell bewährt. Damit verkürzt sich der Aufwand für die Entwicklung leistungsfähiger, qualitativ hochwertiger Lösungen erheblich.

LITERATUR

[1] Seitz, M. (2008): Speicherprogrammierbare Steuerungen. Hanser Fachbuchverlag

[2] SIEMENS (2014): Prozessleitsystem PCS 7: Advanced Process Library (V8.1). A5E33257528-AA. (http://support.automation.siemens.com/WW/view/de/90682917)

SCHRITT-FÜR-SCHRITT-ANLEITUNG

AUFGABENSTELLUNG

Ein Pumpenmotor zum Ablassen der Flüssigkeit aus dem Reaktor R001 soll als erstes Programm im ***Continuous Function Chart (CFC)*** erstellt werden. Der Pumpenmotor hat einen Ausgang zum Ansteuern der Pumpe und eine Rückmeldung zur Kontrolle, ob die Pumpe auch läuft.

Tabelle 3: Zuordnungsliste

Symbol	Adresse	Datentyp	Symbolkommentar
A1.T2.A1T2S003.SO+.O+	E 1.3	BOOL	Pumpe Ablass Reaktor R001 Rückmeldung ein
A1.T2.A1T2S003.SV.C	A 3.4	BOOL	Pumpe Ablass Reaktor R001 Stellsignal

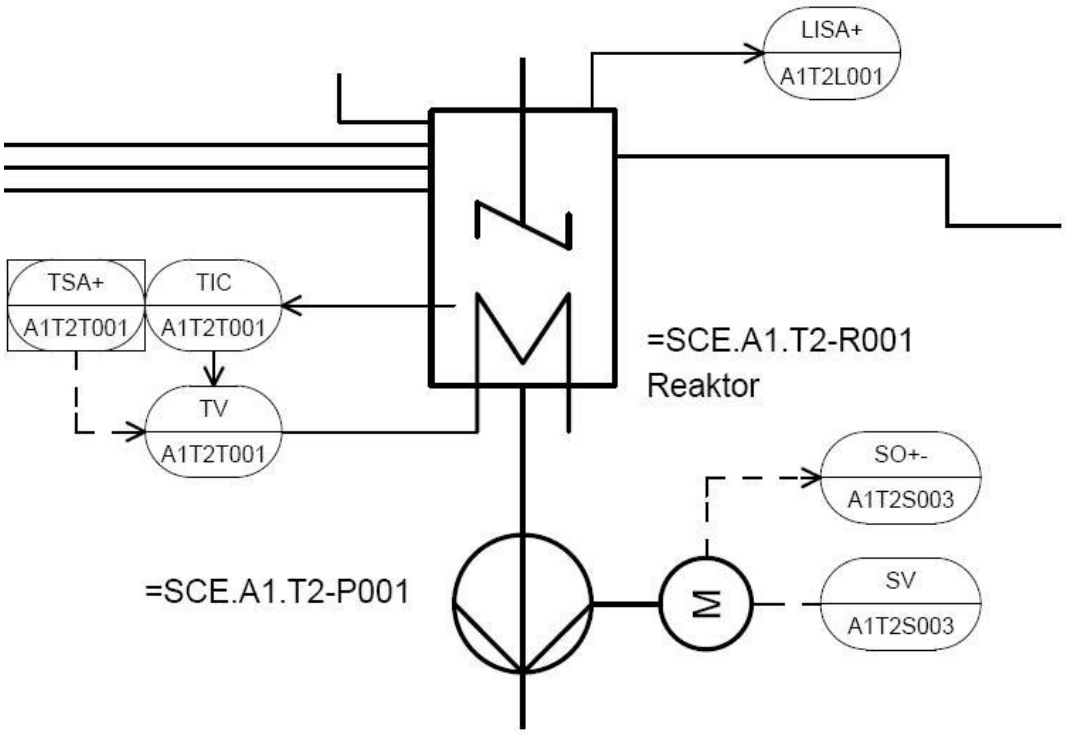


Abbildung 3: Ausschnitt aus dem R&I-Fließbild

Bei der Erstellung des Programms wird ein vorgefertigter Plan ‚Motor_Lean' aus einer ***PCS 7***-Bibliothek verwendet. Dieser wird in die projekteigene Stammdatenbibliothek kopiert und dort angepasst. Anschließend wird das Programm in die SPS-Simulation geladen und getestet.

LERNZIEL

In diesem Kapitel lernt der Studierende:

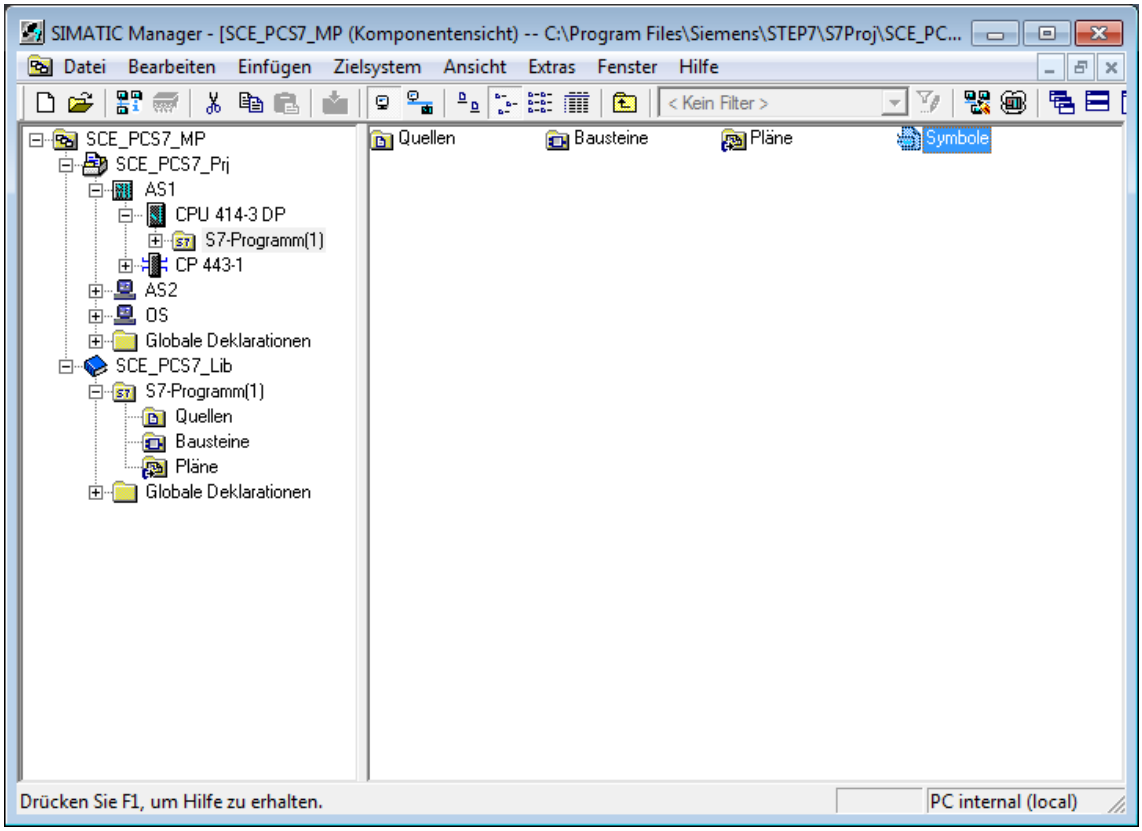
- Anlegen und importieren von Symbolen via Symboltabelle
- Verwendung von Stammdatenbibliotheken
- CFC-Pläne anlegen und bearbeiten
- Projekt zentral übersetzen und laden
- Testen des Programms mit Hilfe der Steuerfunktionen im CFC

Diese Anleitung baut auf dem Projekt ‚PCS7_SCE_0103_Ueb_R1503.zip' auf.

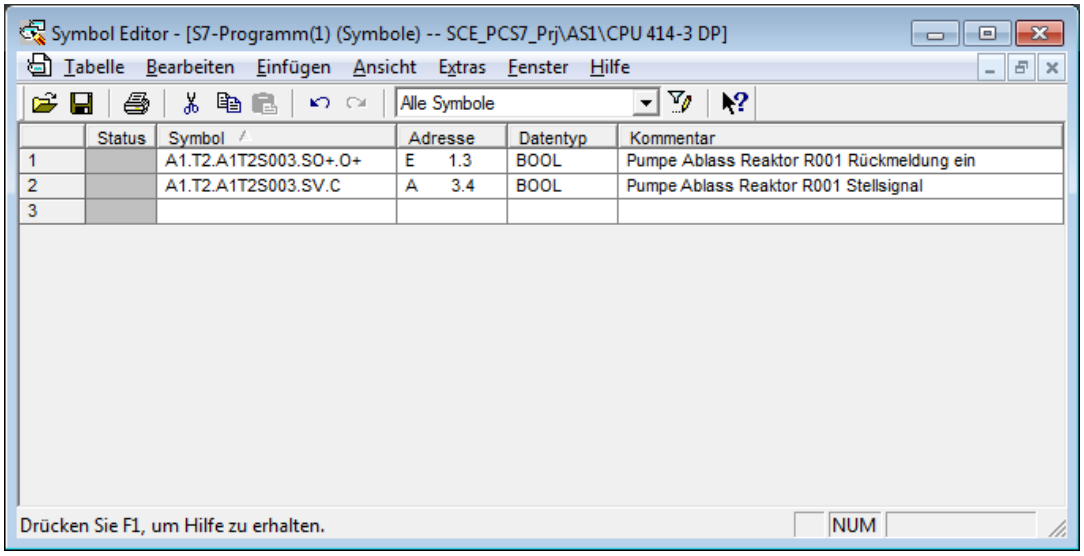
PROGRAMMIERUNG

1. Bevor Sie mit der Programmierung der Einzelsteuerfunktion für den Pumpenmotor beginnen, werden die Symbole für die globalen Variablen angelegt. Sie wählen hierzu die Komponentensicht im SIMATIC Manager, markieren den Ordner ‚S7-Programm(1)' und öffnen mit einem Doppelklick auf Symbole die Symboltabelle.

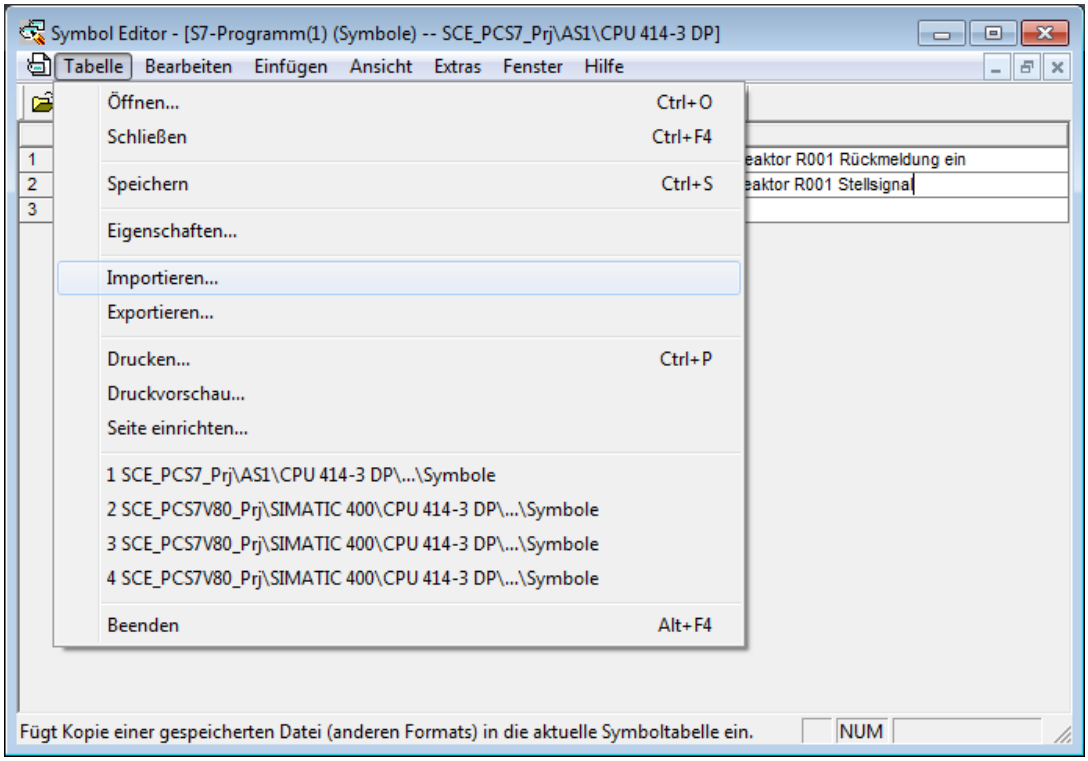
   (→ SIMATIC Manager → Ansicht → Komponentensicht → AS1 → CPU 414-3 DP → S7-Programm(1) → Symbole)



2. In der Symboltabelle können nun Symbol und Symbolkommentar zu jedem Operanden festgelegt werden.

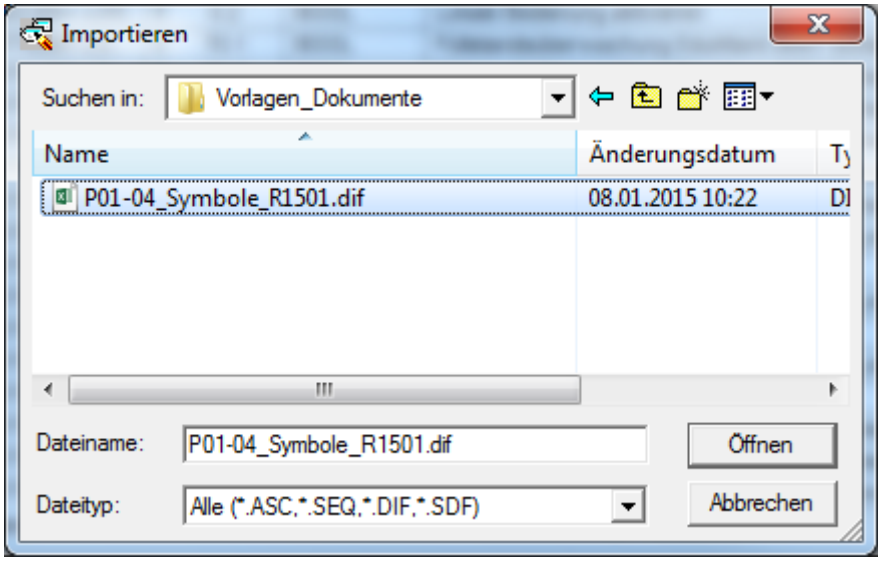


3. Falls vorhanden, kann auch der Inhalt für die komplette Symboltabelle im *.dif- Format importiert werden (→ Tabelle → Importieren). In diesem Fall wird die importierte Tabelle in die bereits vorhandene Tabelle integriert.



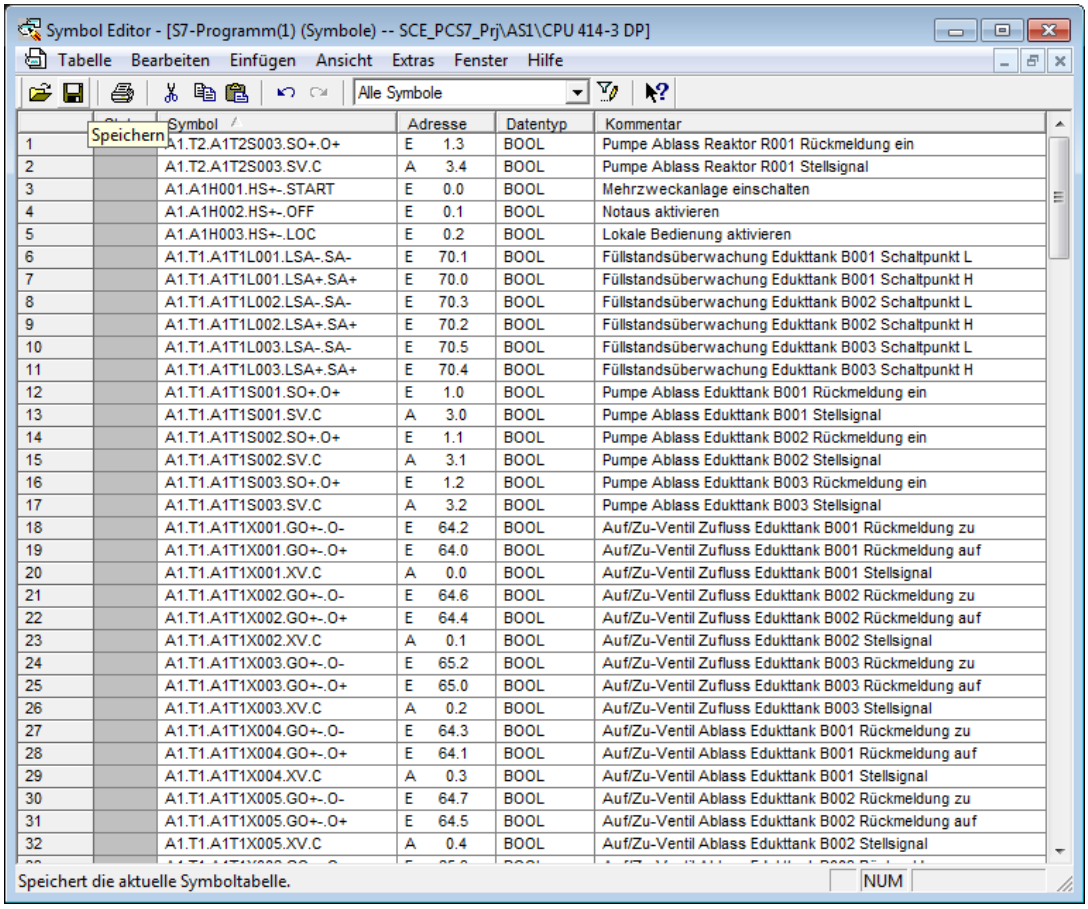
4. Nun erfolgt die Auswahl der Quelldatei im ‚Data Interchange Format' (*.DIF)

   (→ P01-04_Symbole_R1501.dif → Öffnen)



5. Die fertige Symboltabelle muss vor dem Schließen noch gespeichert werden.

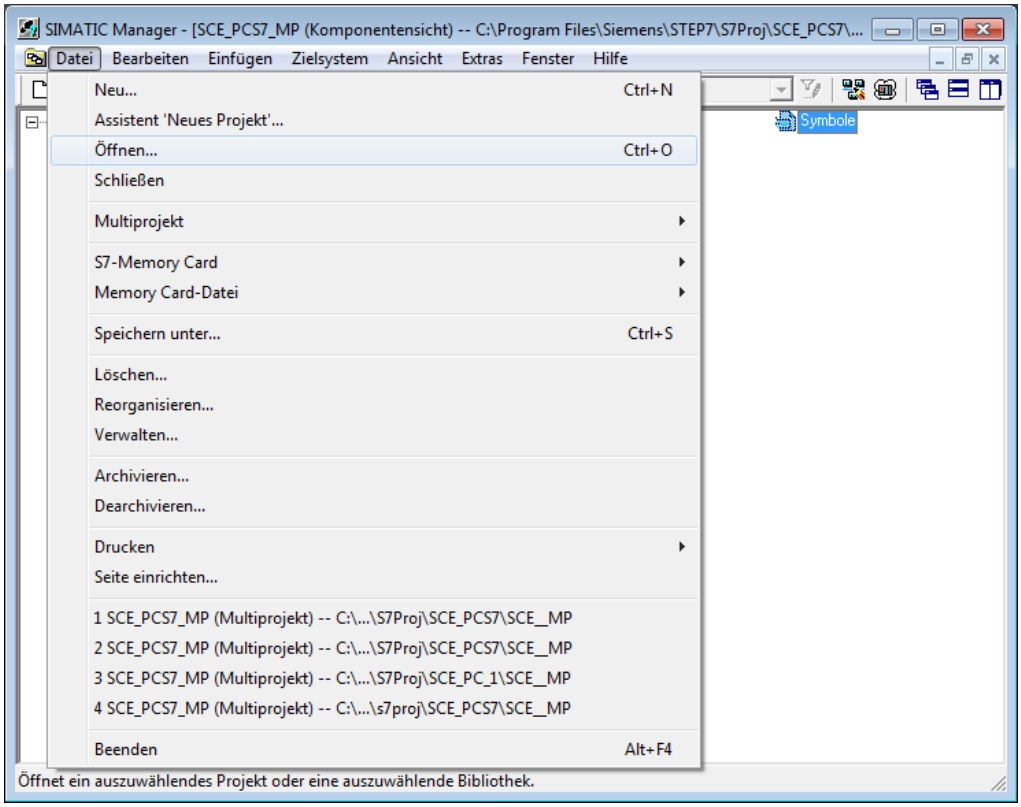
   (→ Speichern → )

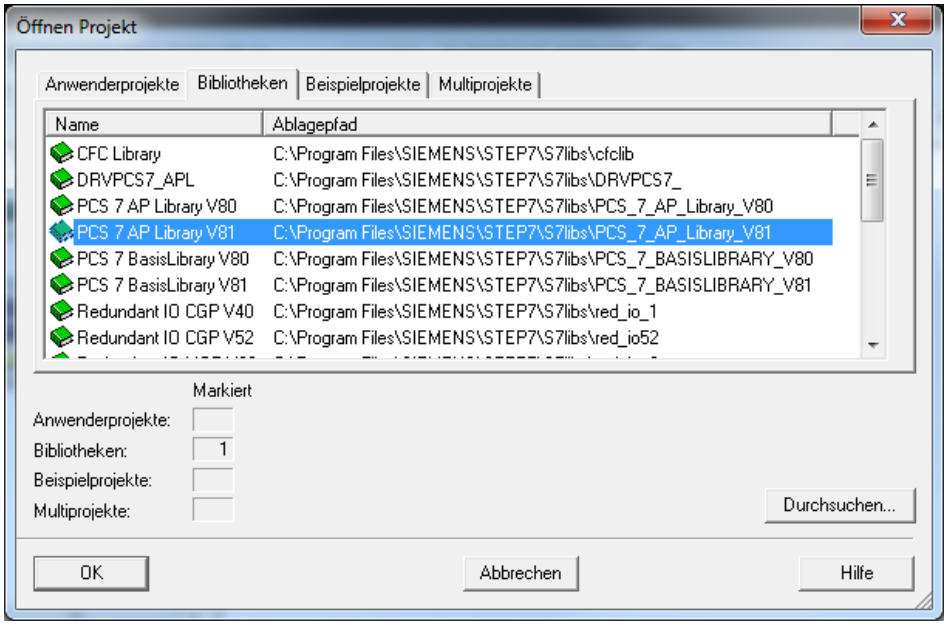


	Symbol	Adresse	Datentyp	Kommentar
1	A1.T2.A1T2S003.SO+.O+	E 1.3	BOOL	Pumpe Ablass Reaktor R001 Rückmeldung ein
2	A1.T2.A1T2S003.SV.C	A 3.4	BOOL	Pumpe Ablass Reaktor R001 Stellsignal
3	A1.A1H001.HS+-.START	E 0.0	BOOL	Mehrzweckanlage einschalten
4	A1.A1H002.HS+-.OFF	E 0.1	BOOL	Notaus aktivieren
5	A1.A1H003.HS+-.LOC	E 0.2	BOOL	Lokale Bedienung aktivieren
6	A1.T1.A1T1L001.LSA-.SA-	E 70.1	BOOL	Füllstandsüberwachung Edukttank B001 Schaltpunkt L
7	A1.T1.A1T1L001.LSA+.SA+	E 70.0	BOOL	Füllstandsüberwachung Edukttank B001 Schaltpunkt H
8	A1.T1.A1T1L002.LSA-.SA-	E 70.3	BOOL	Füllstandsüberwachung Edukttank B002 Schaltpunkt L
9	A1.T1.A1T1L002.LSA+.SA+	E 70.2	BOOL	Füllstandsüberwachung Edukttank B002 Schaltpunkt H
10	A1.T1.A1T1L003.LSA-.SA-	E 70.5	BOOL	Füllstandsüberwachung Edukttank B003 Schaltpunkt L
11	A1.T1.A1T1L003.LSA+.SA+	E 70.4	BOOL	Füllstandsüberwachung Edukttank B003 Schaltpunkt H
12	A1.T1.A1T1S001.SO+.O+	E 1.0	BOOL	Pumpe Ablass Edukttank B001 Rückmeldung ein
13	A1.T1.A1T1S001.SV.C	A 3.0	BOOL	Pumpe Ablass Edukttank B001 Stellsignal
14	A1.T1.A1T1S002.SO+.O+	E 1.1	BOOL	Pumpe Ablass Edukttank B002 Rückmeldung ein
15	A1.T1.A1T1S002.SV.C	A 3.1	BOOL	Pumpe Ablass Edukttank B002 Stellsignal
16	A1.T1.A1T1S003.SO+.O+	E 1.2	BOOL	Pumpe Ablass Edukttank B003 Rückmeldung ein
17	A1.T1.A1T1S003.SV.C	A 3.2	BOOL	Pumpe Ablass Edukttank B003 Stellsignal
18	A1.T1.A1T1X001.GO+-.O-	E 64.2	BOOL	Auf/Zu-Ventil Zufluss Edukttank B001 Rückmeldung zu
19	A1.T1.A1T1X001.GO+-.O+	E 64.0	BOOL	Auf/Zu-Ventil Zufluss Edukttank B001 Rückmeldung auf
20	A1.T1.A1T1X001.XV.C	A 0.0	BOOL	Auf/Zu-Ventil Zufluss Edukttank B001 Stellsignal
21	A1.T1.A1T1X002.GO+-.O-	E 64.6	BOOL	Auf/Zu-Ventil Zufluss Edukttank B002 Rückmeldung zu
22	A1.T1.A1T1X002.GO+-.O+	E 64.4	BOOL	Auf/Zu-Ventil Zufluss Edukttank B002 Rückmeldung auf
23	A1.T1.A1T1X002.XV.C	A 0.1	BOOL	Auf/Zu-Ventil Zufluss Edukttank B002 Stellsignal
24	A1.T1.A1T1X003.GO+-.O-	E 65.2	BOOL	Auf/Zu-Ventil Zufluss Edukttank B003 Rückmeldung zu
25	A1.T1.A1T1X003.GO+-.O+	E 65.0	BOOL	Auf/Zu-Ventil Zufluss Edukttank B003 Rückmeldung auf
26	A1.T1.A1T1X003.XV.C	A 0.2	BOOL	Auf/Zu-Ventil Zufluss Edukttank B003 Stellsignal
27	A1.T1.A1T1X004.GO+-.O-	E 64.3	BOOL	Auf/Zu-Ventil Ablass Edukttank B001 Rückmeldung zu
28	A1.T1.A1T1X004.GO+-.O+	E 64.1	BOOL	Auf/Zu-Ventil Ablass Edukttank B001 Rückmeldung auf
29	A1.T1.A1T1X004.XV.C	A 0.3	BOOL	Auf/Zu-Ventil Ablass Edukttank B001 Stellsignal
30	A1.T1.A1T1X005.GO+-.O-	E 64.7	BOOL	Auf/Zu-Ventil Ablass Edukttank B002 Rückmeldung zu
31	A1.T1.A1T1X005.GO+-.O+	E 64.5	BOOL	Auf/Zu-Ventil Ablass Edukttank B002 Rückmeldung auf
32	A1.T1.A1T1X005.XV.C	A 0.4	BOOL	Auf/Zu-Ventil Ablass Edukttank B002 Stellsignal

6. ***PCS 7*** stellt in umfangreichen Bibliotheken eine Vielzahl an fertigen Bausteinen und auch vorgefertigte Pläne, so genannte Templates, zur Verfügung. Für den Pumpenmotor nutzen Sie dieses Template. Dafür öffnen Sie die ***PCS 7 AP Library V8.1***.

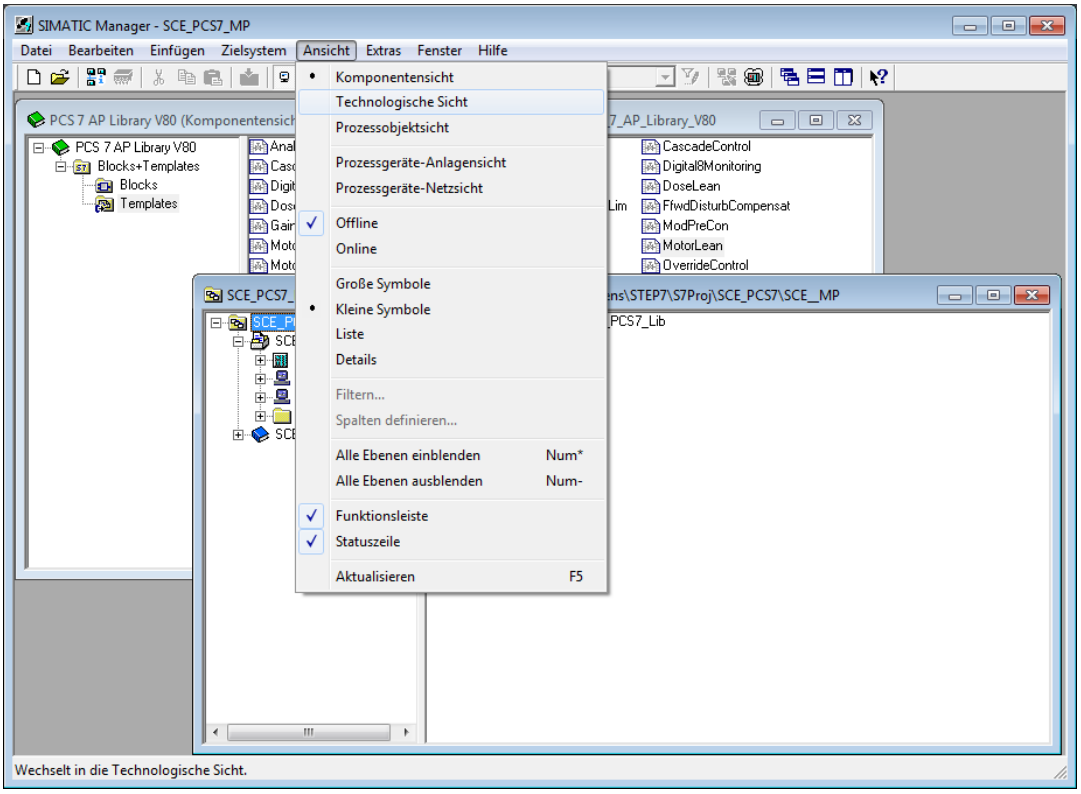
   (→ Datei → Öffnen → Bibliotheken → PCS 7 AP Library V8.1 → OK)





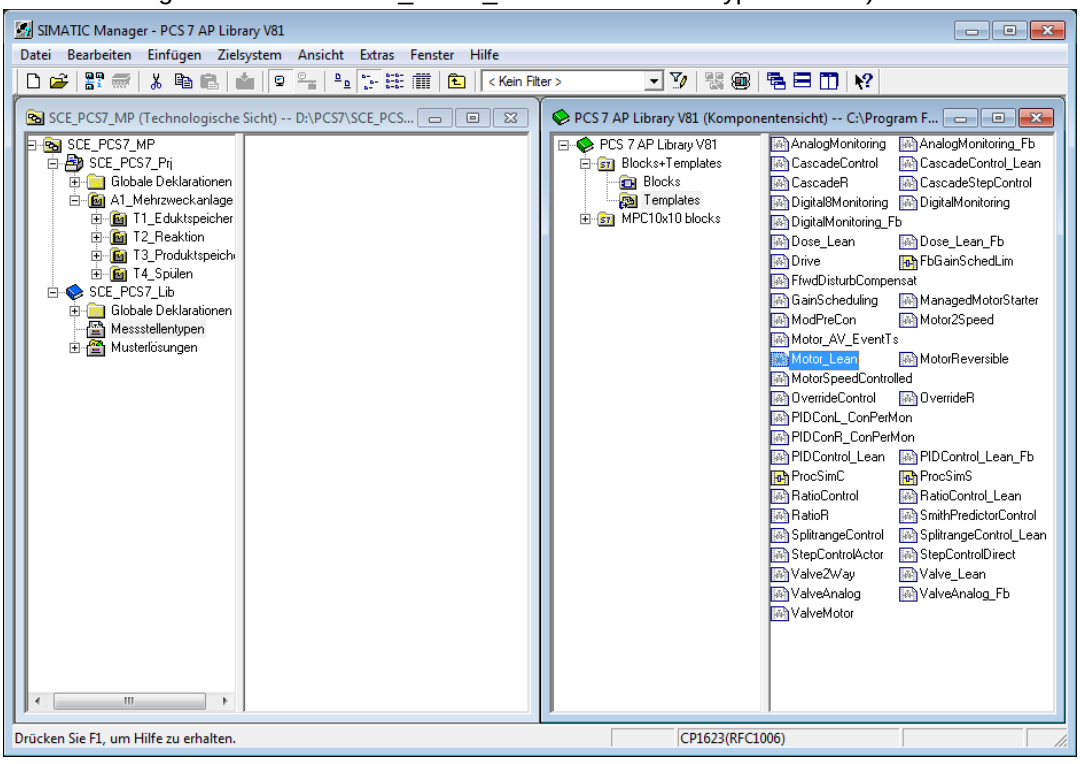
7. Um dieses Template aus der Bibliothek in das Projekt zu ziehen, gehen Sie auf die technologische Sicht des Projektes.

   (→ Ansicht → Technologische Sicht)

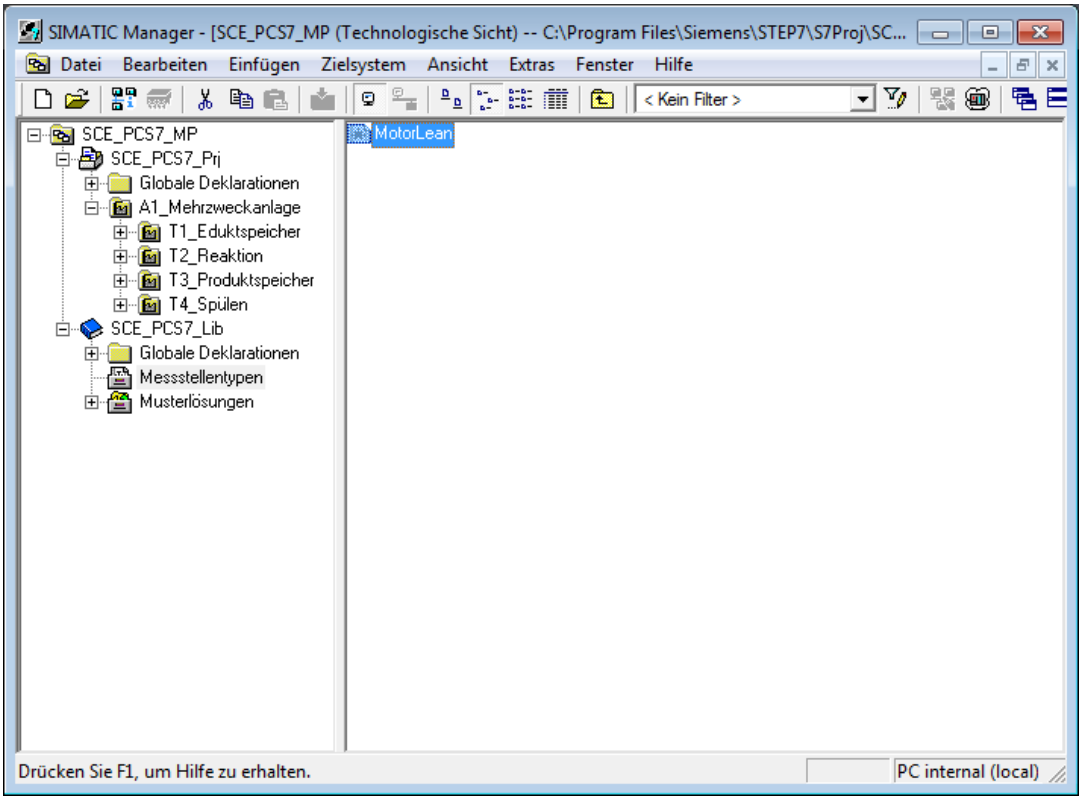


8. Per Drag&Drop wird Motor_Lean (Bibliothek, Templates) nach Messstellentypen (Stammdatenbibliothek) verschoben.

   (→ PCS 7 Library → Blocks+Tamplates → Templates → Motor_Lean nach Technologische Sicht → SCE_PCS7_Lib → Messstellentypen ziehen)



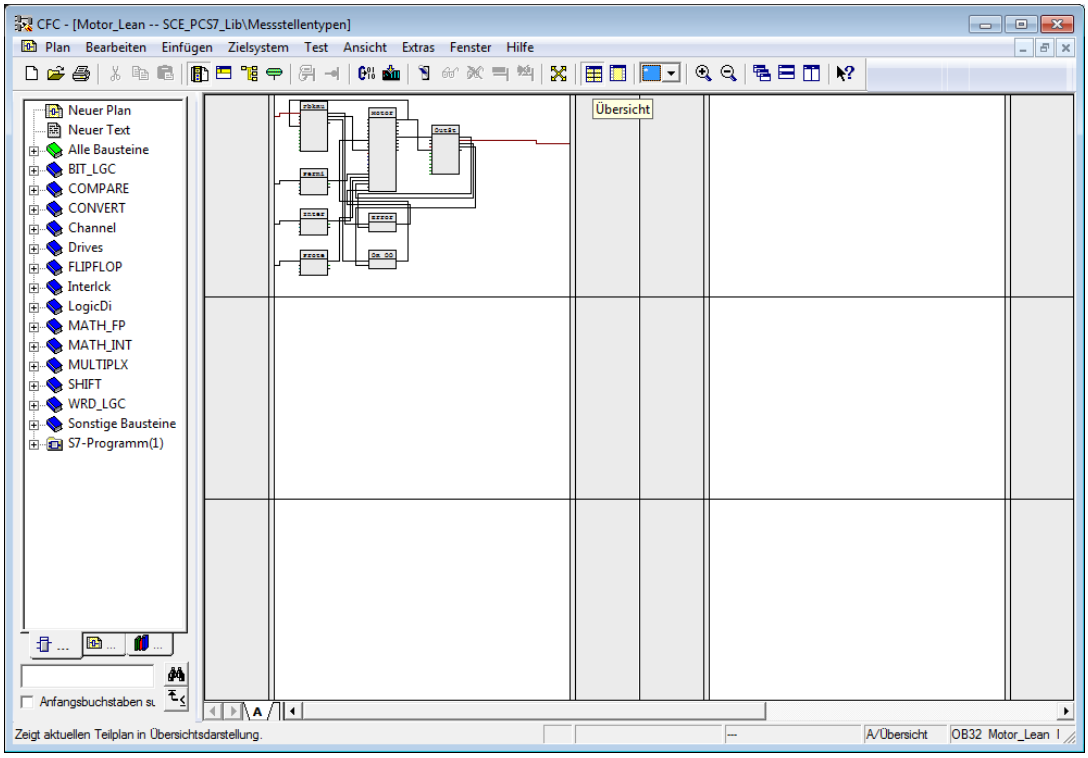
9. Es soll nun eine Änderung zentral für alle Messstellen des Typs ‚Motor_Lean' erfolgen. Dies geschieht, indem der CFC-Plan ‚Motor_Lean' aus der Stammdatenbibliothek per Doppelklick geöffnet wird.
   (→ Motor_Lean)



Hinweis: CFC steht für Continuous Function Chart und ist eine grafische Programmiersprache für die Beschreibung kontinuierlicher Vorgänge. Im CFC werden vorgefertigte Bausteine platziert, parametriert und verschaltet. So erstellt der Programmierer eine Gesamt-Software-Struktur für die Steuerung und Regelung einer Maschine.

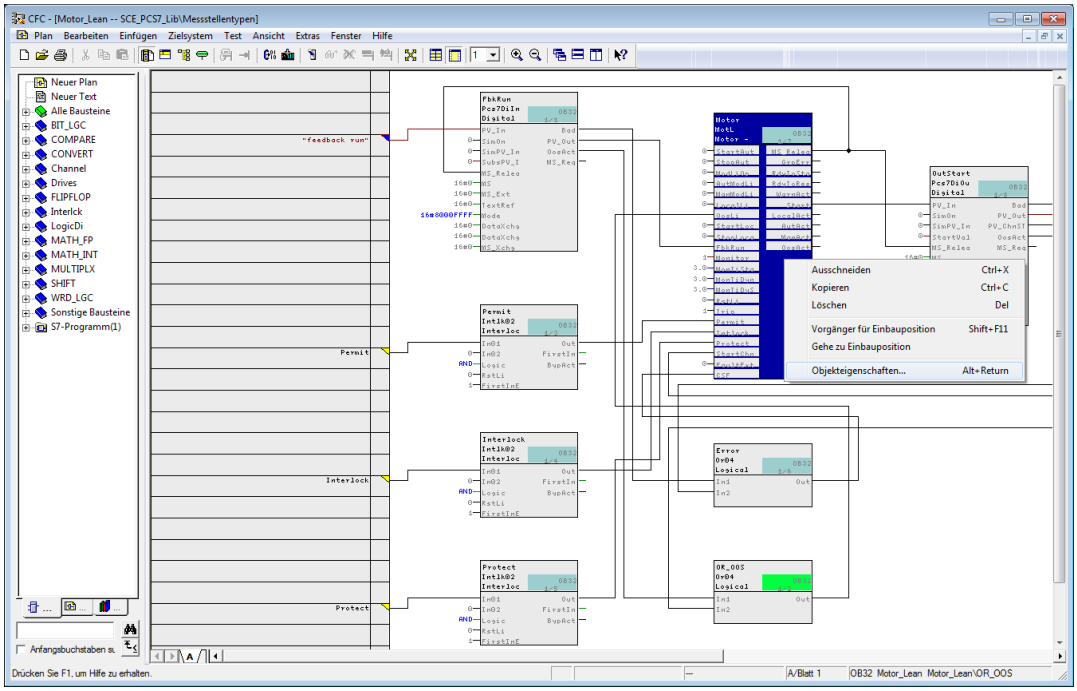
10. Ein CFC-Plan besteht aus Teilplänen mit jeweils sechs Blättern. In der Übersicht werden alle sechs Blätter mit ihren grauen Randleisten angezeigt. An den Randleisten sehen Sie links die kommenden Signale und rechts die gehenden Signale eines Blattes. Mit einem Doppelklick auf ein Blatt können Sie in die Blattsicht wechseln.

    (→ Übersicht → Doppelklick auf erstes Blatt)

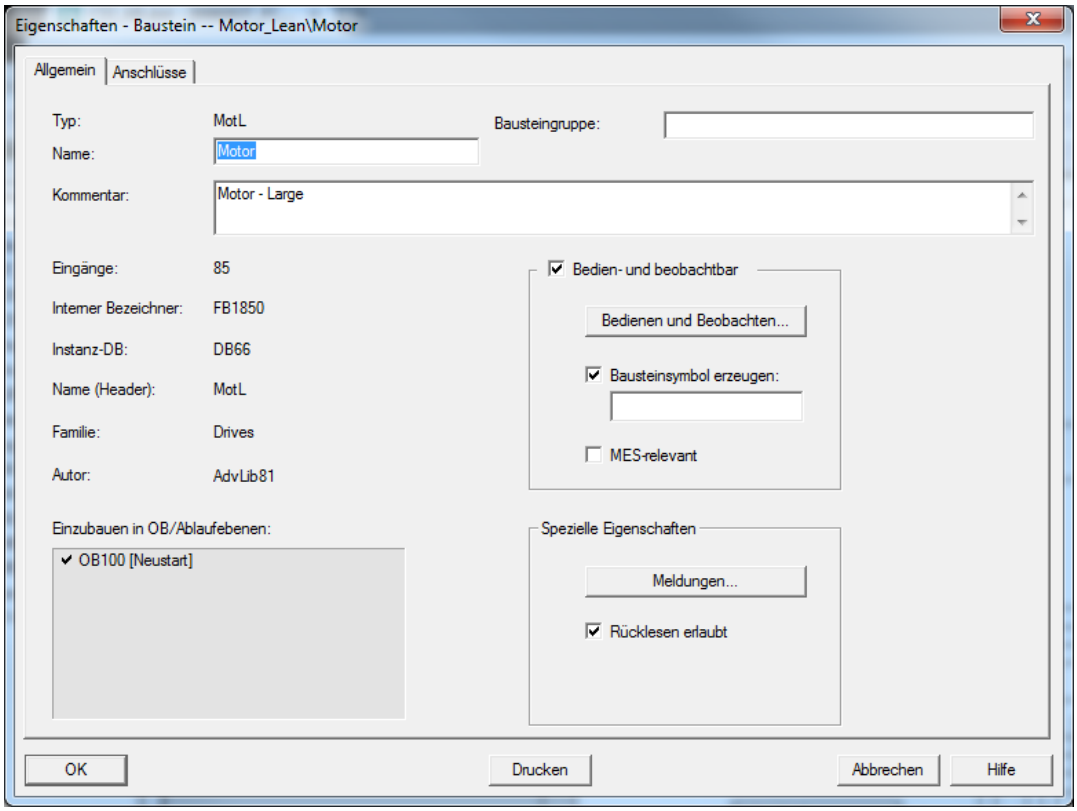


Hinweis: Mit den Registern in der unteren Leiste können Sie zwischen den Teilplänen (maximal A-Z) wechseln. Hier existiert jedoch zuerst nur ein Teilplan A.

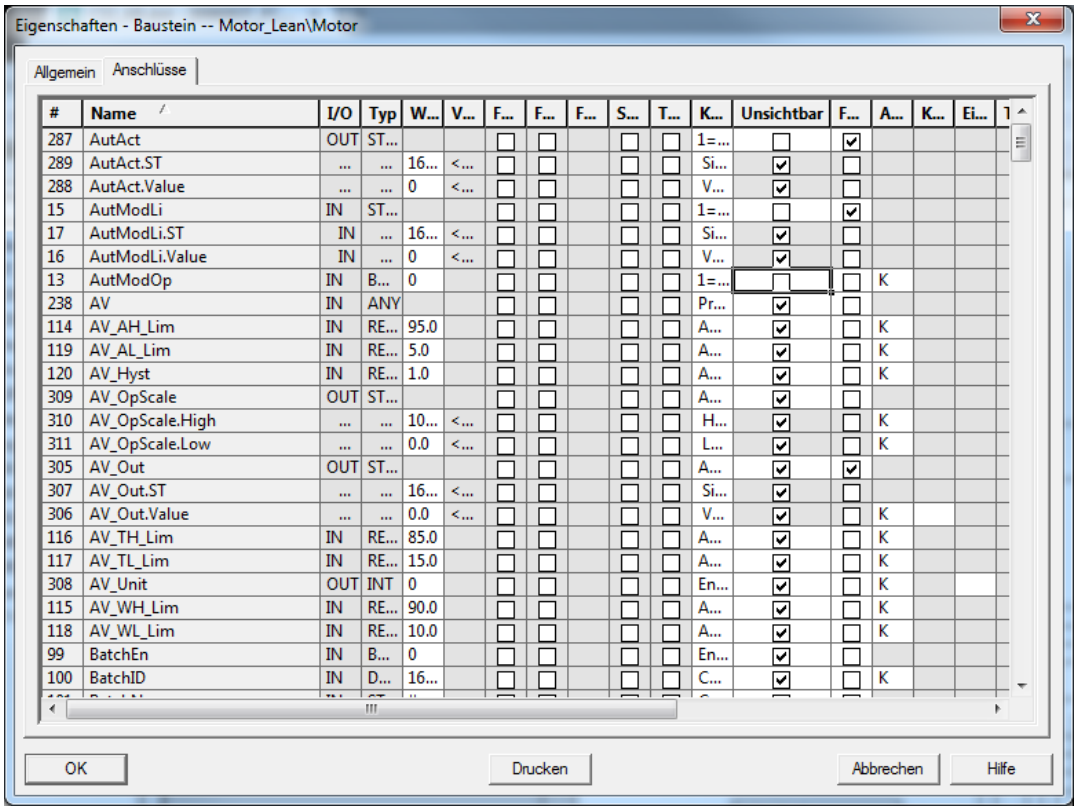
11. In dem Messstellentyp ‚Motor_Lean' wollen Sie an dem Baustein ‚MotL' eine Veränderung vornehmen. Dazu werden dessen Eigenschaften über Objekteigenschaften geöffnet.
(→ MotL → Rechtsklick → Objekteigenschaften)



12. Die allgemeinen Eigenschaften wie Bedien- und Beobachtbarkeit wollen Sie nicht verändern, deshalb wechseln Sie zu den Anschlüssen.
(→ Anschlüsse)



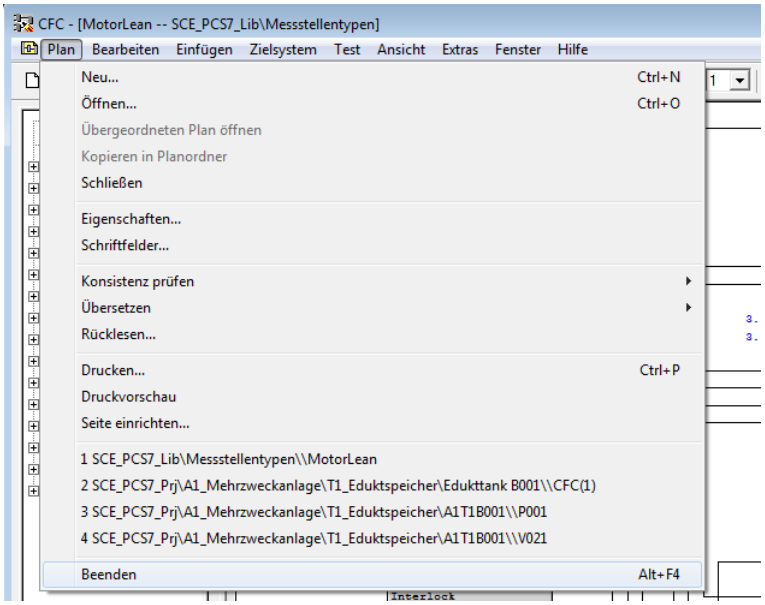
13. Die Anschlüsse werden in einer Tabelle zusammen mit einer Vielzahl an einstellbaren Eigenschaften dargestellt. Die wichtigsten Eigenschaften werden Sie im Folgenden noch kennen lernen. In dem Baustein ‚MotL' wollen Sie hier lediglich bei ‚AutModOP' die Unsichtbarkeit löschen. Dadurch wird dieser Anschluss in dem Blatt dargestellt. Mit ‚OK' verlassen Sie die Eigenschaften.
    (→ AutModOP → ☐ → OK)



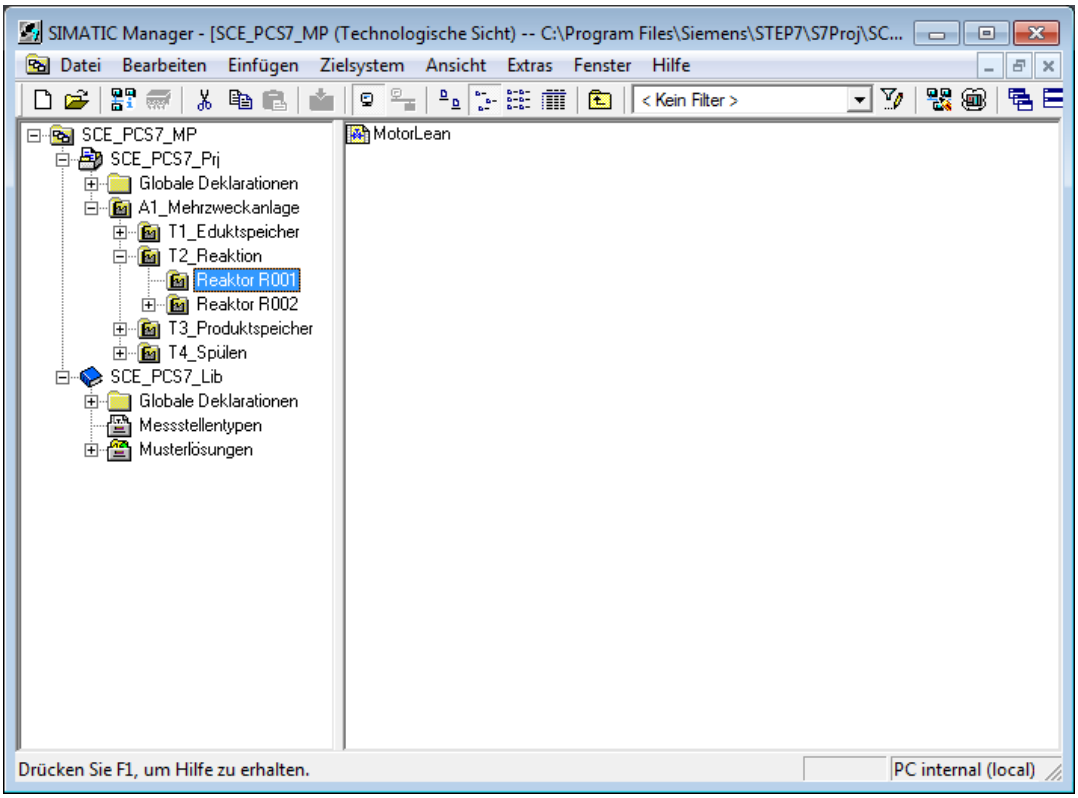
#	Name	I/O	Typ	W...	V...	F...	F...	F...	S...	T...	K...	Unsichtbar	F...	A...	K...	Ei...
287	AutAct	OUT	ST...			☐	☐		☐	☐	1=...	☐	☑			
289	AutAct.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐			
288	AutAct.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐			
15	AutModLi	IN	ST...			☐	☐		☐	☐	1=...	☐	☑			
17	AutModLi.ST	IN	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐			
16	AutModLi.Value	IN	...	0	<...	☐	☐		☐	☐	V...	☑	☐			
13	AutModOp	IN	B...	0		☐	☐		☐	☐	1=...	☐	☐	K		
238	AV	IN	ANY			☐	☐		☐	☐	Pr...	☑	☐			
114	AV_AH_Lim	IN	RE...	95.0		☐	☐		☐	☐	A...	☑	☐	K		
119	AV_AL_Lim	IN	RE...	5.0		☐	☐		☐	☐	A...	☑	☐	K		
120	AV_Hyst	IN	RE...	1.0		☐	☐		☐	☐	A...	☑	☐	K		
309	AV_OpScale	OUT	ST...			☐	☐		☐	☐	A...	☑	☐			
310	AV_OpScale.High	...	...	10...	<...	☐	☐		☐	☐	H...	☑	☐	K		
311	AV_OpScale.Low	...	...	0.0	<...	☐	☐		☐	☐	L...	☑	☐	K		
305	AV_Out	OUT	ST...			☐	☐		☐	☐	A...	☑	☑			
307	AV_Out.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐			
306	AV_Out.Value	...	...	0.0	<...	☐	☐		☐	☐	V...	☑	☐	K		
116	AV_TH_Lim	IN	RE...	85.0		☐	☐		☐	☐	A...	☑	☐	K		
117	AV_TL_Lim	IN	RE...	15.0		☐	☐		☐	☐	A...	☑	☐	K		
308	AV_Unit	OUT	INT	0		☐	☐		☐	☐	En...	☑	☐	K		
115	AV_WH_Lim	IN	RE...	90.0		☐	☐		☐	☐	A...	☑	☐	K		
118	AV_WL_Lim	IN	RE...	10.0		☐	☐		☐	☐	A...	☑	☐	K		
99	BatchEn	IN	B...	0		☐	☐		☐	☐	En...	☑	☐			
100	BatchID	IN	D...	16...		☐	☐		☐	☐	C...	☑	☐	K		

Hinweis: Durch einen Klick mit der Maus auf die Überschrift einer Eigenschaft kann die Darstellung nach dieser Spalte alphabetisch sortiert werden.

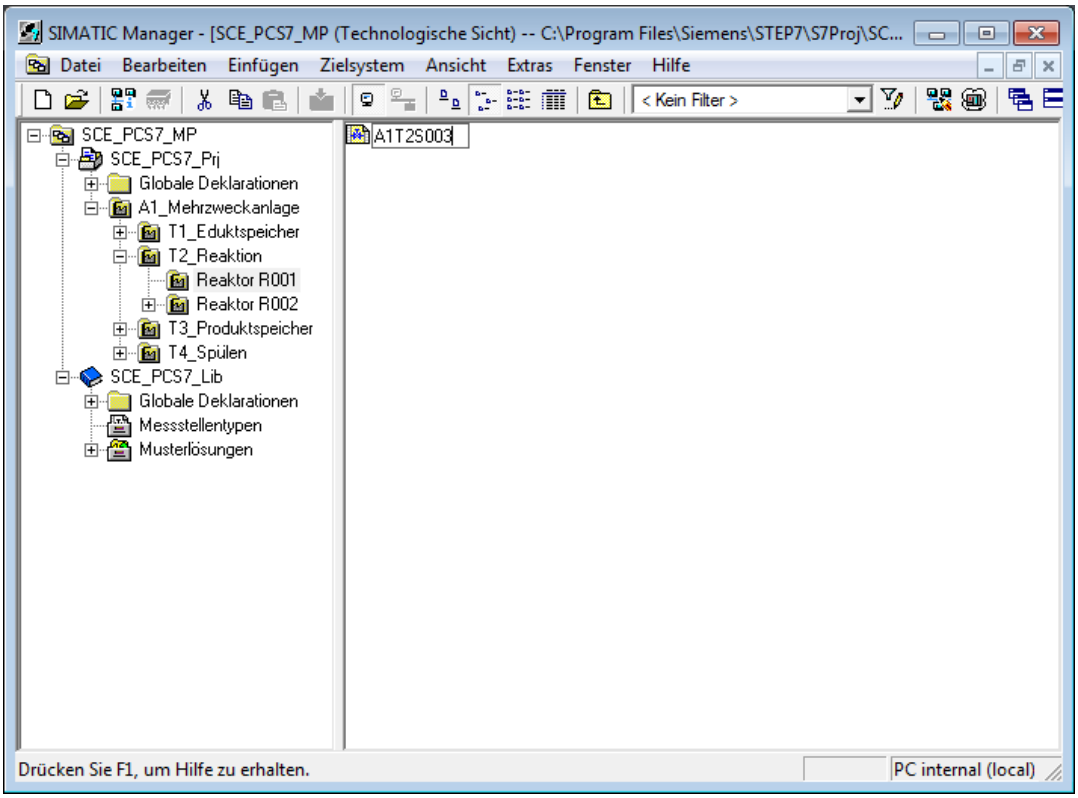
14. Der Messstellentyp kann nun einfach geschlossen werden. Das Speichern erfolgt automatisch bei jeder Änderung.
    (→ Plan → Beenden)



15. Zur Verwendung des Messstellentyps ‚Motor_Lean' wird dieser direkt per Drag&Drop in den Planordner ‚Reaktor R001' geschoben. (→ Motor_Lean → Reaktor R001')

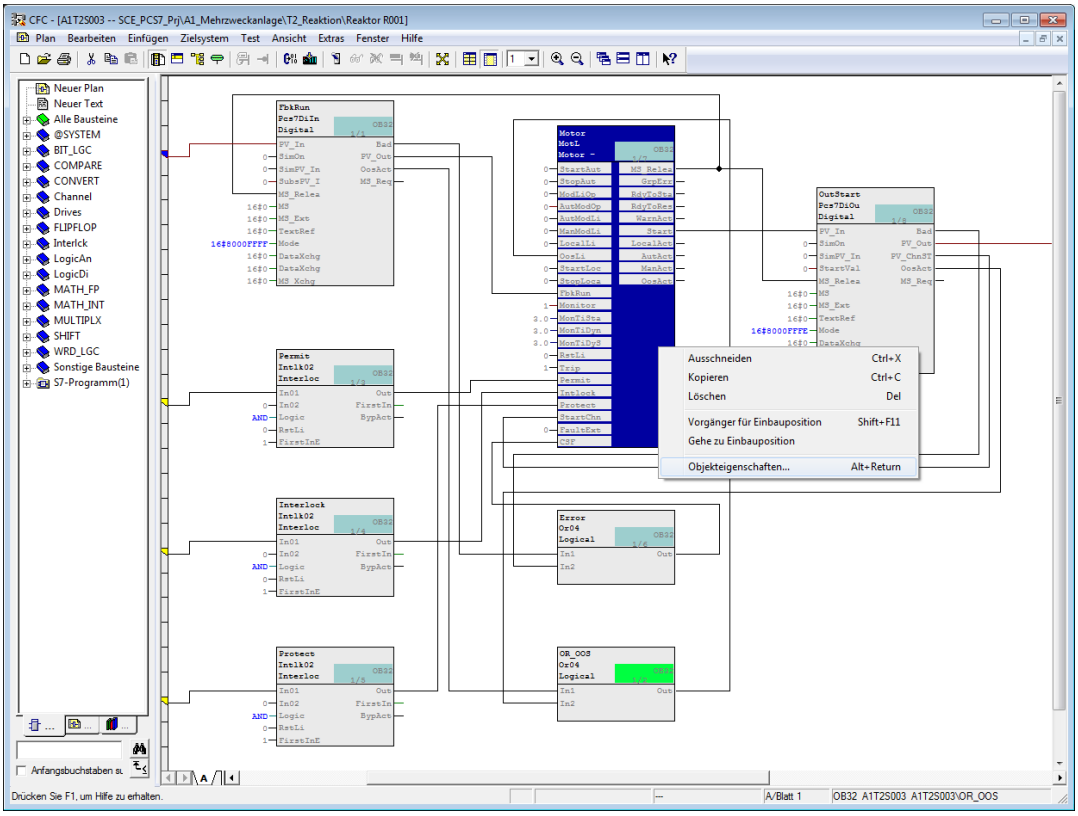


16. Da dieser Plan zur Ansteuerung der Pumpe A1T2S003 dienen soll, wird er nun in A1T2S003 umbenannt und anschließend per Doppelklick geöffnet. (→ Reaktor R001 → A1T2S003 → A1T2S003)



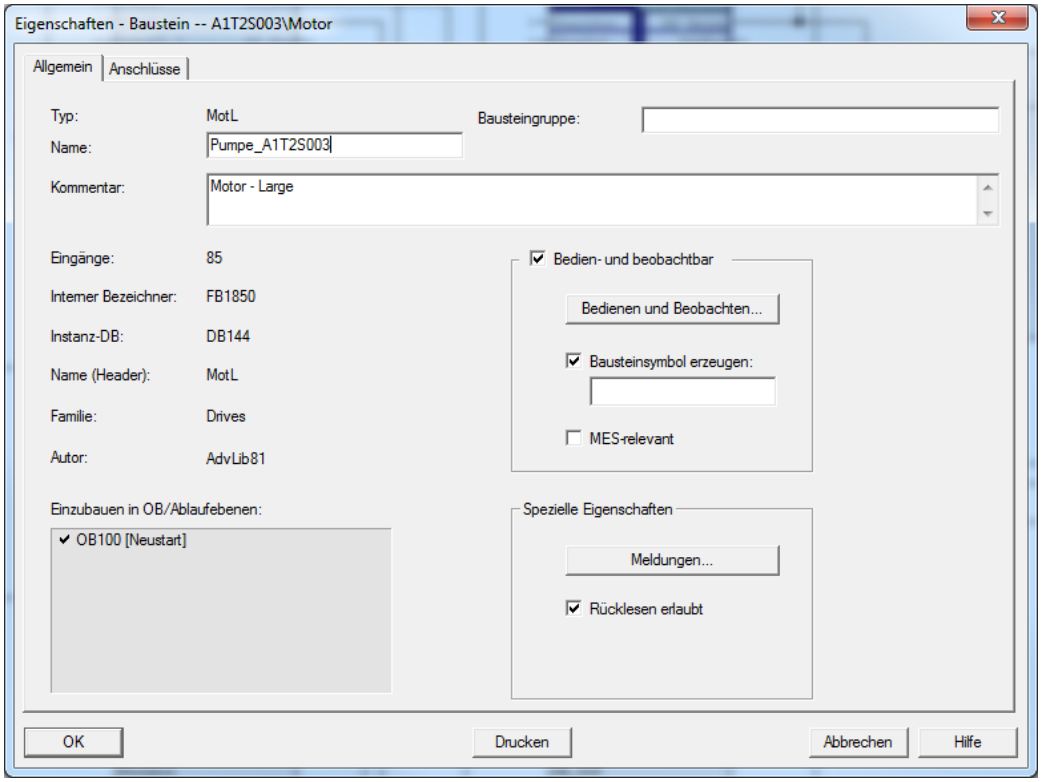
17. Per Rechtsklick werden nun die Eigenschaften des Bausteins ‚MotL' geöffnet.

    (→ Motor_Lean → Objekteigenschaften)



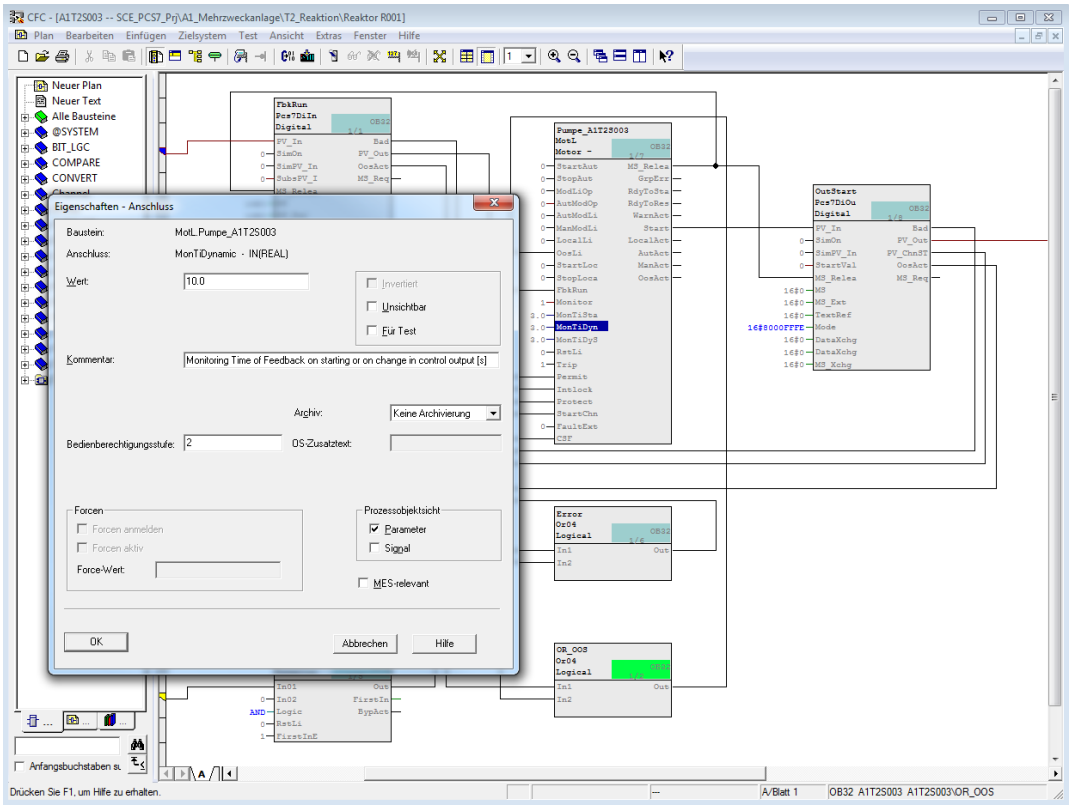
18. Bei den allgemeinen Eigenschaften wird der Name des Bausteins geändert.

    (→ Pumpe_A1T2S003 → OK)



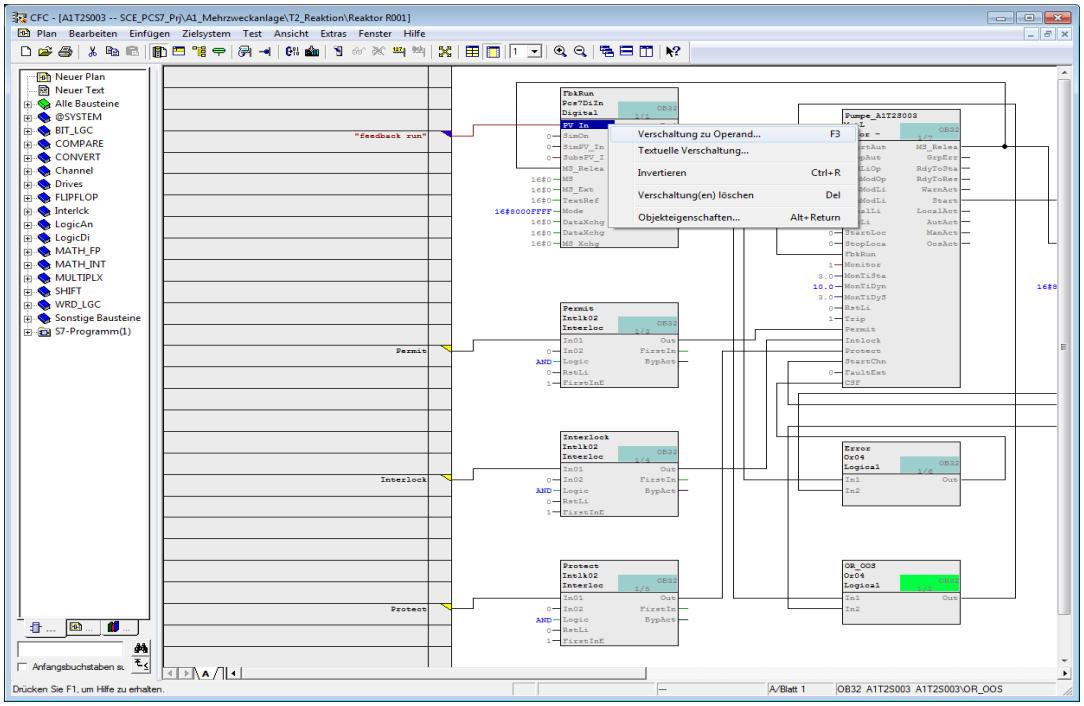
19. Nun wollen Sie die Zeit für die Rückmeldungsüberwachung nach erfolgreicher Bedienung des Motorbausteins auf 10.0 Sekunden ändern. Dafür wird der Eigenschaftsdialog für den Eingang ‚MonTiDyn' mit einem Doppelklick geöffnet und bei Wert 10.0 eingetragen. Mit ‚OK' verlassen Sie den Dialog wieder.

    (→ MonTiDyn → 10.0 → OK)



20. Nun erfolgt noch die Verschaltung der Rückmeldung zum Eingangsoperanden. Dies geschieht, indem der Eingang ‚PV_In' am Baustein ‚PCS7DiIn' mit der rechten Maustaste angeklickt und anschließend Verschaltung zum Operand gewählt wird.

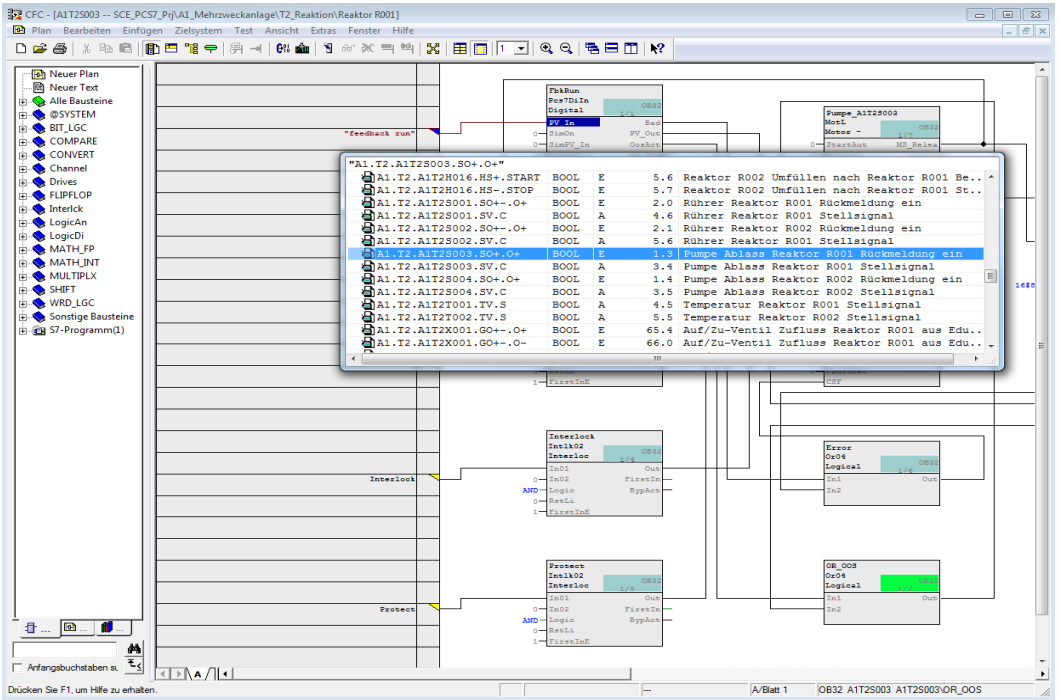
    (→ PV-In → Verschaltung zum Operand)



Hinweis: Der PCS7DiIn Baustein ist ebenso wie der PCS7DiOu ein Treiberbaustein für die Kopplung zur Peripherie der SPS. Wird an einem dieser Bausteine der Wert ‚PV_In' mit einem Operanden verschaltet, der auch in der Hardwarekonfiguration projektiert ist, so wird später beim Übersetzungslauf automatisch der Eingang MODE mit Daten versorgt.

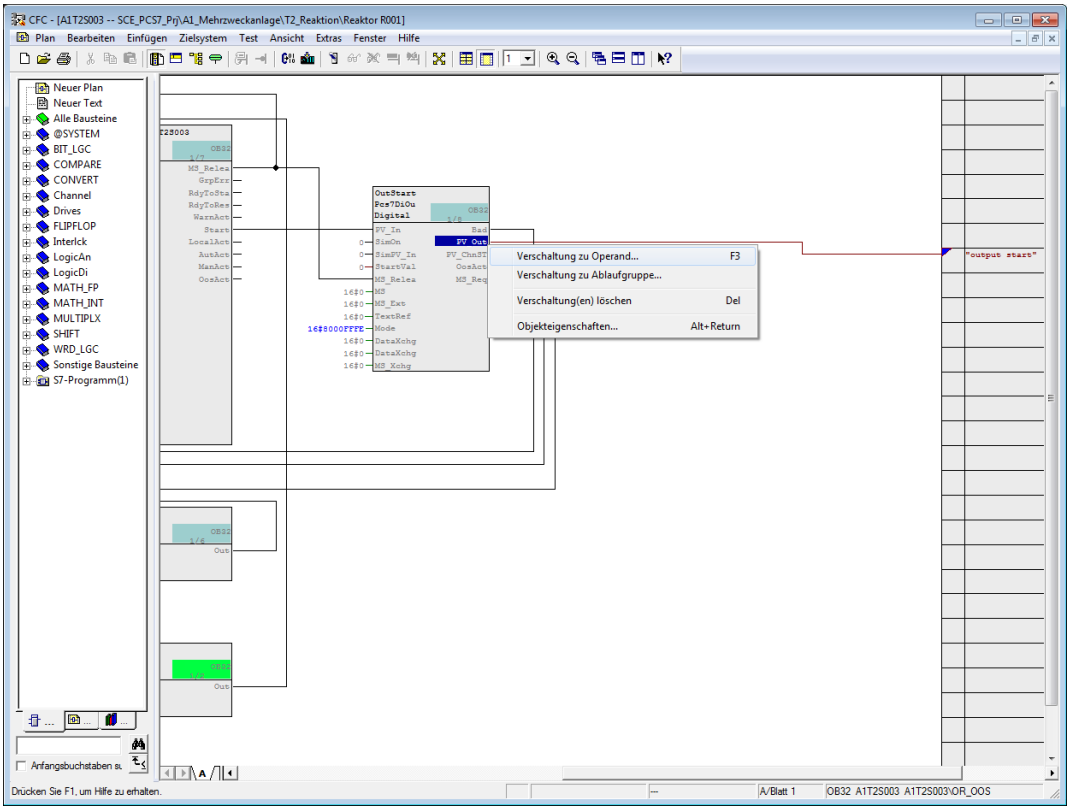
Damit das geschieht, muss später beim Übersetzungslauf ‚Baugruppentreiber erzeugen' angewählt werden.

21. Der Operand, welcher rückmeldet ob die Pumpe läuft, kann komfortabel direkt aus der Symboltabelle ausgewählt werden.
    (→ A1.T2.A1T2S003.S0+.0+)

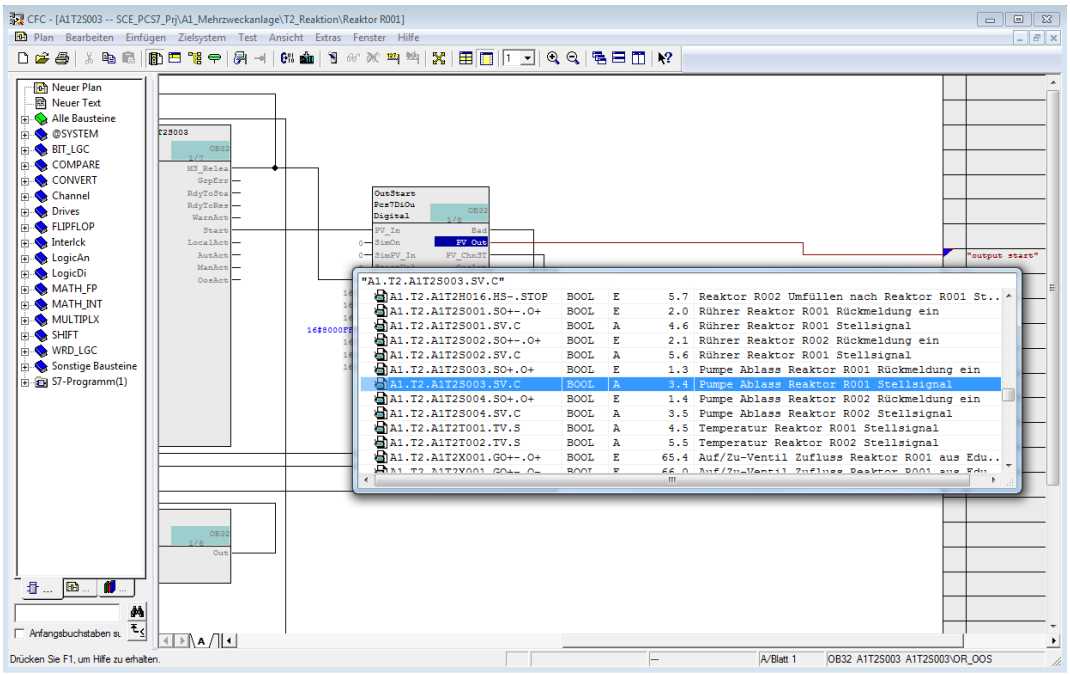


22. Nun erfolgt noch die Verschaltung der Ansteuerung des Ausgangsoperanden. Dies geschieht, indem der Ausgang ‚PV_Out' am Baustein ‚PCS7DiOu' mit der rechten Maustaste angeklickt und anschließend Verschaltung zum Operand gewählt wird.

    (→ PV_Out → Verschaltung zum Operand)

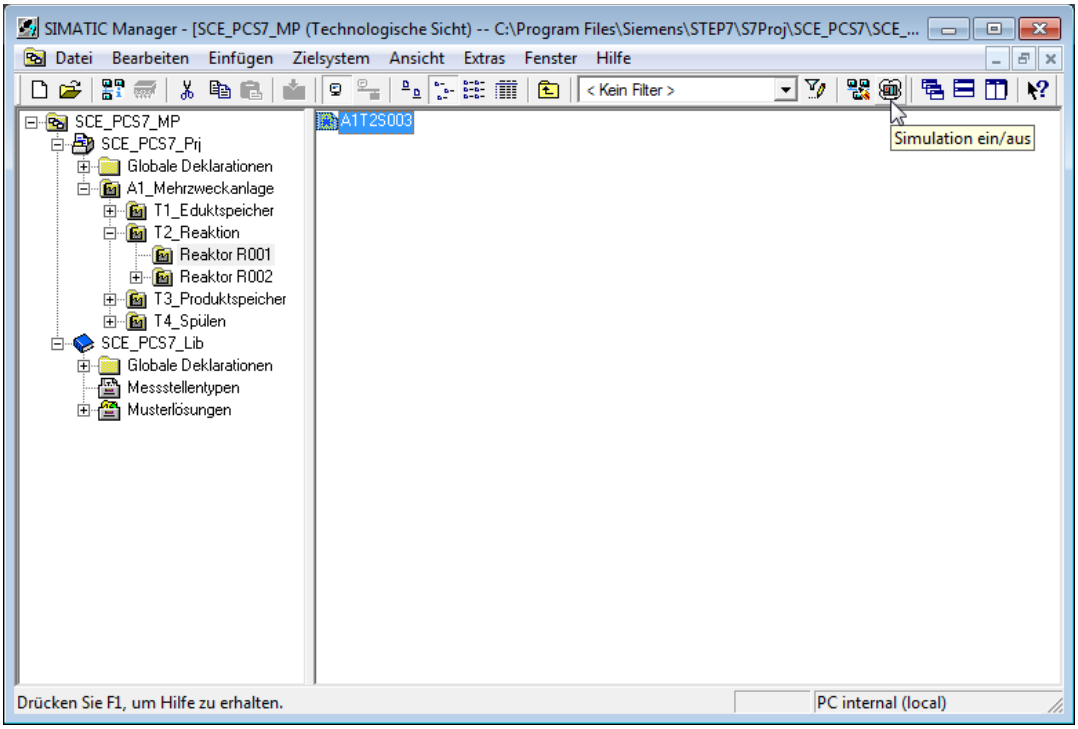


23. Der Operand zur Ansteuerung der Pumpe kann nun wieder komfortabel direkt aus der Symboltabelle ausgewählt werden.
    (→ A1.T2.A1T2S003.SV.C)

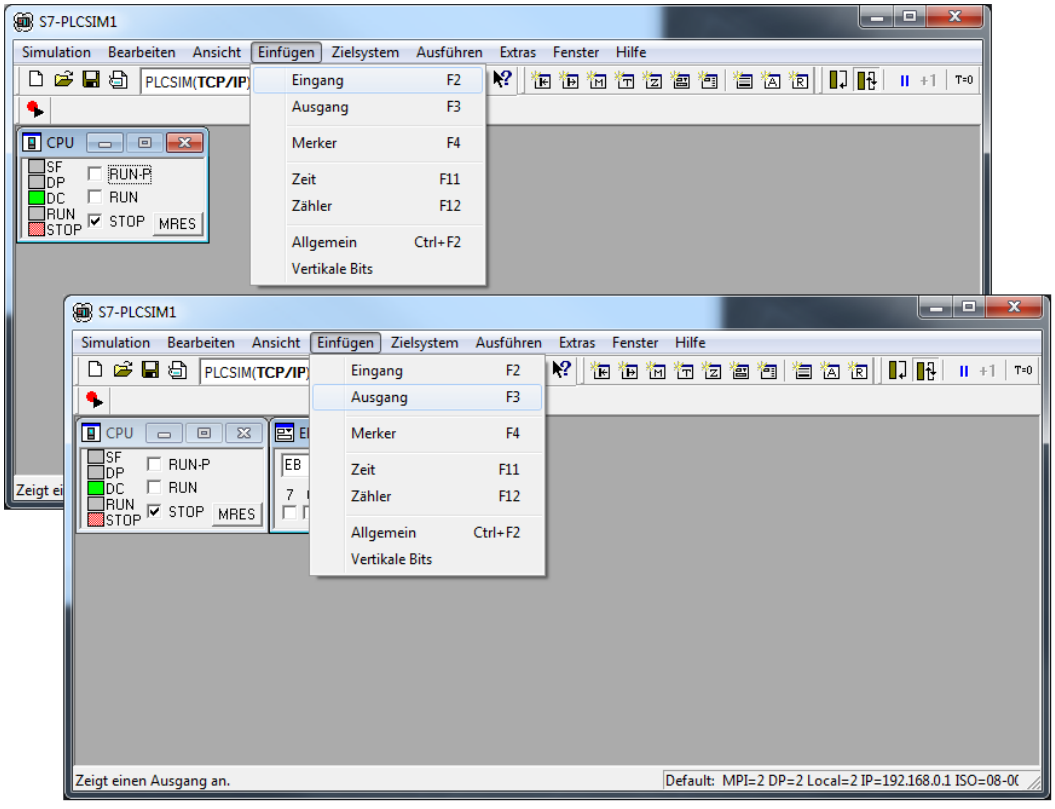


Hinweis: Der Platzhalter am Ausgang von Pcs7DiOu ‚output start' sollte gelöscht werden, sonst gibt es beim Übersetzen eine Warnung!

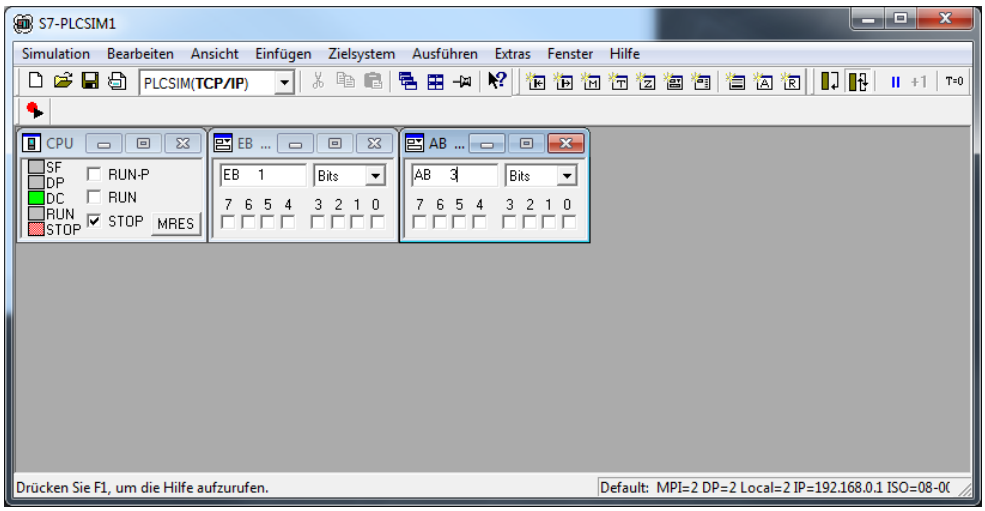
24. Bevor das Programm für den Pumpenmotor übersetzt und geladen werden kann, muss aus dem SIMATIC-Manager heraus die SPS-Simulation ***S7-PLCSIM*** gestartet werden.
    (→ )



25. Die SPS-Simulation verhält sich wie eine richtige SIMATIC S7 CPU. Jedoch müssen Ein- und Ausgänge zuerst eingefügt werden, bevor diese beobachtet und geschaltet werden können.
    (→ Einfügen → Eingang → Einfügen → Ausgang)

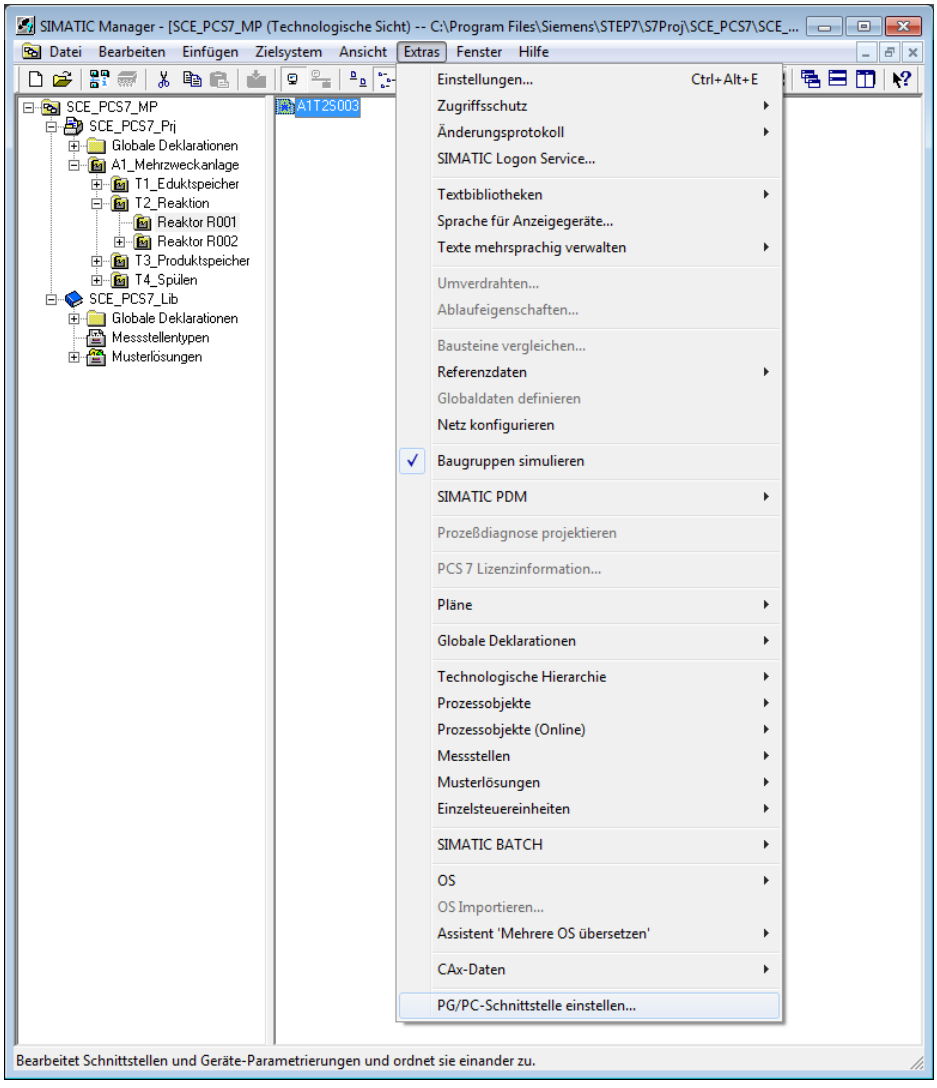


26. Nun müssen noch die richtigen Byte-Adressen eingetragen werden. (→ EB1 → AB3)

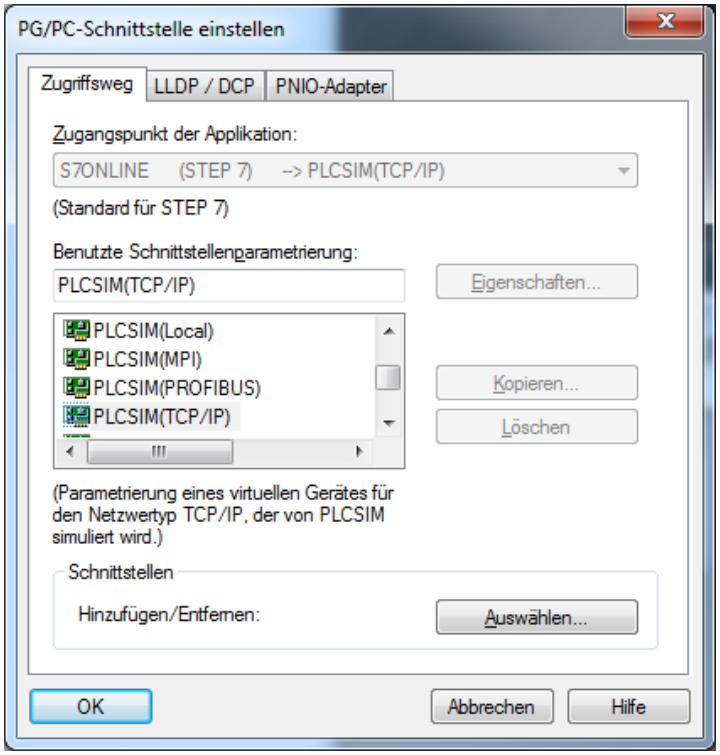


27. Damit aus dem SIMATIC Manager auch über die richtige Schnittstelle in ***S7-PLCSIM*** geladen werden kann, wird anschließend noch die PG/PC-Schnittstelle richtig eingestellt.

    (→ SIMATIC Manager → Extras → PG/PC-Schnittstelle einstellen)

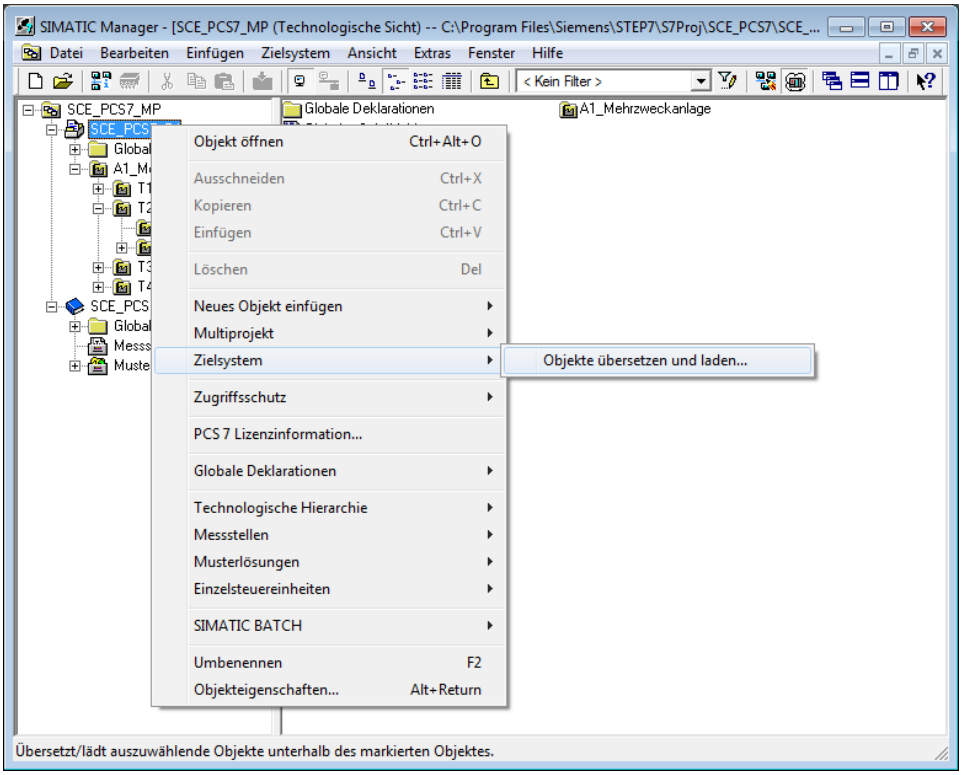


28. Als Schnittstelle wird hier PLCSIM(TCP/IP) eingestellt. (→ PLCSIM(TCP/IP) → OK)

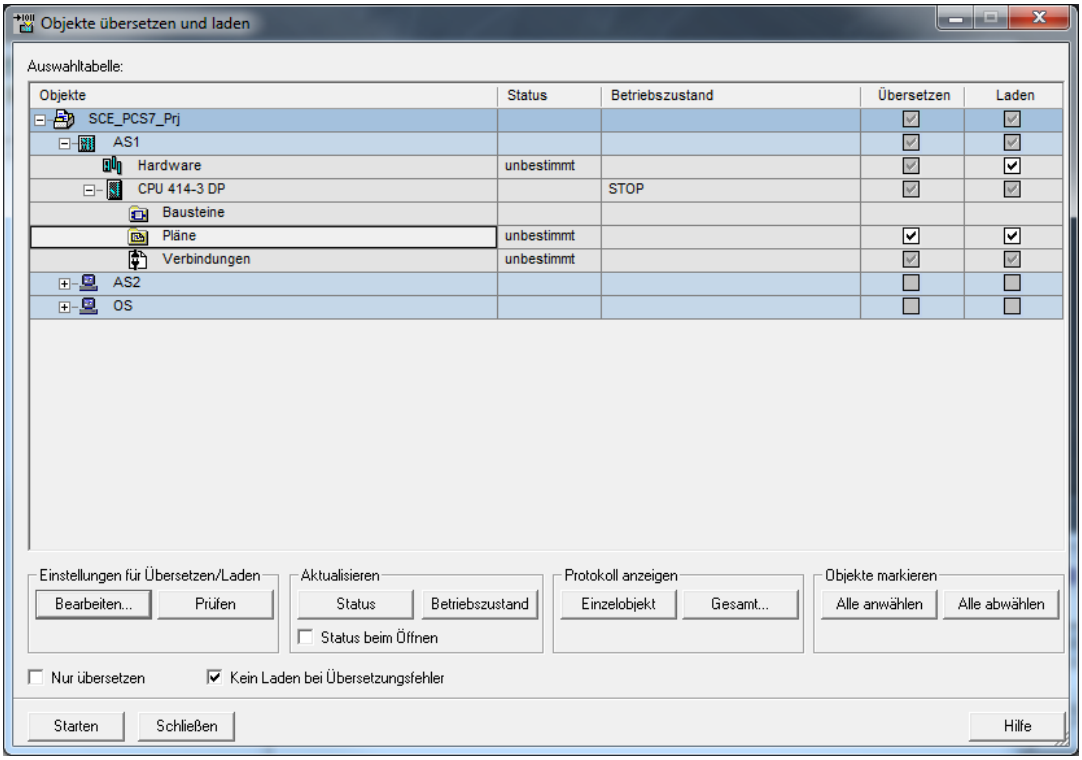


29. Nun kann der Projektordner markiert und mit dem Übersetzen und Laden der Objekte begonnen werden.

    (→ Technologische Sicht → SCE_PCS7_Prj → Zielsystem → Objekte übersetzen und laden)

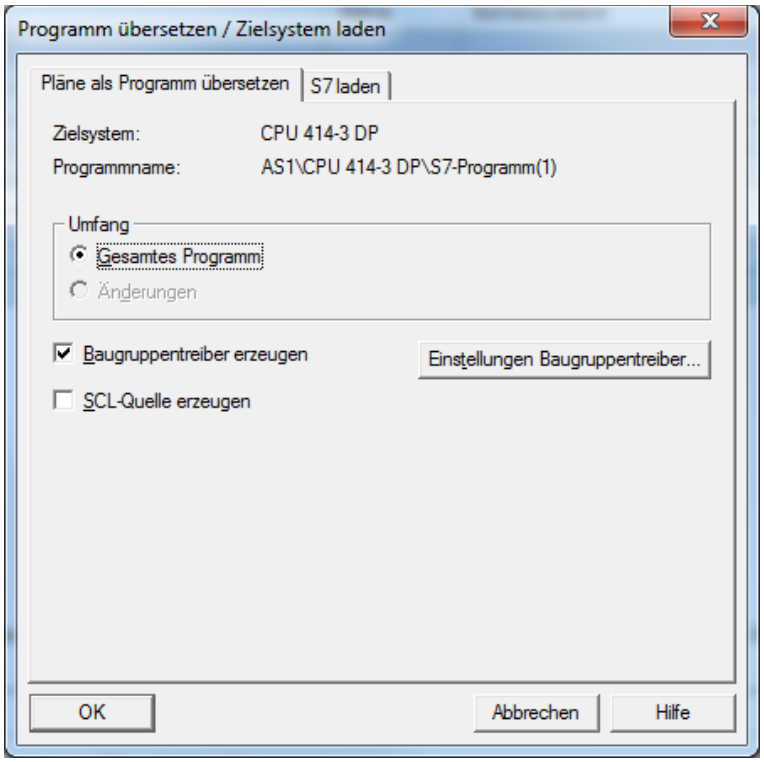


30. In der folgenden Auswahl wird nun für Hardware und die Pläne der AS1 ‚Übersetzen und Laden' angewählt. Darauffolgend wird der Ordner ‚Pläne' markiert und dessen Einstellungen über ‚Bearbeiten' kontrolliert.
    (→ ☑ → ☑ → ☑ → Pläne → Bearbeiten)



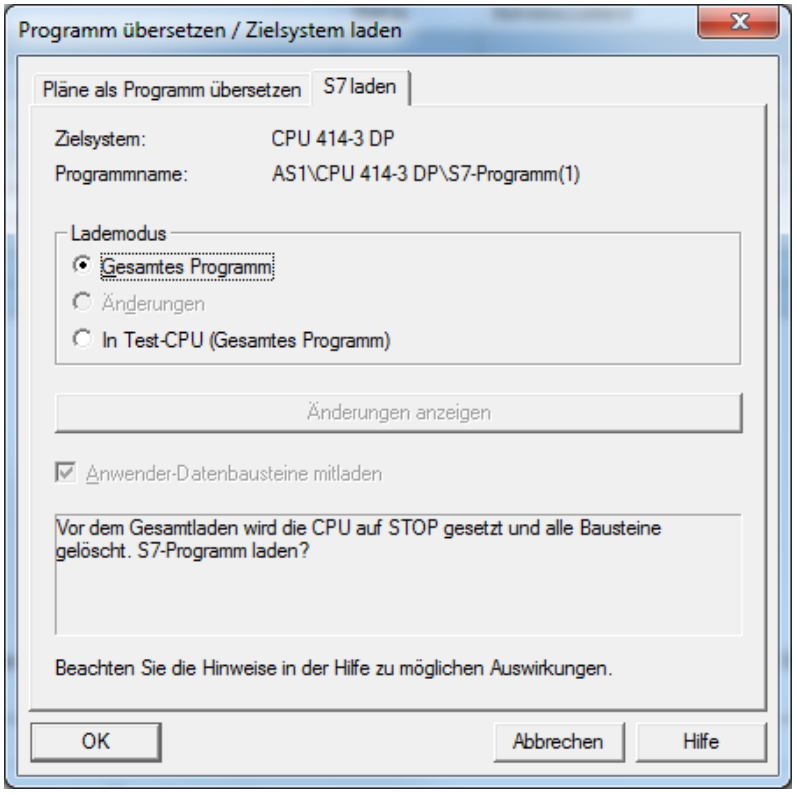
31. Beim Übersetzen der Pläne ist es wichtig das gesamte Programm zu übersetzen und die Baugruppentreiber erzeugen zu lassen.

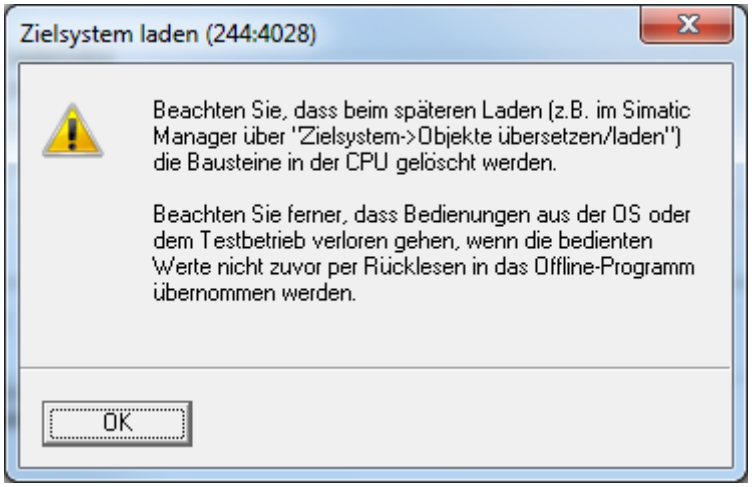
    (→ Gesamtes Programm → Baugruppentreiber erzeugen → S7 laden)



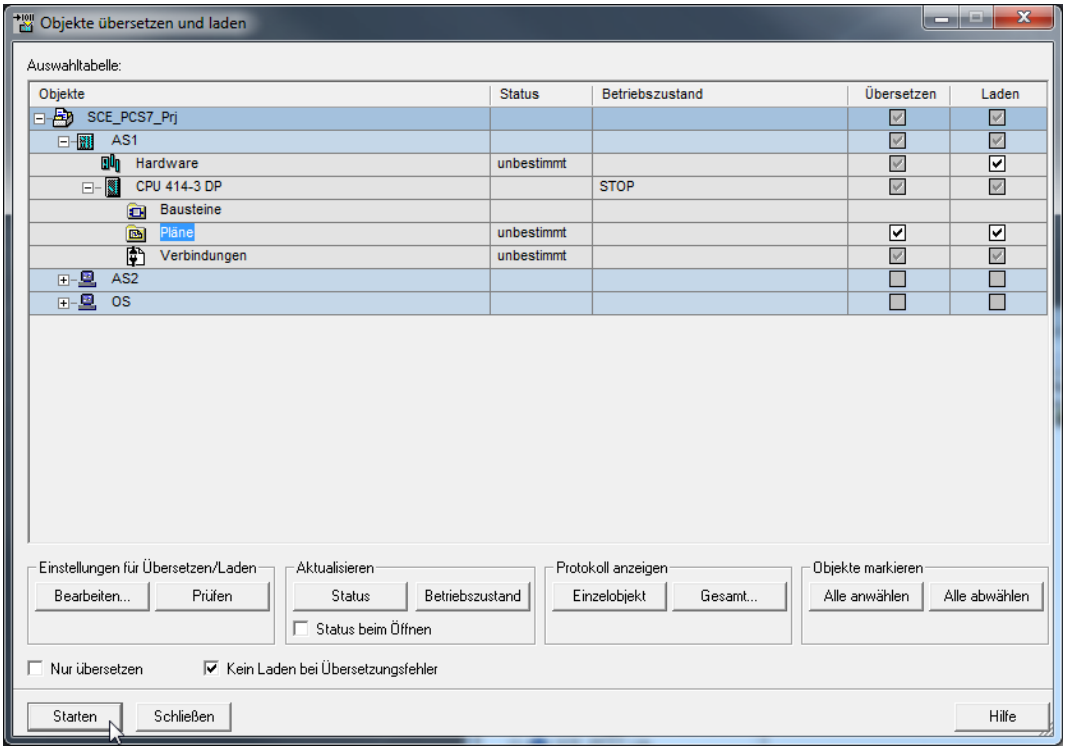
32. Beim Laden der Pläne ist es ebenfalls wichtig das gesamte Programm zu laden.

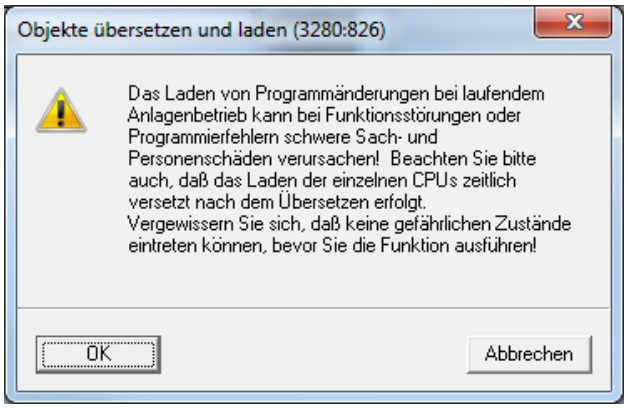
(→ Gesamtes Programm → OK → OK)

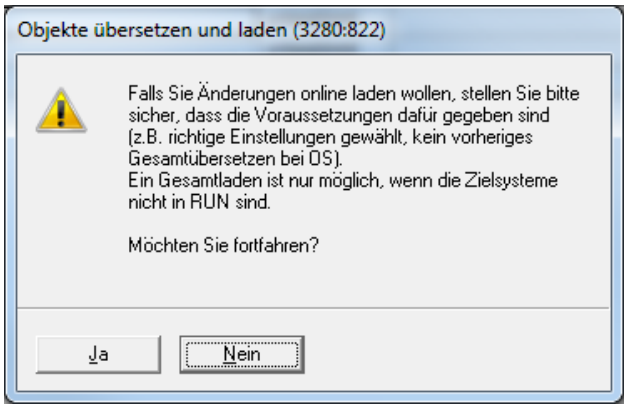




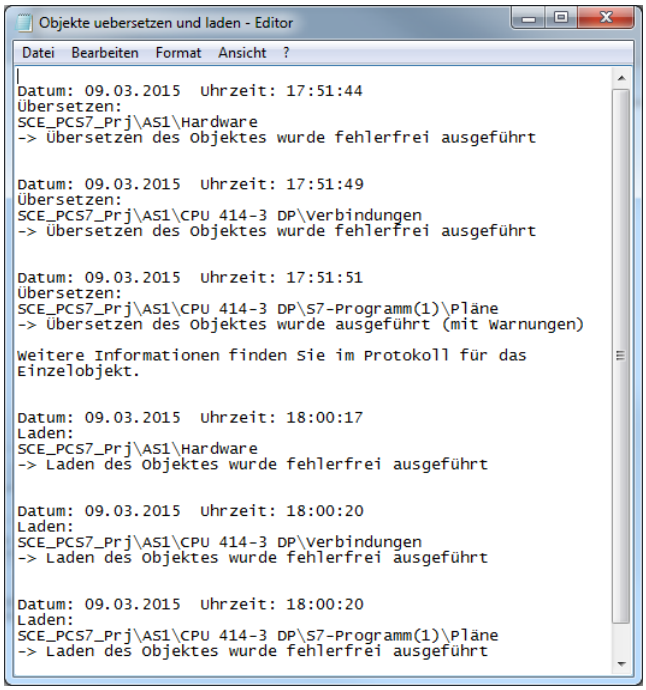
33. Schließlich wird ‚Übersetzen und Laden' gestartet. Die Warnungen und Hinweise zur Anlagensicherheit sollten sorgfältig gelesen werden. Die CPU muss vor ‚Übersetzen und Laden' auf ‚STOP' geschaltet werden.
    (→ Starten → OK → Ja)



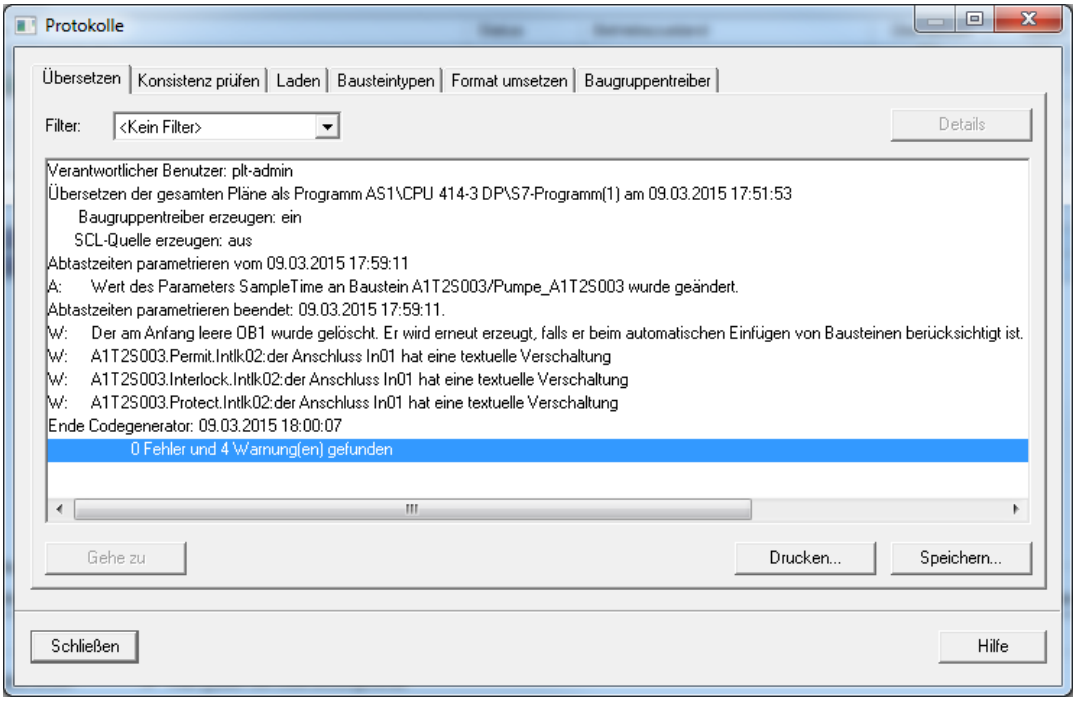




34. Am Ende werden Fehler und Warnungen in einem Protokoll angezeigt. Sie schließen das Fenster. (→ )

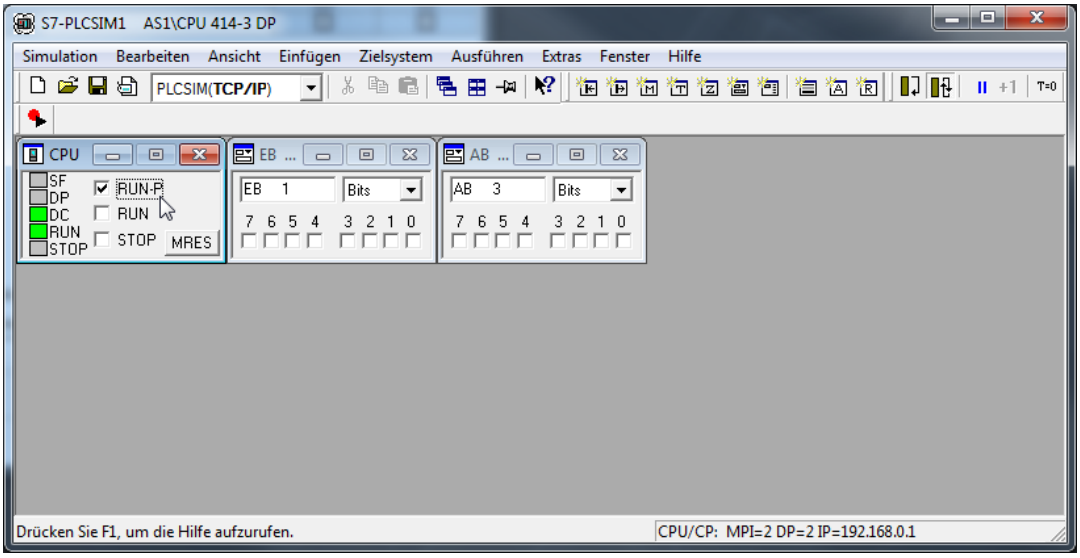


35. Wollen Sie Details zu dem Protokoll ansehen, so müssen Sie bei Protokoll anzeigen auf Einzelobjekt klicken.
(→ Einzelobjekt → Schließen → Schließen)

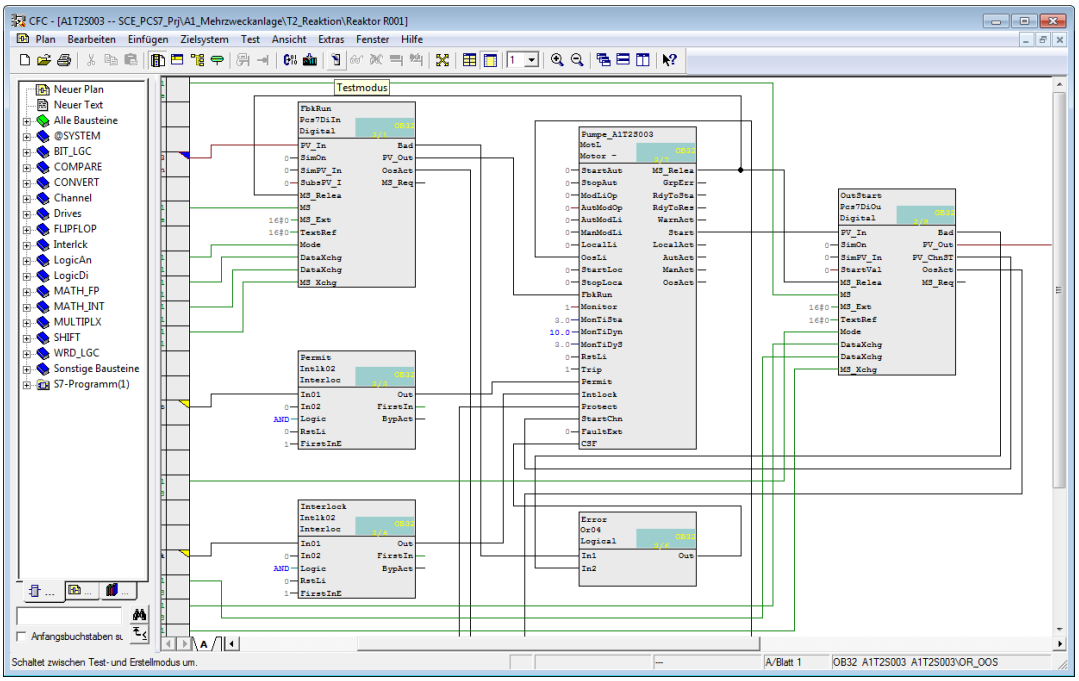


Hinweis: Hier erscheinen vier Warnungen: Löschen des leeren OB 1 und das Vorhandensein von textuellen Verschaltungen. Diese entstehen durch die nicht verknüpften Anschlüsse des Templates. In Kapitel P01-05 werden diese Anschlüsse verbunden.

36. Zum Testen des Programms wird nun die CPU in ***S7-PLCSIM*** auf ‚RUN-P' geschaltet. (→ S7-PLCSIM → RUN-P)

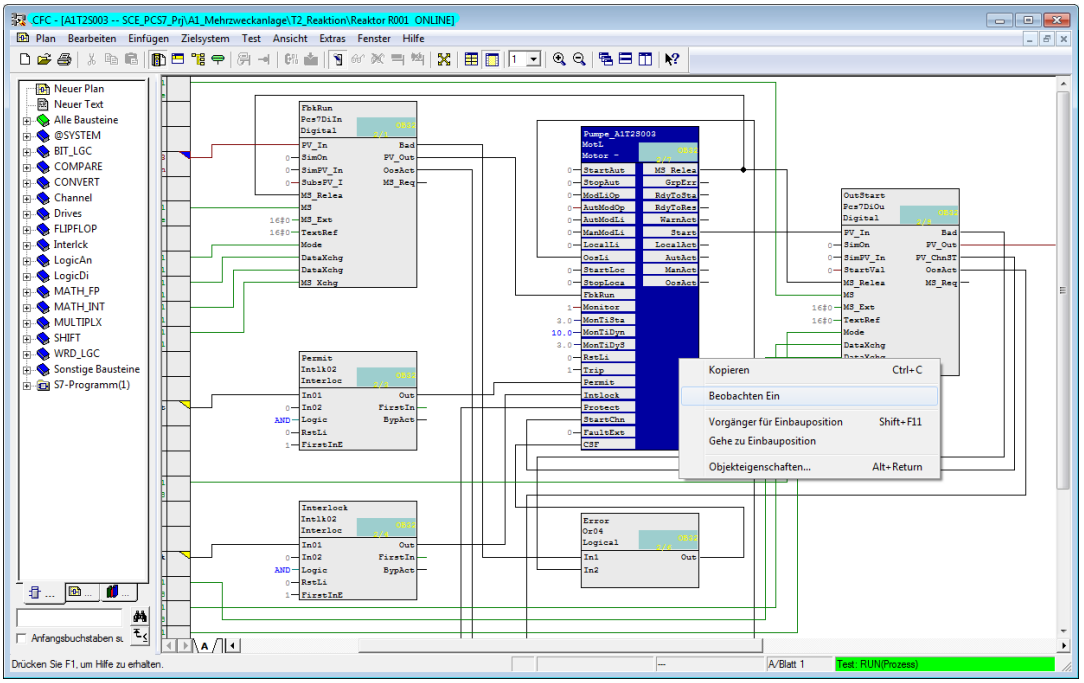


37. Bevor im CFC die einzelnen Bausteine beobachtet werden können, muss der Plan zuerst einmal in den Testmodus geschaltet werden. (→ CFC → )



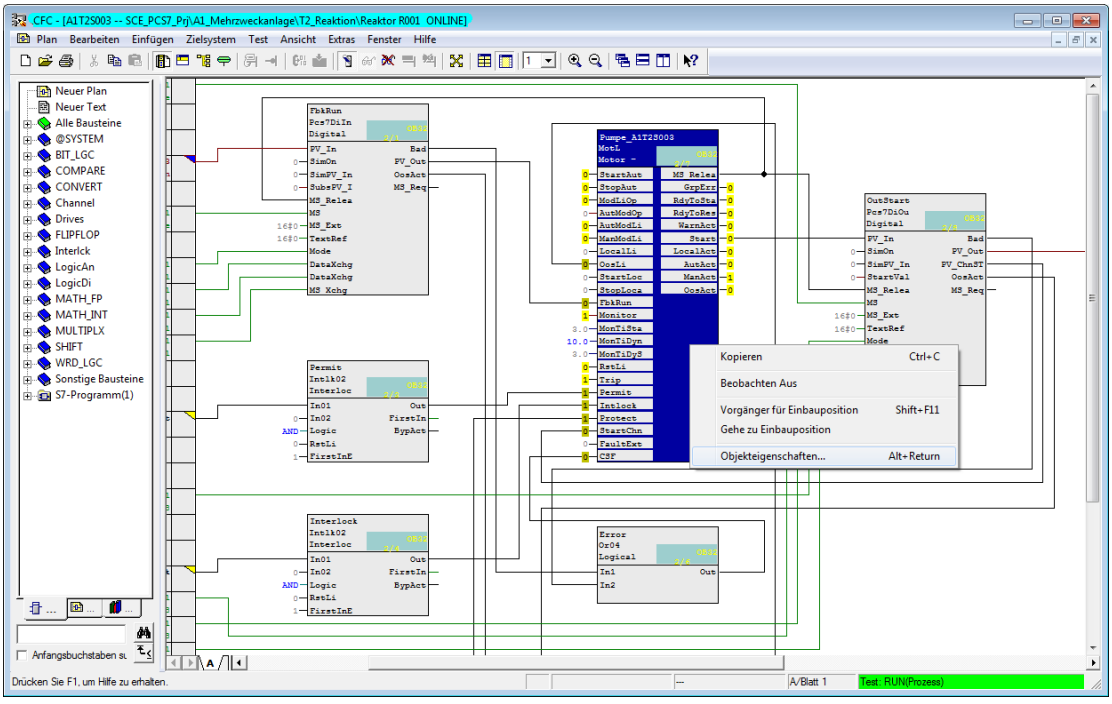
38. Die Bausteine, welche beobachtet werden sollen müssen nun explizit zum Beobachten freigeschaltet werden. Dasselbe gilt anschließend für einzelne Anschlüsse des Bausteins.

    (→ Pumpe_A1T2S003 → Beobachten Ein)

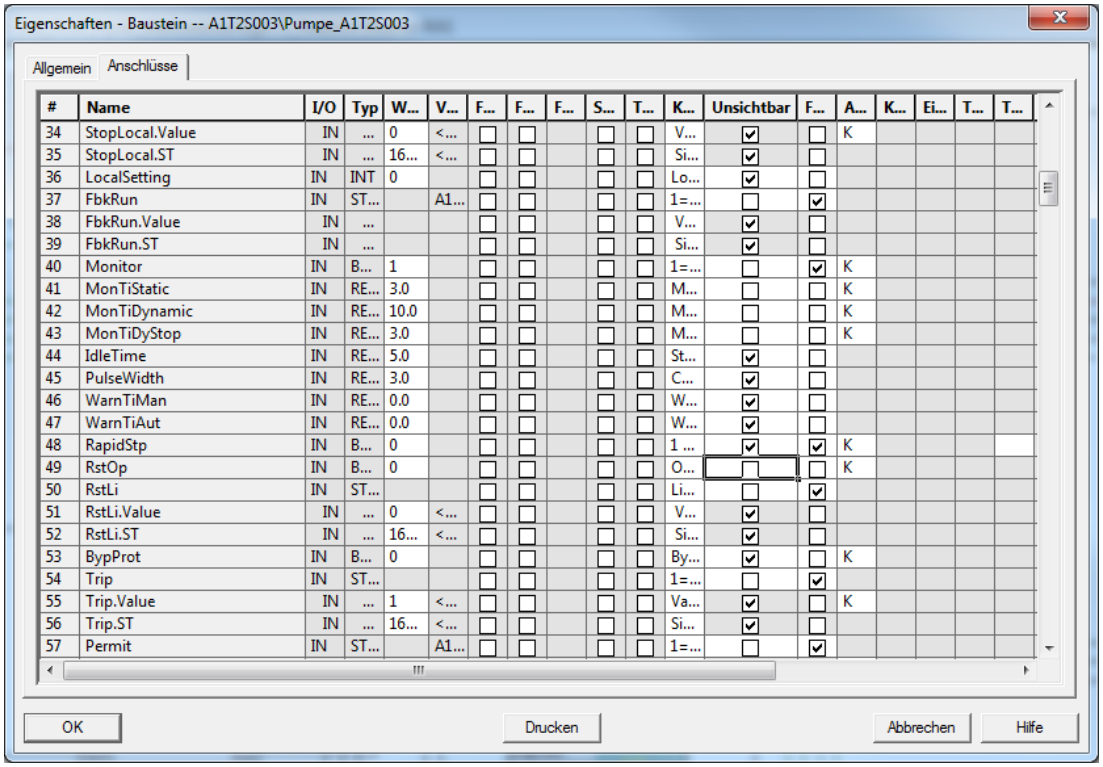


39. Für das weitere Vorgehen ist es notwendig, dass die Anschlüsse für die Automatiksteuerung ‚StartAut' und ‚StopAut' des Bausteins ‚MotL' sichtbar sind. Für den Fehlerfall sollten auch ‚RstOp' und ‚MonDynErr' sichtbar geschaltet werden.

    (→ Objekteigenschaften)

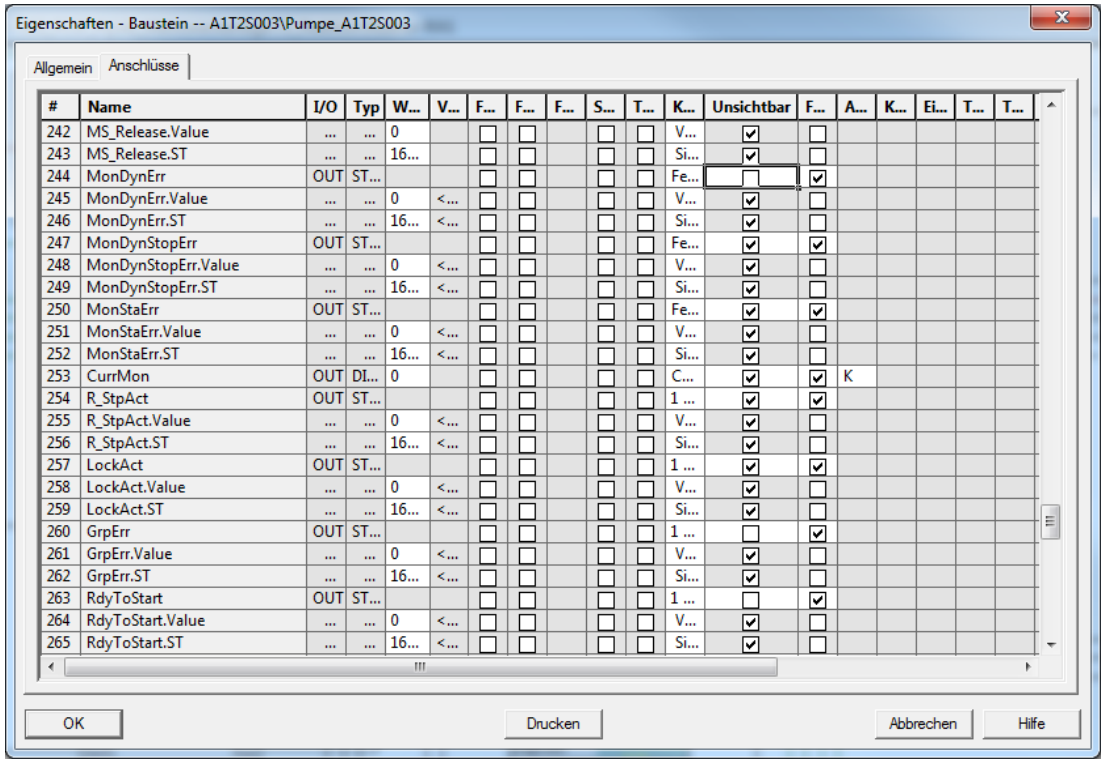


(→ RstOp)



#	Name	I/O	Typ	W...	V...	F...	F...	F...	S...	T...	K...	Unsichtbar	F...	A...	K...	Ei...	T...	T...
34	StopLocal.Value	IN	...	0	<...	☐	☐		☐	☐	V...	☑	☐	K				
35	StopLocal.ST	IN	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
36	LocalSetting	IN	INT	0		☐	☐		☐	☐	Lo...	☑	☐					
37	FbkRun	IN	ST...		A1...	☐	☐		☐	☐	1=...	☐	☑					
38	FbkRun.Value	IN	...			☐	☐		☐	☐	V...	☑	☐					
39	FbkRun.ST	IN	...			☐	☐		☐	☐	Si...	☑	☐					
40	Monitor	IN	B...	1		☐	☐		☐	☐	1=...	☐	☑	K				
41	MonTiStatic	IN	RE...	3.0		☐	☐		☐	☐	M...	☐	☐	K				
42	MonTiDynamic	IN	RE...	10.0		☐	☐		☐	☐	M...	☐	☐	K				
43	MonTiDyStop	IN	RE...	3.0		☐	☐		☐	☐	M...	☐	☐	K				
44	IdleTime	IN	RE...	5.0		☐	☐		☐	☐	St...	☑	☐					
45	PulseWidth	IN	RE...	3.0		☐	☐		☐	☐	C...	☑	☐					
46	WarnTiMan	IN	RE...	0.0		☐	☐		☐	☐	W...	☑	☐					
47	WarnTiAut	IN	RE...	0.0		☐	☐		☐	☐	W...	☑	☐					
48	RapidStp	IN	B...	0		☐	☐		☐	☐	1 ...	☑	☑	K				
49	RstOp	IN	B...	0		☐	☐		☐	☐	O...	☐	☐	K				
50	RstLi	IN	ST...			☐	☐		☐	☐	Li...	☐	☑					
51	RstLi.Value	IN	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
52	RstLi.ST	IN	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
53	BypProt	IN	B...	0		☐	☐		☐	☐	By...	☑	☐	K				
54	Trip	IN	ST...			☐	☐		☐	☐	1=...	☐	☑					
55	Trip.Value	IN	...	1	<...	☐	☐		☐	☐	Va...	☑	☐	K				
56	Trip.ST	IN	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
57	Permit	IN	ST...		A1...	☐	☐		☐	☐	1=...	☐	☑					

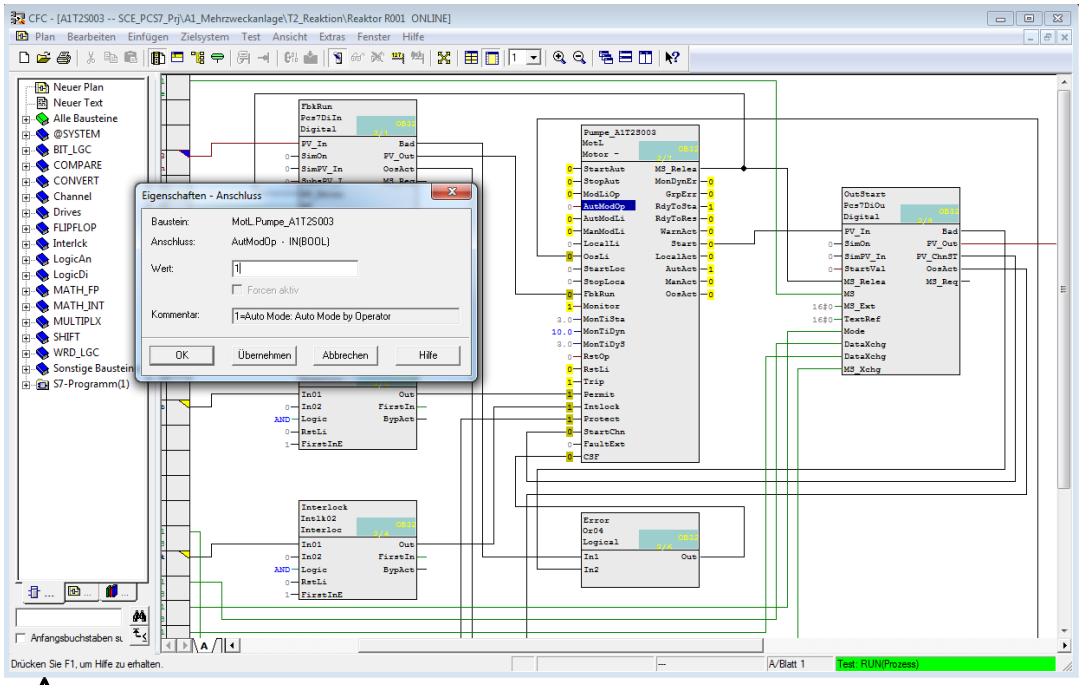
(→ MonDynErr)



#	Name	I/O	Typ	W...	V...	F...	F...	F...	S...	T...	K...	Unsichtbar	F...	A...	K...	Ei...	T...	T...
242	MS_Release.Value	...	...	0		☐	☐		☐	☐	V...	☑	☐					
243	MS_Release.ST	...	...	16...		☐	☐		☐	☐	Si...	☑	☐					
244	MonDynErr	OUT	ST...			☐	☐		☐	☐	Fe...	☐	☑					
245	MonDynErr.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
246	MonDynErr.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
247	MonDynStopErr	OUT	ST...			☐	☐		☐	☐	Fe...	☑	☑					
248	MonDynStopErr.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
249	MonDynStopErr.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
250	MonStaErr	OUT	ST...			☐	☐		☐	☐	Fe...	☑	☑					
251	MonStaErr.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
252	MonStaErr.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
253	CurrMon	OUT	DI...	0		☐	☐		☐	☐	C...	☑	☑	K				
254	R_StpAct	OUT	ST...			☐	☐		☐	☐	1 ...	☑	☑					
255	R_StpAct.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
256	R_StpAct.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
257	LockAct	OUT	ST...			☐	☐		☐	☐	1 ...	☑	☑					
258	LockAct.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
259	LockAct.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
260	GrpErr	OUT	ST...			☐	☐		☐	☐	1 ...	☐	☑					
261	GrpErr.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
262	GrpErr.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					
263	RdyToStart	OUT	ST...			☐	☐		☐	☐	1 ...	☐	☑					
264	RdyToStart.Value	...	...	0	<...	☐	☐		☐	☐	V...	☑	☐					
265	RdyToStart.ST	...	...	16...	<...	☐	☐		☐	☐	Si...	☑	☐					

40. Als Nächstes wird die Umschaltung für den Hand-/Automatikbetrieb auf Automatik durch ‚AutModOp' == 1 vorgenommen.

    (→ AutModOp → Eigenschaften → „1")



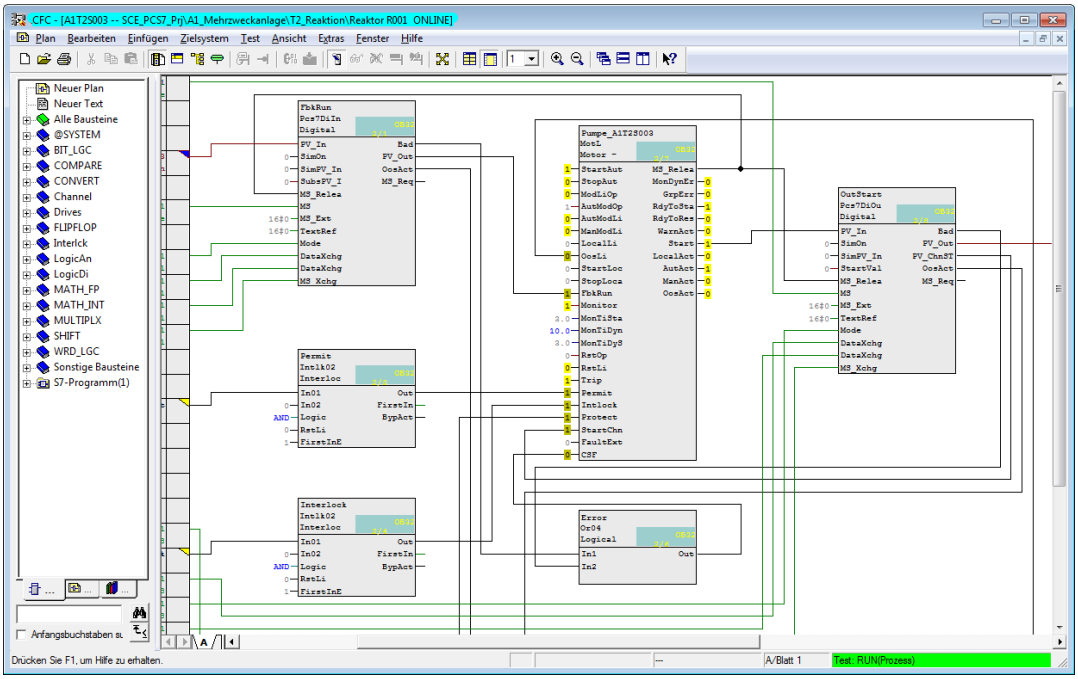
Hinweis: Beim Testen sollte nicht vergessen werden innerhalb von 10 Sekunden nach Ansteuerung des Ausgangs A 3.4 in ***S7-PLCSIM*** die Rückmeldung E 1.3 zu setzen. Wird dies vergessen, so schaltet der Baustein Pumpe_A1T2S003 ab und gibt einen Fehler aus.

Hinweis: Der Pumpenbaustein schaltet bei fehlender Rückmeldung nicht nur das Stellsignal START auf 0, sondern zeigt durch Setzen des Ausgangs ‚ModDynErr' auf 1 auch an, dass die Laufmeldung der Pumpe nicht rechtzeitig eingegangen war. Dazu müssen alle Anschlüsse sichtbar gemacht werden. Um Schäden durch wiederholte Einschaltversuche zu vermeiden, muss der Pumpenbaustein erst zurückgesetzt werden, bevor ein neuerlicher Versuch gestartet werden kann.

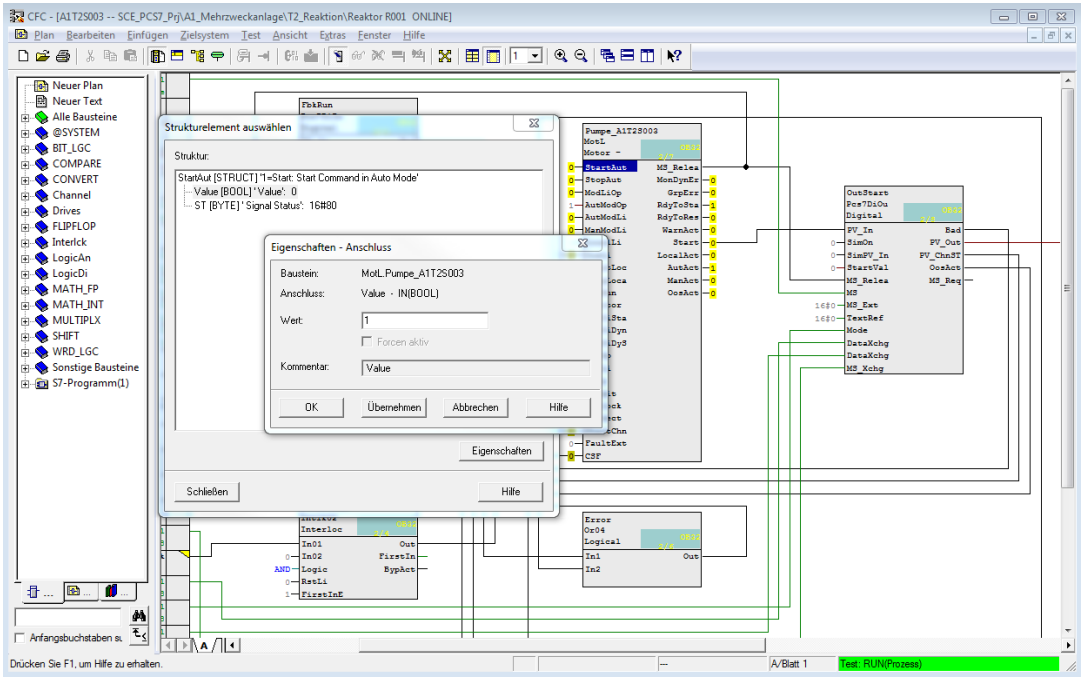
Hierzu ist der Eingang ‚RstOp' kurz auf 1 und anschließend wieder auf 0 zu setzen!

Mit Doppelklick auf diesen Eingangsparameter des Pumpenbausteins Pumpe_A1T2S003 wird das oben gezeigte Dialogfenster geöffnet. Im Feld Wert wird zunächst eine 1 eingetragen, dieser Wert wird mit Click auf den Button ‚Übernehmen' an das Leitsystem übertragen und die Fehlerausgänge auf 0 gesetzt. Um zur normalen Funktionsweise zurückzukehren muss ‚RstOp' durch Eingabe von 0 und nochmals ‚Übernehmen' abschließend auf den ursprünglichen Zustand zurückgesetzt werden.

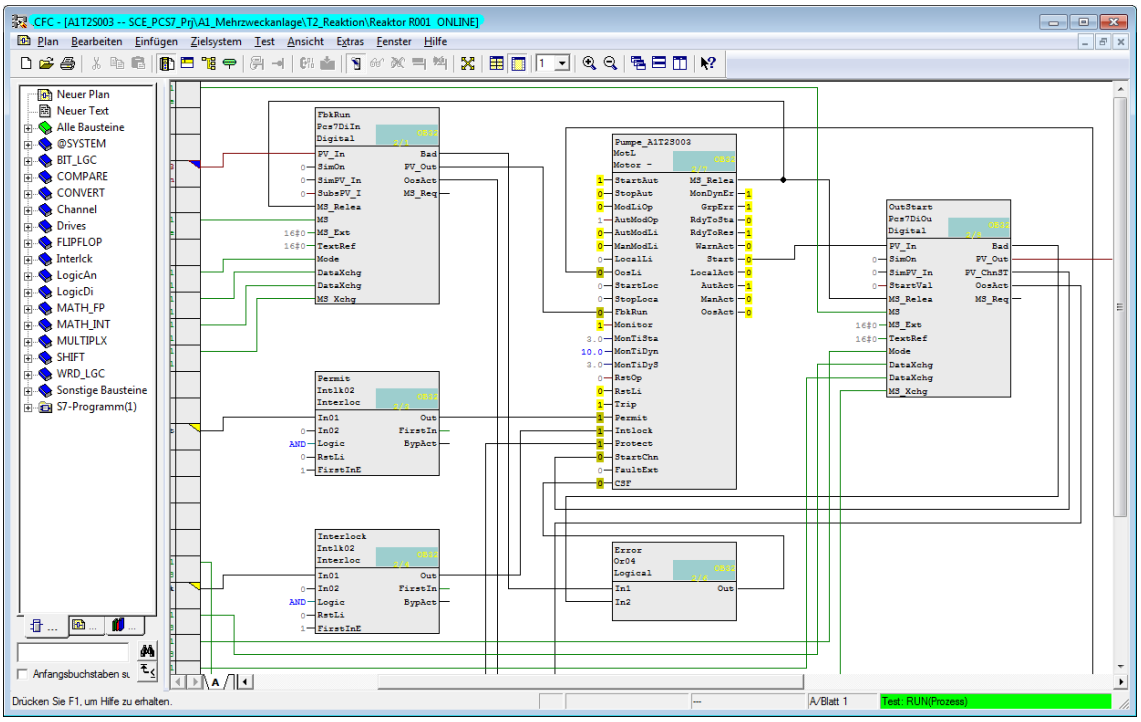
41. Im nächsten Schritt wird die Pumpe mit (StartAut == 1 und StopAut == 0) gestartet und kann jetzt mit (StartAut == 0 und StopAut == 1) wieder gestoppt werden.



(→ StartAut → Eigenschaften → Wert „1“)



42. Falls die Laufrückmeldung nicht rechtzeitig im PLCSIM angeschaltet wurde, wird im Anschluss ‚MonDynErr' ein Fehler angezeigt. Dieser kann mit ‚RstOp' == 1 quittiert werden.



TESTEN DER AUTOMATISIERUNGSLOGIK MIT DER SIMULATION

Die manuelle Eingabe von Prozesszuständen an das simulierte Leitsystem ist beim Testen kleiner Funktionen noch mit vertretbarem Aufwand möglich. Bei aufwändigeren Abläufen mit mehreren dynamischen Prozessgrößen sind jedoch schnell die Grenzen des machbaren erreicht. Hier empfiehlt sich der Einsatz einer Prozesssimulation.

Für diesen Kurs wurden deshalb die wesentlichen Zusammenhänge des hier zu automatisierenden Prozesses mit der Simulationssoftware ***SIMIT*** abgebildet. Das Modell bildet das dynamische Verhalten der Pumpen, Ventile, Behälter, Reaktoren sowie das Vor-Ort Bedienpanel mit Hauptschalter, NOTAUS, Umschaltung auf lokale Vor-Ort Bedienung und die entsprechenden Bedienelemente ab. Die dynamischen Vorgänge sind gegenüber der Realität um den Faktor 5 bis 50 beschleunigt, um die Wartezeiten kurz zu halten.

Die Bedienoberfläche des Simulators ist in Abbildung 4 abgebildet. Sie stellt auf der linken Seite das Prozessschema sowie die Signalpegel von Stell- und Messgrößen dar. Auf der rechten Seite ist oben das ockerfarben hinterlegte Vor-Ort-Bedienpanel dargestellt, unten sind eine Reihe Steuerelemente für die Simulation angebracht.

Die Anwendung des Simulators ist denkbar einfach – es muss lediglich darauf geachtet werden, dass die Belegung der Ein- und Ausgänge nicht verändert wurde.

Mit dem Simulator kann die Pumpenansteuerung nun sehr einfach überprüft werden:

1. Nach ***S7-PLCSIM*** wird das Simulationsprogramm gestartet.
2. Die Simulation beginnt mit 75 % gefüllten Edukttanks, alle anderen sind entleert. Dieser Zustand kann jederzeit mit der Option ‚RESET' in der Simulationssteuerung wieder hergestellt werden. Die Option ‚RESET 50 %' füllt alle Tanks, wie in Abbildung 4 dargestellt, zu 50 %.
3. Zum Testen wird der Motorbaustein, wie im vorherigen Abschnitt unter Schritt 41 beschrieben, angesteuert – in der Simulation leuchtet das Stellsignal der Pumpe grün auf.

4. Der simulierte Motoranlauf dauert etwa 2 Sekunden. Danach leuchtet zum einen die Laufanzeige des Motors in der Simulation grün auf, zum anderen wird der Signalpegel für den Binäreingang E 1.3 des Leitsystems gesetzt.
5. Um den Förderweg zu öffnen, muss zudem das Ventil zum Produkttank T3.B001 geöffnet werden. Dieses Ventil wird in der anschließenden Übung über eine geeignete Einzelsteuerfunktion angesprochen. In der Simulation kann bei eingeschalteter Pumpe und offenem Ventil beobachten werden, wie der Inhalt des Reaktors T2.R001 in den Produkttank T3.B001 gepumpt wird.
6. Über die Simulationssteuerungen können die Produkttanks entleert und die Edukttanks befüllt werden. Beim Befüllen über die Simulationssteuerung ist zunächst die Simulation von dem Leitsystem zu trennen – dazu wird der Button mit dem waagerechten Strich (−) einmal angeklickt. Anschließend kann das Ventil mit dem Knopf rechts daneben geöffnet werden. Das Stellsignal leuchtet grün auf, nach etwa einer Sekunde folgt das Signal des Endlagenschalters für die Offen-Position, nach weiteren 5 bis 10 Sekunden sind erste Änderungen im Füllstand sichtbar.
7. Durch ‚RESET' und ‚RESET 50 %' kann die Simulation jederzeit wieder in einen definierten Zustand gebracht werden.

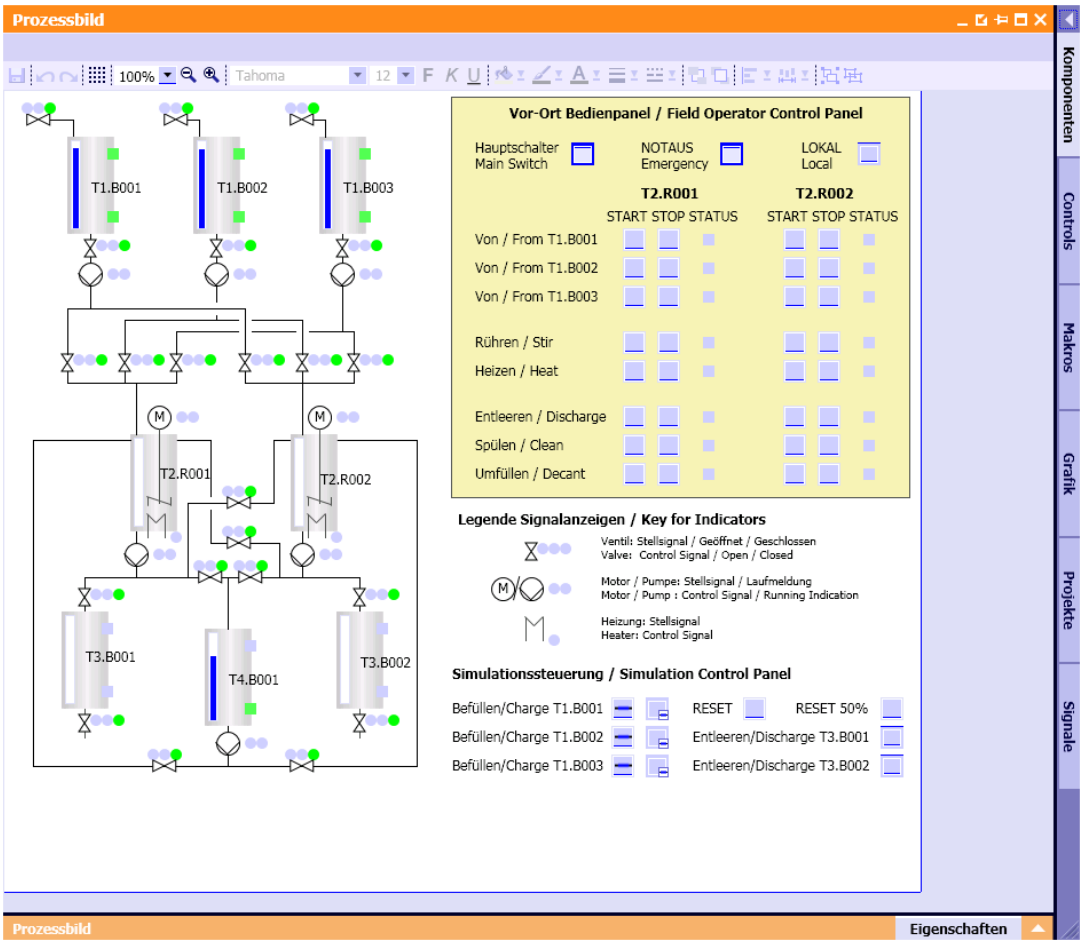


Abbildung 4: Bedienoberfläche der Prozesssimulation

ÜBUNGEN

In den Übungsaufgaben soll Gelerntes aus der Theorie und der Schritt-für-Schritt-Anleitung umgesetzt werden. Hierbei soll das schon vorhandene Multiprojekt aus der Schritt-für-Schritt-Anleitung (PCS7_SCE_0104_R1503.zip) genutzt und erweitert werden.

Ziel dieser Übung ist es einen CFC zu erstellen, mit dem die Ventile der Anlage gesteuert werden können. Hierbei soll auf das Wissen aus der Schritt-für-Schritt-Anleitung aufgebaut werden, in der ein ähnlicher CFC zur Steuerung des Motors erstellt wurde.

Außerdem wird ein CFC zur Normierung des Füllstandes, also eines analogen Eingangswertes, vom digitalisierten Wert auf den physischen Wert erstellt. Diese Aufgabe erweitert das Wissen, da der CFC ohne ein Template aber trotzdem mit einem Baustein aus der Bibliothek angelegt wird. Dieser CFC wird für das Kapitel Anlagensicherung benötigt.

ÜBUNGSAUFGABEN

Die folgenden Übungen orientieren sich an der Schritt-für-Schritt-Anleitung. Für jede Übungsaufgabe können die entsprechenden Schritte der Anleitung als Hilfestellung genutzt werden.

1. Fügen Sie das Template ‚Valve_Lean' als Vorlage in die Messstellentypen ein (analog zu ‚Motor_Lean'). Diese Vorlage dient der Implementierung der Ventile.
2. Im Planordner ‚Produktbehälter B001' der Teilanlage T3_Produktspeicher soll nun eine Objektinstanz des Ventiltemplates eingefügt und in A1T3X001 umbenannt werden. Öffnen Sie den CFC-Plan und passen Sie auch den Namen des Bausteins ‚VlvL' an. Schließen Sie nun die Rückmeldungs- und Steuersignale (siehe Abbildung 5 und Tabelle 4) an.
3. Laden und Testen Sie nun Ihre Implementierung mit dem SIMIT-Modell. Dafür sollten die folgenden Anschlüsse sichtbar sein: ‚ModLiOp', ‚AutModOp', ‚ManModOp', ‚OpenAut', ‚CloseAut' und ‚RstOp'.
4. Zum Einbinden des analogen Füllstandsensors A1T2L001 (siehe Abbildung 5) erstellen Sie einen neuen CFC im Planordner ‚Reaktor R001'. Benennen Sie diesen mit A1T2L001 und öffnen Sie ihn. Ziehen Sie den Baustein ‚Pcs7AnIn' (FB1869) per Drag/Drop aus dem Katalog in den CFC. Wählen Sie dafür im linken Rahmen das Register Bibliotheken an und nutzen nun entweder die Suchfunktion ganz unten oder öffnen PCS 7 AP Library V8.1/Blocks+Templates\Blocks/Channel. Sobald der Baustein eingefügt wurde, benennen Sie ihn in Stand_A1T2L001 um.
5. Parametrieren Sie den Baustein ‚Pcs7AnIn' nun, indem Sie die Eingangswerte ‚Scale' auf High = 0.0 und Low = 1158.0 und ‚PV_InUni' auf 1040 (für die Einheit ml) setzen. Verbinden Sie den Eingang ‚PV_In' des Bausteins ‚Pcs7AnIn' mit dem Symbol für den Füllstandistwert (siehe Tabelle 4) des Reaktors R001.
6. Realisieren Sie nun den oberen und unteren Füllstandsensor vom Produktbehälter B001. Legen Sie dafür einen CFC-Plan im Planordner ‚Produkttank B001' an und benennen Sie ihn mit A1T3L001. Öffnen Sie den Plan und fügen Sie zweimal den Baustein ‚Pcs7DiIn' aus dem Katalog (analog zu Aufgabe 5) hinzu. Benennen Sie den einen Baustein mit A1T3L001_LSA+ und den anderen mit A1T3L001_LSA-. Verschalten Sie jeweils ‚PV_In' mit den Sensorsignalen.
7. Legen Sie nun CFC-Pläne für den Hauptschalter, den NOTAUS-Schalter sowie den Schalter für die lokale Bedienung an. Dazu wird wie in Aufgabe 7 jeweils ein CFC-Plan für A1H001, A1H002 und A1H003 im Planordner ‚Mehrzweckanlage' angelegt, der Baustein ‚Pcs7DiIn' hinzugefügt, benannt und mit dem jeweiligen Operanden nach ‚PV_In' verschaltet.

Tabelle 4: Die Symbole für die Realisierung der Ventilsteuerung und des Füllstandsensors

		Symbol	Adresse	Datentyp	Kommentar
Aufgabe	2	A1.T3.A1T3X001.XV.C	A 0.6	BOOL	Auf/Zu-Ventil Zufluss Produkttank B001 Stellsignal
		A1.T3.A1T3X001.GO+-.O+	E 15.4	BOOL	Auf/Zu-Ventil Zufluss Produkttank B001 Rückmeldung auf/ein
		A1.T3.A1T3X001.GO+-.O-	E 15.5	BOOL	Auf/Zu-Ventil Zufluss Produkttank B001 Rückmeldung zu
	5	A1.T2.A1T2L001.LISA+.M	EW 512	WORD	Füllstandistwert Reaktor R001
	6	A1.T3.A1T3L001.LSA+.SA+	E 18.6	BOOL	Füllstandsüberwachung Produkttank B001 Schaltpunkt H
		A1.T3.A1T3L001.LSA-.SA-	E 18.7	BOOL	Füllstandsüberwachung Produkttank B001 Schaltpunkt L
	7	A1.A1H001.HS+-.START	E 0.0	BOOL	Mehrzweckanlage einschalten
		A1.A1H002.HS+-.OFF	E 0.1	BOOL	NOTAUS aktivieren (Schließer)
		A1.A1H003.HS+-.LOC	E 0.2	BOOL	Lokale Bedienung aktivieren

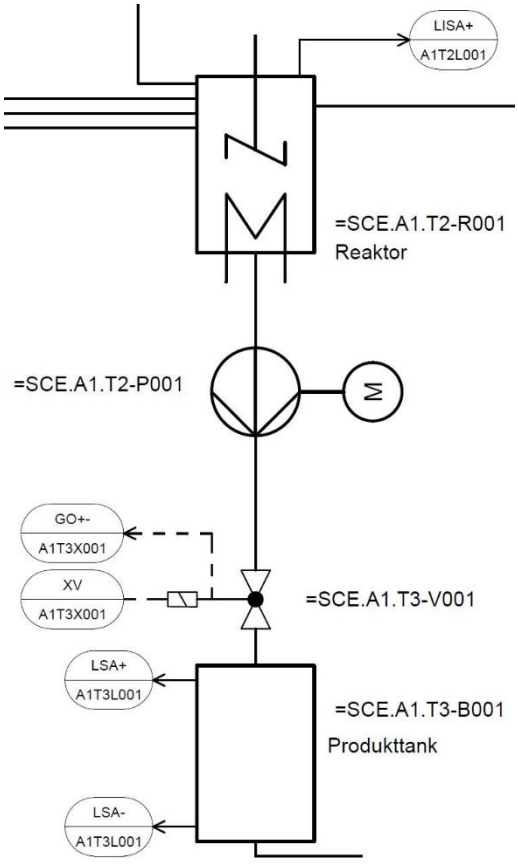


Abbildung 5: Ausschnitt aus dem R&I-Fließbild