SIEMENS

SCE Lehrunterlagen

Siemens Automation Cooperates with Education | 05/2017

TIA Portal Modul 032-100

Grundlagen der FC-Programmierung
mit SIMATIC S7-1500

Cooperates with Education
Automation
SIEMENS

Passende SCE Trainer Pakete zu diesen Lehrunterlagen

SIMATIC Steuerungen

- **SIMATIC ET 200SP Open Controller CPU 1515SP PC F und HMI RT SW**
  Bestellnr.: 6ES7677-2FA41-4AB1
- **SIMATIC ET 200SP Distributed Controller CPU 1512SP F-1 PN Safety**
  Bestellnr.: 6ES7512-1SK00-4AB2
- **SIMATIC CPU 1516F PN/DP Safety**
  Bestellnr.: 6ES7516-3FN00-4AB2
- **SIMATIC S7 CPU 1516-3 PN/DP**
  Bestellnr.: 6ES7516-3AN00-4AB3
- **SIMATIC CPU 1512C PN mit Software und PM 1507**
  Bestellnr.: 6ES7512-1CK00-4AB1
- **SIMATIC CPU 1512C PN mit Software, PM 1507 und CP 1542-5 (PROFIBUS)**
  Bestellnr.: 6ES7512-1CK00-4AB2
- **SIMATIC CPU 1512C PN mit Software**
  Bestellnr.: 6ES7512-1CK00-4AB6
- **SIMATIC CPU 1512C PN mit Software und CP 1542-5 (PROFIBUS)**
  Bestellnr.: 6ES7512-1CK00-4AB7

SIMATIC STEP 7 Software for Training

- **SIMATIC STEP 7 Professional V14 SP1 - Einzel-Lizenz**
  Bestellnr.: 6ES7822-1AA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1- 6er Klassenraumlizenz**
  Bestellnr.: 6ES7822-1BA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1 - 6er Upgrade-Lizenz**
  Bestellnr.: 6ES7822-1AA04-4YE5
- **SIMATIC STEP 7 Professional V14 SP1 - 20er Studenten-Lizenz**
  Bestellnr.: 6ES7822-1AC04-4YA5

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

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. Die 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, der Fa. Michael Dziallas Engineering und allen weiteren Beteiligten für die Unterstützung bei der Erstellung dieser SCE Lehrunterlage.

Inhaltsverzeichnis

GRUNDLAGEN DER FC-PROGRAMMIERUNG

1 Zielstellung

In diesem Kapitel lernen Sie die grundlegenden Elemente eines Steuerungsprogrammes – die ***Organisationsbausteine (OB),*** die ***Funktionen (FC)***, die **Funktionsbausteine (FB)** und die **Datenbausteine (DB)** kennen. Zusätzlich werden Ihnen die ***bibliotheksfähige*** Funktions- und Funktionsbausteinprogrammierung vorgestellt. Sie lernen die Programmiersprache ***Funktionsplan (FUP)*** kennen und nutzen diese zur Programmierung einer Funktion FC1 und eines Organisationsbausteins OB1.

Es können die unter Kapitel 3 aufgeführten SIMATIC S7-Steuerungen eingesetzt werden.

2 Voraussetzung

Dieses Kapitel baut auf der Hardwarekonfiguration einer SIMATIC S7 auf. Es kann mit beliebigen Hardwarekonfigurationen, die digitale Eingangs- und Ausgangskarten besitzen, realisiert werden. Zur Durchführung dieses Kapitels können Sie z.B. auf das folgende Projekt zurückgreifen:

„SCE_DE_012_101__Hardwarekonfiguration_CPU1516F…..zap13“

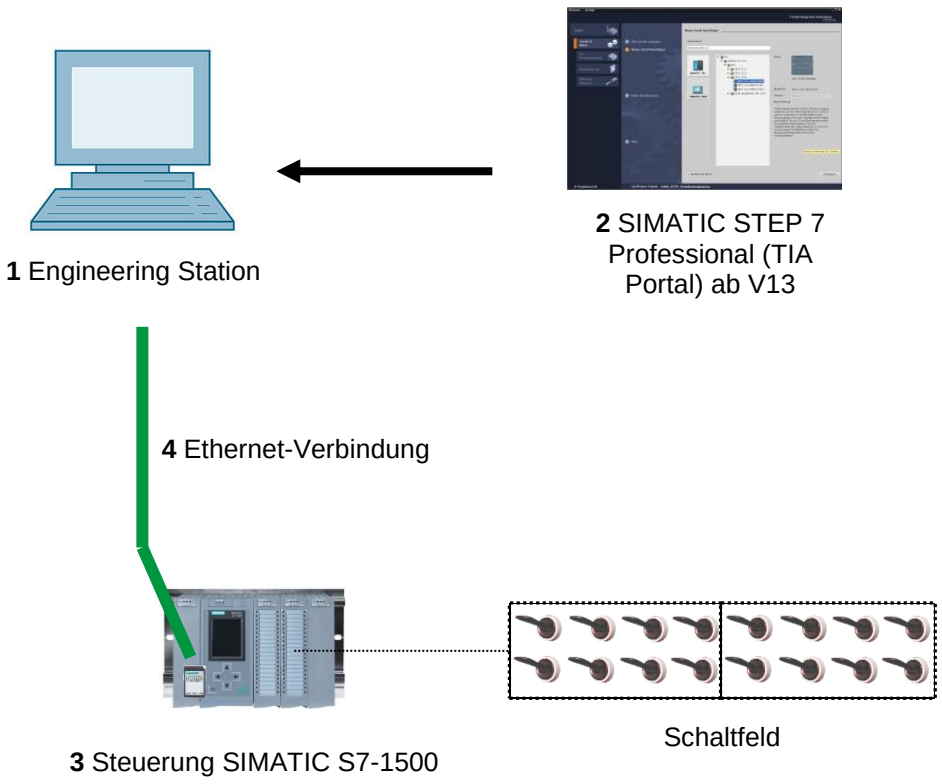
3 Benötigte Hardware und Software

1 Engineering Station: Voraussetzungen sind Hardware und Betriebssystem
(weitere Informationen siehe Readme/Liesmich auf den TIA Portal Installations-DVDs)

2 Software SIMATIC STEP 7 Professional im TIA Portal – ab V13

3 Steuerung SIMATIC S7-1500/S7-1200/S7-300, z.B. CPU 1516F-3 PN/DP –
ab Firmware V1.6 mit Memory Card und 16DI/16DO sowie 2AI/1AO
Hinweis: Die digitalen Eingänge sollten auf ein Schaltfeld herausgeführt sein.

4 Ethernet-Verbindung zwischen Engineering Station und Steuerung



4 Theorie

4.1 Betriebssystem und Anwendungsprogramm

Das ***Betriebssystem*** ist in jeder Steuerung (CPU) enthalten und organisiert alle Funktionen und Abläufe der CPU, die nicht mit einer spezifischen Steuerungsaufgabe verbunden sind. Zu den Aufgaben des Betriebssystems gehören z. B.:

- Abwickeln von Neustart (Warmstart)
- Aktualisieren des Prozessabbilds der Eingänge und des Prozessabbilds der Ausgänge
- Zyklisches Aufrufen des Anwenderprogramms
- Erfassen von Alarmen und Aufrufen der Alarm-OBs
- Erkennen und Behandeln von Fehlern
- Verwalten von Speicherbereichen

Das Betriebssystem ist Bestandteil der CPU und ist bei der Auslieferung bereits auf dieser enthalten.

Das ***Anwenderprogramm*** enthält alle Funktionen, die zur Bearbeitung Ihrer spezifischen Automatisierungsaufgabe erforderlich sind. Zu den Aufgaben des Anwenderprogramms gehören:

- Prüfung der Vorbedingungen für einen Neustart (Warmstart) mithilfe von Anlauf-OBs
- Bearbeiten von Prozessdaten d.h. Ansteuerung der Ausgangssignale in Abhängigkeit von den Zuständen der Eingangssignale
- Reaktion auf Alarme und Alarmeingänge
- Bearbeiten von Störungen im normalen Programmablauf

4.2 Organisationsbausteine

Die Organisationsbausteine (OB) bilden die Schnittstelle zwischen dem Betriebssystem der Steuerung (CPU) und dem Anwendungsprogramm. Sie werden vom Betriebssystem aufgerufen und steuern folgende Vorgänge:

- Zyklische Programmbearbeitung (z.B. OB1)
- Anlaufverhalten der Steuerung
- Alarmgesteuerte Programmbearbeitung
- Fehlerbehandlung

In einem Projekt muss mindestens ***ein Organisationsbaustein für die zyklische Programmbearbeitung*** vorhanden sein. Ein OB wird durch ein ***Startereignis*** aufgerufen, wie in Abbildung 1 dargestellt. Dabei haben die einzelnen OBs festgelegte Prioritäten, damit z.B. ein OB82 zur Fehlerbehandlung den zyklischen OB1 unterbrechen kann.

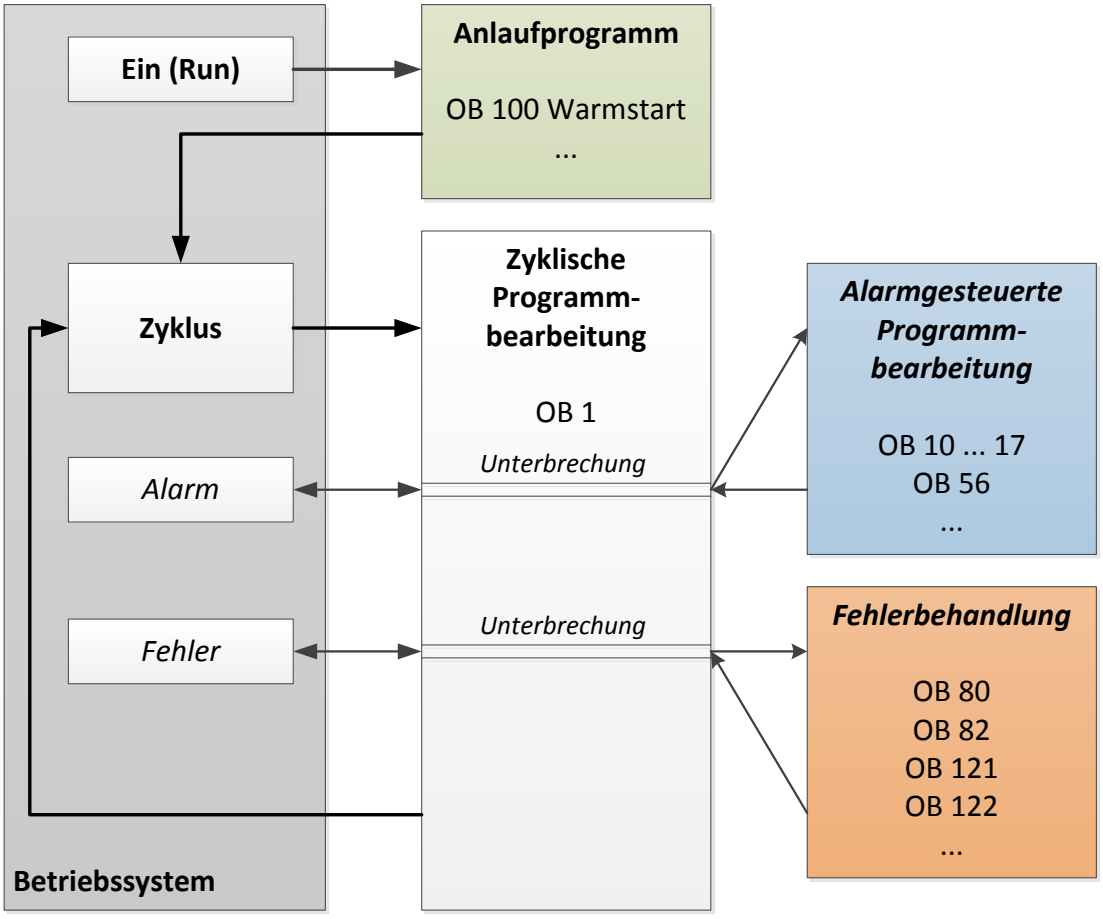


Abbildung 1: Startereignisse im Betriebssystem und OB-Aufruf

Nach dem Auftreten eines Startereignisses sind folgenden Reaktionen möglich:

- Falls dem Ereignis ein OB zugeordnet wurde, stößt dieses Ereignis die Ausführung des zugeordneten OB an. Ist die Priorität des zugeordneten OB höher als die Priorität des gerade ausgeführten OBs, wird dieser sofort ausgeführt (Interrupt). Ist dies nicht der Fall, wird zuerst noch gewartet bis der OB mit der höheren Priorität ausgeführt werden konnte.
- Falls dem Ereignis kein OB zugeordnet haben, wird die voreingestellte Systemreaktion durchgeführt.

Tabelle 1 gibt für eine SIMATIC S7-1500 ein paar Beispiele für Startereignisse, deren mögliche OB-Nummer(n) und die voreingestellte Systemreaktion sollte der Organisationsbaustein nicht in der Steuerung vorhanden sein.

Startereignis	**Mögliche OB-Nummer**	**Voreingestellte Systemreaktion**
Anlauf	100, ≥ 123	Ignorieren
Zyklisches Programm	1, ≥ 123	Ignorieren
Uhrzeitalarm	10 bis 17, ≥ 123	-
Update-Alarm	56	Ignorieren
Zyklusüberwachungszeit einmal überschritten	80	STOP
Diagnosealarm	82	Ignorieren
Programmierfehler	121	STOP
Peripheriezugriffsfehler	122	Ignorieren

Tabelle 1: OB-Nummern für unterschiedliche Startereignisse

4.3 Prozessabbild und zyklische Programmbearbeitung

Wenn im zyklischen Anwenderprogramm die Eingänge (E) und Ausgänge (A) angesprochen werden, so werden die Signalzustände normalerweise nicht direkt von den Ein-/Ausgabemodulen abgefragt, sondern es wird auf einen Speicherbereich der CPU zugegriffen. Dieser Speicherbereich enthält ein Abbild der Signalzustände und wird als **Prozessabbild** bezeichnet.

Die zyklische Programmbearbeitung geschieht mit folgendem Ablauf:

1. Am Anfang des zyklischen Programms wird abgefragt, ob die einzelnen Eingänge Spannung führen oder nicht. Dieser Status der Eingänge wird in dem **Prozessabbild der Eingänge (PAE)** gespeichert. Dabei wird für die Spannung führenden Eingänge die Information 1 oder „High“, für die keine Spannung führenden die Information 0 oder „Low“ hinterlegt.
2. Der Prozessor arbeitet nun das im zyklischen Organisationsbaustein hinterlegte Programm ab. Dabei wird für die benötigte Eingangsinformation auf das bereits vorher eingelesene **Prozessabbild der Eingänge (PAE**) zugegriffen und die Verknüpfungsergebnisse in ein sogenanntes **Prozessabbild der Ausgänge** (**PAA**) geschrieben.
3. Am Ende des Zyklus wird das **Prozessabbild der Ausgänge** (**PAA**) als Signalzustand zu den Ausgabemodulen übertragen und diese ein- bzw. ausgeschaltet. Danach geht es wieder weiter mit Punkt 1.

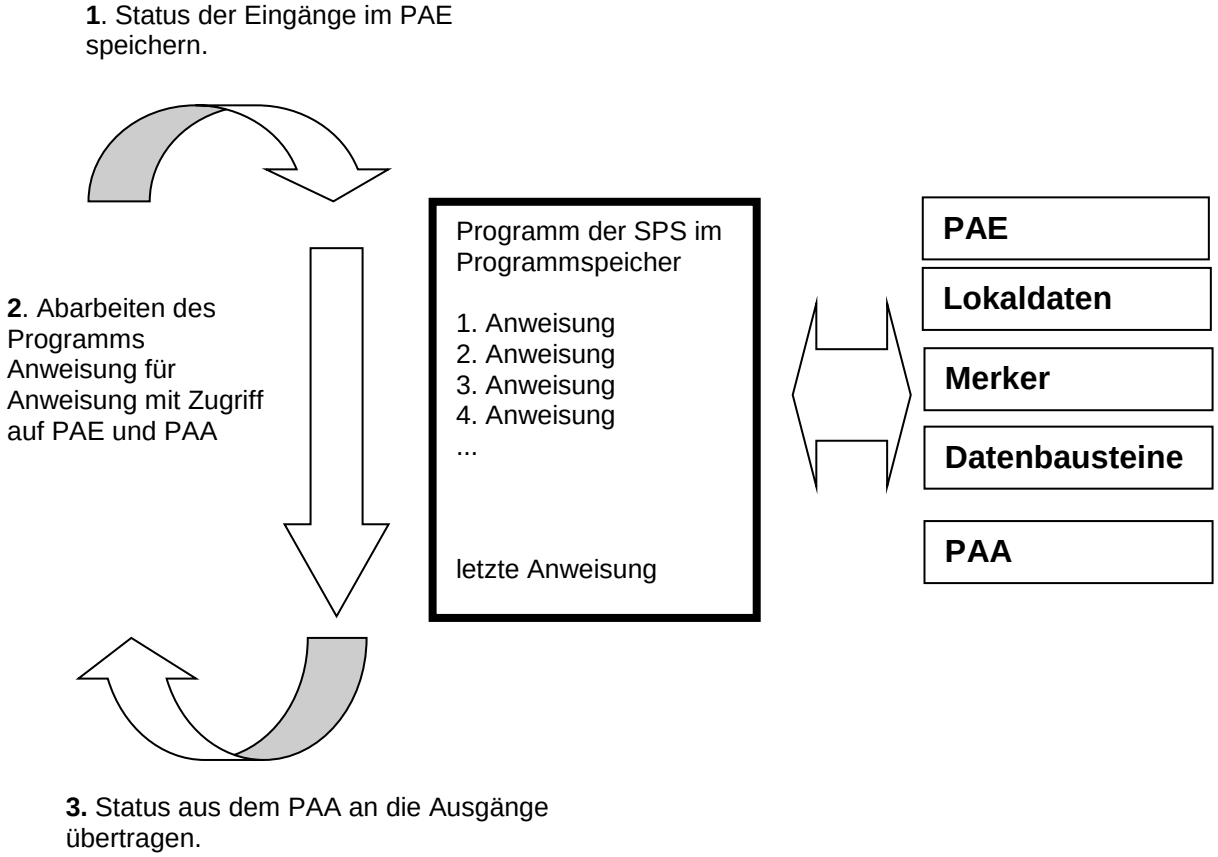


Abbildung 2: Zyklische Programmbearbeitung

Hinweis: *Die Zeit die der Prozessor für diesen Ablauf benötigt nennt man Zykluszeit. Diese ist wiederum abhängig von Anzahl und Art der Anweisungen und der Prozessorleistung der Steuerung.*

4.4 Funktionen

Funktionen (FCs) sind Codebausteine ohne Gedächtnis. Sie ***haben keinen Datenspeicher***, in denen Werte von Bausteinparametern gespeichert werden könnten. Deshalb müssen beim Aufruf einer Funktion alle Schnittstellenparameter beschaltet werden. Um Daten dauerhaft zu speichern müssen zuvor globale Datenbausteine angelegt werden.

Eine Funktion enthält ein Programm, das immer ausgeführt wird, wenn die Funktion von einem anderen Codebaustein aufgerufen wird.

Funktionen können z.B. zu folgenden Zwecken eingesetzt werden:

- Mathematische Funktionen die in Abhängigkeit von Eingangswerten ein Ergebnis zurückgeben.
- Technologische Funktionen wie Einzelansteuerungen mit Binärverknüpfungen

Eine Funktion kann auch mehrmals an verschiedenen Stellen innerhalb eines Programms aufgerufen werden.

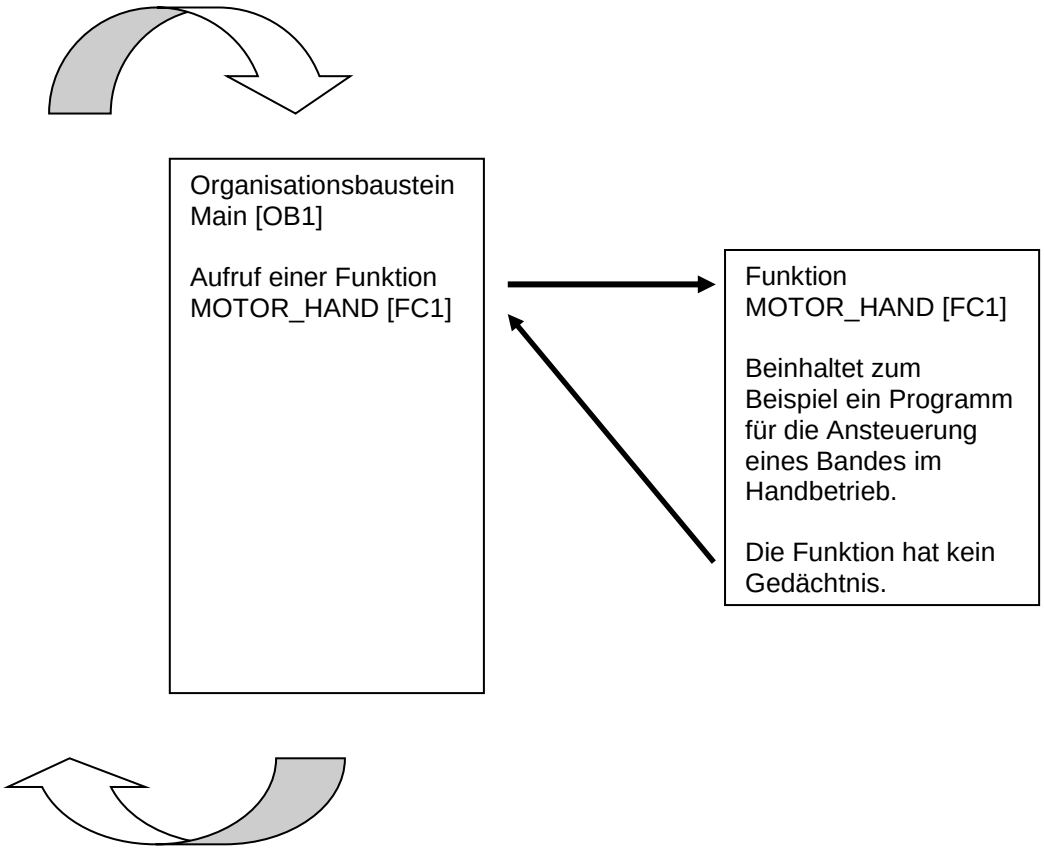


Abbildung 3: Funktion mit Aufruf aus dem Organisationsbaustein Main[OB1]

4.5 Funktionsbausteine und Instanz-Datenbausteine

Funktionsbausteine sind Codebausteine, die ihre Eingangsvariablen, Ausgangsvariablen, Durchgangsvariablen und auch die statischen Variablen dauerhaft in Instanz-Datenbausteinen ablegen, sodass sie auch ***nach der Bausteinbearbeitung zur Verfügung stehen***. Deshalb werden sie auch als Bausteine mit "Gedächtnis" bezeichnet.

Funktionsbausteine können auch mit temporären Variablen arbeiten. Die temporären Variablen werden jedoch nicht im Instanz-DB abgespeichert, sondern stehen nur einen Zyklus lang zur Verfügung.

Funktionsbausteine werden bei Aufgaben verwendet die mit Funktionen nicht realisierbar sind:

- Immer wenn in den Bausteinen Zeiten und Zähler benötigt werden.
- Immer wenn eine Information in dem Programm gespeichert werden muss. Zum Beispiel eine Vorwahl der Betriebsart mit einem Taster.

Funktionsbausteine werden immer dann ausgeführt, wenn ein Funktionsbaustein von einem anderen Codebaustein aufgerufen wird. Ein Funktionsbaustein kann auch mehrmals an verschiedenen Stellen innerhalb eines Programms aufgerufen werden. Sie erleichtern so die Programmierung häufig wiederkehrender, komplexer Funktionen.

Ein Aufruf eines Funktionsbausteins wird als Instanz bezeichnet. Jeder Instanz eines Funktionsbausteins wird ein Speicherbereich zugeordnet, der die Daten enthält, mit denen der Funktionsbaustein arbeitet. Dieser Speicher wird von Datenbausteinen zur Verfügung gestellt, die automatisch von der Software erstellt werden.

Es ist auch möglich den Speicher für mehrere Instanzen in einem Datenbaustein als **Multiinstanz** zur Verfügung zu stellen. Die maximale Größe von Instanz-Datenbausteinen variiert abhängig von der CPU. Die im Funktionsbaustein deklarierten Variablen bestimmen die Struktur des Instanz- Datenbausteins.

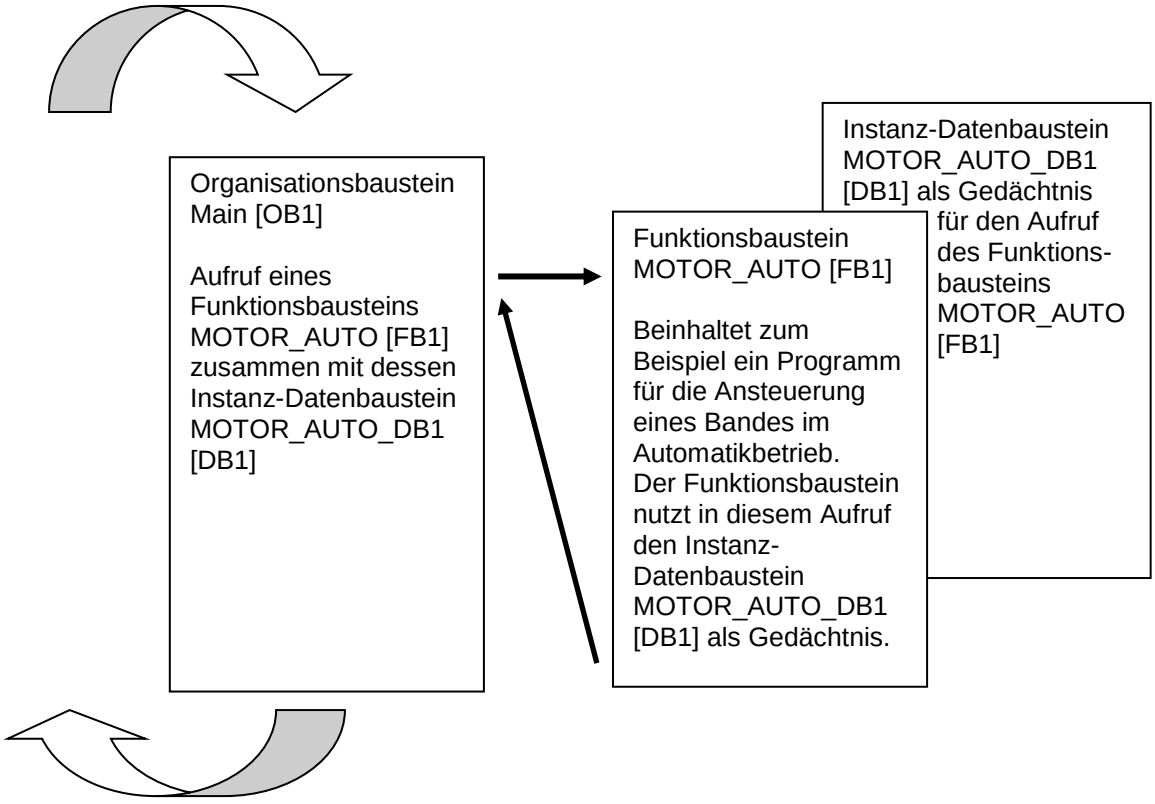


Abbildung 4: Funktionsbaustein und Instanz mit Aufruf aus dem Organisationsbaustein Main[OB1]

4.6 Globale Datenbausteine

Datenbausteine enthalten im Gegensatz zu Codebausteinen keine Anweisungen, sondern dienen der Speicherung von Anwenderdaten.

In Datenbausteinen stehen also variable Daten, mit denen das Anwenderprogramm arbeitet. Die Struktur globaler Datenbausteine können Sie beliebig festlegen.

Globale Datenbausteine nehmen Daten auf, die ***von allen anderen Bausteinen*** aus verwendet werden können (siehe Abbildung 5). Auf Instanz- Datenbausteine sollte nur der zugehörige Funktionsbaustein zugreifen. Die maximale Größe von Datenbausteinen variiert abhängig von der CPU.

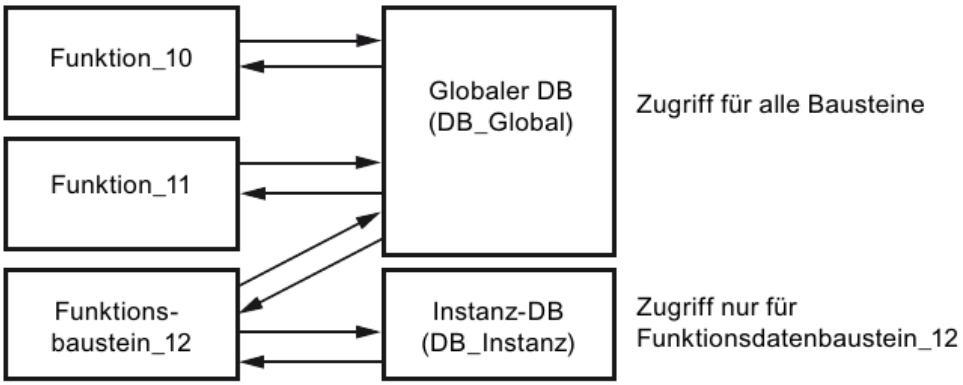


Abbildung 5: Unterschied zwischen globalem DB und Instanz-DB.

Anwendungsbeispiele für **globale Datenbausteine** sind:

- Speicherung der Informationen zu einem Lagersystem. „Welches Produkt liegt wo?“
- Speicherung von Rezepturen zu bestimmten Produkten.

4.7 Bibliotheksfähige Codebausteine

Die Erstellung eines Anwenderprogramms kann linear oder strukturiert erfolgen. Die ***lineare Programmierung*** schreibt das gesamte Anwenderprogramm in den Zyklus-OB, eignet sich jedoch nur für sehr einfache Programme bei denen inzwischen andere, günstigere Steuerungssysteme z.B. LOGO! zum Einsatz kommen.

Bei komplexeren Programmen ist immer eine ***strukturierte Programmierung*** zu empfehlen. Hier kann die gesamte Automatisierungsaufgabe in kleine Teilaufgaben zerlegt werden, um diese nun in Funktionen und Funktionsbausteinen zu lösen.

Dabei sollten bevorzugt bibliotheksfähige Codebausteine erstellt werden. Das heißt, dass die Eingangs- und Ausgangsparameter einer Funktion oder eines Funktionsbausteins allgemein festgelegt werden und erst bei der Nutzung des Bausteins mit den aktuellen globalen Variablen (Eingänge/Ausgänge) versehen werden.

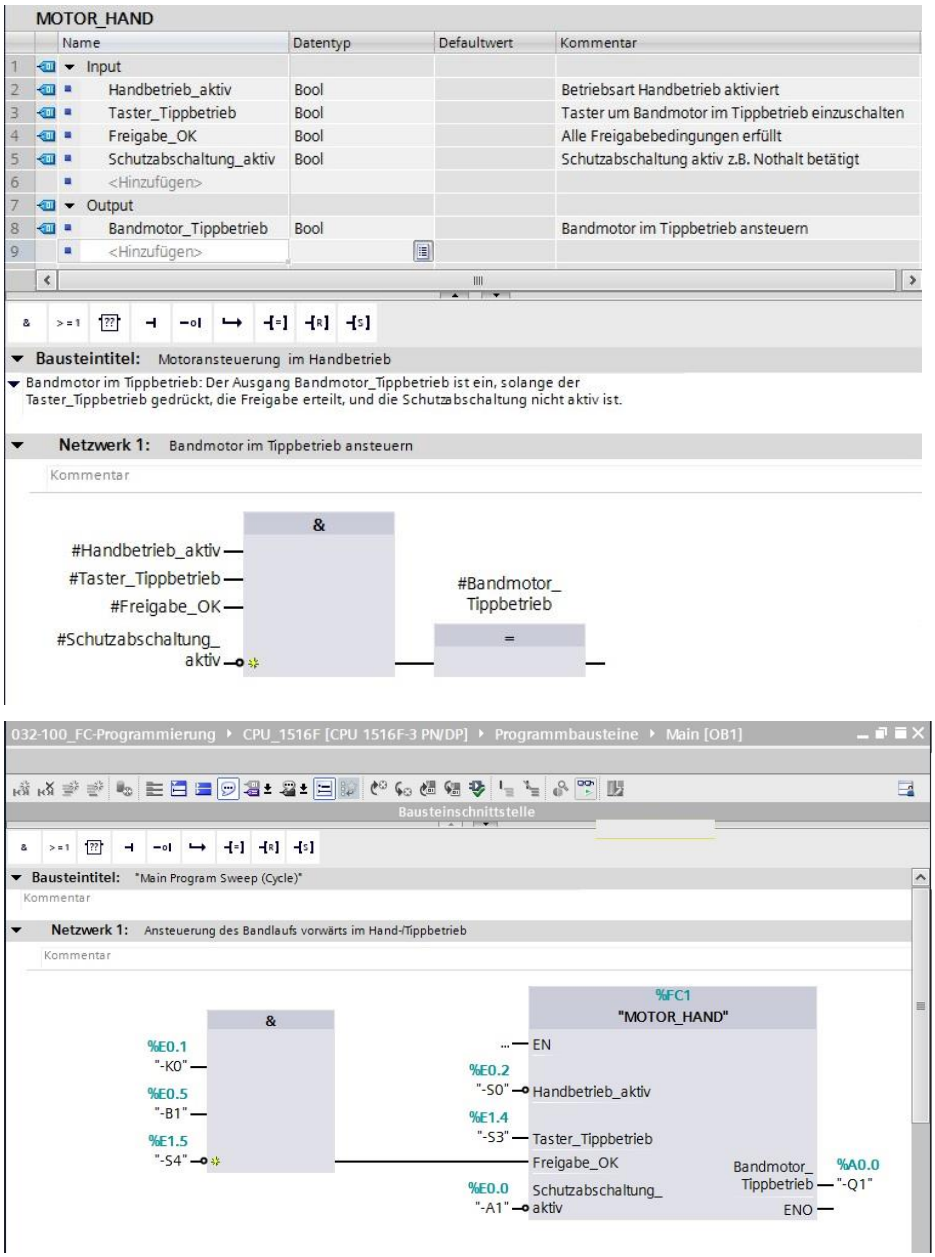


Abbildung 6: Bibliotheksfähige Funktion mit Aufruf im OB1

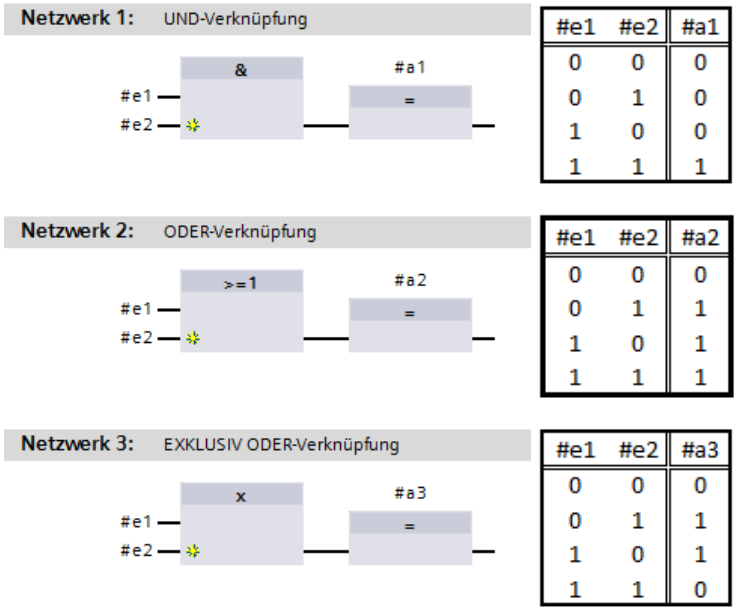
4.8 Programmiersprachen

Zur Programmierung von Funktionen stehen die Programmiersprachen Funktionsplan (FUP), Kontaktplan (KOP), Anweisungsliste (AWL) und Structured Control Language (SCL) zur Verfügung. Für Funktionsbausteine gibt es zusätzlich die Programmiersprache GRAPH zur Programmierung grafischer Schrittketten.

Im Folgenden wird die Programmiersprache ***Funktionsplan (FUP)*** vorgestellt.

FUP ist eine grafische Programmiersprache. Die Darstellung ist elektronischen Schaltkreissystemen nachempfunden. Das Programm wird in Netzwerken abgebildet. Ein Netzwerk enthält einen oder mehrere Verknüpfungspfade. Binäre und analoge Signale werden durch Boxen miteinander verknüpft. Zur Darstellung der binären Logik werden die von der booleschen Algebra bekannten grafischen Logiksymbole verwendet.

Mit binären Funktionen können Sie Binäroperanden abfragen und deren Signalzustände verknüpfen. Beispiele für binäre Funktionen sind die Anweisungen "UND-Verknüpfung", "ODER-Verknüpfung" und "EXKLUSIV ODER-Verknüpfung" wie in Abbildung 7 dargestellt.



#e1	#e2	#a1
0	0	0
0	1	0
1	0	0
1	1	1

#e1	#e2	#a2
0	0	0
0	1	1
1	0	1
1	1	1

#e1	#e2	#a3
0	0	0
0	1	1
1	0	1
1	1	0

Abbildung 7: Binäre Funktionen in FUP und zugehörige Logiktabelle

Mit einfachen Anweisungen können Sie so z.B. binäre Ausgänge steuern, Flanken auswerten oder Sprungfunktionen im Programm ausführen.

Komplexe Anweisungen stellen Programmelemente wie z.B. IEC-Zeiten und IEC-Zähler zur Verfügung.

Die Leerbox dient als Platzhalter, in dem Sie die gewünschte Anweisung auswählen können.

Freigabeeingang EN (enable) / Freigabeausgang ENO (enable output) -Mechanismus:

- Eine Anweisung ohne EN-/ENO-Mechanismus wird unabhängig vom Signalzustand an den Box-Eingängen ausgeführt.
- Anweisungen mit EN-/ENO-Mechanismus werden nur ausgeführt, wenn der Freigabeeingang "EN" den Signalzustand "1" führt. Bei ordnungsgemäßer Bearbeitung der Box führt der Freigabeausgang "ENO" den Signalzustand "1". Sobald während der Bearbeitung ein Fehler auftritt, wird der Freigabeausgang "ENO" zurückgesetzt. Wenn der Freigabeeingang EN nicht verschaltet ist, wird die Box immer ausgeführt.

5 Aufgabenstellung

In diesem Kapitel sollen die folgenden Funktionen der Prozessbeschreibung Sortieranlage geplant, programmiert und getestet werden:

- Handbetrieb – Bandmotor im Tippbetrieb

6 Planung

Die Programmierung aller Funktionen im OB1 wird aus Gründen der Übersichtlichkeit und Wiederverwendbarkeit nicht empfohlen. Der Programmcode wird deshalb größtenteils in Funktionen (FCs) und Funktionsbausteine (FBs) ausgelagert. Diese Entscheidung, welche Funktionen in FCs ausgelagert werden und welche im OB1 ablaufen sollen, wird im Folgenden geplant.

6.1 NOTHALT

Das NOTHALT benötigt keine eigene Funktion. Ebenso wie die Betriebsart kann der aktuelle Zustand des NOTHALT-Relais direkt an den Bausteinen genutzt werden.

6.2 Handbetrieb – Bandmotor im Tippbetrieb

Der Tippbetrieb des Bandmotors soll in einer Funktion (FC) „MOTOR_HAND“ gekapselt werden. Damit ist zum einen die Übersichtlichkeit im OB1 gewahrt, zum anderen ist bei einer Erweiterung der Anlage um ein weiteres Förderband, die Wiederverwendung möglich. In Tabelle 2 sind die geplanten Parameter aufgeführt.

Input	Datentyp	Kommentar
Handbetrieb_aktiv	BOOL	Betriebsart Handbetrieb aktiviert
Taster_Tippbetrieb	BOOL	Taster um Bandmotor im Tippbetrieb einzuschalten
Freigabe_OK	BOOL	Alle Freigabebedingungen erfüllt
Schutzabschaltung_aktiv	BOOL	Schutzabschaltung aktiv z.B. Not Halt betätigt
Output		
Bandmotor_Tippbetrieb	BOOL	Bandmotor im Tippbetrieb ansteuern

Tabelle 2: Parameter für FC "MOTOR_HAND"

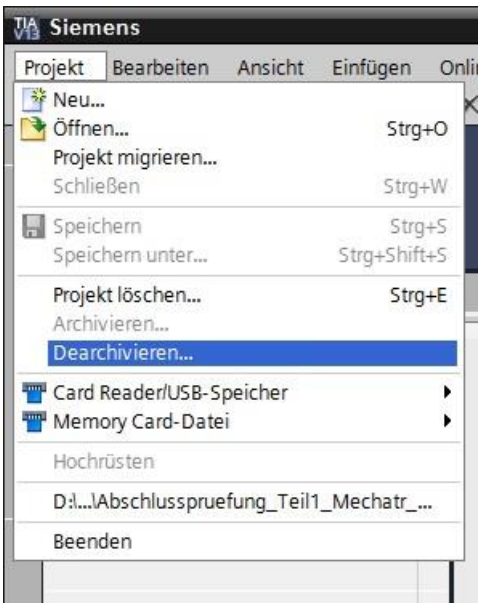
Der Ausgang Bandmotor_Tippbetrieb ist ein, solange der Taster_Tippbetrieb gedrückt, die Betriebsart Handbetrieb aktiviert, die Freigabe erteilt, und die Schutzabschaltung nicht aktiv ist.

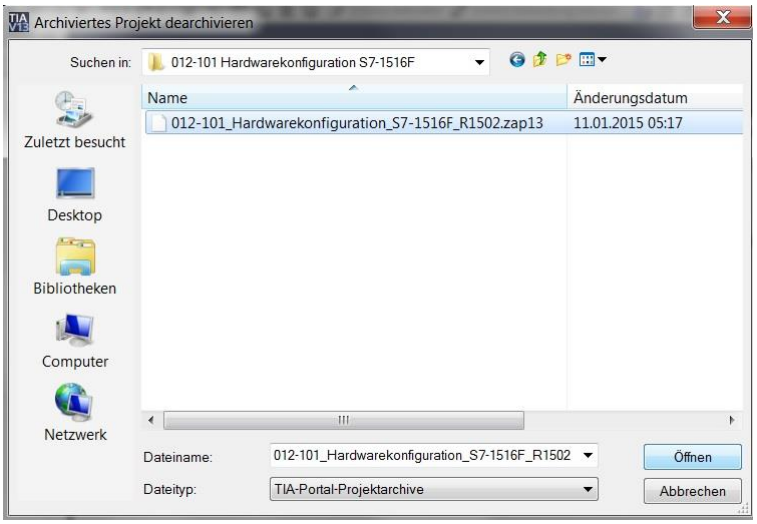
7 Strukturierte Schritt-für-Schritt-Anleitung

Im Folgenden finden Sie eine Anleitung wie Sie die Planung umsetzen können. Sollten Sie schon gut klarkommen reichen Ihnen die nummerierten Schritte zur Bearbeitung aus. Ansonsten folgen Sie einfach den folgenden detaillierten Schritten der Anleitung.

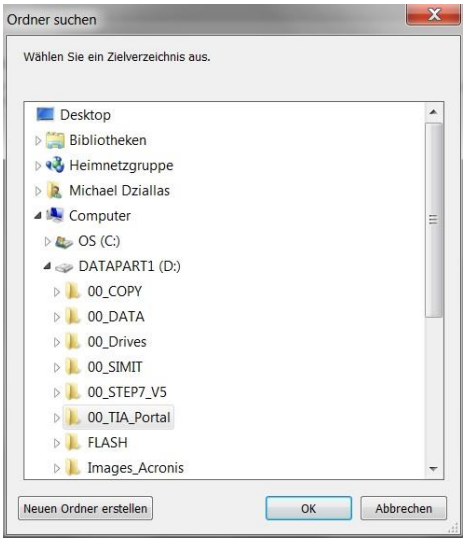
7.1 Dearchivieren eines vorhandenen Projekts

→ Bevor wir mit der Programmierung der Funktion (FC) „MOTOR_HAND“ beginnen können, benötigen wir ein Projekt mit einer Hardwarekonfiguration. (z.B. SCE_DE_012-101_Hardwarekonfiguration_S7-1516F_R1502.zap) Zum Dearchivieren eines vorhandenen Projekts müssen Sie aus der Projektansicht heraus unter →Projekt →Dearchivieren das jeweilige Archiv aussuchen. Bestätigen Sie Ihre Auswahl anschließend mit öffnen. (→ Projekt → Dearchivieren → Auswahl eines .zap-Archivs → öffnen)



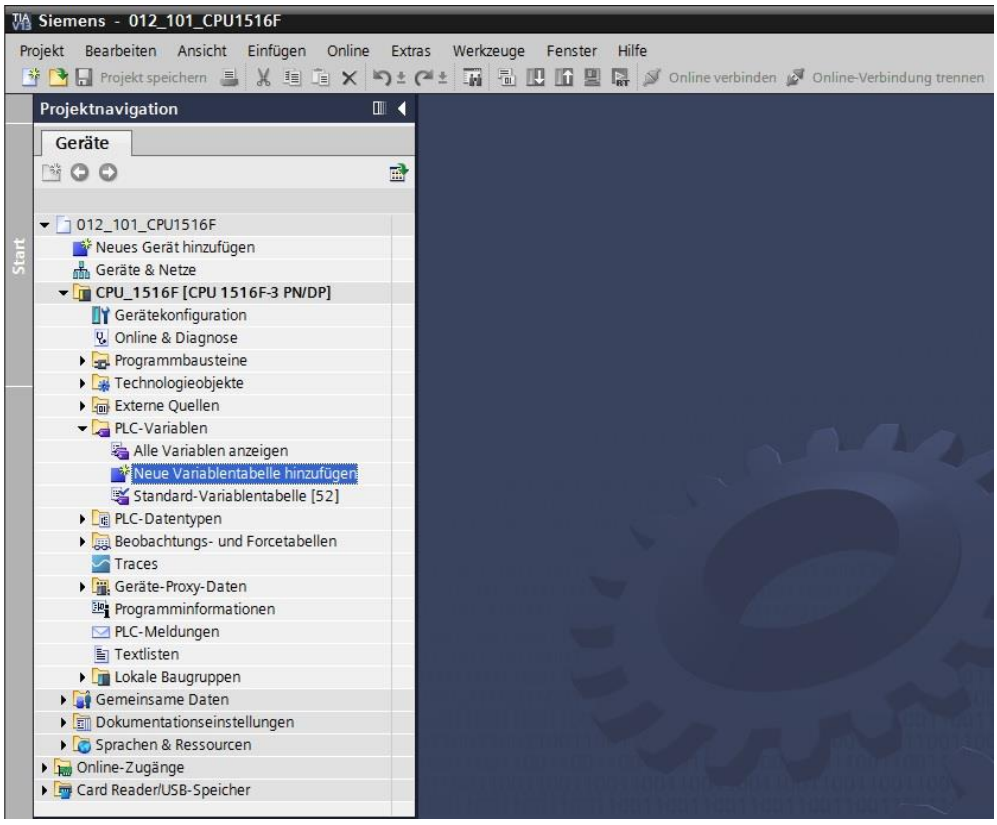


→ Als nächstes kann das Zielverzeichnis ausgewählt werden, in welches das dearchivierte Projekt gespeichert werden soll. Bestätigen Sie Ihre Auswahl mit „OK“. (→ Zielverzeichnis → OK)

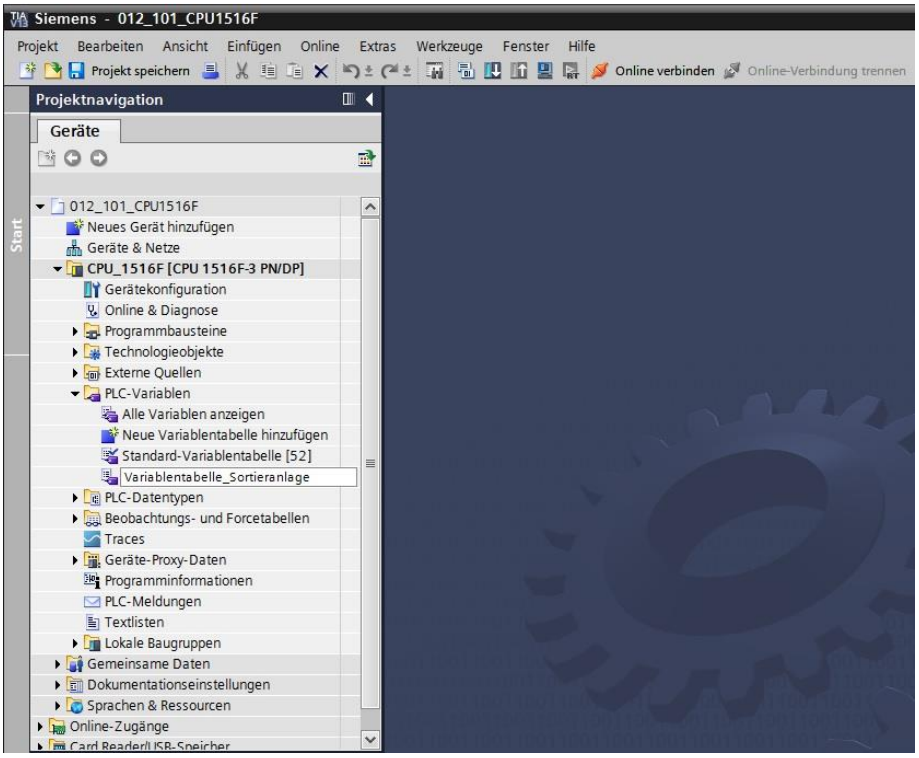


7.2 Anlegen einer neuen Variablentabelle

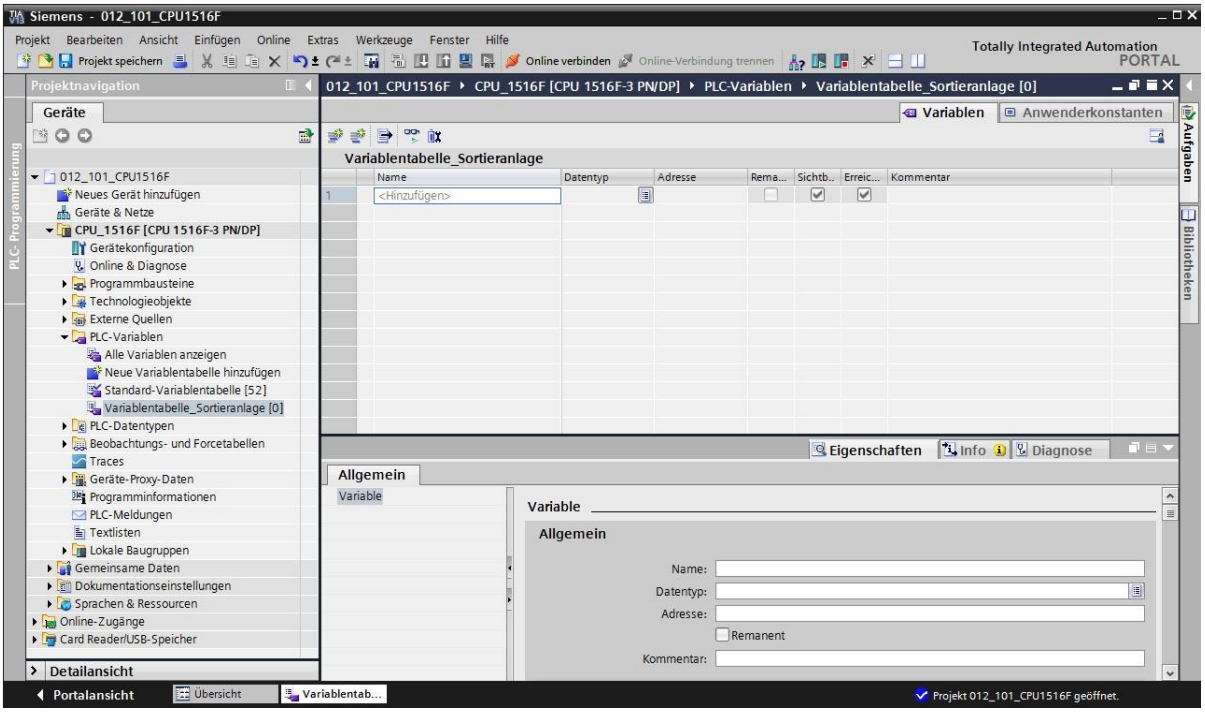
→ Navigieren Sie in der Projektansicht zu den → PLC-Variablen Ihrer Steuerung und erstellen Sie eine neue Variablentabelle, indem Sie auf → „Neue Variablentabelle hinzufügen“ doppelklicken.



→ Benennen Sie die soeben erstellte Variablentabelle in „Variablentabelle_Sortieranlage“ um. (→ Rechtsklick auf „Variablentabelle_1“ → „Umbenennen“ → Variablentabelle_Sortieranlage)

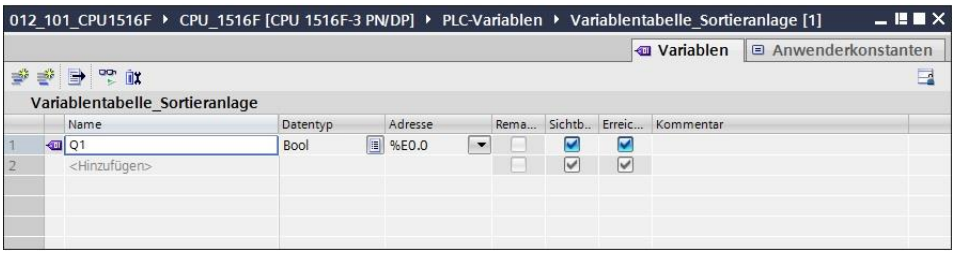


→ Öffnen Sie sie anschließend durch einen Doppelklick. (→ Variablentabelle_Sortieranlage)

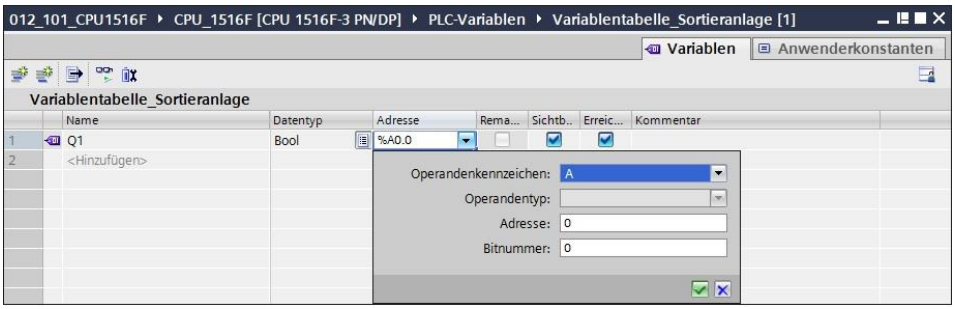


7.3 Anlegen neuer Variablen innerhalb einer Variablentabelle

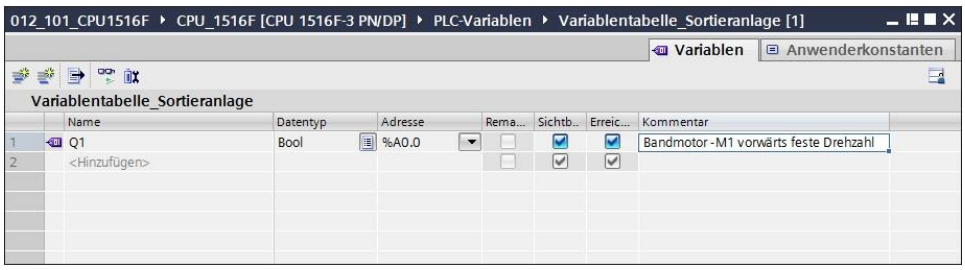
→ Fügen Sie den Namen Q1 hinzu und bestätigen Sie die Eingabe mit der Enter-Taste. Wenn Sie noch keine weiteren Variablen erstellt haben, hat TIA Portal nun automatisch den Datentyp „Bool“ und die Adresse %E0.0 (I 0.0) vergeben. (→ <Hinzufügen> → Q1 → Enter)



→ Ändern Sie die Adresse auf %A0.0 (Q0.0), indem Sie diese direkt eingeben oder per Klick auf den Dropdown-Pfeil das Menü zur Adressierung öffnen, dort das Operandenkennzeichen auf A ändern und mit Enter oder einem Klick auf das Häkchen bestätigen. (→ %E0.0 → Operationskennzeichen → A →)

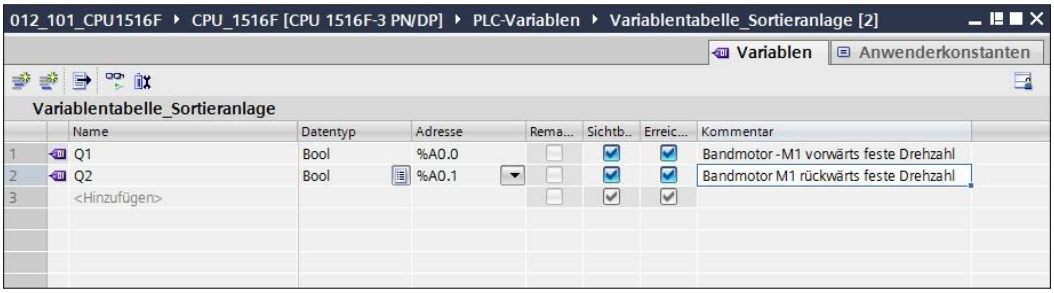


→ Vergeben Sie für die Variable den Kommentar „Bandmotor -M1 vorwärts feste Drehzahl“.



→ Fügen Sie in Zeile 2 eine neue Variable Q2 hinzu. TIA Portal hat automatisch denselben Datentyp wie in Zeile 1 vergeben und die Adresse um 1 hochgezählt auf %A0.1 (Q0.1). Geben Sie den Kommentar „Bandmotor M1 rückwärts feste Drehzahl“ ein.

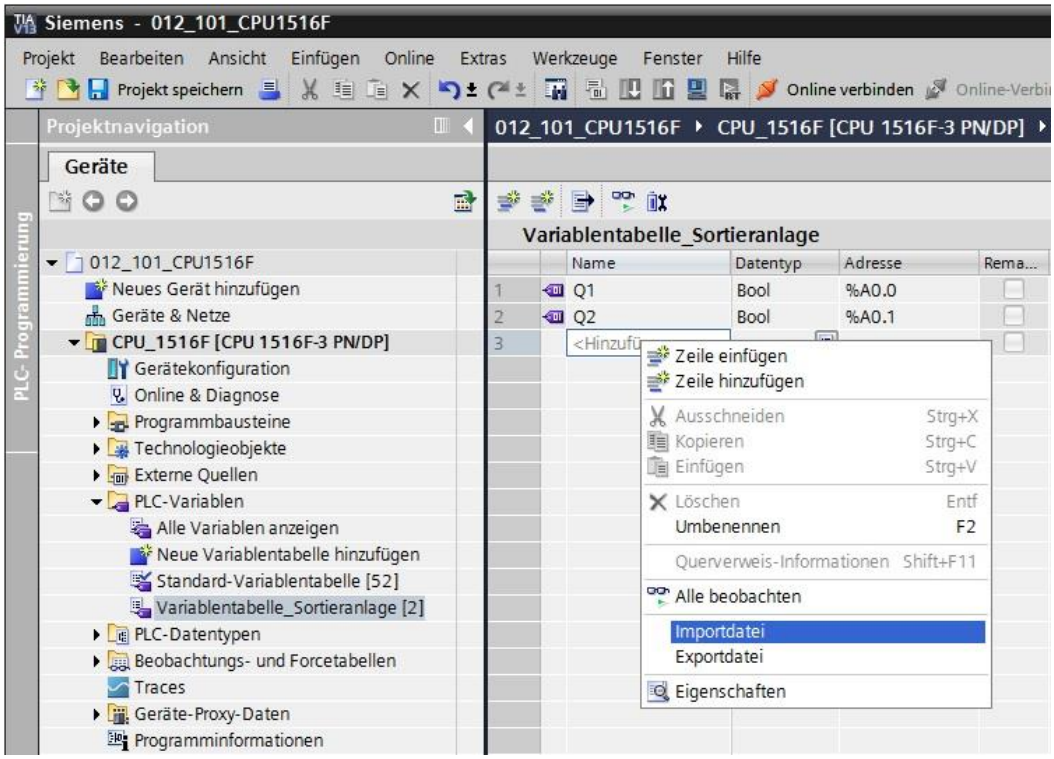
(→ <Hinzufügen> → Q2 → Enter → Kommentar → Bandmotor M1 rückwärts feste Drehzahl)



7.4 Importieren der „Variablentabelle_Sortieranlage“

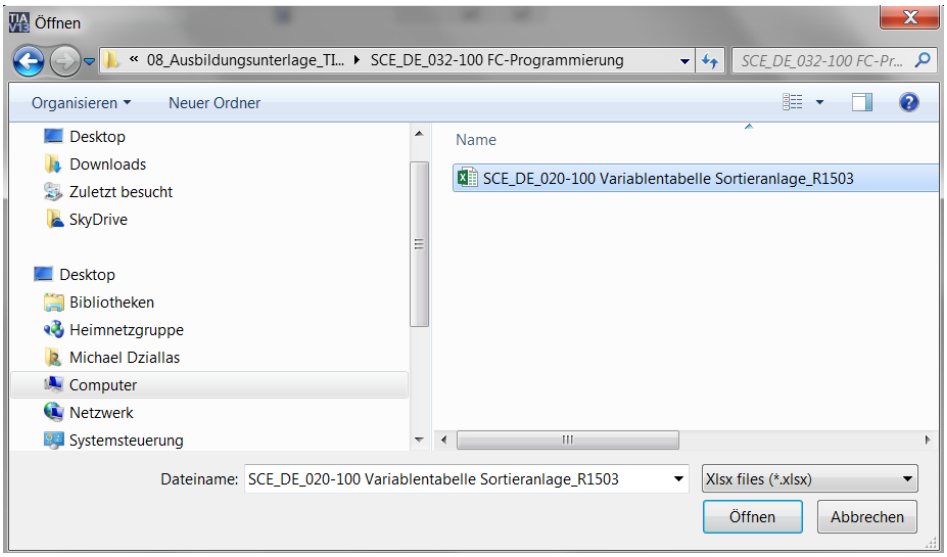
→ Zum Einfügen einer bereits vorhandenen Symboltabelle klicken Sie mit der rechten Maustaste auf ein leeres Feld der angelegten „Variablentabelle_Sortieranlage“. Im Kontextmenü wählen Sie „Importdatei“ aus.

(→ Rechtsklick in ein leeres Feld der Variablentabelle → Importdatei)



→ Wählen Sie die gewünschte Symboltabelle aus (z.B. im .xlsx-Format) und bestätigen die Auswahl mit „Öffnen“.

(→ SCE_DE_020-100_Variablentabelle_Sortieranlage… → Öffnen)

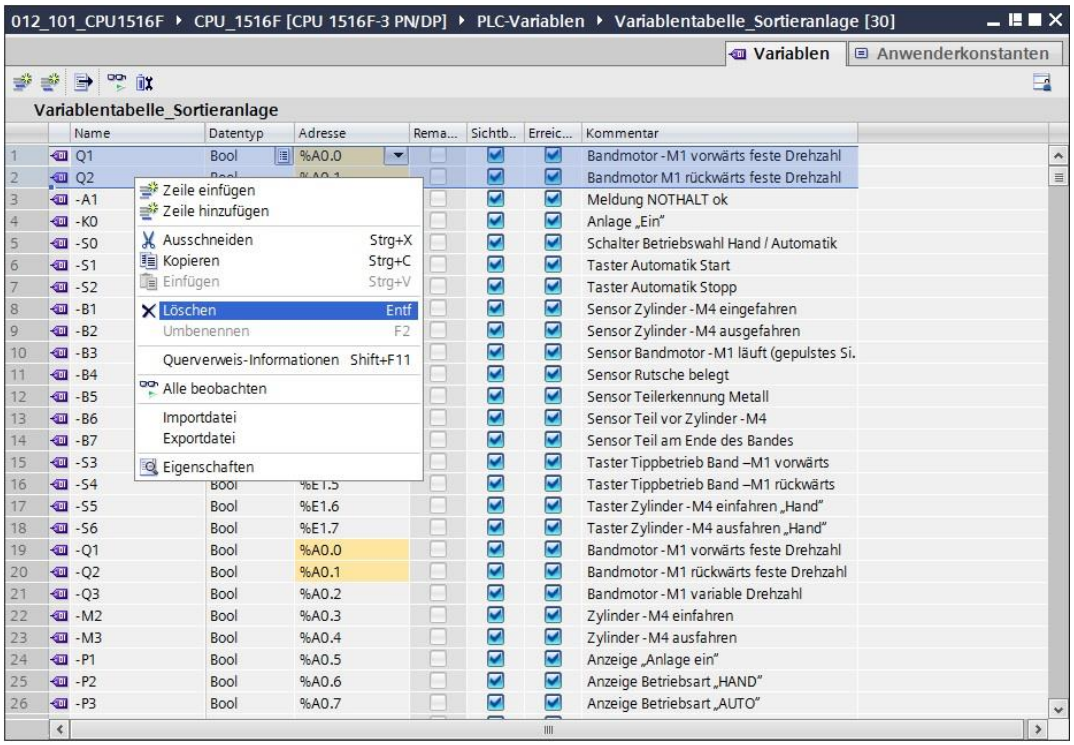


→ Ist der Import abgeschlossen erhalten Sie ein Bestätigungsfenster mit der Möglichkeit sich die Protokolldatei zum Import anzusehen. Klicken Sie hier auf → OK.

→ Sie werden feststellen, dass einige Adressen orange hervorgehoben wurden. Diese sind doppelt vorhanden und die Namen der zugehörigen Variablen wurden automatisch nummeriert, um Uneindeutigkeiten zu vermeiden.

→ Löschen Sie die doppelt vorhandenen Variablen, indem Sie die Zeilen markieren und die Taste Entf auf ihrer Tastatur drücken oder im Kontextmenü den Punkt Löschen auswählen.

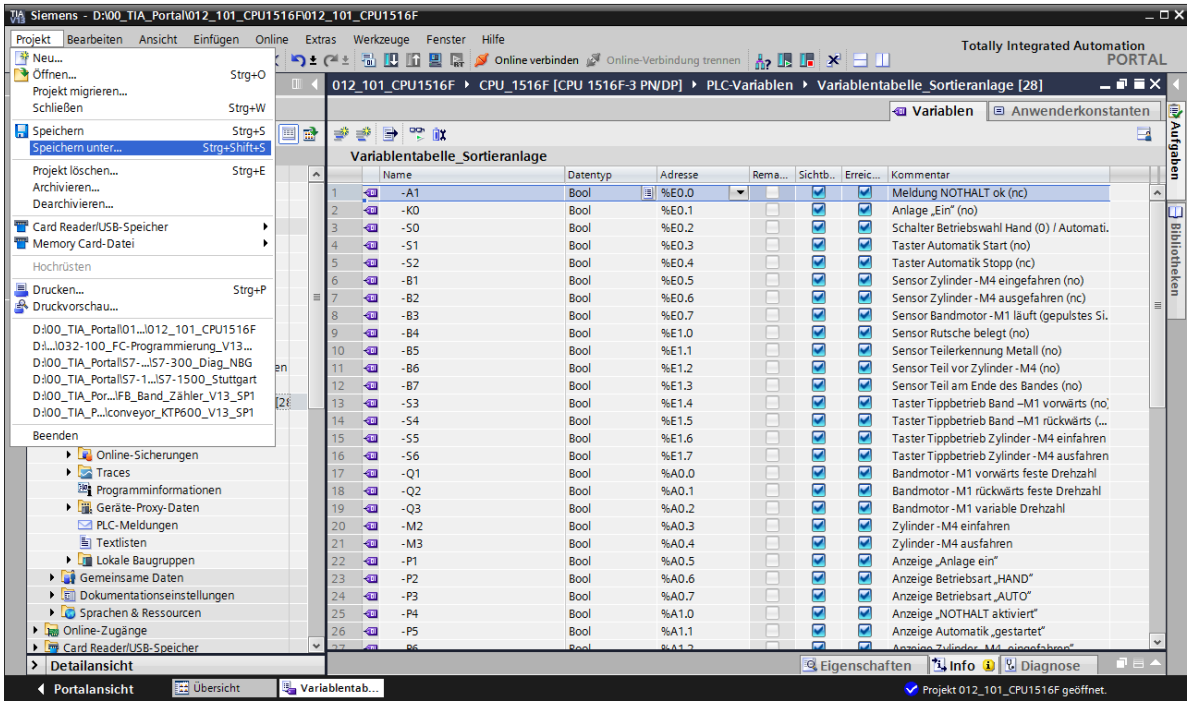
(→ Rechtsklick auf markierte Variablen → Löschen)

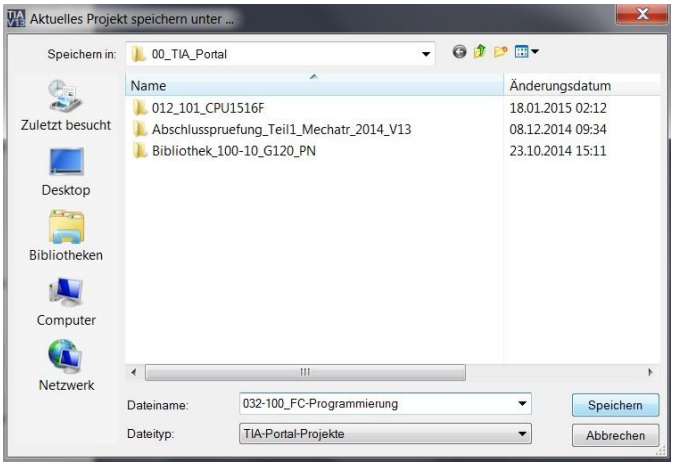


	Name	Datentyp	Adresse	Rema...	Sichtb..	Erreic...	Kommentar
1	Q1	Bool	%A0.0		☑	☑	Bandmotor -M1 vorwärts feste Drehzahl
2	Q2				☑	☑	Bandmotor M1 rückwärts feste Drehzahl
3	-A1				☑	☑	Meldung NOTHALT ok
4	-K0				☑	☑	Anlage „Ein"
5	-S0				☑	☑	Schalter Betriebswahl Hand / Automatik
6	-S1				☑	☑	Taster Automatik Start
7	-S2				☑	☑	Taster Automatik Stopp
8	-B1				☑	☑	Sensor Zylinder -M4 eingefahren
9	-B2				☑	☑	Sensor Zylinder -M4 ausgefahren
10	-B3				☑	☑	Sensor Bandmotor -M1 läuft (gepulstes Si.
11	-B4				☑	☑	Sensor Rutsche belegt
12	-B5				☑	☑	Sensor Teilerkennung Metall
13	-B6				☑	☑	Sensor Teil vor Zylinder -M4
14	-B7				☑	☑	Sensor Teil am Ende des Bandes
15	-S3				☑	☑	Taster Tippbetrieb Band –M1 vorwärts
16	-S4				☑	☑	Taster Tippbetrieb Band –M1 rückwärts
17	-S5	Bool	%E1.6		☑	☑	Taster Zylinder -M4 einfahren „Hand"
18	-S6	Bool	%E1.7		☑	☑	Taster Zylinder -M4 ausfahren „Hand"
19	-Q1	Bool	%A0.0		☑	☑	Bandmotor -M1 vorwärts feste Drehzahl
20	-Q2	Bool	%A0.1		☑	☑	Bandmotor -M1 rückwärts feste Drehzahl
21	-Q3	Bool	%A0.2		☑	☑	Bandmotor -M1 variable Drehzahl
22	-M2	Bool	%A0.3		☑	☑	Zylinder -M4 einfahren
23	-M3	Bool	%A0.4		☑	☑	Zylinder -M4 ausfahren
24	-P1	Bool	%A0.5		☑	☑	Anzeige „Anlage ein"
25	-P2	Bool	%A0.6		☑	☑	Anzeige Betriebsart „HAND"
26	-P3	Bool	%A0.7		☑	☑	Anzeige Betriebsart „AUTO"

→ Sie haben nun eine vollständige Symboltabelle der digitalen Ein- und Ausgänge vor sich. Speichern Sie Ihr Projekt nun unter dem Namen 032-100_FC-Programmierung.

(→ Projekt → Speichern unter … → 032-100_FC-Programmierung → Speichern)



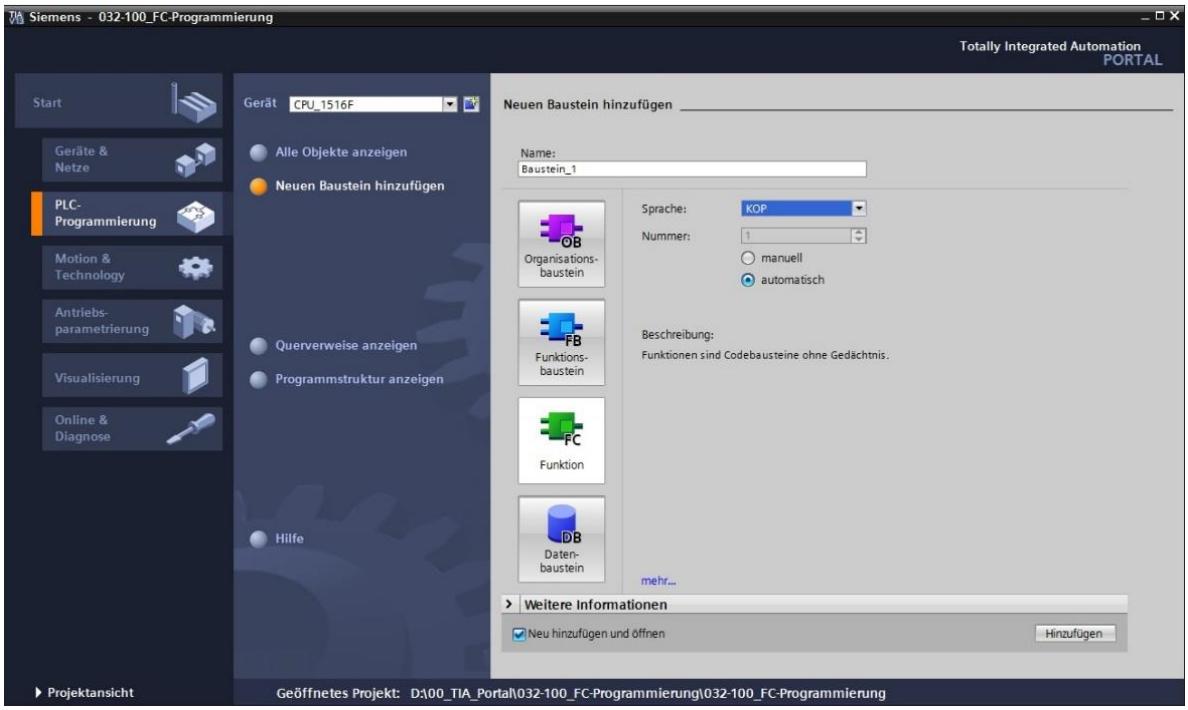


7.5 Erstellen der Funktion FC1 „MOTOR_HAND“ für den Bandmotor im Tippbetrieb

→ Klicken Sie in der Portalansicht im Abschnitt PLC-Programmierung auf „Neuen Baustein hinzufügen“ um dort eine neue Funktion anzulegen.

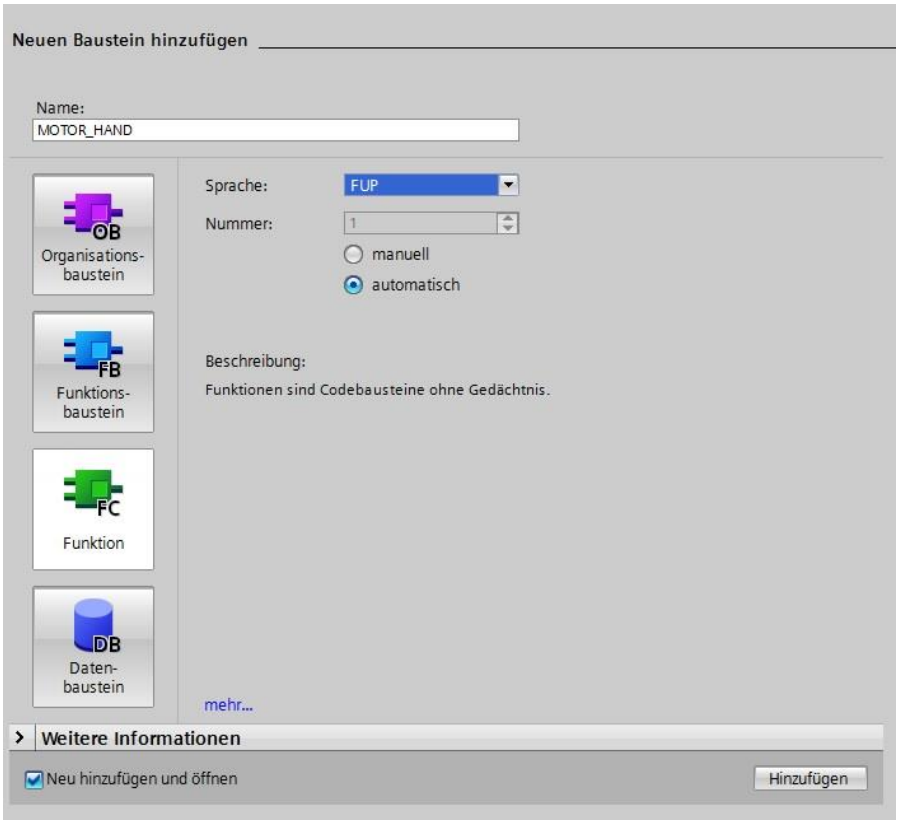
(→ PLC-Programmierung → Neuen Baustein hinzufügen → 

)



→ Benennen Sie Ihren neuen Baustein mit dem Name: „MOTOR_HAND“, stellen Sie die Sprache auf FUP und lassen Sie die Nummer automatisch vergeben. Aktivieren Sie das Häkchen „Neu hinzufügen und öffnen“, so gelangen Sie automatisch in der Projektansicht in Ihren erstellten Funktionsbaustein. Klicken Sie nun auf „Hinzufügen“.

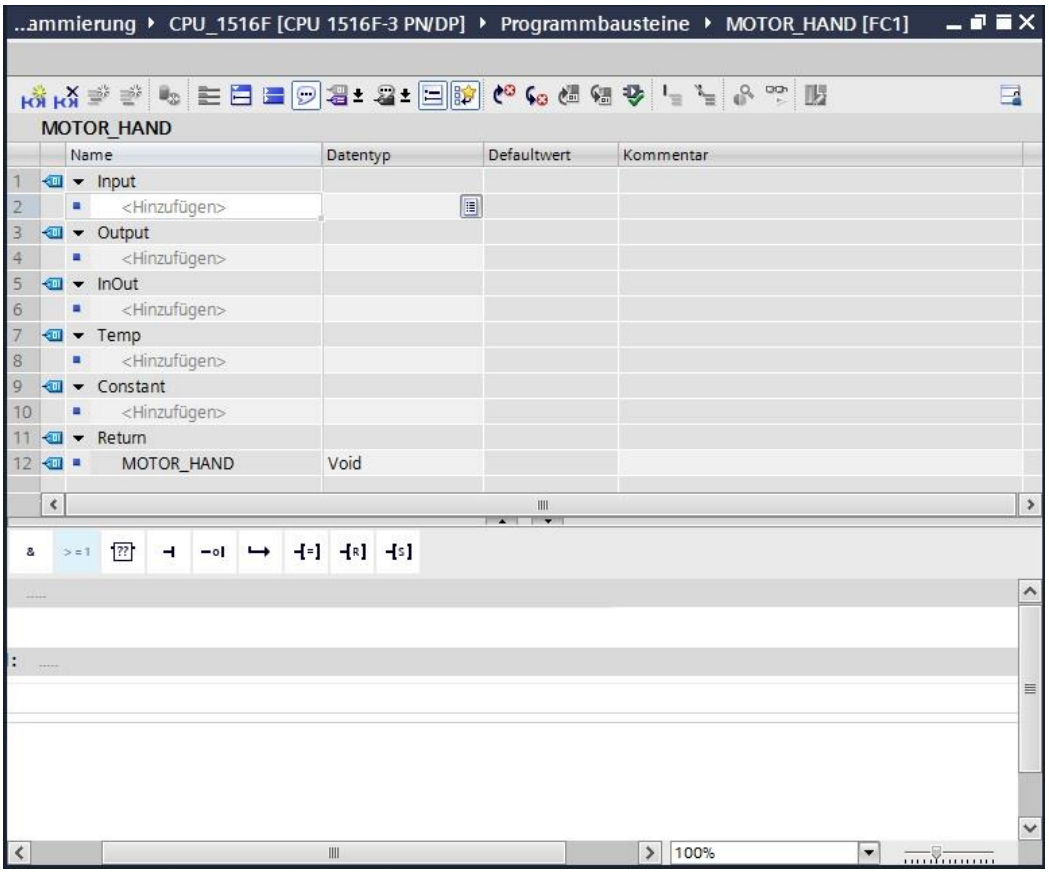
(→ Name: MOTOR_HAND→ Sprache: FUP → Nummer: automatisch → ☑ Neu hinzufügen und öffnen → Hinzufügen)



7.6 Schnittstelle der Funktion FC1 „MOTOR_HAND“ festlegen

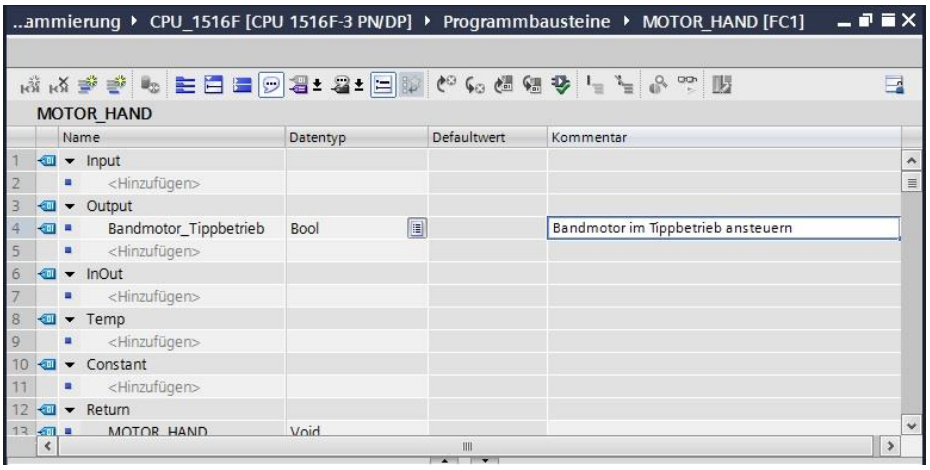
→ Haben Sie „Neu hinzufügen und öffnen“ angeklickt, öffnet sich die Projektansicht mit einem Fenster zum Erstellen des eben angelegten Bausteins.

→ Im oberen Abschnitt ihrer Programmieransicht finden Sie die Schnittstellenbeschreibung Ihrer Funktion.



→ Zur Ansteuerung des Bandmotors wird ein binäres Ausgangssignal benötigt. Deshalb legen wir zuerst die lokale Output- Variable #Bandmotor_Tippbetrieb vom Typ „Bool“ an. Zu dem Parameter vergeben Sie sie den Kommentar „Bandmotor im Tippbetrieb ansteuern“.

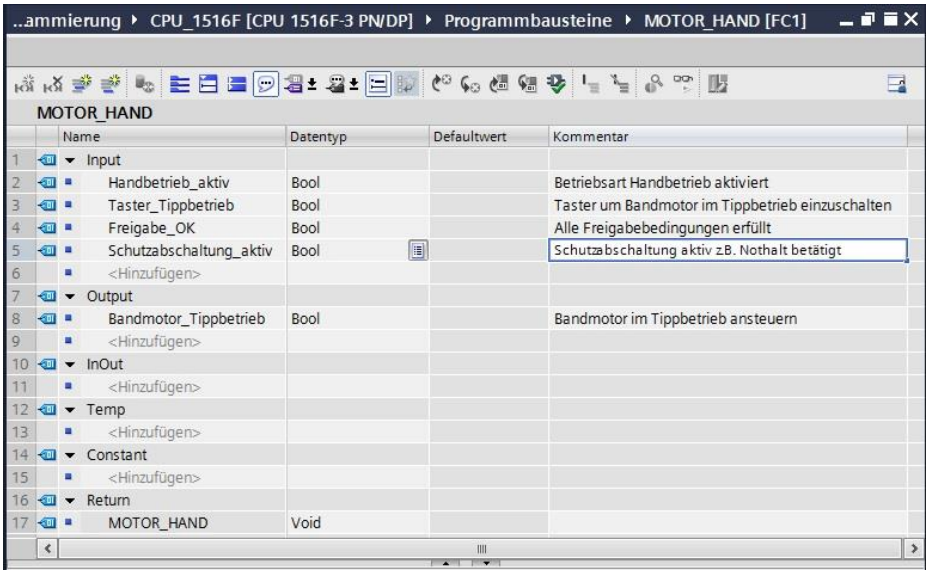
(→ Output: Bandmotor_Tippbetrieb → Bool → Bandmotor im Tippbetrieb ansteuern)



→ Fügen Sie als Eingangsschnittstelle unter Input zuerst den Parameter #Handbetrieb_aktiv hinzu und bestätigen Sie die Eingabe mit der Enter-Taste oder indem Sie das Eingabefeld verlassen. Es wird automatisch der Datentyp „Bool“ vergeben. Dieser wird beibehalten. Geben Sie anschließend den zugehörigen Kommentar „Betriebsart Handbetrieb aktiviert“ ein.

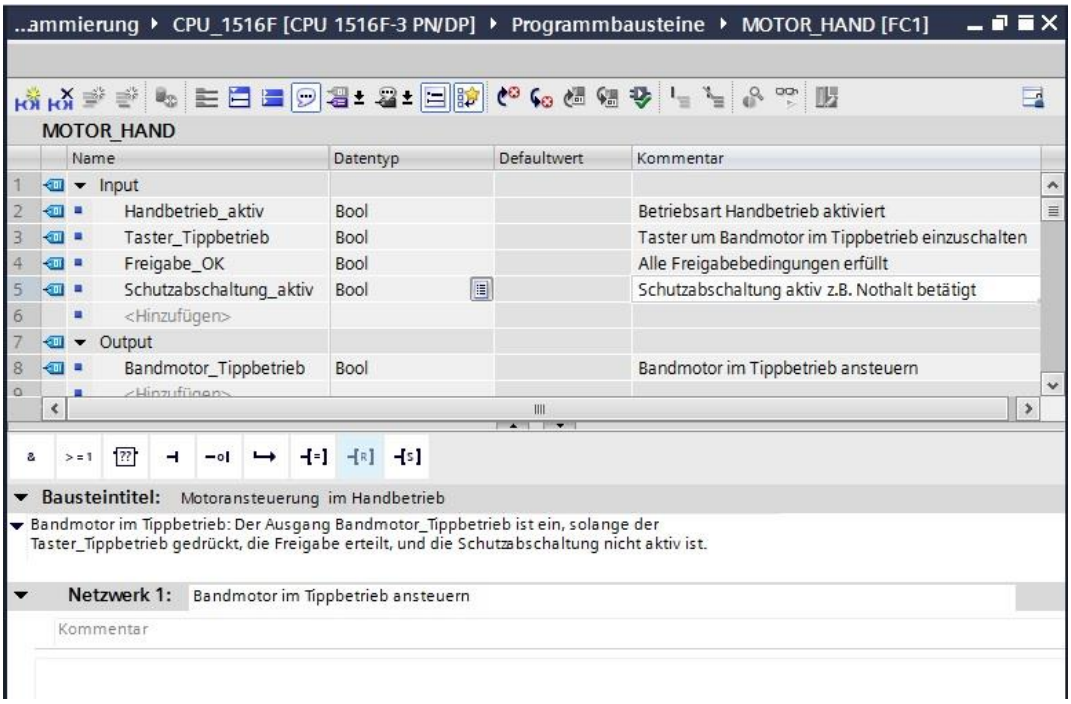
(→ Handbetrieb_aktiv → Enter → Bool → Betriebsart Handbetrieb aktiviert)

→ Fügen Sie unter Input als weitere binäre Eingangsparameter #Taster_Tippbetrieb, #Freigabe_OK und #Schutzabschaltung_aktiv hinzu und überprüfen Sie deren Datentypen. Ergänzen Sie mit sinnvollen Kommentaren.



→ Vergeben Sie zur Programmdokumentation den Bausteintitel, einen Bausteinkommentar und für das Netzwerk 1 einen hilfreichen Netzwerktitel.

(→ Bausteintitel: Motoransteuerung im Handbetrieb → Netzwerk 1: Bandmotor im Tippbetrieb ansteuern)

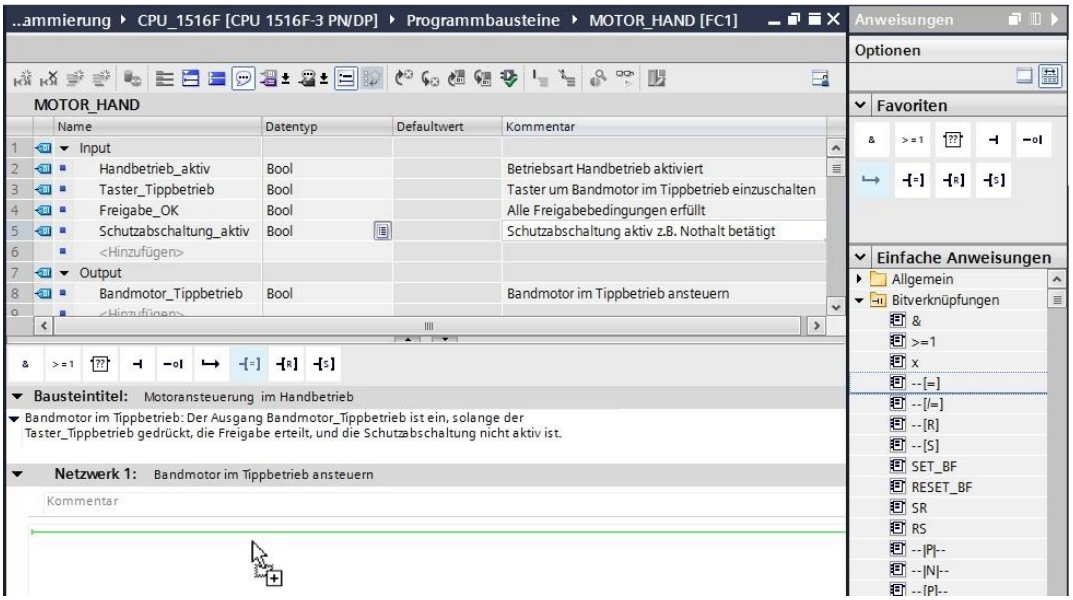


7.7 Programmierung des FC1: MOTOR_HAND

→ Unterhalb der Schnittstellenbeschreibung sehen Sie in dem Programmierfenster eine Symbolleiste mit verschiedenen Logikfunktionen und darunter einen Bereich mit Netzwerken. Dort haben wir bereits den Bausteintitel und den Titel für das erste Netzwerk festgelegt. Innerhalb der Netzwerke erfolgt die Programmierung unter Verwendung einzelner Logikbausteine. Die Aufteilung auf mehrere Netzwerke dient dabei der Wahrung der Übersichtlichkeit. Die verschiedenen Möglichkeiten, Logikbausteine einzufügen, werden sie im Folgenden kennenlernen.

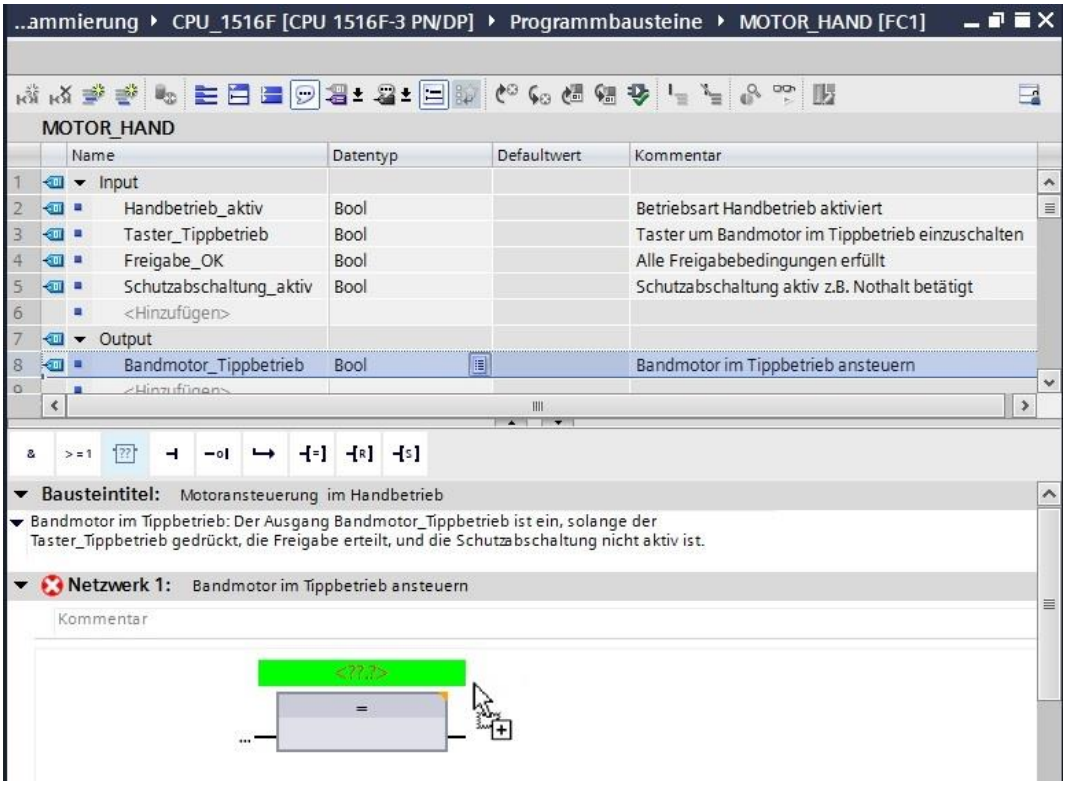
→ Auf der rechten Seite ihres Programmierfensters sehen Sie eine Liste von Anweisungen, die Sie im Programm verwenden können. Suchen Sie unter → Einfache Anweisungen → Bitverknüpfungen nach der Funktion –[=] (Zuweisung) und ziehen Sie diese per Drag and Drop in ihr Netzwerk 1 (grüne Linie erscheint, Mauszeiger mit + Symbol).

(→ Anweisungen → Einfache Anweisungen → Bitverknüpfung → –[=])



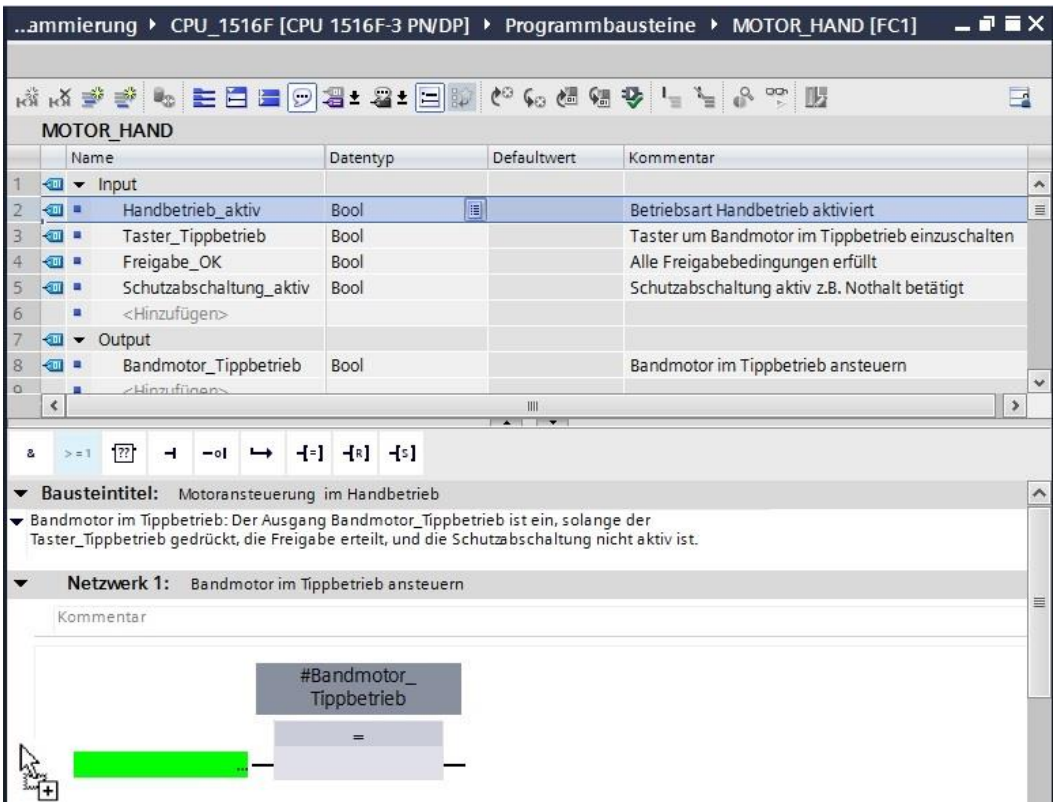
→ Ziehen Sie nun Ihren Output-Parameter #Bandmotor_Tippbetrieb per Drag and Drop auf <??.?> über ihrem soeben eingefügten Block. Sie können einen Parameter in der Schnittstellenbeschreibung am besten anwählen, indem Sie ihn an dem blauen Symbol anfassen.

(→ Bandmotor_Tippbetrieb)



→ Dadurch wird bestimmt, dass der Parameter #Bandmotor_Tippbetrieb durch diesen Block geschrieben wird. Es fehlen allerdings noch die Eingangs-Bedingungen, damit dies auch tatsächlich geschieht. Ziehen Sie dazu den Input-Parameter #Handbetrieb_aktiv per Drag and Drop auf „…“ auf der linken Seite des Zuweisungs-Blocks.

(→ Handbetrieb_aktiv)

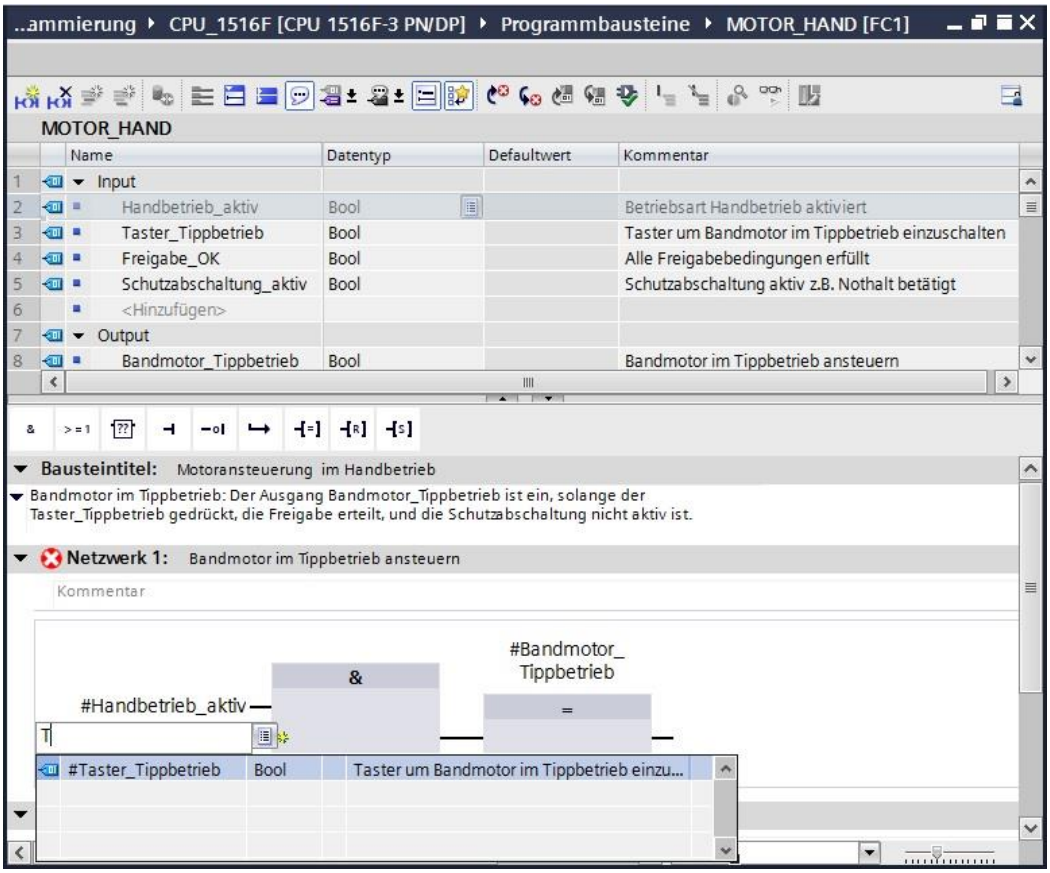


→ Der Eingang des Zuweisungs-Blocks soll zusätzlich mit weiteren Parametern UND-verknüpft werden. Klicken Sie dazu zunächst auf den Eingang des Blocks, an dem bereits #Handbetrieb_aktiv verschaltet ist, so dass der Eingangsstrich blau hinterlegt ist.

→ Klicken Sie auf das Symbol & in Ihrer Logik-Symbolleiste, um eine UND-Verknüpfung zwischen der Variable #Handbetrieb_aktiv und ihrem Zuweisungs-Baustein einzufügen.

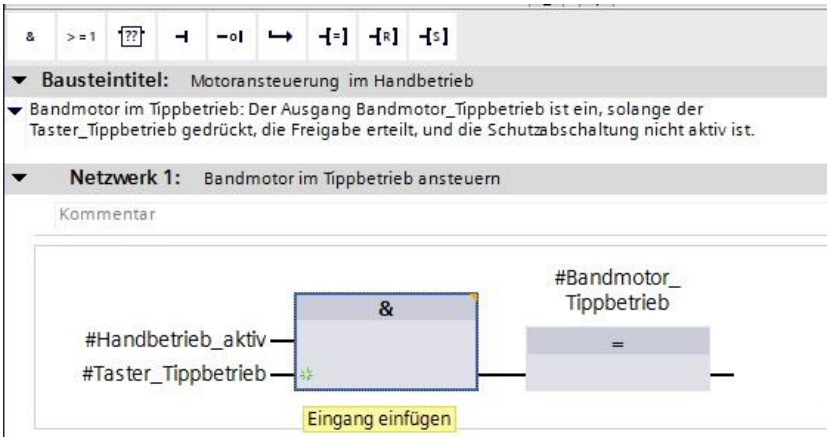
→ Klicken Sie doppelt auf den zweiten Eingang der &-Verknüpfung <??.?> und geben Sie im daraufhin erscheinenden Feld den Buchstaben „T“ ein, um eine Liste der verfügbaren Variablen, die mit „T“ beginnen, zu sehen. Klicken Sie auf die Variable #Taster_Tippbetrieb und übernehmen Sie mit → Enter.

(→ &-Block → <??.?> → T → #Taster_Tippbetrieb → Enter)

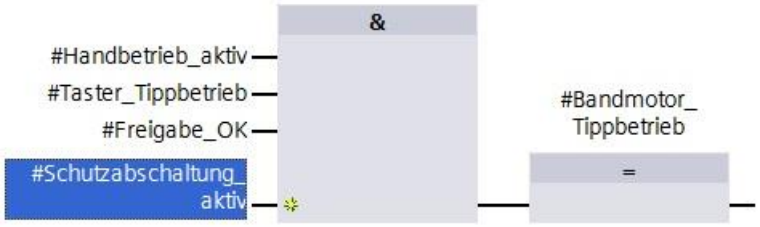


Hinweis: *Bei dieser Variante der Variablenzuordnung besteht die Gefahr einer Verwechslung mit den globalen Variablen aus der Variablentabelle. Deshalb sollte die vorher gezeigte Variante mit Drag and Drop aus der Schnittstellenbeschreibung bevorzugt werden.*

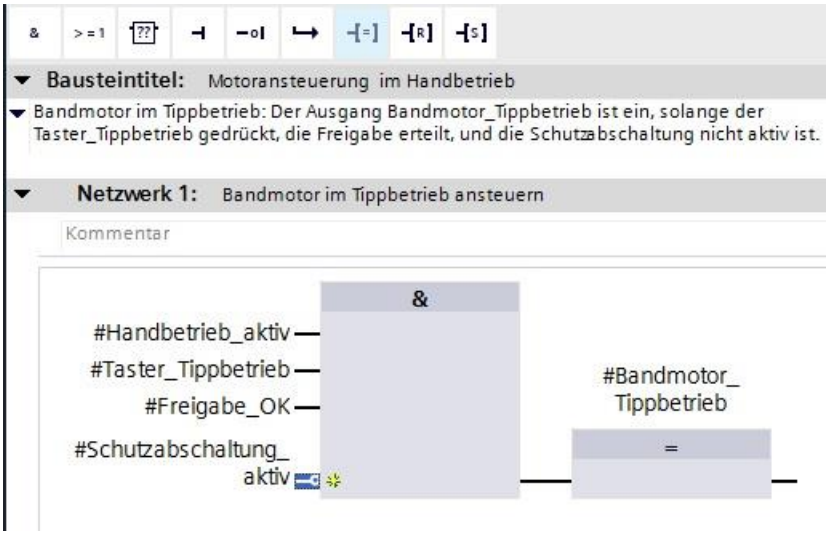
→ Damit der Ausgang nur angesteuert werden kann, wenn die Freigabe erteilt wurde und die Schutzabschaltung nicht aktiv ist, sollen zusätzlich die Eingangs- Variablen #Freigabe_OK und #Schutzabschaltung_aktiv mit dem UND verknüpft werden. Klicken Sie dazu zweimal auf den gelben Stern Ihres UND-Glieds um zwei weitere Eingänge hinzuzufügen.



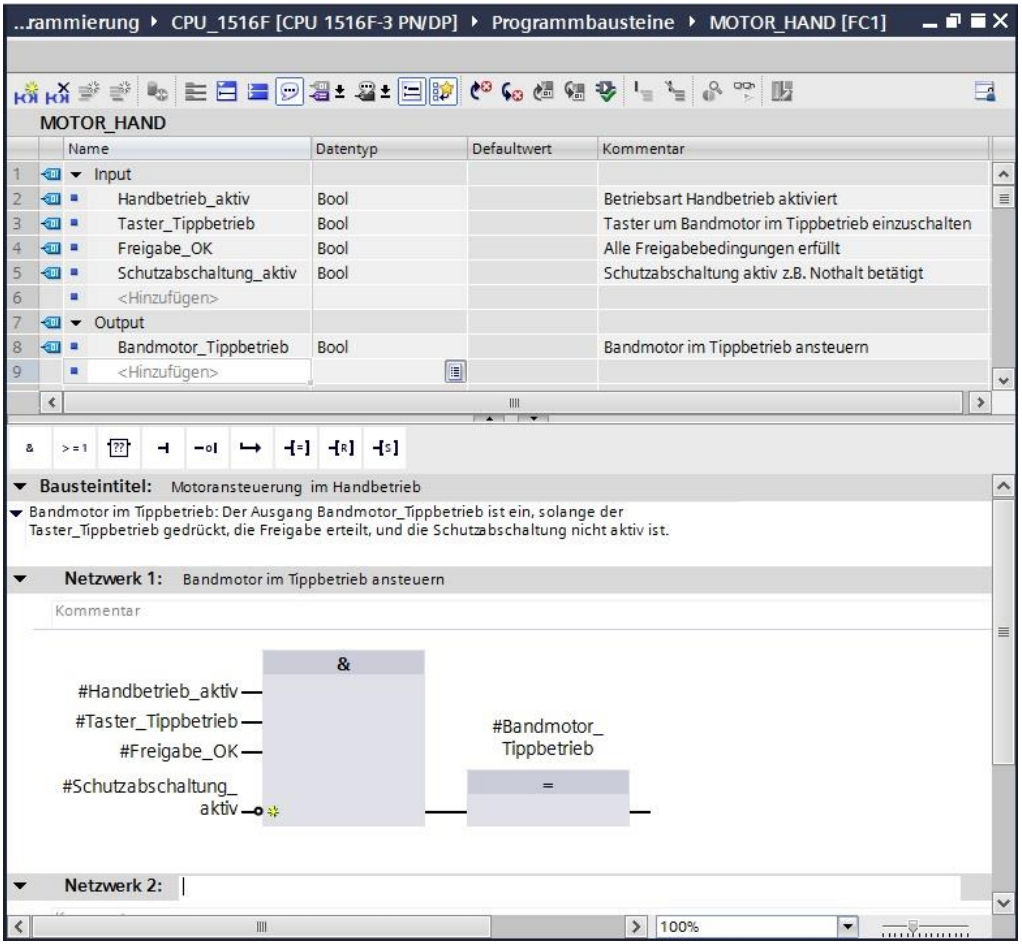
→ Fügen Sie an Ihren neu erstellten Eingängen des UND-Glieds die Eingangs- Variablen #Freigabe_OK und #Schutzabschaltung_aktiv hinzu.



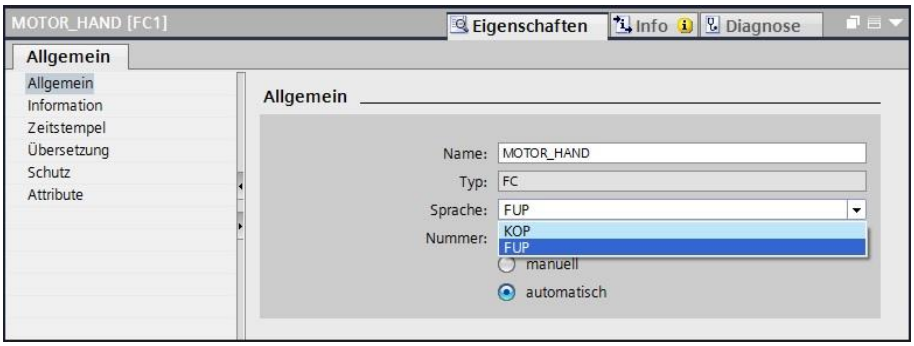
→ Negieren Sie den mit dem Parameter #Schutzabschaltung_aktiv beschalteten Eingang, indem Sie ihn markieren und anschließend auf klicken.



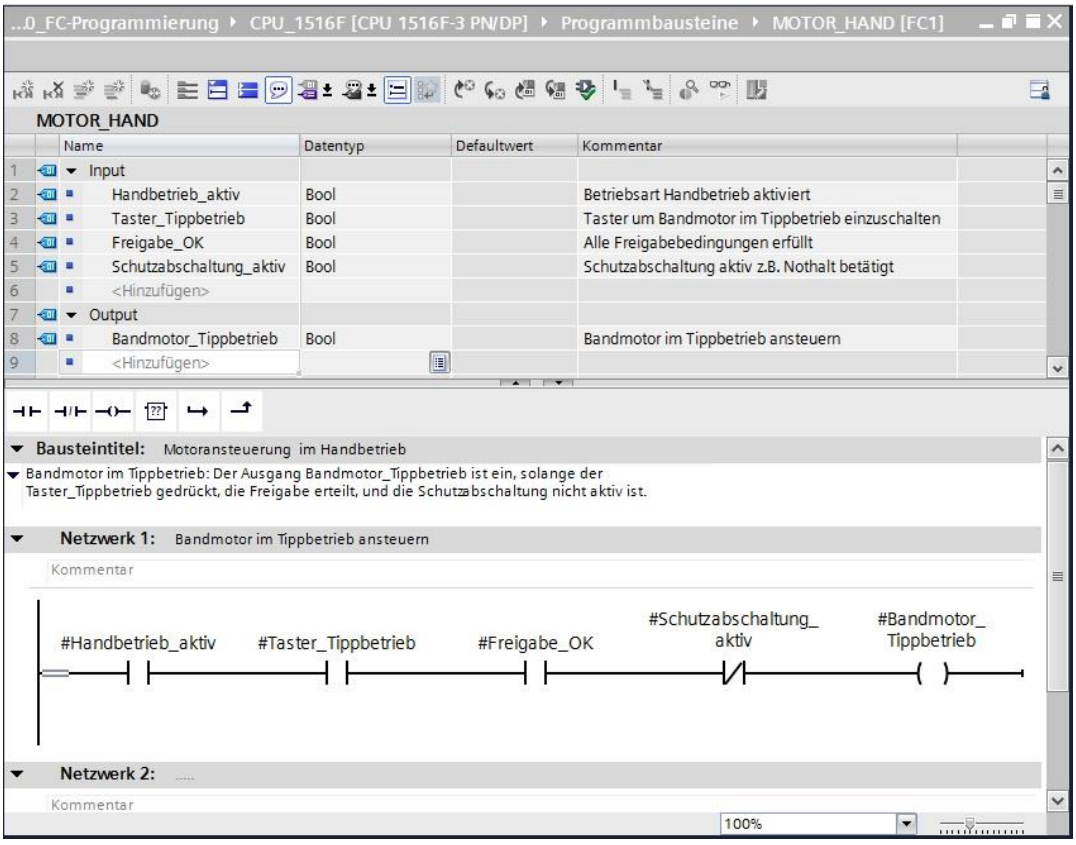
→ Vergessen Sie nicht auf Projekt speichern zu klicken. Die fertige Funktion „MOTOR_HAND [FC1] in FUP ist nachfolgend dargestellt.



→ Bei den Eigenschaften des Bausteins können Sie im Punkt „Allgemein“ die „Sprache“ auf KOP (Kontaktplan) umstellen. (→ Eigenschaften → Allgemein → Sprache: KOP)



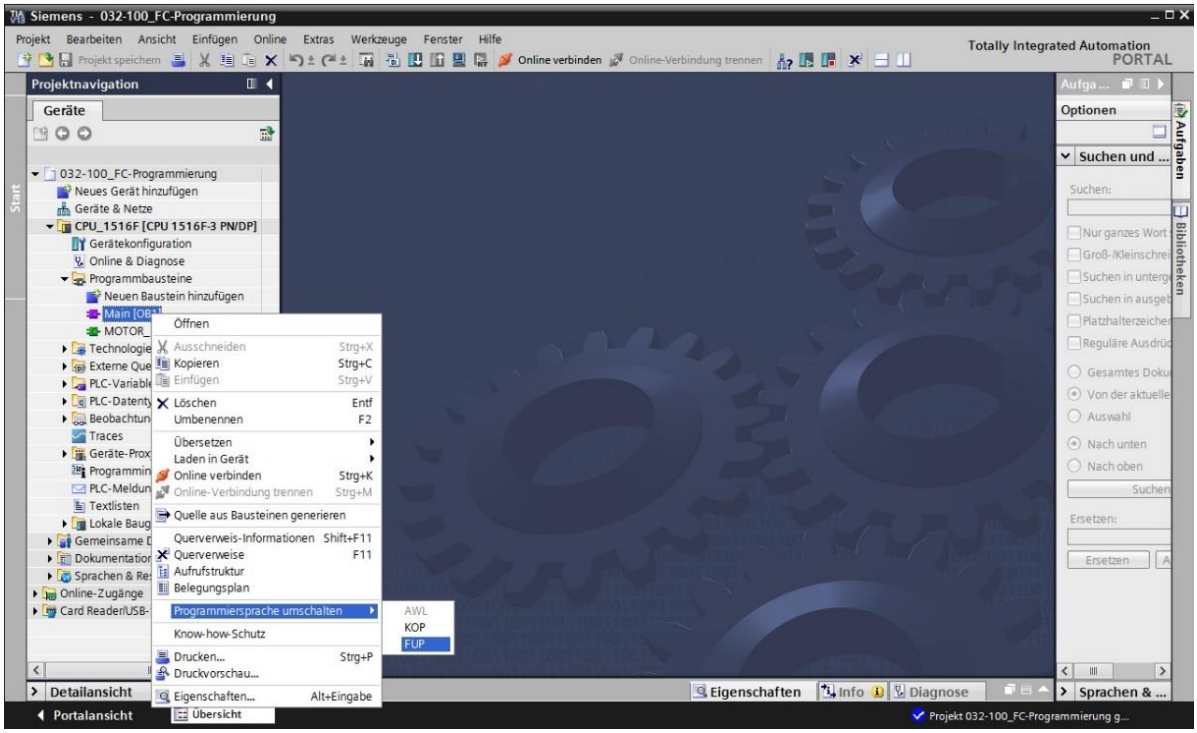
→ In KOP sieht das Programm wie folgt aus.



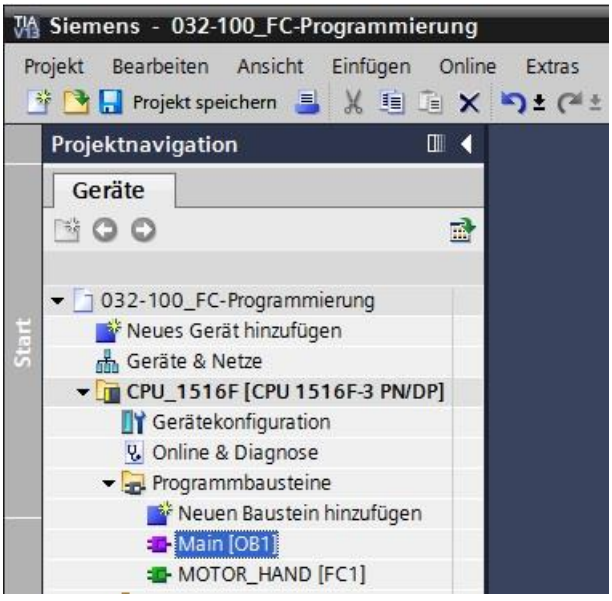
7.8 Programmierung des Organisationsbausteins OB1 – Steuerung des Bandlaufs vorwärts im Handbetrieb

→ Vor der Programmierung des Organisationsbausteins „Main[OB1]“ stellen wir dort die Programmiersprache auf FUP (Funktionsplan) um. Klicken Sie hierzu vorher mit der linken Maustaste im Ordner „Programmbausteine“ auf „Main[OB1)“.

(→ CPU_1516F[CPU 1516F-3 PN/DP → Programmbausteine → Main [OB1] → Programmiersprache umschalten → FUP)

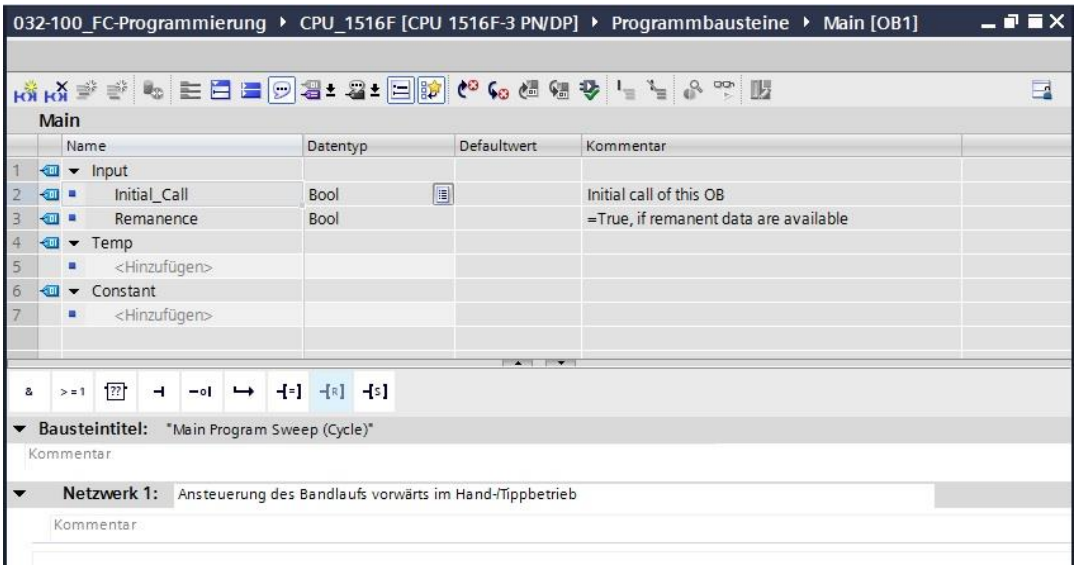


→ Öffnen Sie nun den Organisationsbaustein „Main [OB1]“ mit einem Doppelklick.

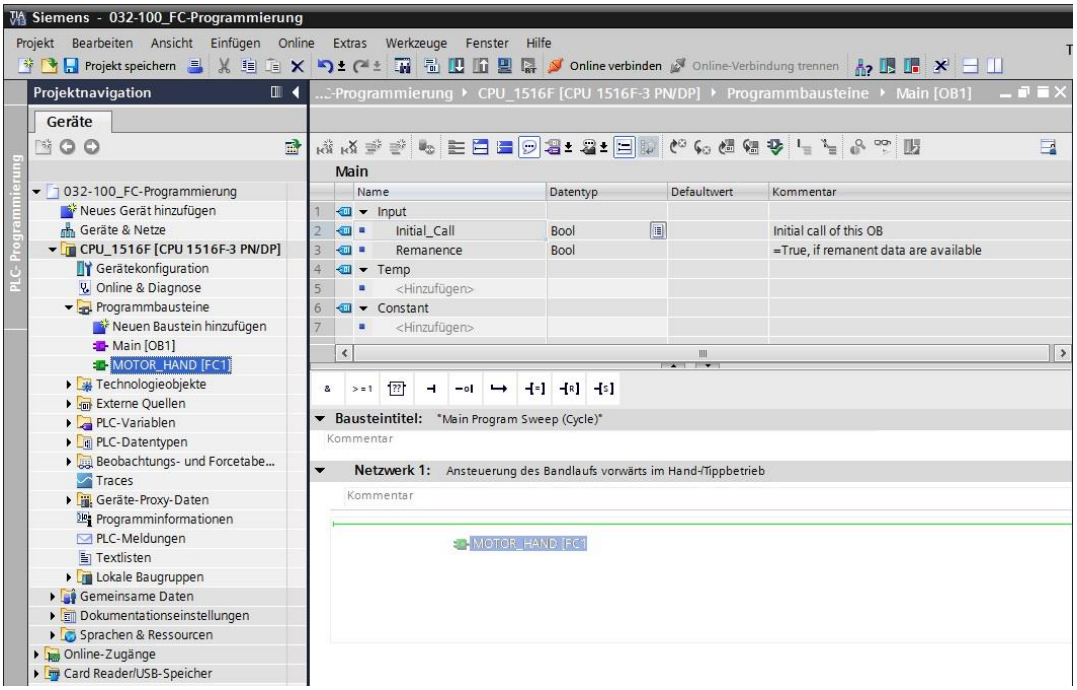


→ Geben Sie dem Netzwerk 1 den Namen „Ansteuerung des Bandlaufs vorwärts im Hand-/Tippbetrieb“.

(→ Netzwerk 1:… → Ansteuerung des Bandlaufs vorwärts im Hand-/Tippbetrieb)

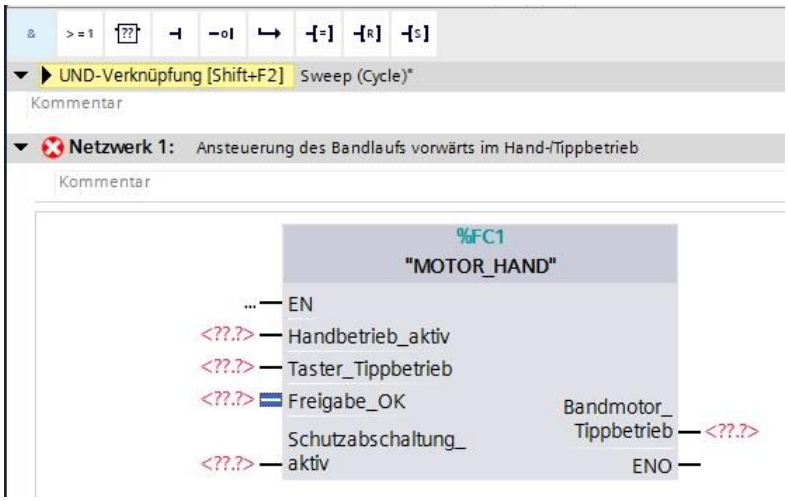


→ Ziehen Sie nun ihre Funktion „MOTOR_HAND [FC1]“ per Drag and Drop in das Netzwerk 1 auf die grüne Linie.

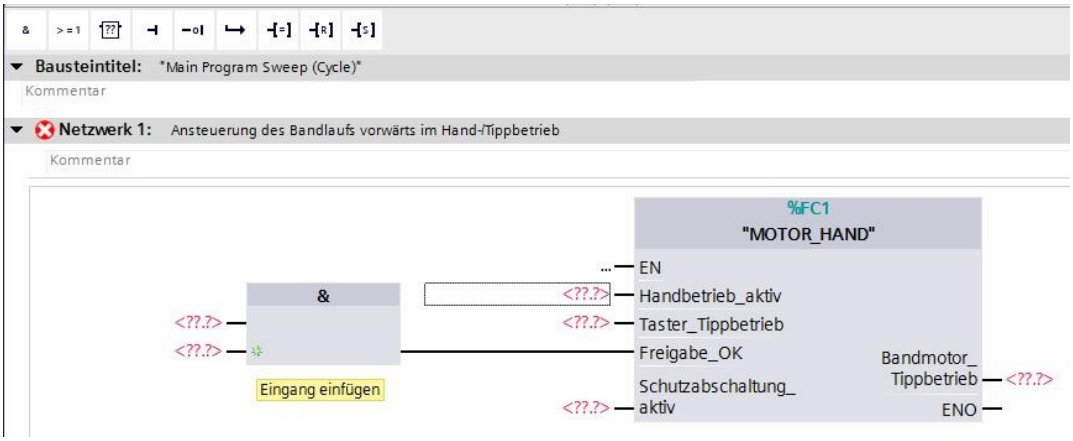


→ Es wird ein Block mit der von Ihnen festgelegten Schnittstelle und den Anschlüssen EN und ENO im Netzwerk 1 eingefügt.

→ Um ein UND vor dem Eingangsparameter „Freigabe_OK“ einzufügen, markieren Sie diesen Eingang und fügen das UND mit einem Klick auf das Symbol & in Ihrer Logik-Symbolleiste ein. (→ &)

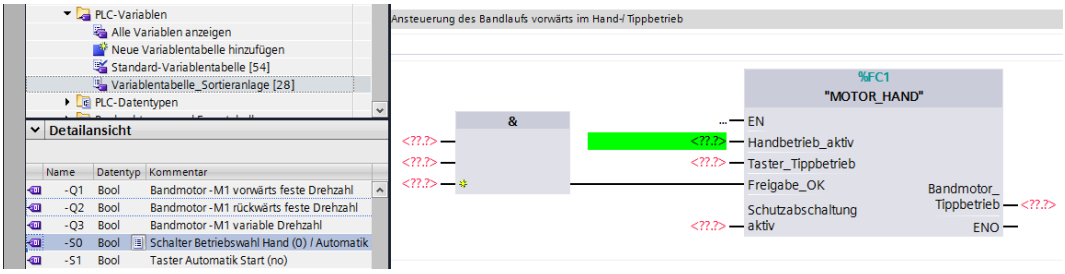


→ Klicken Sie auf den gelben Stern Ihres UND-Glieds um einen weiteren Eingang hinzuzufügen. (→)

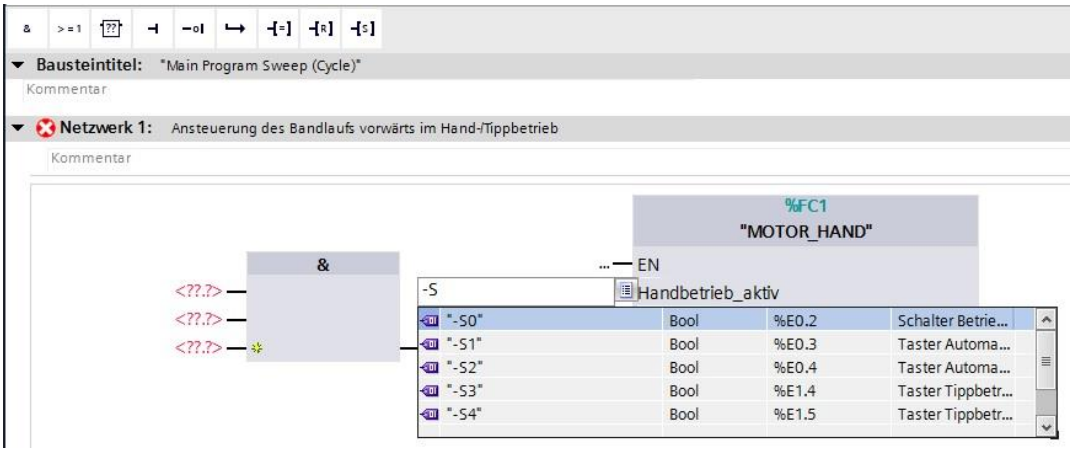


→ Um den Baustein mit den globalen Variablen aus der „Variablentabelle_Sortieranlage“ zu verschalten, haben wir 2 Möglichkeiten:

→ Entweder Sie markieren in der Projektnavigation die „Variablentabelle_Sortieranlage“ und ziehen dann die gewünschte globale Variable per Drag and Drop aus der Detailansicht auf die Schnittstelle des FC1 (→ Variablentabelle_Sortieranlage → Detailansicht → -S0 → Handbetrieb_aktiv)

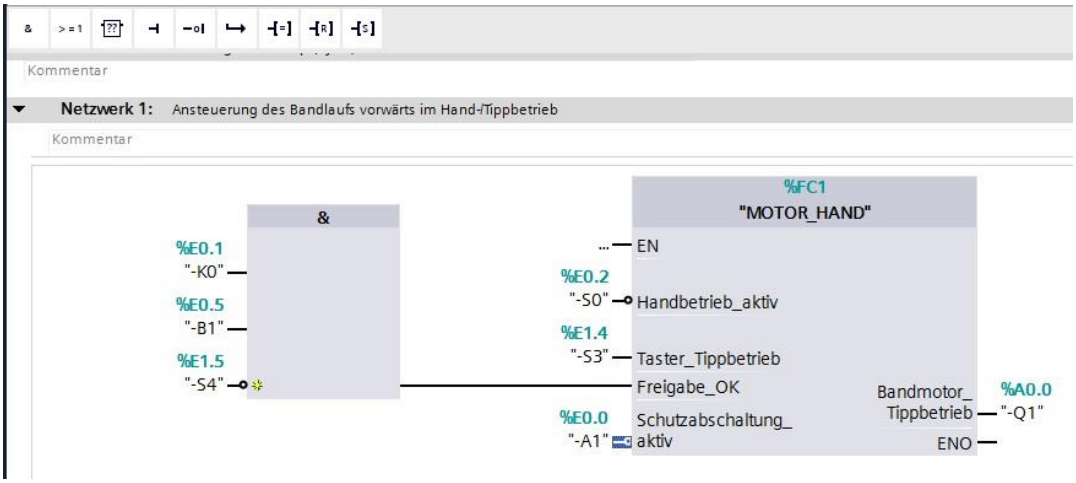


→ Oder Sie geben bei <??.?> die Anfangsbuchstaben (z.B.: „-S“) der gewünschten globalen Variable ein und wählen aus der eingeblendeten Liste die globale Eingangs-Variable „-S0“ (%E0.2) aus. (→ Handbetrieb_aktiv → -S → -S0)



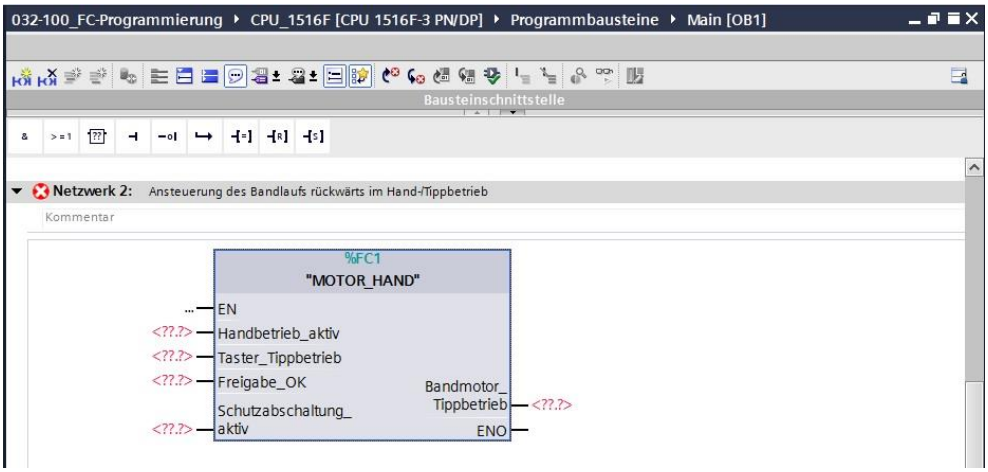
→ Fügen Sie die weiteren Eingangsvariablen „-S3“, „-K0“, „-B1“, „-S4“ und „-A1“ sowie am Ausgang „Bandmotor_Tippbetrieb“ die Ausgangsvariable „-Q1“ (%A0.0) ein.

→ Negieren Sie die Abfragen der Eingangsvariablen „-S4“ und „-A1“ indem Sie diese markieren und anschließend auf −o| klicken. (→ -S4 → −o| → -A1 → −o|)

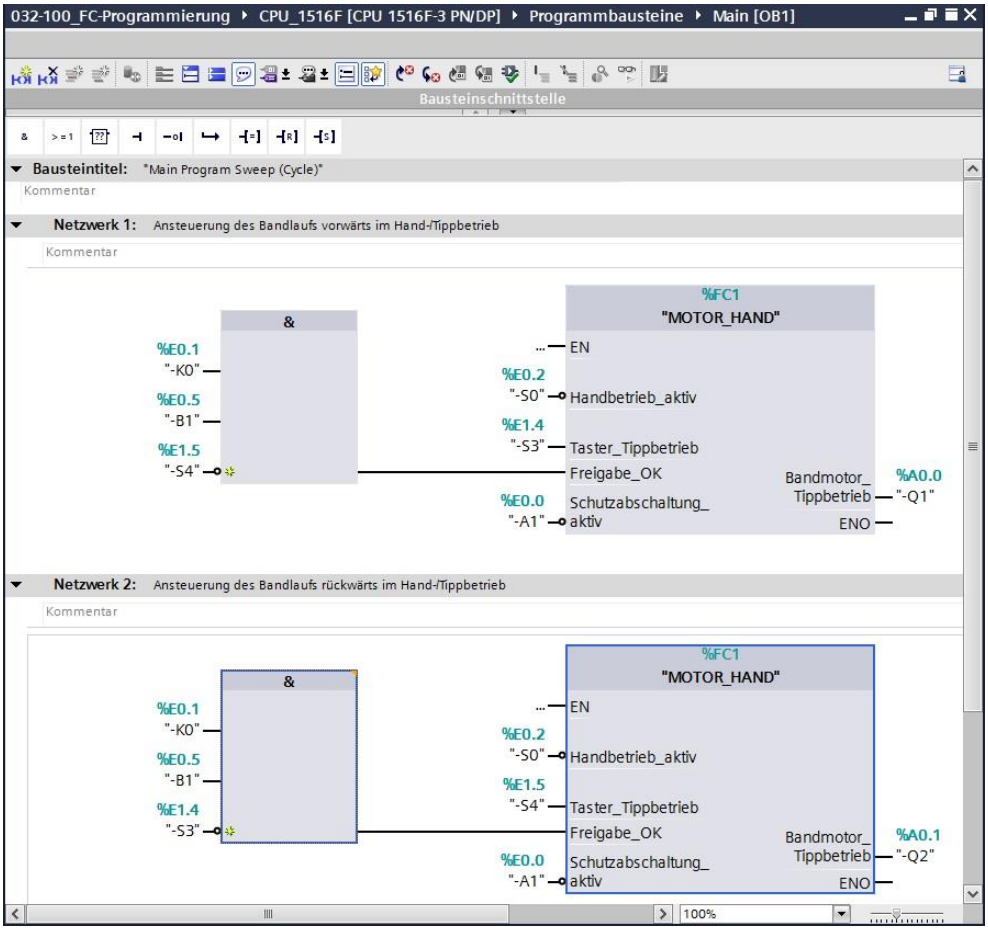


7.9 Programmierung des Organisationsbausteins OB1 – Steuerung des Bandlaufs rückwärts im Handbetrieb

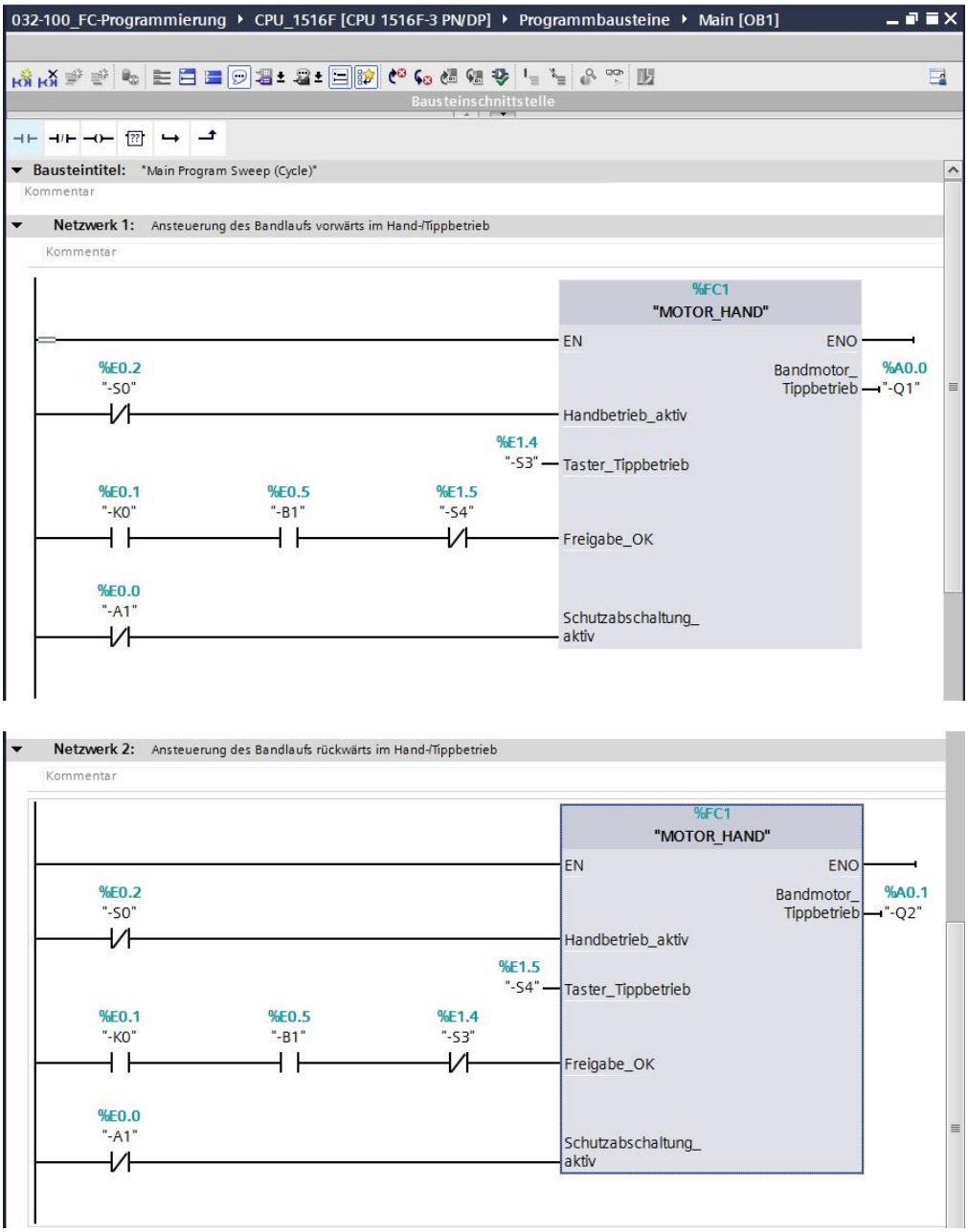
→ Geben Sie Netzwerk 2 den Namen „Ansteuerung des Bandlaufs rückwärts im Hand-/Tippbetrieb“ und fügen Sie wie bereits vorher in Netzwerk 1 ihre Funktion „MOTOR_HAND [FC1]“ per Drag and Drop ein.



→ Beschalten Sie Ihre Funktion so wie hier gezeigt. In der Programmiersprache FUP (Funktionsplan) erhalten Sie das folgende Ergebnis.



→ In der Programmiersprache KOP (Kontaktplan) sieht das Ergebnis folgendermaßen aus.

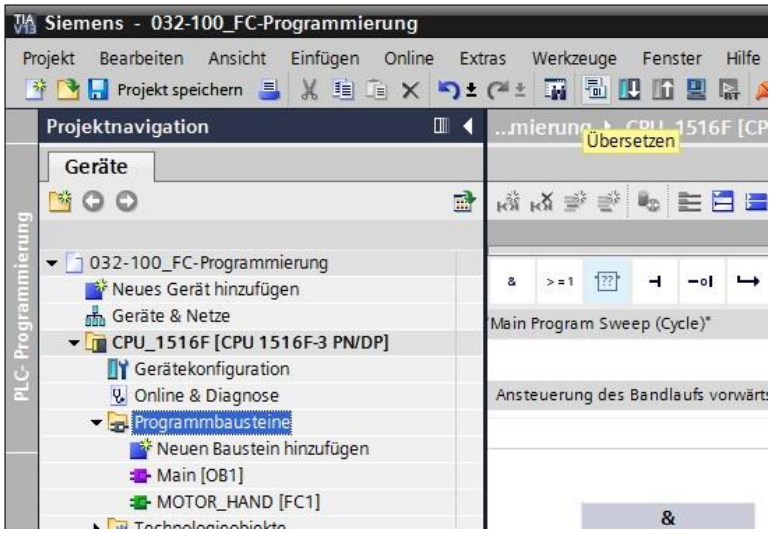


7.10 Programm speichern und übersetzen

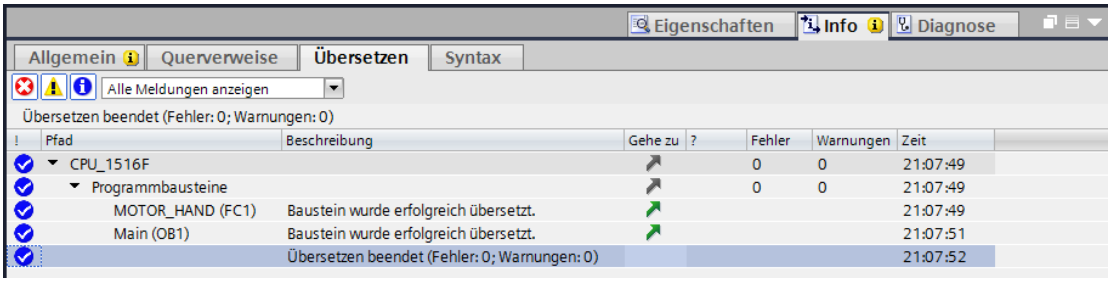
→ Zum Speichern Ihres Projektes wählen Sie im Menü den Button 

. Zum Übersetzen aller Bausteine klicken Sie auf den Ordner „Programmbausteine“ und wählen im Menü das Symbol  für Übersetzen an. (→ 

→ Programmbausteine → )

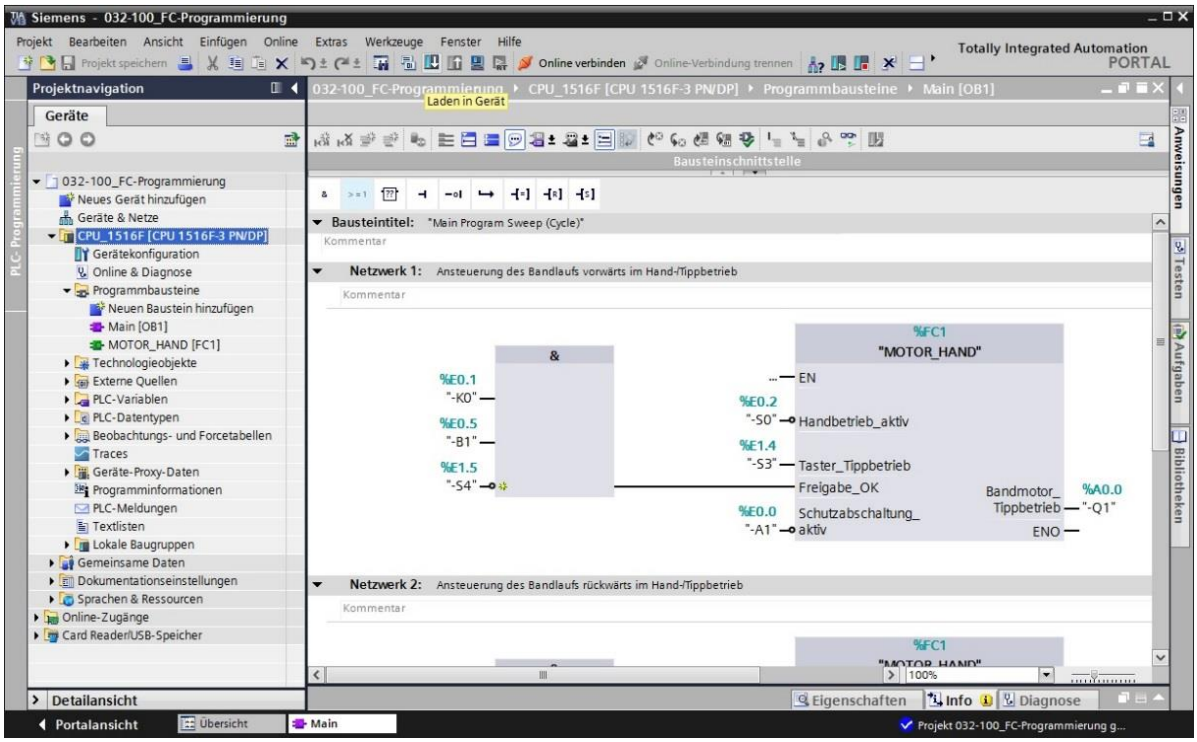


→ Im Bereich „Info“ „Übersetzen“ wird anschließend angezeigt, welche Bausteine erfolgreich übersetzt werden konnten.



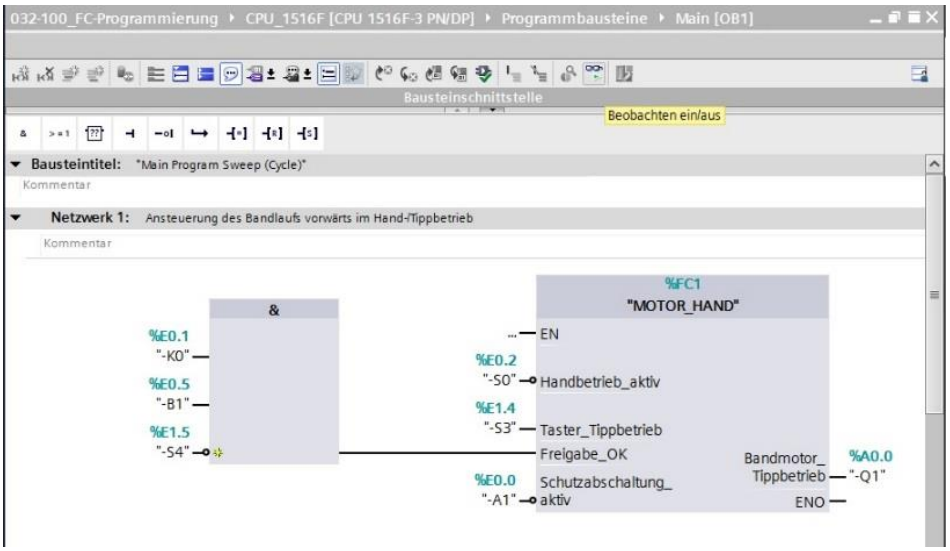
7.11 **Programm laden**

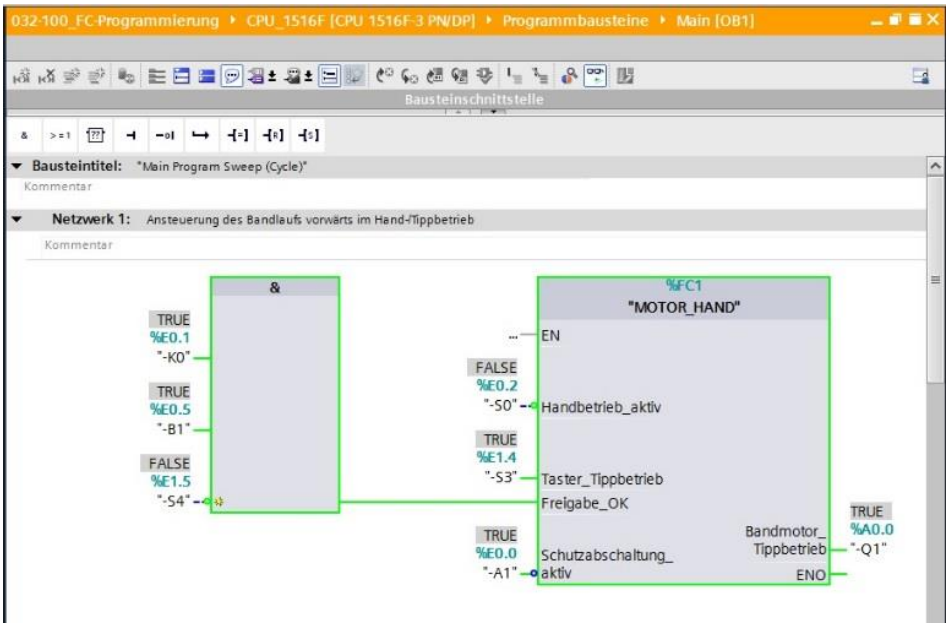
→ Nach erfolgreichem Übersetzen kann die gesamte Steuerung mit dem erstellten Programm, wie in den Modulen zur Hardwarekonfiguration bereits beschrieben, geladen werden. (→ )



7.12 Programmbausteine beobachten

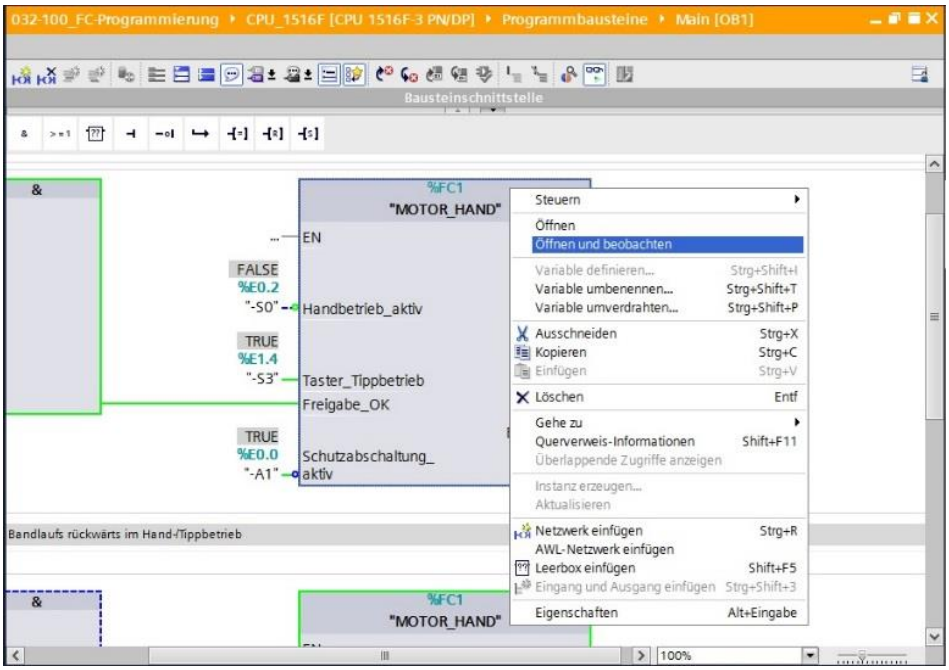
→ Zum Beobachten des geladenen Programms muss der gewünschte Baustein geöffnet sein. Mit einem Klick auf das Symbol kann das Beobachten ein/ausgeschaltet werden. (→ Main [OB1] →)

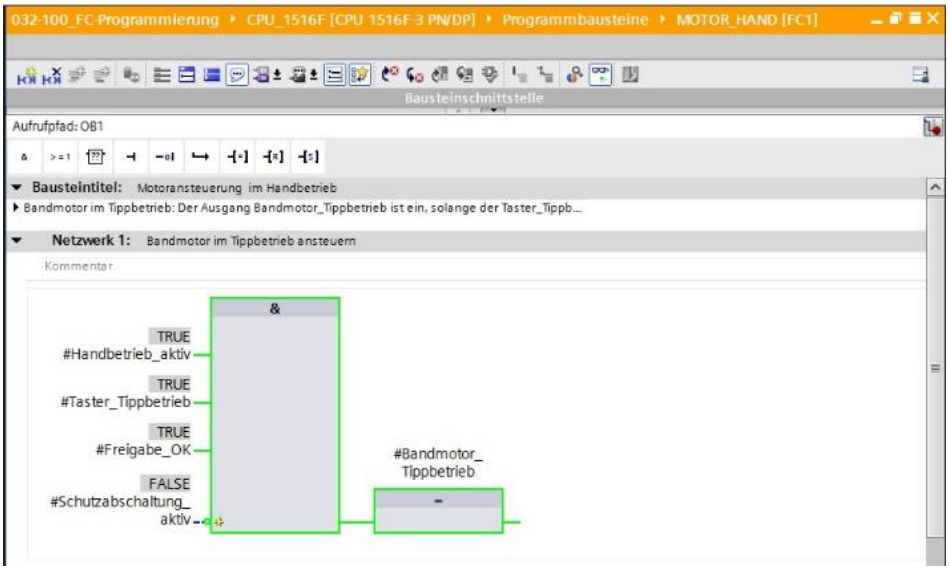




Hinweis: *Das Beobachten erfolgt hier signalbezogen und steuerungsabhängig. Die Signalzustände an den Klemmen werden mit TRUE bzw. FALSE angezeigt.*

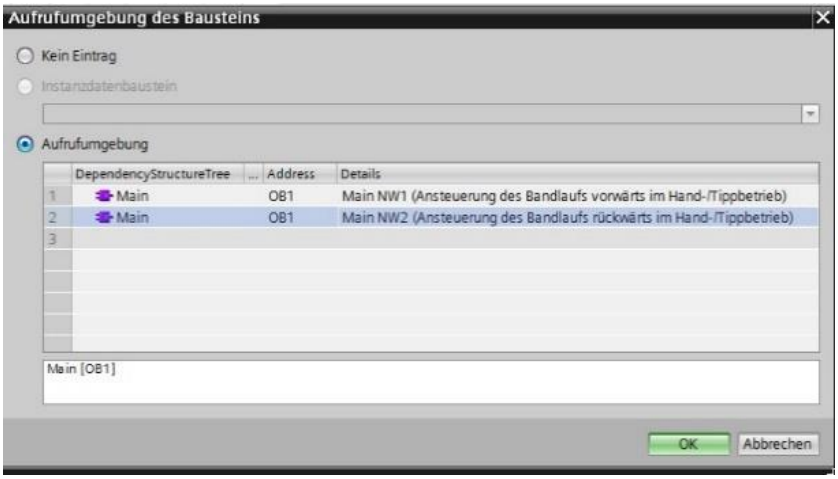
→ Die im Organisationsbaustein „Main [OB1]“ aufgerufene Funktion „MOTOR_HAND“ [FC1] kann nach einem Rechtsklick mit der Maus direkt zum „Öffnen und Beobachten“ ausgewählt werden. (→ „MOTOR_HAND“ [FC1] → Öffnen und beobachten)





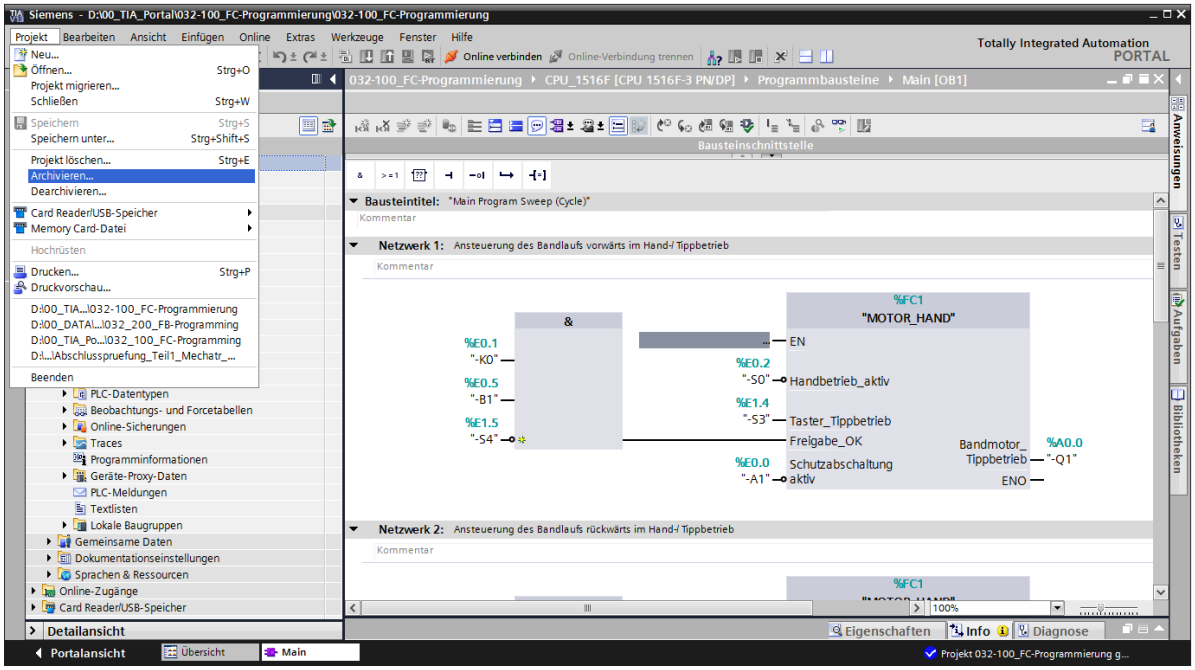
Hinweis: *Das Beobachten erfolgt hier funktionsbezogen und steuerungsunabhängig. Die Betätigung der Geber oder der Anlagenzustand werden hier mit TRUE bzw. FALSE dargestellt.*

→ Soll eine bestimmte Verwendungsstelle der Funktion „MOTOR_HAND“ [FC1] beobachtet werden, so kann über das Symbol die Aufrufumgebung ausgewählt werden. (→ → Aufrufumgebung → OK)



7.13 Archivieren des Projektes

→ Zum Abschluss wollen wir das komplette Projekt noch archivieren. Wählen Sie bitte im Menüpunkt → „Projekt“ den Punkt → „Archivieren …“ aus. Wählen Sie einen Ordner, in dem Sie ihr Projekt archivieren wollen und speichern Sie es als Dateityp „TIA Portal-Projektarchive“. (→ Projekt → „Archivieren → TIA Portal-Projektarchive → 032-100_FC-Programmierung…. → Speichern)



8 Checkliste

Nr.	Beschreibung	Geprüft
1	Übersetzen erfolgreich und ohne Fehlermeldung	
2	Laden erfolgreich und ohne Fehlermeldung	
3	Anlage einschalten (-K0 = 1) Zylinder eingefahren / Rückmeldung aktiviert (-B1 = 1) NOTAUS (-A1 = 1) nicht aktiviert Betriebsart HAND (-S0 = 0) Tippbetrieb Band vorwärts aktivieren (-S3 = 1) dann Bandmotor vorwärts feste Drehzahl (-Q1 = 1)	
4	wie 3 aber NOTAUS (-A1 = 0) aktivieren → -Q1 = 0	
5	wie 3 aber Betriebsart AUTO (-S0 = 1) → -Q1 = 0	
6	wie 3 aber Anlage ausschalten (-K0 = 0) → -Q1 = 0	
7	wie 3 aber Zylinder nicht eingefahren (-B1 = 0) → -Q1 = 0	
8	Anlage einschalten (-K0 = 1) Zylinder eingefahren / Rückmeldung aktiviert (-B1 = 1) NOTAUS (-A1 = 1) nicht aktiviert Betriebsart HAND (-S0 = 0) Tippbetrieb Band rückwärts aktivieren (-S4 = 1) dann Bandmotor rückwärts feste Drehzahl (-Q2 = 1)	
9	wie 8 aber NOTAUS (-A1 = 0) aktivieren → -Q2 = 0	
10	wie 8 aber Betriebsart AUTO (-S0 = 1) → -Q2 = 0	
11	wie 8 aber Anlage ausschalten (-K0 = 0) → -Q2 = 0	
12	wie 8 aber Zylinder nicht eingefahren (-B1 = 0) → -Q2 = 0	
13	wie 8 aber ebenfalls Tippbetrieb Band vorwärts aktivieren (-S3 = 1) → -Q1 = 0 und auch -Q2 = 0	
14	Projekt erfolgreich archiviert	

9 Übung

9.1 Aufgabenstellung – Übung

In dieser Übung sollen die folgenden Funktionen der Prozessbeschreibung Sortieranlage geplant, programmiert und getestet werden:

- Handbetrieb – Zylinder ausfahren
- Handbetrieb – Zylinder einfahren

Hinweis: *Achten Sie dabei auf die Wiederverwendbarkeit oder Kapselung der Funktionen.*

9.2 Planung

Planen Sie nun selbstständig die Umsetzung der Aufgabenstellung.

9.3 Checkliste – Übung

Nr.	Beschreibung	Geprüft
1	Funktion FC: ZYLINDER_HAND erstellt	
2	Schnittstellen definiert	
3	Funktion programmiert	
4	Funktion FC2 ins Netzwerk 3 des OB1 eingefügt	
5	Eingangsvariablen für Zylinder einfahren verschaltet	
6	Ausgangsvariablen für Zylinder einfahren verschaltet	
7	Übersetzen erfolgreich und ohne Fehlermeldung	
8	Funktion FC2 ins Netzwerk 4 des OB1 eingefügt	
9	Eingangsvariablen für Zylinder ausfahren verschaltet	
10	Ausgangsvariablen für Zylinder ausfahren verschaltet	
11	Übersetzen erfolgreich und ohne Fehlermeldung	
12	Laden erfolgreich und ohne Fehlermeldung	
13	Anlage einschalten (-K0 = 1) Zylinder eingefahren / Rückmeldung aktiviert (-B1 = 1) NOTAUS (-A1 = 1) nicht aktiviert Betriebsart HAND (-S0 = 0) Zylinder einfahren nicht aktivieren (-S5 = 0) Zylinder ausfahren aktivieren (-S6 = 1) dann Zylinder ausfahren (-M3 = 1) erfolgreich	
14	Anlage einschalten (-K0 = 1) Zylinder ausgefahren / Rückmeldung aktiviert (-B2 = 0) NOTAUS (-A1 = 1) nicht aktiviert Betriebsart HAND (-S0 = 0) Zylinder ausfahren nicht aktivieren (-S6 = 0) Zylinder einfahren aktivieren (-S5 = 1) dann Zylinder einfahren (-M2 = 1) erfolgreich	
15	Zylinder ein- und ausfahren nicht gleichzeitig aktivierbar	
16	Projekt erfolgreich archiviert	

10 Weiterführende Information

Zur Einarbeitung bzw. Vertiefung finden Sie als Orientierungshilfe weiterführende Informationen, wie z.B.: Getting Started, Videos, Tutorials, Apps, Handbücher, Programmierleitfaden und Trial Software/Firmware, unter nachfolgendem Link:

www.siemens.de/sce/s7-1500