MOELLER

Kurzanleitung

Sucosoft S40
Programmiersoftware

02/02 AWB27-1307-D

1. Auflage 1997, Redaktionsdatum 12/97
2. Auflage 1998, Redaktionsdatum 07/98
3. Auflage 1999, Redaktionsdatum 06/99
4. Auflage 2002, Redaktionsdatum 02/02,
siehe Änderungsprotokoll auf
der nächsten Seite

Autoren: Arno Dielmann, Frank Stober
Redaktion: Thomas Kracht

Änderungsprotokoll zum Handbuch AWB27-1307-D

Redaktions-datum	Seite	Stichwort	neu	Ände-rung	ent-fällt
04/99	allg.	Sucosoft S30-S4			×
		Sucosoft S4 → S40		×	
		AWB27-1185/1186			×
		AWB27-1280-D → AWB2700-1305-D		×	
		AWB27-1281-D → AWB2700-1306-D		×	
	14	Legende ③	×		
	41	Slaveadresse		×	
	52	Hinweis	×		
	52/53	Grafik/Tabelle		×	
	83	EMV: RFI, Surge		×	
06/99	gesamtes Handbuch	Überarbeitung für Version 4.0, Schwerpunkte: Kapitel 5 – 7, 9 – 10		×	
02/02	gesamtes Handbuch	Komplette Überarbeitung für Version 5.0		×	

Warnung!
Gefährliche elektrische Spannung!

Vor Beginn der Installationsarbeiten

- Gerät spannungsfrei schalten
- Gegen Wiedereinschalten sichern
- Spannungsfreiheit feststellen
- Erden und kurzschließen
- Benachbarte, unter Spannung stehende Teile abdecken oder abschranken.
- Die für das Gerät angegebenen Montagehinweise (AWA) sind zu beachten.
- Nur entsprechend qualifiziertes Personal gemäß EN 50110-1/-2 (VDE 0105 Teil 100) darf Eingriffe an diesem Gerät/System vornehmen.
- Achten Sie bei Installationsarbeiten darauf, dass Sie sich statisch entladen, bevor Sie das Gerät berühren.
- Die Funktionserde (FE) muss an die Schutzerde (PE) oder den Potentialausgleich angeschlossen werden. Die Ausführung dieser Verbindung liegt in der Verantwortung des Errichters.
- Anschluss- und Signalleitungen sind so zu installieren, dass induktive und kapazitive Einstreuungen keine Beeinträchtigung der Automatisierungsfunktionen verursachen.
- Einrichtungen der Automatisierungstechnik und deren Bedienelemente sind so einzubauen, dass sie gegen unbeabsichtigte Betätigung geschützt sind.
- Damit ein Leitungs- oder Aderbruch auf der Signalseite nicht zu undefinierten Zuständen in der Automatisierungseinrichtung führen kann, sind bei der E/A-Kopplung hard- und softwareseitig entsprechende Sicherheitsvorkehrungen zu treffen.
- Bei 24-Volt-Versorgung ist auf eine sichere elektrische Trennung der Kleinspannung zu achten. Es dürfen nur Netzgeräte verwendet werden, die die Forderungen der IEC 60364-4-41 bzw. HD 384.4.41 S2 (VDE 0100 Teil 410) erfüllen.
- Schwankungen bzw. Abweichungen der Netzspannung vom Nennwert dürfen die in den technischen Daten angegebenen Toleranzgrenzen nicht überschreiten, andernfalls sind Funktionsausfälle und Gefahrenzustände nicht auszuschließen.
- NOT-AUS-Einrichtungen nach IEC/EN 60204-1 müssen in allen Betriebsarten der Automatisierungseinrichtung wirksam bleiben. Entriegeln der NOT-AUS-Einrichtungen darf keinen Wiederanlauf bewirken.
- Einbaugeräte für Gehäuse oder Schränke dürfen nur im eingebauten Zustand, Tischgeräte oder Portables nur bei geschlossenem Gehäuse betrieben und bedient werden.

- Es sind Vorkehrungen zu treffen, dass nach Spannungseinbrüchen und -ausfällen ein unterbrochenes Programm ordnungsgemäß wieder aufgenommen werden kann. Dabei dürfen auch kurzzeitig keine gefährlichen Betriebszustände auftreten. Ggf. ist NOT-AUS zu erzwingen.
- An Orten, an denen in der Automatisierungseinrichtung auftretende Fehler Personen- oder Sachschäden verursachen können, müssen externe Vorkehrungen getroffen werden, die auch im Fehler- oder Störfall einen sicheren Betriebszustand gewährleisten beziehungsweise erzwingen (z. B. durch unabhängige Grenzwertschalter, mechanische Verriegelungen usw.).

Inhalt

02/02 AWB27-1307-D

02/02 AWB27-1307-D

Einleitung

Geräte und Software

Die Kompaktsteuerungen der Reihe PS4 sind leistungsfähige Mehrzweckgeräte für kleine bis mittlere Automatisierungsaufgaben. Sie zeichnen sich aus durch ihre platzsparende Bauweise, durch die umfangreiche Grundausstattung und die dezentrale und – je nach Typ – zentrale Erweiterbarkeit. Aufgrund dieser Merkmale werden die Steuerungen der Reihe PS4 in nahezu allen Bereichen der Automatisierungstechnik eingesetzt: Von einfachen Einzelanwendungen über Maschinensteuerungen direkt an der Maschine und intelligenter Vorverarbeitung bis zu vernetzten Systemen z. B. zur Gebäudeautomatisierung oder als dezentrale Stationen zum Fernwirken beispielsweise im Kläranlagenbereich.

Die Steuerung PS416 ist ein modulares Mehrzwecksystem für mittlere bis große Automatisierungsaufgaben. Sie zeichnet sich aus durch eine Platz sparende Bauweise, kurze Bearbeitungszeiten sowie ihren großen Speicher. Das modulare Konzept mit verschiedenen Zentraleinheiten, Baugrößen und vielfältigen Baugruppen ermöglicht einen maßgeschneiderten Einsatz für individuelle Automatisierungsaufgaben. Aufgrund ihrer schnellen Bearbeitungsgeschwindigkeit wird die PS416 besonders in Bereichen eingesetzt, in denen schnelle Informationsverarbeitung gefordert ist.

Alle Steuerungen werden mit der Sucosoft S40 programmiert und in Betrieb genommen. Die Sucosoft S40 ist eine Programmiersoftware nach der internationalen Norm IEC/EN 61131-3 und stellt in den vier angebotenen Programmiersprachen Anweisungsliste (AWL), Strukturierter Text (ST), Kontaktplan (KOP) und Funktionsbausteinsprache (FBS) einen umfangreichen Befehlssatz zur Verfügung.

Die Programmiersprachen AWL, KOP und FBS sind untereinander umschaltbar, sodass Sie nach Ihren Vorlieben und nach Bedarf die jeweils geeignete Sprache wählen können. ST kann bei abgeschlossenen Anweisungen mit AWL-Sequenzen, nicht aber mit KOP- oder FBS-Grafikelementen gemischt werden. Die Steuerungen verarbeiten alle gängigen elementaren Datentypen sowie abgeleitete Datentypen. Zusätzlich zu zahlreichen IEC-Standardfunktionen und -Funktionsbausteinen stehen Ihnen weitere Funktionen und Funktionsbausteine von Moeller zur Verfügung, die Sie durch selbst erstellte, einfach wieder verwendbare Funktionsbausteine ergänzen können.

Dokumentation

Dieses Handbuch verschafft Ihnen einen Schnelleinstieg in das Programmierpaket Sucosoft S40. Anhand einer einfachen Steuerungsaufgabe lernen Sie das Arbeiten mit der Sucosoft S40: Vom Anlegen eines eigenen Projekts bis zum Testen des erstellten Programms in der Steuerung. Nach der Code-Erzeugung wird das fertige Programm in die Steuerung übertragen und getestet. Dabei lernen Sie die verschiedenen Möglichkeiten kennen, die das Werkzeug TEST & INBETRIEBNAHME bietet. Ein weiteres Programmbeispiel verdeutlicht, wie Sie mit der Programmiersoftware Sucosoft S40 fertige Funktionsbausteine verwenden und eigene Funktionsbausteine programmieren.

Sie benötigen ca. zwei Stunden, um das erste und ca. drei Stunden, um das zweite Programmbeispiel durchzuspielen.

Eine vollständige Beschreibung aller Möglichkeiten, die Sie bei der Programmierung der Steuerungen PS4 und PS416 einsetzen können, finden Sie in den Nachschlagewerken AWB2700-1305-D „Programmiersoftware S40, Benutzeroberfläche“ und AWB2700-1306-D „Programmiersoftware S40, Sprachelemente für PS4-200, PS4-300 und PS416“.

Schreibkonventionen

Wählen Sie ‹ Projekt → Neu› bedeutet: Aktivieren Sie den Befehl „Neu“ aus dem Menü „Projekt“.

Kursive Kleinbuchstaben kennzeichnen Texte, die Sie genau wie gezeigt eingeben müssen.
Beispiel: *c:\projekte\beispiel*

Handlungsanweisungen sind mit einem Pfeil ▶ gekennzeichnet. Alle anderen Abschnitte geben Ihnen lediglich Informationen, ohne dass Sie etwas tun müssen.

Informationen und Tipps geben Ihnen zusätzliche Hinweise zum Thema und stellen kapitelübergreifende Zusammenhänge dar.

Hardware-Voraussetzungen

PS4

Für das erste Beispiel benötigen Sie eine Steuerung PS4-141-MM1, PS4-151-MM1, PS4-201-MM1, PS4-271-MM1 oder PS4-341-MM1, für das zweite Beispiel zusätzlich ein Erweiterungsmodul EM4-101-DD1 und ein Verbindungskabel KPG1-PS3. Weiterhin benötigen Sie in jedem Fall ein Programmierkabel ZB4-303-KB1 (Verbindungskabel zwischen PC und PS4).

PS416

Sie benötigen eine PS416, bestückt mit einer Stromversorgungsbaugruppe, einer CPU-300 oder CPU-400, einer Digital-Eingabe- und einer PS416-OUT-400-Ausgabe-Baugruppe, dazu ein Programmierkabel PS416-ZBK-210 oder einen Schnittstellenumsetzer UM1.5. Weitere Baugruppen, die möglicherweise in der Steuerung gesteckt sind, werden für das Beispiel nicht benötigt und auch nicht berücksichtigt.

02/02 AWB27-1307-D

1 PS4 installieren und verdrahten

Adresscodierung

Bevor die PS4-141-MM1 mit dem EM4-101-DD1 vernetzt und verdrahtet wird, muss die Teilnehmer-Adresse des EM4-101-DD1 über den Adresscodierschalter S2 eingestellt werden. Für die hier gewählte Teilnehmeradresse 1 gilt folgende Codierung (siehe auch AWB27-1257-D):

S2	1	2	3	4	5	6	7	8
	1	0	1	1	1	1	0	1

1 = ON, 0 = OFF

Busabschluss-widerstände

Für den physikalisch ersten und letzten Busteilnehmer am Strang müssen außerdem die Busabschlusswiderstände geschlossen sein. Deshalb muss der Schalter S1 des EM4-101-DD1 wie folgt eingestellt werden:

S1: 1 = ON
2 = ON

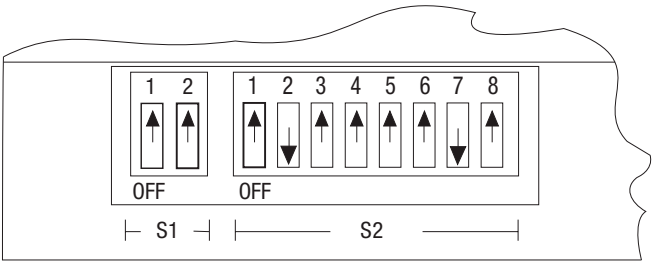


Abbildung 1: Busabschlusswiderstände einschalten

Die Abbildung zeigt die Verdrahtung der PS4-201-MM1, PS4-141-MM1 oder PS4-341 und des EM4-101-DD1.

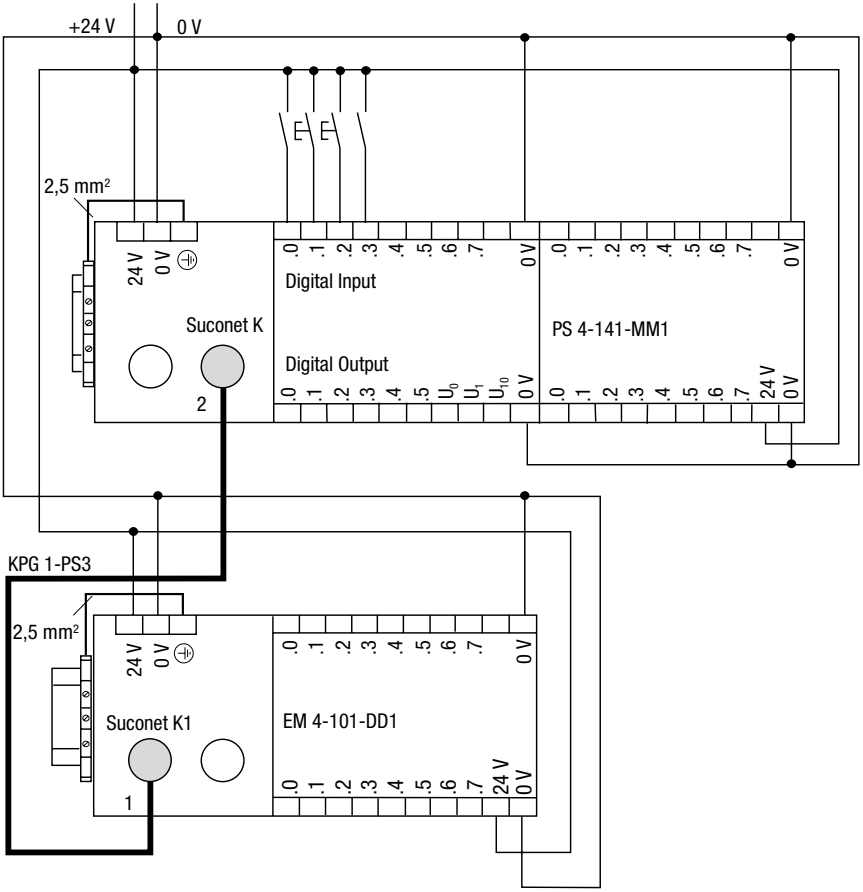


Abbildung 2: Verdrahtungsbeispiel PS4 – EM4

02/02 AWB27-1307-D

2 PS416 installieren und verdrahten

Adresscodierung

Bevor Sie die digitale Eingabebaugruppe und die Ausgabebaugruppe PS416-OUT-400 in den Baugruppenträger einsetzen, müssen Sie die Baugruppen per DIP-Schalter adressieren. Für die Beispiele dieser Kurzanleitung werden Digital-Baugruppen wie folgt auf die Eingangs- und Ausgangsbyteadresse „0“ codiert (siehe auch AWA27-1304):

S1	S2	S3	S4	S5	S6	S7	S8
1	1	1	1	1	1	x	x

1 = ON, 0 = OFF, x = nicht ausgewertet

Baugruppen stecken

Stecken Sie die Stromversorgungsbaugruppe auf die Steckplätze 0 und 1 ganz links in den Baugruppenträger. Die PS416-CPU wird direkt anschließend auf die Plätze 2 und 3 gesteckt. Schließlich folgen noch die codierten Digitalbaugruppen PS416-INP-40x und PS416-OUT-400 auf den Steckplätzen 4 und 5.

Die Abbildung zeigt die Verdrahtung der PS416. Für das Beispiel kann ein beliebiger Baugruppenträger PS416-BGT verwendet werden.

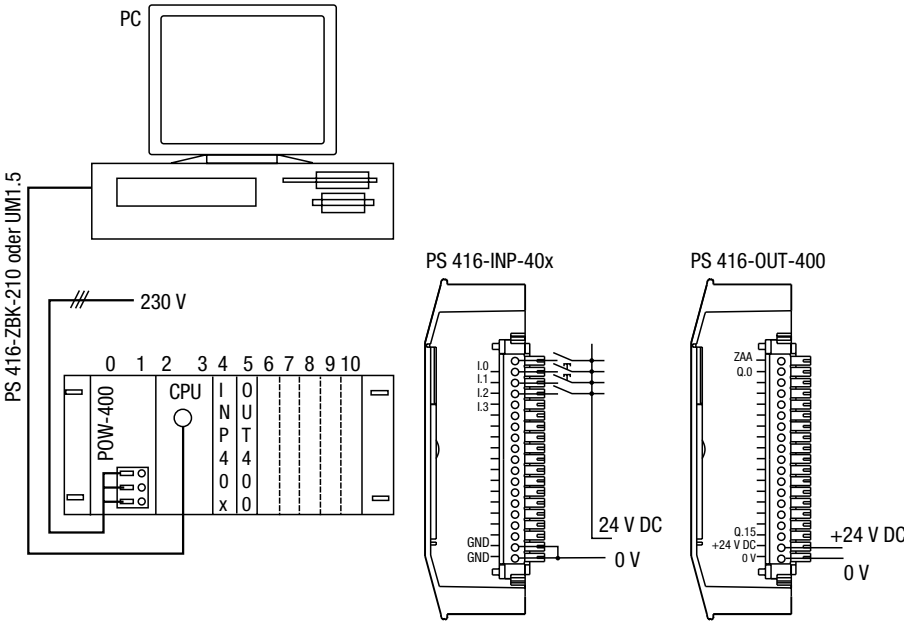


Abbildung 3: Verdrahtungsbeispiel PS416

Je nach verwendeter Stromversorgungsbaugruppe werden 230 V AC oder 24 V DC angeschlossen. Über die steckbare Schraubklemme der PS416-OUT-400 werden die LEDs der Baugruppe mit 24 V DC versorgt. Die Eingänge 1 und 2 der INP-40x werden mit Tastern, die Eingänge 0 und 3 mit Schaltern beschaltet, die an 24 V DC liegen.

Der Schalter der Programmierschnittstelle der CPU muss auf RS 232 gestellt werden, wenn Sie ein Programmierkabel PS416-ZBK-210 verwenden. Bei Verwendung eines Schnittstellenumsetzers UM1.5 muss der Schalter in Position RS 485 stehen.

02/02 AWB27-1307-D

3 Programmieraufgabe 1

Lesen Sie zuerst die Aufgabenstellung und die Umsetzung in ein AWL-Programm. Anschließend werden Sie Schritt für Schritt angeleitet, mit der Sucosoft S40

- ein Projekt anzulegen
- die Programmdatei zu erstellen
- die Konfiguration einzugeben
- den Programmcode zu erzeugen
- die Test und die Inbetriebnahme durchzuführen.

Aufgabenstellung

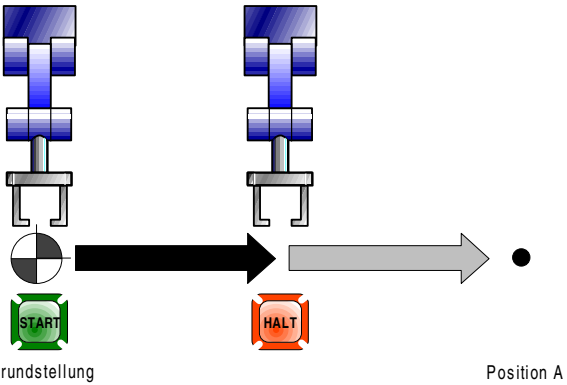


Abbildung 4: Aufgabenstellung

Bei Grundstellung der Anlage soll der Roboterarm durch Betätigen der Taste „START“ in Position A bewegt werden. Die Taste „HALT“ dient der Unterbrechung der Aktion.

Basisinformationen zur Programmierung nach IEC/EN 61131-3

Entsprechend der IEC/EN 61131-3 werden die benutzten Variablen zwischen zwei Schlüsselwörtern deklariert. Abhängig vom Gültigkeitsbereich werden verschiedene Schlüsselwörter verwendet. Die in der ersten Programmieraufgabe verwendeten lokalen Variablen werden zwischen den Schlüsselwörtern VAR und END_VAR deklariert.

Der Deklarationsteil einer Programmorganisationseinheit (POE) steht immer vor dem Anweisungsteil. Je nach Gültigkeitsbereich müssen im Deklarationsblock bestimmte Angaben gemacht werden. Für lokale Variablen sind ein Variablenname, der Datentyp und optional die physikalische Adresse, ein Initialwert, ein Attribut und ein Kommentar anzugeben. Variablen als Zwischenmerker werden rein symbolisch deklariert, das heißt ohne Adressangabe.

Binäre Eingangs- und Ausgangsoperanden, also Variablen mit einer Zuordnung zu physikalischen Adressen, werden nach der IEC/EN 61131-3 als „direkt dargestellte Variablen“ bezeichnet. Für direkt dargestellte Variablen gibt es bestimmte Konventionen, die bei der Deklaration der Adresse eingehalten werden müssen:

Jede direkt dargestellte Variable muss mit dem Schlüsselwort „AT“ gekennzeichnet werden, die Adresse wird mit dem Prozentzeichen, einem Kennbuchstaben (I = Eingang, Q = Ausgang) sowie der fünfstelligen Adresse angegeben. Die ersten drei Stellen der Adresse sind topologische Adressen, die sich aus der Gerätekonfiguration ergeben. Bei einem Steuerungssystem, das nur aus einer Steuerung besteht ohne weitere Teilnehmer, sind diese drei Stellen immer „0“. Die letzten beiden Stellen bezeichnen die Byte- und die Bitadresse.

Beispiel: AT %I0.0.0.0.3 kennzeichnet den Eingangsoperanden Bit 3 im Byte 0.

02/02 AWB27-1307-D

In der Beispielaufgabe kommen nur binäre Eingangs-/Ausgangsvariablen vor, für die wir die angegebenen Adressen annehmen:

Grundstellung:	Eingang 0.0
Taste Start:	Eingang 0.1
Taste Halt:	Eingang 0.2
Position A:	Eingang 0.3
Motor:	Ausgang 0.0

Die Variablennamen sind frei wählbar. Die Variablendeklaration sieht folgendermaßen aus:

```
VAR
  grundstellung   AT %I0.0.0.0.0:BOOL;     (*Anlagengrundstellung*)
  taste_start     AT %I0.0.0.0.1:BOOL;     (*Taste start*)
  taste_halt      AT %I0.0.0.0.2:BOOL;
  position_A      AT %I0.0.0.0.3:BOOL;     (*Position A erreicht*)
  motor           AT %Q0.0.0.0.0:BOOL;
END_VAR
```

Der binäre Eingangsoperand „grundstellung“ wird gelesen und nach der UND-Funktion mit dem Operanden „taste_start“ verknüpft.

Lesebefehl: LD
UND-Verknüpfung: AND

Ist das Verknüpfungsergebnis (VKE) = 1, wird der Ausgangsoperand „Motor“ auf den Wert „1“ gesetzt. (Die LED am Ausgang „motor“ leuchtet.)
Setzbefehl, wenn VKE = 1: S

Anschließend werden die Eingangsoperanden „taste_halt“ und „position_A“ gelesen und mit einem logischen „ODER“ verknüpft. Ist das VKE = 1, wird der Ausgangsoperand „motor“ rückgesetzt.

(Das Lämpchen am Ausgang „motor“ erlischt.)
Rücksetzbefehl, wenn VKE = 1: R

Die Anweisungsliste sieht daher folgendermaßen aus:

```
LD   grundstellung
AND  taste_start
S    motor       (*Schaltet den Motor ein*)
LD   taste_halt
OR   position_A
R    motor       (*Schaltet den Motor aus*)
```

Vor den Deklarationsblock muss das Schlüsselwort „PROGRAM“ sowie der Programmname zur Kennzeichnung eines Programms gesetzt werden, am Ende des Anweisungsteils muss das Schlüsselwort END_PROGRAM zur Kennzeichnung des Programmendes stehen.

Das Beispielprogramm hat also folgende Form:

```
PROGRAM position
VAR
  grundstellung   AT %I0.0.0.0.0:BOOL;     (*Anlagengrundstellung*)
  taste_start     AT %I0.0.0.0.1:BOOL;     (*Taste start*)
  taste_halt      AT %I0.0.0.0.2:BOOL;
  position_A      AT %I0.0.0.0.3:BOOL;     (*Position A erreicht*)
  motor           AT %Q0.0.0.0.0:BOOL;
END_VAR
  LD   grundstellung
  AND  taste_start
  S    motor         (*Schaltet den Motor ein*)
  LD   taste_halt
  OR   position_A
  R    motor         (*Schaltet den Motor aus*)
END_PROGRAM
```

Geben Sie die Schlüsselwörter PROGRAM und END_PROGRAM und damit die erste und die letzte Zeile des Beispiels nicht ein. Dies erfolgt automatisch durch den POE-EDITOR.

02/02 AWB27-1307-D

4 NAVIGATOR

Überblick

Starten Sie Windows und aktivieren Sie das NAVIGATOR-Symbol. Die Sucosoft S40 wird gestartet, und die NAVIGATOR-Oberfläche öffnet sich:

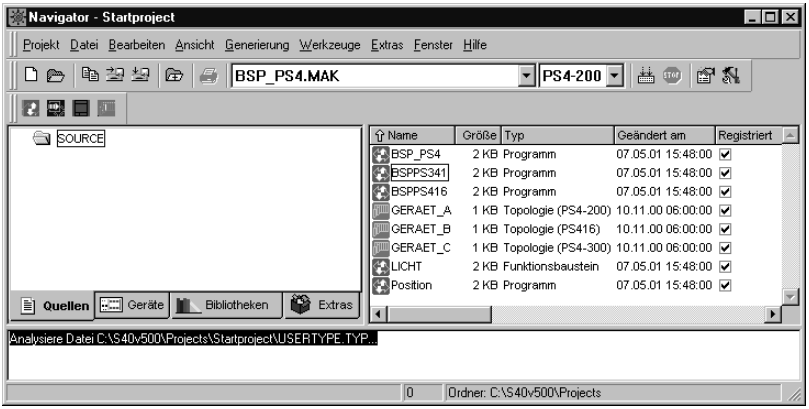


Abbildung 5: NAVIGATOR-Oberfläche

Im Fenster „Navigator“ sind die Symbole zum Öffnen der einzelnen Werkzeuge integriert.

Der NAVIGATOR unterstützt Sie bei allen Arbeiten, die von der Erstellung eines Anwenderprogramms bis zu dessen Ausführung in der SPS anfallen:

Er beinhaltet die Projektverwaltung, über die Sie Projekte anlegen und strukturieren.

Er ermöglicht Ihnen den Zugriff auf den POE-EDITOR, mit dem Sie Ihre POEs (Programme, Funktionsbausteine und Funktionen) editieren.

Er ermöglicht Ihnen den Zugriff auf den TOPOLOGIE-KONFIGURATOR , mit dem Sie die Hardware-Konfiguration definieren.

Er beinhaltet die Programmcode-Erzeugung, die Ihr Anwenderprogramm in ein ablauffähiges, steuerungsspezifisches Maschinenprogramm übersetzt.

Sie können direkt auf die TEST & INBETRIEB-NAHME zugreifen, mit der Sie Ihr Anwender-programm in der Steuerung testen können.

Er stellt einen FORMULAR-EDITOR zur Verfü-gung, mit dem Sie Druck-Formulare ansehen und gestalten können.

Im Hauptfenster des NAVIGATORs sehen Sie oben die Titelleiste und unten die Statusleiste. Unterhalb der Titelleiste erscheint die Menüleiste, dann die Symbolleiste, an die sich die Werkzeugleiste anschließt. Der dazwischen liegende Arbeitsbereich ist in drei Fenster aufgeteilt:

Das obere linke Fenster (Browser-Fenster) beinhaltet die „Baumansicht“ mit den jeweiligen Ordnern. Den unteren Abschluss dieses Fensters bilden die vier Karteikarten-Reiter „Quellen“, „Geräte“, „Bibliotheken“ und „Extras“.

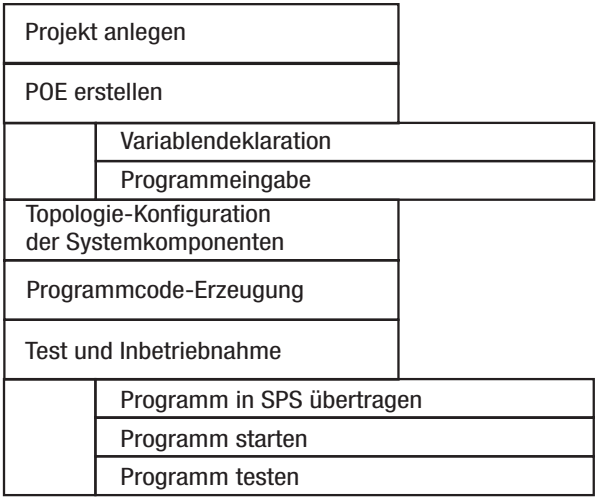
Im rechten Fenster befindet sich die „Datei-ansicht“.

Unterhalb dieser Fenster befindet sich das Ausgabefenster für Status- bzw. Fehlermel-dungen, z. B. bei der Programmcode-Erzeugung.

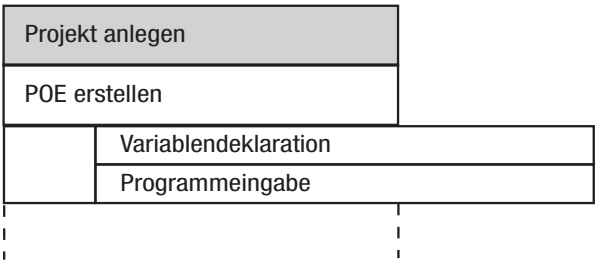
Die Größe der Fenster können Sie durch Ziehen der Trennlinien mit der Maus ändern.

02/02 AWB27-1307-D

Die untenstehende Grafik zeigt Ihnen für die Programmieraufgabe 1 die Schritte von der Eingabe bis zum Testlauf des fertigen Programms. Die Darstellung wird Sie als Orientierung durch die folgenden Kapitel begleiten.



5 Ein neues Projekt anlegen



Für das hier behandelte Beispiel soll folgender Pfad gelten:

C:\PROJEKTE\BEISPIEL

Ein neues Projekts legen Sie über das Menü mit ‹ Projekt → Neu...› oder die entsprechende Schaltfläche an.

 *Neues Projekt anlegen*

Es öffnet sich das Dialogfeld „Neues Projekt anlegen“:

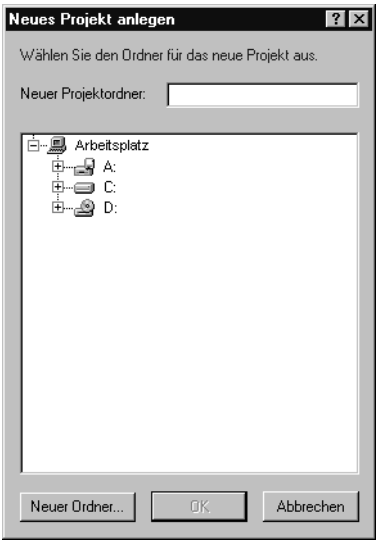


Abbildung 6: Neues Projekt anlegen

▶ Wählen Sie zuerst das Laufwerk „C“ aus.

Da Sie Ihr Projekt in dem neu zu erstellenden Unterordner „Projekte“ anlegen möchten:

▶ Betätigen Sie die Schaltfläche „Neuer Ordner“, geben Sie anschließend im Dialogfeld „Neuer Ordner“ den neuen Ordnernamen ein. Bestätigen Sie die Eingabe mit der Schaltfläche OK.

▶ Tragen Sie anschließend im Eingabefeld „Neuer Projektordner“ den Namen „Beispiel“ für das neue Projekt ein. Bestätigen Sie die Eingabe mit der Schaltfläche OK.

Es wird zunächst eine Projektstruktur mit den Ordnern „Devices“ und „Source“ von der Sucosoft S40 eingerichtet.

02/02 AWB27-1307-D

In der Titelleiste wird Ihnen „Beispiel“ als Name des geöffneten Projektes und in der Statusleiste als Projekt-Laufwerk, mit Pfadangabe, „C:\Projekte“ angezeigt.

Bevor Sie mit der Erstellung Ihrer POEs beginnen, stellen Sie sicher, dass der richtige SPS-Typ ausgewählt ist. Dieser wird in einem Auswahlfenster in der Symbolleiste des NAVIGATORs angezeigt. Abhängig vom gewähltem SPS-Typ stehen Ihnen im POE-EDITOR die entsprechenden Datentypen und Hersteller-Funktionsbausteine bzw. Funktionen zur Verfügung.

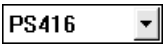


Abbildung 7: Auswahlfenster für den SPS-Typ

SPS-Typ auswählen
Derzeit sind unter den drei wählbaren SPS-Typen folgende Steuerungen gruppenweise zusammengefasst:

PS4-200:

PS4-141-MM1,
PS4-151-MM1,
PS4-201-MM1,
PS4-271-MM1

PS4-300:

PS4-341-MM1

PS416:

PS416-CPU-200,
PS416-CPU-300,
PS416-CPU-400

▶ Wählen Sie den gewünschten SPS-Typ im Auswahlfenster innerhalb der Symbolleiste.

02/02 AWB27-1307-D

6 Ein Anwenderprogramm erstellen

Anwenderprogramme erstellen Sie mit dem POE-EDITOR; sie können aus einer oder mehreren POEs (Dateien) bestehen. Der Oberbegriff „POE“ steht gemäß der internationalen Norm IEC/EN 61131-3 für ProgrammOrganisationsEinheit und bezeichnet die drei POE-Typen „Programm“, „Funktionsbaustein“ und „Funktion“. „Funktion“ oder „Funktionsbaustein“ wählt man für Programmteile, die man öfter verwenden möchte.

Alle POEs (Dateien), die Sie speichern, werden vom übergeordneten NAVIGATOR automatisch unter dem aktuellen Projekt registriert und im Ordner „Source“ abgelegt. Ebenfalls in diesen Ordner werden Topologie-Dateien, die Sie mit dem TOPOLOGIE-KONFIGURATOR erstellen, abgelegt. Im Fenster „Dateiansicht“ werden dann alle Dateien der Typen „Topologie“ und „POE“ (Programm, Funktion und Funktionsbaustein) angezeigt.

Die Grundeinstellung des POE-EDITORs beim Anlegen einer neuen POE, und unter anderem damit die Programmiersprache, für die sich der Editor öffnet, können Sie unter ‹ Extras → Einstellungen → Editor› entsprechend Ihren Wünschen vornehmen.

In der folgenden Beschreibung werden die Schaltflächen der Standard-Symbolleiste verwendet. Aktivieren Sie diese Symbolleiste – wenn nötig – über ‹ Ansicht → Symbolleiste› .

Kontextmenüs
Eine bequeme Arbeitshilfe bieten Ihnen die Kontextmenüs. Dies sind Kurzmenüs, welche die wichtigsten Befehle zu einer bestimmten Sucosoft-S40-Funktion enthalten.

Sie werden durch Klicken mit der rechten Maustaste auf das ausgesuchte Objekt oder Fenster geöffnet. Der Inhalt der Kurzmenüs ist abhängig von der Umgebung, dem Kontext, wo sich der Mauszeiger befindet und vom markierten Element.

Abbildung 8: Kontextmenü zum Deklarationsteil im Syntax-Modus

Für unser Beispiel benötigen Sie ein ablauffähiges Programm, es muss daher der POE-Typ „Programm“ gewählt werden.

Benutzen Sie zur Erstellung des Anwenderprogramms das NAVIGATOR-Menü:

▶ Wählen Sie ‹ Werkzeuge → POE-Editor› oder die entsprechende Schaltfläche in der Werkzeugleiste.

 *„POE-EDITOR“ im Offline-Modus*

Es öffnet sich der POE-EDITOR:

▶ Wählen Sie – dem benötigten POE-Typ „Programm“ entsprechend – den Menüpunkt ‹ Datei → POE neu → Programm› oder die Schaltfläche „P“ in der Standard-Symbolleiste.

 *„Programm“*

02/02 AWB27-1307-D

Wenn Sie eine neue Programm-POE anlegen, fragt Sie der POE-EDITOR zuerst, ob Sie Variablen aus einer Topologie erzeugen bzw. deklarieren möchten. Dabei werden die von Ihnen im TOPOLOGIE-KONFIGURATOR vereinbarten physikalischen SPS-Adressen in den Deklarationsteil der Programm-POE übernommen und im Gültigkeitsbereich „Global“ eingetragen. Sie brauchen lediglich einen Variablennamen zuzuordnen.

Da diese Funktion Ihre Editierarbeit reduziert, bietet es sich an, dass Sie zuerst das Kapitel 7 durchlesen, Ihre Beispiel-Topologie erstellen und diese Frage mit „Ja“ beantworten.

Es öffnen sich die beiden Fenster „Deklarations- und Anweisungsteil“, die entsprechend der Grundeinstellung des POE-EDITORs auf dem Bildschirm angeordnet sind.

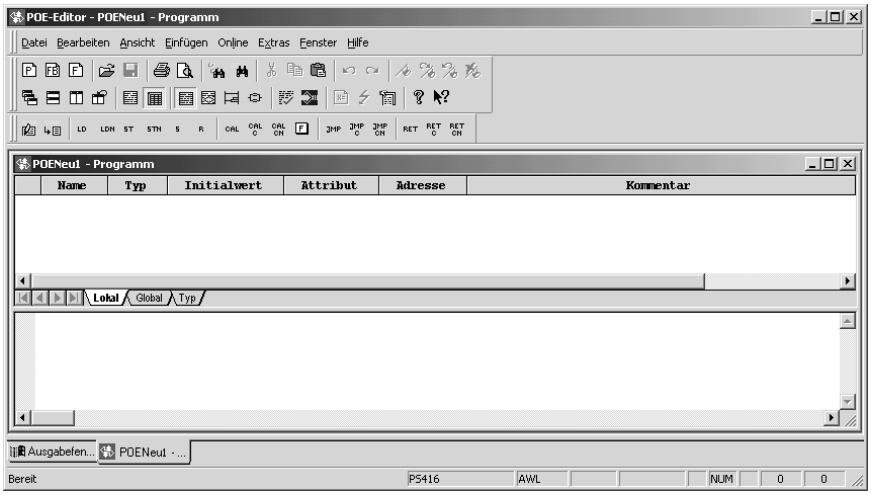


Abbildung 1: POE-EDITOR im Syntax-Modus

Die Programmeingabe erfolgt in zwei Phasen:

Im Deklarationsteil deklarieren Sie die Variablen, die Sie im Anweisungsteil verwenden wollen.

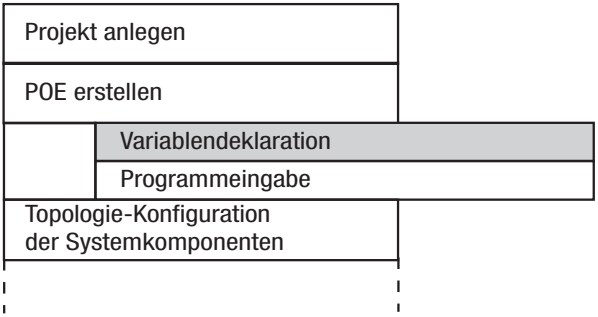
Im Anweisungsteil erstellen Sie Ihr Anwenderprogramm.

Für die Eingabe Ihrer Variablen im Deklarationsteil bietet Ihnen die Sucosoft S40 einen tabellenorientierten, syntaxgesteuerten Variableneditor (Syntax-Modus) für eine komfortable, benutzergeführte Variablendeklaration in vorgegebenen Eingabefeldern. Daneben gibt es einen rein textuellen, freien Variableneditor (Freier Modus) ohne Benutzerführung für geübte SPS-Programmierer. Für den Einstieg deklarieren Sie die Variablen im syntaxgesteuerten Modus – so brauchen Sie sich nicht um die Schlüsselwörter und die Syntax zu kümmern.

Für die Erstellung Ihres Anwenderprogramms können Sie zwischen den textuellen Programmiersprachen „Anweisungsliste“ (AWL), „Strukturierter Text“ (ST) oder einer der grafischen Programmiersprachen „KOntaktPlan“ (KOP) oder „FunktionsBausteinSprache“ (FBS) wählen. Das Beispielprogramm erstellen Sie in AWL.

Es ist Ihnen freigestellt, ob Sie zuerst die Variablen deklarieren oder zuerst die Anweisungsliste schreiben – Sie können auch beides parallel machen. Für das Beispiel deklarieren Sie zuerst die Variablen und geben anschließend den Anweisungsteil ein.

Deklarieren der Variablen



Die Eingabe der Variablen nehmen Sie, für jeden Gültigkeitsbereich getrennt, unter einer eigenen Karteikarte vor. Beim gewählten POE-Typ „Programm“ sind nur Karteikarten der zulässigen Gültigkeitsbereiche „Typ“, „Lokal“ und „Global“ vorhanden.

Abbildung 2: Gültigkeitsbereiche für den POE-Typ „Programm“

Die Gültigkeitsbereiche wählen Sie mit einem Klick auf einen der angebotenen Karteikarten-Reiter am unteren Fensterrand;

Beginnen Sie die Variablendeklaration mit dem voreingestellten Gültigkeitsbereich „Lokal“. Sie brauchen daher den Karteikarten-Reiter nicht zu aktivieren. Es sind nur die Zellen vorhanden, die für den gewählten Gültigkeitsbereich zulässig sind. Deklarieren Sie alle Variablen dieses Typs.

Wenn in einer POE noch andere Gültigkeitsbereiche vorkommen, wählen Sie den Karteikarten-Reiter des nächsten Typs an und deklarieren Sie diese Variablen. In der Liste werden immer nur die deklarierten Variablen eines Typs angezeigt.

Die Eingabe der Variablen erfolgt durch Ausfüllen der Eingabe-Zellen der Bearbeitungszeile. Zwischen den Zellen wechseln Sie mit

Cursor links/rechts,

TAB/Umschalt + TAB,

Cursor auf/ab,

Seite auf/Seite ab oder

über Mausklick.

Fehlerhafte Variablendeklarationen werden bei der Syntaxprüfung oder beim Speichern festgestellt und im Fehlerprotokoll-Fenster angezeigt. Springen Sie in dem Fall durch Doppelklick auf die Fehlermeldung an die fehlerhafte Stelle und korrigieren Sie diese entsprechend.

Wie oben erwähnt, erfordern die im Programmbeispiel eingesetzten Variablen den Gültigkeitsbereich „Lokal“.

▶ Geben Sie zur Deklaration der ersten Variable aus dem Programmbeispiel der Reihe nach die folgenden Angaben ein:

Name: grundstellung

Typ: Wählen Sie den Datentyp der Variablen unter ‹ Einfügen → Variablendeklaration› oder dem Kontextmenü über die rechte Maustaste aus. Markieren Sie oben die Gruppe „Datentypen“ und wählen Sie aus der darunter stehenden Liste den gewünschten Typ aus.

Initialwert: Eine Eintragung ist nicht erforderlich – damit wird beim Programmstart der Variable „grundstellung“ der Wert „0“ zugewiesen.

02/02 AWB27-1307-D

Attribut: Keine Eintragung.
An dieser Stelle kann RETAIN für eine remanente Variable oder CONSTANT, wenn es sich um eine Konstante handelt, eingegeben werden. In Zusammenhang mit Ein- und Ausgangsvariablen (I/Q) wäre keine der Eingaben sinnvoll.

Adresse: I0.0.0.0.0
Physikalische Adresse der Variablen „grundstellung".

Kommentar: Kann bei Bedarf eingegeben werden. Geben Sie in diesem Fall „Anlagengrundstellung" ein.

▶ Beenden Sie die Eingabe in der Zeile, indem Sie diese durch Drücken der „Enter"-Taste verlassen.

Sie sehen die Zeile mit der vollständig definierten Variable:

Name	Typ	Adresse	Kommentar
grundstellung	BOOL	I0.0.0.0.0	Anlagengrundstellung

▶ Deklarieren Sie auf die gleiche Weise die restlichen Variablen des Beispielprogramms:

Name	Typ	Adresse	Kommentar
taste_start	BOOL	I0.0.0.0.1	Taste start
taste_halt	BOOL	I0.0.0.0.2	
position_A	BOOL	I0.0.0.0.3	Position A erreicht
motor	BOOL	Q0.0.0.0.0	

Damit ist die Variablenliste vollständig.

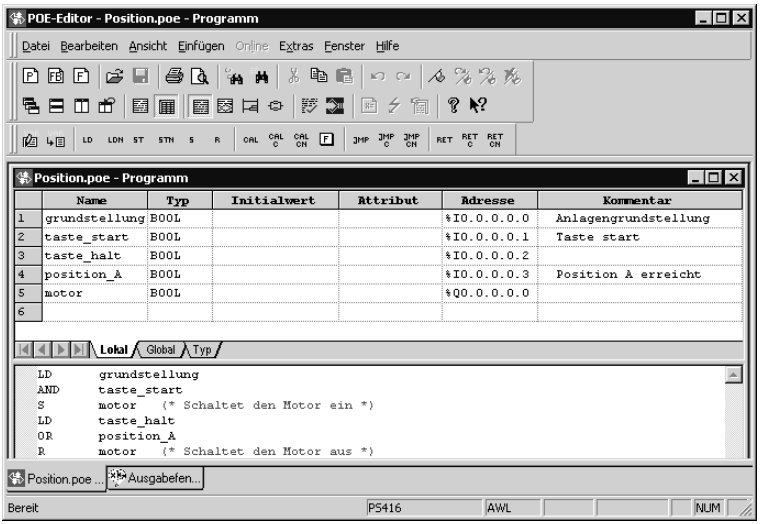


Abbildung 3: Deklarierte Variablen im Syntax-Modus

Die fertige Variablendeklaration können Sie unter „Freier Modus“ ansehen:

▶ Wählen Sie ‹ Extras → Variableneditor → Freier-Modus› oder die Schaltfläche der Standard-Symbolleiste.

 *„Freier Modus“*

Auf dem Bildschirm erscheint das Editierfenster mit der Variablendeklaration, die Sie im syntaxgesteuerten Variablen-Editor erstellt haben:

02/02 AWB27-1307-D

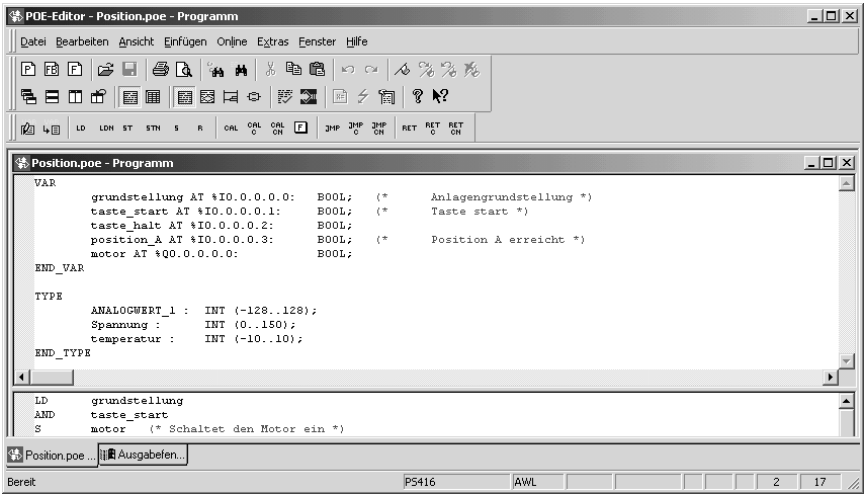


Abbildung 9: Deklarierte Variablen im Freien Modus

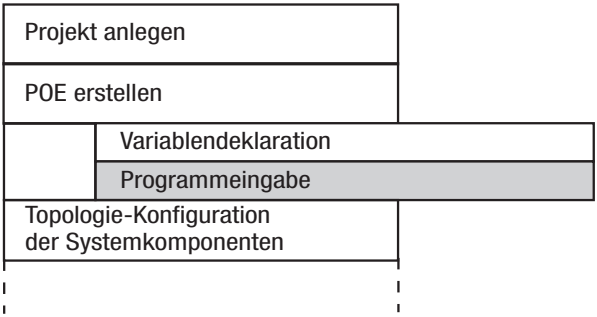
Im freien Variablen-Editor sind die deklarierten Variablen in der Form sichtbar, wie es der in der IEC-Norm beschriebenen Darstellung der Deklarationsblöcke entspricht: Mit den Schlüsselwörtern VAR, END_VAR und AT, mit den Trennzeichen (:) sowie den Kennzeichen für den Kommentar (*Kommentar*).

Bei der Deklaration im Syntax-Modus werden die Schlüsselwörter automatisch eingesetzt. Auch die Reihenfolge der Deklaration der einzelnen Gültigkeitsbereiche wird automatisch geregelt. Die Deklaration im Syntax-Modus ist daher sehr unkompliziert. Sie hat jedoch den Nachteil, dass nur die Variablenliste für einen Gültigkeitsbereich sichtbar ist, z. B. nur Inputvariablen oder lokale Variablen.

Im freien Variablen-Editor kann man die Variablen ebenfalls deklarieren, jedoch muss man dazu Schlüsselwörter, Trennzeichen und Syntaxregeln kennen. Der freie Variablen-Editor bietet Ihnen die komplette Übersicht über alle deklarierten Variablen

aller nutzbaren Gültigkeitsbereiche und die Eingabe geht schneller, sofern man die erforderlichen Kenntnisse hat. Im freien Variablen-Editor können Sie auch bequem Korrekturen vornehmen.

Programmeingabe im AWL-Editor



Ist die Variablendeklaration abgeschlossen, beginnen Sie mit der Eingabe des Anweisungsteils des Programms.

▶ Wechseln Sie durch Klicken mit der linken Maustaste in den Anweisungsteil.

Die Programmiersprache AWL ist bereits voreingestellt oder kann über die Symbol-Schaltfläche oder über das POE-EDITOR-Menü ‹ Extras → Programmiersprache → AWL› gewählt werden.

„Verwendung des AWL-Editors“

Die Programmiersprache AWL ist eine textuelle, zeilenorientierte Sprache und hat folgenden Aufbau:

```
Label (optional) – Operator – Operand – Kommentar (optional)
```

Zwischen den einzelnen Elementen ist mindestens ein Tabulator oder ein Leerzeichen einzugeben. Jede Anweisung muss mit einer neuen Zeile beginnen. Der Kommentar wird zwischen die Zeichenkombinationen „öffnende Klammer/Stern“ (* und „Stern/

schließende Klammer“ *) geschrieben. Setzen Sie vor jeden Kommentar einen Tabulator, damit der Kommentar von der Befehlszeile abgesetzt wird. Zeilen- und Spaltennnummer der aktuellen Cursorposition sind rechts in der Statusleiste am unteren Fensterrand angezeigt.

Eine Syntaxcolorierung, durch die Operatoren und Kommentare farbig darstellt werden, erleichtert Ihnen das Editieren der Anweisungsliste.

Operatoren einfügen

Die verfügbaren Operatoren sind als Symbole in der Werkzeugleiste dargestellt. Diese ändert sich mit der von Ihnen eingestellten Programmiersprache.

Abbildung 10: Toolbar AWL

Abbildung 11: Toolbar ST

Abbildung 12: Toolbar KOP

Abbildung 13: Toolbar FBS

Ein Tooltip (erscheint, wenn Sie mit dem Mauszeiger auf die Schaltfläche gehen) gibt Ihnen eine kurze Beschreibung des Operators.

Sie können die Befehle natürlich auch über die Tastatur eingeben.

Variablen einfügen

Bereits deklarierte Variablen können Sie im Anweisungsteil einfügen über die rechte Maustaste und „Einfügen Variable...“ bzw. den entsprechenden Menüpunkt unter „Einfügen“.

Es öffnet sich ein Dialog, in dem die von Ihnen erstellten Variablen und ggf. Funktionsbausteine, abhängig von deren jeweiligem Gültigkeitsbereich, angezeigt werden.

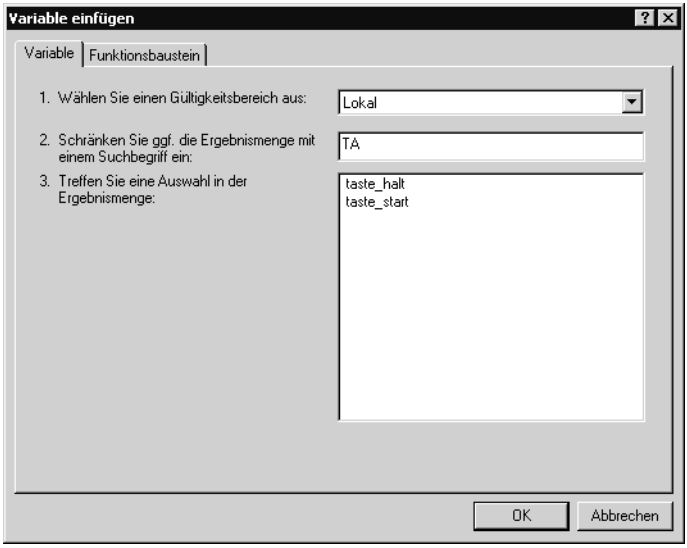


Abbildung 14: Variable einfügen

▶ Geben Sie das Beispiel-Programm wie dargestellt ein und berücksichtigen Sie dabei die genannten Regeln.

02/02 AWB27-1307-D

```
LD  grundstellung
AND taste_start
S   motor       (*Schaltet den Motor ein*)
LD  taste_halt
OR  position_A
R   motor       (*Schaltet den Motor aus*)
```

Nach der Programmeingabe hat das Fenster „Anweisungsteil“ folgenden Inhalt:

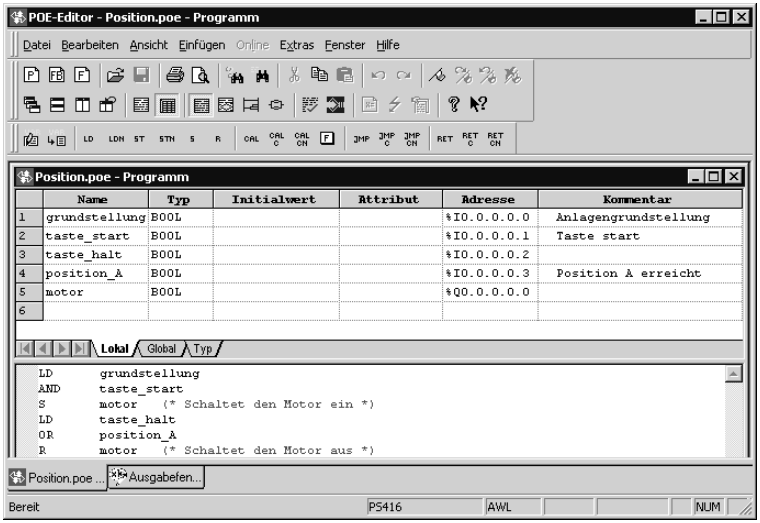


Abbildung 15: Das fertige Beispiel-Programm

Programm speichern

POEs werden innerhalb eines Projekts immer im Ordner „source“ (voreingestellt), bzw. dessen Unterordnern gespeichert. Falls eine POE, z. B. ein Funktionsbaustein, in einem Unterordner gespeichert werden soll, müssen Sie diesen zuvor im NAVIGATOR über den Menüpunkt „Bearbeiten“ oder das Kontextmenü zum Quellen-Dateibaum im Fenster „Baumansicht“ anlegen.

▶ Speichern Sie das fertige Beispiel-Programm mit ‹ Datei → Speichern unter...› oder durch Betätigung der Symbol-Schaltfläche:

„Aktuelle POE speichern“

In dem sich öffnenden Fenster ist der dargestellte Ordner „SOURCE“ des aktuellen Projekts der Zielordner für die POE.

▶ Geben Sie den POE-Namen „position“ ein und bestätigen Sie die Eingabe mit der Schaltfläche OK.

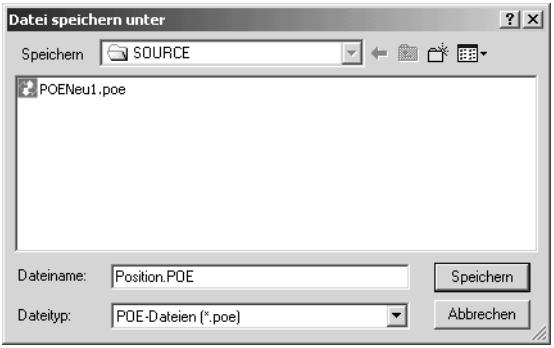


Abbildung 16: POE speichern unter

02/02 AWB27-1307-D

Syntaxprüfung
Bei der Syntaxprüfung wird die gerade bearbeitete POE automatisch gespeichert. Wurde die POE neu angelegt, so erscheint die Aufforderung, die POE mit einem neuen Namen zu speichern.

▶ Wählen Sie ‹ Datei → Syntaxprüfung› oder die entsprechende Schaltfläche.


„Syntaxüberprüfung“

Wenn Sie alles wie vorgeschlagen eingegeben haben, erscheint in der Statuszeile eine Meldung über die fehlerfrei beendete Syntaxprüfung, andernfalls öffnet sich das POE-EDITOR-Ausgabefenster mit einer Fehlerliste:

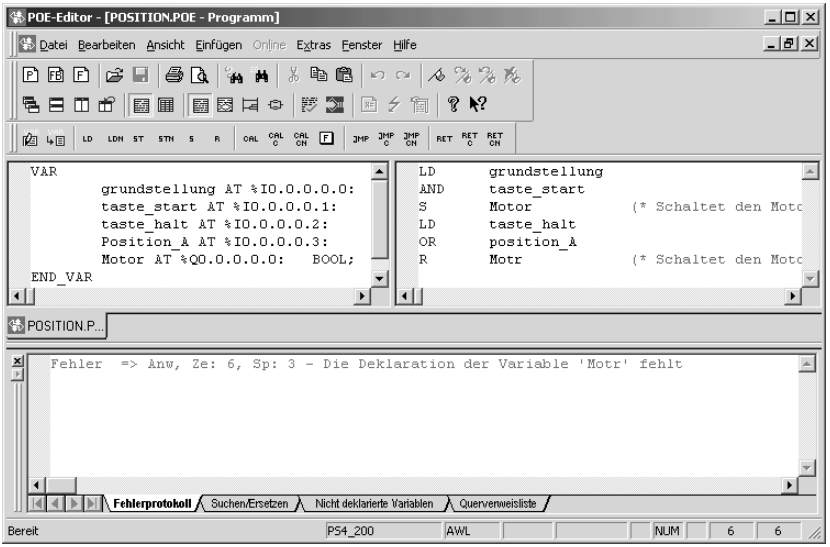


Abbildung 17: Programmcode-Erzeugung mit Fehlerliste

Von links nach rechts sind angegeben:

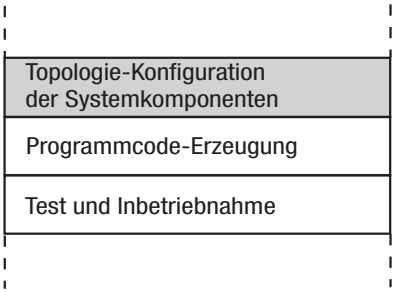
Fehlerort (Deklarationsteil oder Anweisungsteil),
Zeilennummer,
Spaltennummer und
eine Kurzbeschreibung des Fehlers.

Falls die Kurzbeschreibung zur Identifikation des Fehlers nicht ausreicht, markieren Sie die Fehlerzeile und drücken F1, um weitere Informationen zu einem Fehler anzufordern.

Fehlerbehandlung: Wenn Sie auf eine Fehlerzeile doppelt klicken oder diese markieren und anschließend die Eingabetaste betätigen, springt der Cursor an den Fehlerort.

02/02 AWB27-1307-D

7 Topologie-Konfiguration



Vor der Programmcode-Erzeugung müssen Sie Ihr Hardware-System mit dem „TOPOLOGIE-KONFIGURATOR“ einstellen. Dabei geben Sie die erforderlichen Informationen zum Aufbau des Systems – der Topologie – an und konfigurieren die Systemkomponenten.

▶ Starten Sie den TOPOLOGIE-KONFIGURATOR unter dem Menüpunkt „Werkzeuge“ oder das entsprechende Symbol im NAVIGATOR.

 *„TOPOLOGIE-KONFIGURATOR“*

Der TOPOLOGIE-KONFIGURATOR wird gestartet.

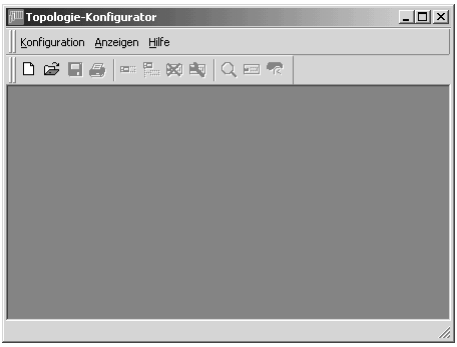


Abbildung 18: TOPOLOGIE-KONFIGURATOR

Topologie-Konfiguration für PS4 erstellen

Die Erstellung einer Topologie-Konfiguration wird am folgenden Beispiel erläutert. Für das Beispiel wird eine Steuerung PS4-141-MM1, PS4-151-MM1, PS4-201-MM1, PS4-271-MM1 oder PS4-341 benötigt.

Alle verwendeten Baugruppen des Automatisierungssystems und die erforderlichen Informationen zu jeder Komponente müssen angegeben werden. Für die Beispiel-Topologie ist jedoch nur die entsprechende Kompaktsteuerung PS4 relevant. Wenn an Ihre Steuerung weitere Baugruppen angeschlossen sind, können Sie sie für das Beispiel unberücksichtigt lassen.

Alle Aktionen zur Konfiguration einer Topologie können Sie über das Menü oder über die Schaltflächenleiste des TOPOLOGIE-KONFIGURATORs aufrufen. Die Erklärungen zu den einzelnen Schaltflächen können Sie den Tooltips entnehmen.

02/02 AWB27-1307-D

Deklarieren Sie zuerst die CPU:

▶ Legen Sie eine neue Topologie-Konfiguration an:

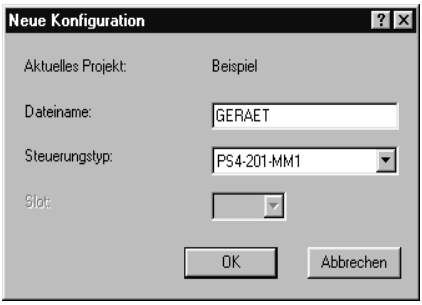


Abbildung 19: Dialogfeld „Neue Konfiguration“

▶ Geben Sie den Namen „GERAET“ in das Feld „Dateiname“ ein, wählen Sie „PS4-141-MM1“ oder Ihren vorhandenen PS4-Typ aus dem Listenfeld „Steuerungstyp“ und bestätigen Sie mit der Schaltfläche OK.

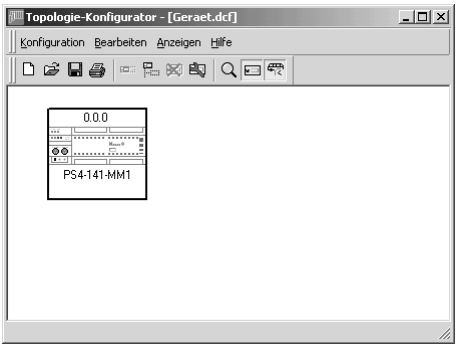


Abbildung 20: Topologie mit PS4-141-MM1

Die SPS muss für das vorliegende Beispiel nicht als Slave oder Master parametriert werden. Nur wenn die Steuerung in einem vernetzten Steuerungssystem über die Suconet-K-Schnittstelle an ein Netzwerk angeschlossen ist, muss die CPU als Slave oder Master eingerichtet werden. Sie können die Einstellungen der SPS im Parameterfenster festlegen. Das Fenster öffnet sich über das Kontextmenü, die Aktion ‹ Bearbeiten → Parametrieren› oder über die Schaltfläche „Parametrieren“.

▶ Speichern Sie die Konfiguration.

Topologie-Konfiguration für PS416 erstellen

Für dieses Beispiel wird die Steuerung PS416-CPU-400 eingesetzt.

Alle verwendeten Baugruppen des Automatisierungssystems und die erforderlichen Informationen zu jeder Komponente müssen angegeben werden. Für die Beispiel-Topologie sind jedoch nur die CPU und die digitale Ausgabebaugruppe PS416-OUT-400 sowie die Eingabebaugruppe relevant.

Gehen Sie bei der Erstellung der PS416-Konfiguration analog zur Erstellung der PS4-Konfiguration vor:

▶ Legen Sie eine neue Konfiguration an.

▶ Wählen Sie entsprechend Ihrer Hardware eine PS416-CPU-200/-300/-400 aus.

02/02 AWB27-1307-D

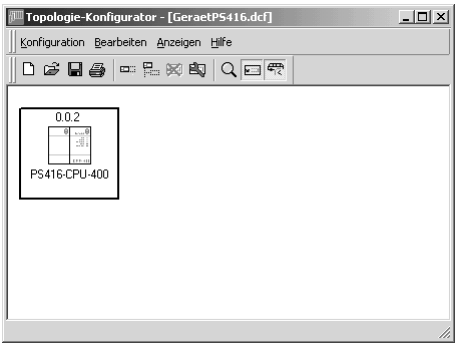


Abbildung 21: Topologie mit PS416

- Betätigen Sie die Schaltfläche „Lokal erweitern“ und wählen Sie aus der Liste die Eingabebaugruppe „PS416-INP-400“ oder „PS416-INP-401“ und bestätigen Sie mit OK.
- Gehen Sie ebenso vor, um die Ausgabebaugruppe „PS416-OUT-400“ in die Konfiguration aufzunehmen.
- Markieren Sie nacheinander beide Baugruppen und geben Sie nach Betätigung der Schaltfläche „Parametrieren“ im Dialogfeld die Byte-Adressen ein, die Sie auf den Baugruppen per DIP-Schalter codiert haben.

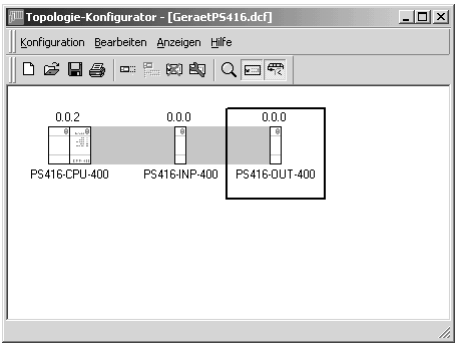
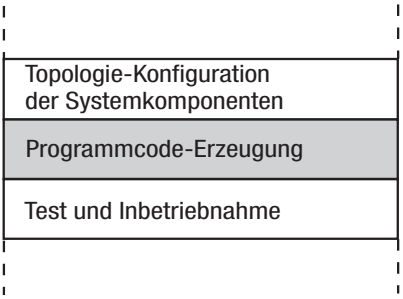


Abbildung 22: Erweiterte Konfiguration

▶ Speichern Sie die Konfiguration.

02/02 AWB27-1307-D

8 Programmcode-Erzeugung



Die Programmcode-Erzeugung nehmen Sie in zwei Phasen vor:

Erstellung einer Generierliste, wobei die Elemente (Dateien) anzugeben sind, die bei der Programmcode-Erzeugung berücksichtigt werden sollen: Die Programm-POE (mit zugehörigen Funktionsbausteinen und Funktionen), die Topologie-Datei und die Programmparametrierung. Es kann pro Programm-POE eine Generierlste erzeugt werden. Sie erhält automatisch den Namen dieser Programm-POE. Anschließend aktualisiert die Sucosoft S40 die Liste selbstständig, sobald Sie eine Projekt-POE, die Programmparametrierung oder die Topologie ändern.

Generierung des Programmcodes anhand der Generierliste.

Alle Aktionen zum Erzeugen eines ablauffähigen Anwenderprogramms können über die Bearbeitungsschaltflächen der Symbolleiste ausgewählt werden.

Generierliste erstellen

- ▶ Legen Sie in der NAVIGATOR-Werkzeugleiste die Steuerung fest, für die Sie den Programmcode erzeugen wollen.
- ▶ Wählen Sie ‹ Generierung → Generierliste neu...› oder die entsprechende Schaltfläche.

 *„Eine Generierliste erzeugen“*

Es öffnet sich das Dialogfeld „Generierliste neu“:

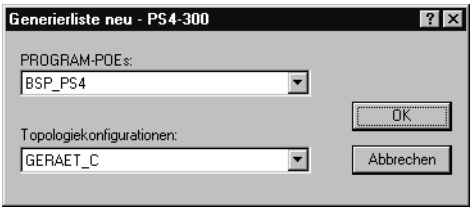


Abbildung 4: Dialogfeld „Generierliste neu“

In diesem Dialog wählen Sie die Programm-POE und die Topologie aus. Es werden nur Topologien angezeigt, die zu der Steuerung passen, die Sie im NAVIGATOR eingestellt haben.

- ▶ Betätigen Sie die Schaltfläche OK.

Die Generierliste wird erstellt.

02/02 AWB27-1307-D

Programmparametrierung vornehmen

Programmparameter, wie z. B. Compiler-Optionen für den jeweils gewählten Steuerungstyp oder relevante Bedingungen für die Programmausführung wie maximale Zykluszeit, Passwort und Merkerbereich sind von Ihnen festzulegen. Dies geschieht über die entsprechende Schaltfläche.

 *„Programmparametrierung vornehmen“*

In unserem Beispiel genügen jedoch die voreingestellten Standardparameter.

Programmcode generieren

▶ Wählen Sie ‹ Generierung → Programmcode generieren› oder die entsprechende Schaltfläche.

 *„Programmcode generieren“*

Das ablauffähige Programm wird anhand der aktuell ausgewählten Generierliste erzeugt. Im Ausgabefenster kann der Fortschritt der Generierung beobachtet werden. Nach einem fehlerfreien Ablauf erscheint die entsprechende Meldung.

Treten während der Programmcode-Erzeugung Fehler auf, werden diese im Ausgabefenster mit der entsprechenden Meldung ausgegeben.

Gemeldete Fehler können mit ‹ Projekt → Ausgabe drucken› gedruckt werden.

Fehlerbehandlung

Von links nach rechts wird angegeben:

Fehlerort (Deklarationsteil oder Anweisungsteil),

Zeilennummer,

Spaltennummer und

Name und Pfad der POE.

▶ Betätigen Sie bei markierter Fehlerzeile die Eingabetaste oder klicken Sie doppelt auf eine Fehlerzeile.

Es öffnet sich der POE-EDITOR. Der Cursor befindet sich an der Stelle der POE, in welcher der Fehler gefunden wurde. Die fehlerhafte POE-Zeile ist der Grundeinstellung entsprechend rot markiert.

▶ Beheben Sie den Fehler und starten Sie erneut die Programmcode-Erzeugung.

02/02 AWB27-1307-D

9 TEST & INBETRIEBNAHME

Mit Hilfe der TEST & INBETRIEBNAHME (T & I) können Sie Ihr Programm in die Steuerung übertragen und testen. Dazu sind folgende Einzelschritte notwendig:

Verbindung zwischen Programmiergerät und SPS definieren und aufbauen

Bei der PS416 oder PS4-300 erstmalig ein Betriebssystem in die Steuerung übertragen.

Programm in die SPS übertragen

Programm starten

Programm testen.

▶ Starten Sie die T & I.

„T & I“

Es öffnet sich das Hauptfenster T & I mit dem Fenster „Verbindungsliste“:

Abbildung 23: Fenster „Verbindungsliste“

Programmiergerät und Steuerung verbinden

Verbindung definieren

Zunächst müssen Sie die Verbindung zwischen dem Programmiergerät und der Steuerung definieren. Im Fenster „Verbindungsliste“ sind dazu einige voreingestellte Standard-Werte eingeblendet, die Sie ergänzen oder bei Bedarf ändern können.

▶ Klicken Sie in das Feld in der Spalte „Gerätename“ und geben Sie z. B. den Namen „Beispiel_1“ für die anzukoppelnde Steuerung ein.

Sollte Ihr Programmierkabel nicht an die standardmäßig eingestellte Schnittstelle COM1 angeschlossen sein, können Sie eine andere serielle Schnittstelle einstellen. Klicken Sie dazu auf das Feld in der Spalte „Schnittstelle“ und wählen aus der eingeblendeten Liste den passenden Anschluss.

Die Felder in den Spalten „Strang“ und „Teilnehmer“ sind für unser Beispiel nicht von Bedeutung und werden daher leer gelassen.

▶ Speichern Sie die eingegebenen Daten mit ‹ Datei → speichern› . Die Datei erhält automatisch den Namen „CONNECT.CCF“.

02/02 AWB27-1307-D

Verbindung aufbauen

So verbinden Sie Ihr Programmiergerät mit der Steuerung.

Voraussetzung: In der T & I ist nur das Fenster „Verbindungsliste“ geöffnet.

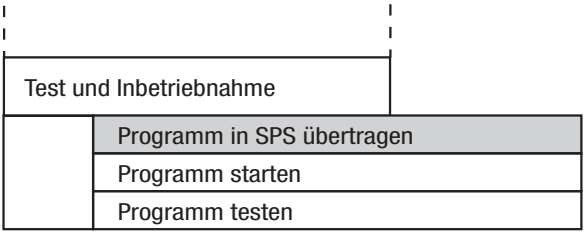
▶ Koppeln Sie die Steuerung über ‹ Gerät → Ankoppeln› oder die entsprechende Schaltfläche an.

 *„Gerät ankoppeln“*

Die erfolgreiche Verbindung wird in der Verbindungsliste durch ein geändertes Geräte-Symbol angezeigt.

Abbildung 24: Darstellung einer erfogreichen Verbindung

Betriebssystem und Programm übertragen



Wenn sich in Ihrer Steuerung kein Betriebssystem oder ein älteres Betriebssystem befindet, müssen Sie zuerst eine neue Version des Betriebssystems in die Steuerung übertragen. Die jeweils aktuelle Version des Betriebssystems wird mit der Sucosoft S40 ausgeliefert.

Betriebssystem von der Sucosoft in die Steuerung übertragen

Der Transfer eines Betriebssystems ist beim Steuerungstyp PS4-200 nicht notwendig. Dieser Abschnitt bezieht sich nur auf die Steuerungen vom Typ PS4-300 und PS416.

▶ Wählen Sie ‹ Gerät → Transfer/Datei-Manager› oder die entsprechende Schaltfläche an.

„Transfer/Datei-Manager“

Es öffnet sich das Dialogfeld „Transfer/Datei-Manager“. Die Karteikarte „Programmiergerät (PRG)“ ist voreingestellt. In der Liste stehen die Namen der ausführbaren Dateien vom Typ „.PCD“ (Programmcode-Dateien).

▶ Um das Betriebssystem auf die Steuerung zu übertragen, wählen Sie im Listenfeld „Dateiformat“ den Eintrag „Betriebssystem (*.OSF)“ und markieren das gewünschte Betriebssystem.

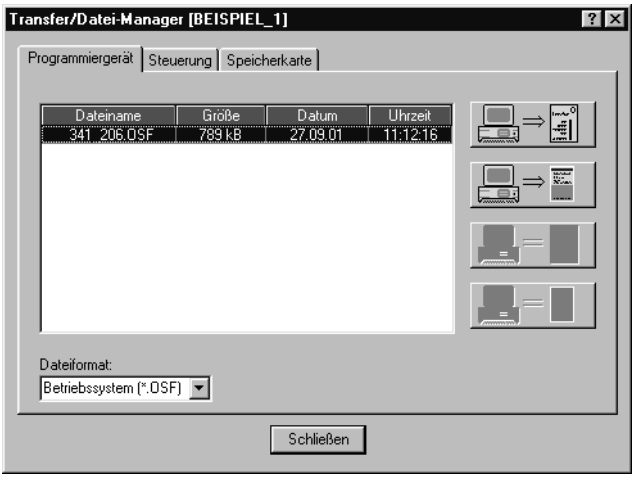


Abbildung 25: Ausgewähltes Betriebssystem für Transfer

▶ Wählen Sie die Schaltfläche „Übertragen → SPS“.

„Übertragen zur SPS“

Achtung!
Wenn Sie ein vorhandenes Betriebssystem mit einer neuen Version überschreiben, werden gleichzeitig alle auf der Steuerung vorhandenen Anwenderprogramme und Daten gelöscht.

Die Übertragung dauert einige Minuten, abhängig von der eingestellten Baudrate. Eine Fortschrittsanzeige unterrichtet Sie über den Stand des Transfers.

Programm von der Sucosoft in die Steuerung übertragen

Dieser Abschnitt gilt für alle Steuerungstypen gleichermaßen.

▶ Wechseln Sie wie zuvor beschrieben in das Dialogfeld „Transfer/Datei-Manager“.

Die Karteikarte „Programmiergerät (PRG)“ ist voreingestellt. In der Liste stehen die Namen der ausführbaren Dateien vom Typ „.PCD“ (Programmcode-Dateien).

▶ Wählen Sie ggf. im Listenfeld „Dateiformat“ den Eintrag „Programm (*.PCD)“.

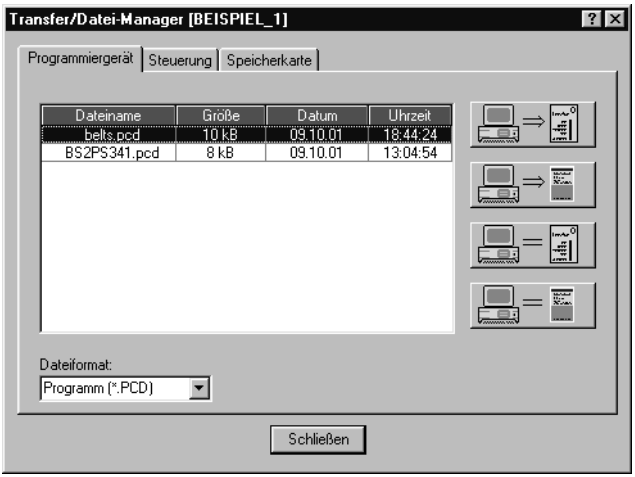


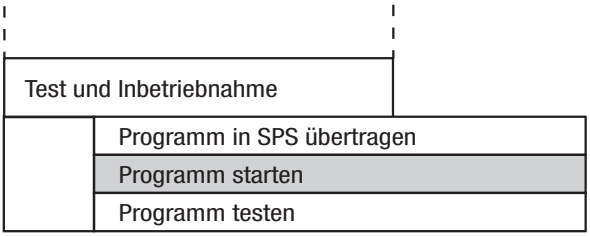
Abbildung 26: Programmdatei auswählen

▶ Markieren Sie eine Programmcode-Datei, die Sie in die Steuerung übertragen wollen und wählen Sie die Schaltfläche „Übertragen → SPS“.

„Übertragen zur SPS“

Eine Fortschrittsanzeige unterrichtet Sie über den Stand des Transfers.

Programm starten



02/02 AWB27-1307-D

▶ Wechseln Sie nun im Dialogfeld „Transfer/Datei-Manager“ zur Karteikarte „Steuerung“ und betätigen Sie die Schaltfläche „Kaltstart“, um das Programm zu starten.

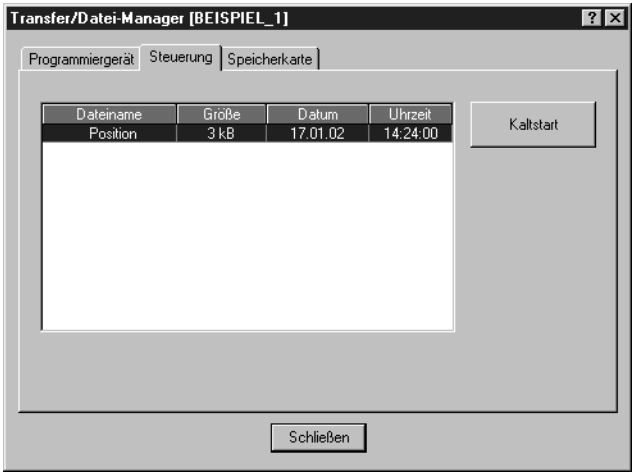


Abbildung 27: Programm starten

Nach erfolgreichem Transfer zeigt die CPU-Baugruppe mit der grünen „Ready“-Anzeige ihre Betriebsbereitschaft an. Konnte das Programm nicht gestartet werden, erhalten Sie eine Systemmeldung mit dem Hinweis auf die mögliche Ursache.

Weitere Informationen zum Zustand des Programms und der CPU bekommen Sie im Dialogfeld „Status und Diagnose“, das im folgenden Abschnitt beschrieben ist.

Status und Diagnose

▶ Wählen Sie ‹ Gerät → CPU-Status› oder die entsprechende Schaltfläche.

 *„CPU-Status“*

Es erscheint das Dialogfeld „Status und Diagnose“. Hier können Sie z. B. überprüfen,

ob das Anwenderprogramm ordnungsgemäß abgearbeitet wird (CPU-Status),

welche möglichen Ursachen ein fehlerhaftes Anwenderprogramm hat (CPU-Diagnose),

welche Eigenschaften Ihr Anwenderprogramm besitzt (Programmstatus).

Detaillierte Informationen zum Dialogfeld „Status und Diagnose“ bekommen Sie im Handbuch „S40-Benutzeroberfläche“ (AWB2700-1305-D). Im Folgenden erhalten Sie nur einen kurzen Überblick.

CPU-Status

▶ Klicken Sie ggf. auf die Karteikarte „CPU-Status“.

Je nach eingestellter Steuerung erscheint eines der nachfolgenden Dialogfelder:

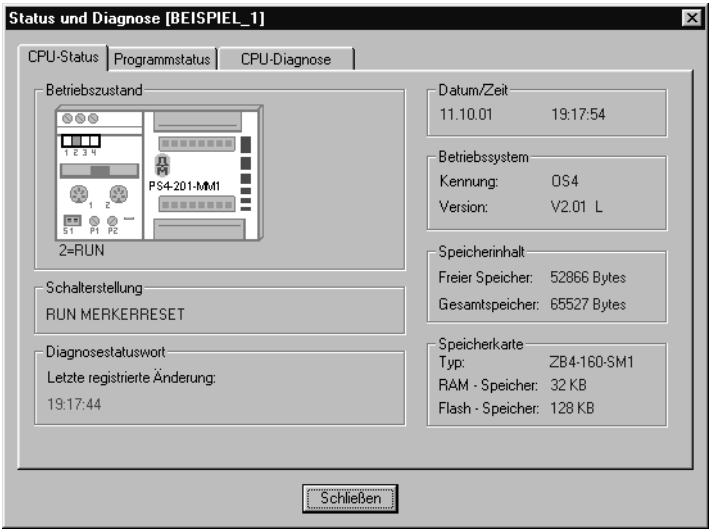


Abbildung 28: CPU-Status für PS4-200 (PS4-300 analog)

Abbildung 29: CPU-Status für PS416

Der Betriebszustand RUN zeigt an, dass das Anwenderprogramm abgearbeitet wird. Falls der Betriebszustand NOT READY angezeigt wird, liegt ein Fehler vor. Mögliche Ursachen für diesen Fehler erfahren Sie in der CPU-Diagnose.

02/02 AWB27-1307-D

CPU-Diagnose

▶ Klicken Sie im Dialogfeld „Status und Diagnose“auf die Karteikarte „CPU-Diagnose“.

Je nach eingestellter Steuerung erscheint eines der nachfolgenden Dialogfelder mit einer Auflistung der Diagnosebits.

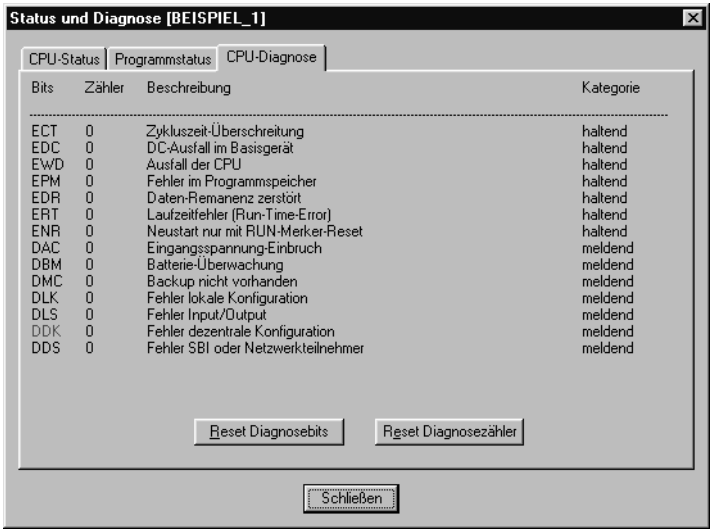


Abbildung 30: CPU-Diagnose für PS4-200 (analog für PS4-300)

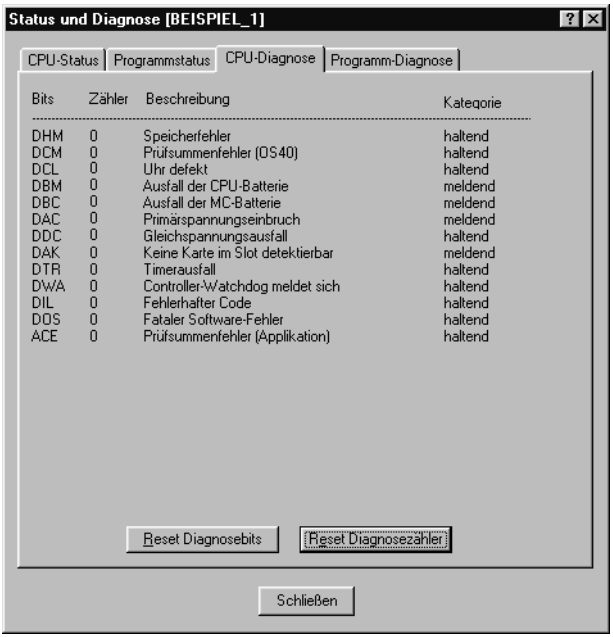


Abbildung 31: CPU-Diagnose für PS416

Bei der PS416 können Sie über die Karteikarte „Programm-Diagnose“ noch weitere Informationen bekommen:

02/02 AWB27-1307-D

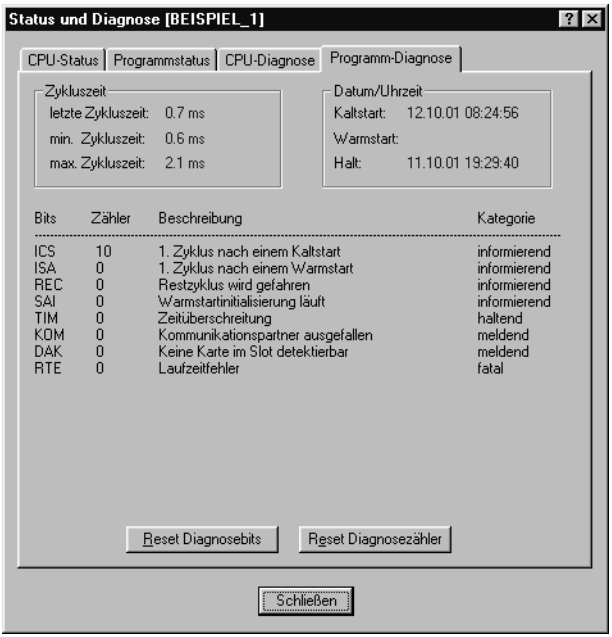


Abbildung 32: Programm-Diagnose PS416

Programmstatus

▶ Klicken Sie auf die Karteikarte „Programmstatus“.

Je nach eingestellter Steuerung erscheint eines der nachfolgenden Dialogfelder. Hier können Sie z. B.

die Steuerung – abhängig von der Stellung des CPU-Betriebsartenschalters – kalt- bzw. warmstarten,

die Steuerung per Mausklick anhalten oder

anhand der Zykluszeit ablesen, wie lange Ihr Programm für die Abarbeitung benötigt.

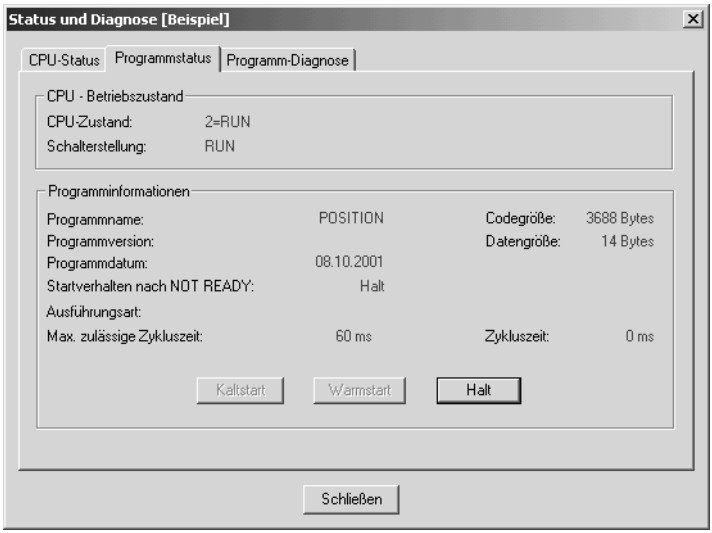
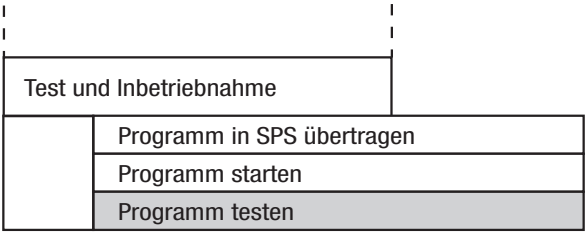


Abbildung 33: Programmstatus PS4-200

02/02 AWB27-1307-D

Abbildung 34: Programmstatus PS416

Programm testen



Voraussetzung: In der T & I ist nur das Fenster „Verbindungsliste“ geöffnet.

▶ Wählen Sie ‹ Gerät → Programm› oder die entsprechende Schaltfläche an.

Programm

Die Menü- und Symbolleiste werden entsprechend angepasst.

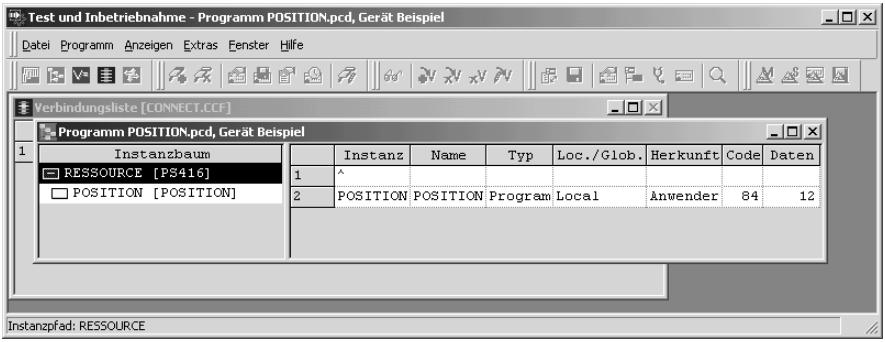


Abbildung 35: T & I mit geöffneter Programm-POE

Das Fenster „Programm“ besteht aus zwei Teilen. Im linken Teil „Instanzbaum“ steht der Begriff „Ressource“. „Ressource“ bezeichnet die oberste Ebene der Programmstruktur und steht für die CPU. Im rechten Teil des Dialogfeldes steht der Name der Programm-POE „POSITION“.

02/02 AWB27-1307-D

▶ Klicken Sie doppelt auf die Zeile „Ressource“, um die weitere Programmstruktur anzuzeigen, in diesem Fall nur die Programm-POE „POSITION“.

Werden in einer Programm-POE weitere POEs – Funktionsbausteine und Funktionen – aufgerufen, wird dies durch ein Pluszeichen neben dem Programmnamen markiert. Auf diese Weise können Sie alle Bestandteile einer Programmstruktur auflisten und auswählen sowie online ansehen und Änderungen vornehmen.

Wenn Sie eine POE online bearbeiten, muss das dazugehörige Projekt geladen sein. Außerdem dürfen Sie an den Versionen der POEs und der Gerätekonfiguration, aus denen der Programmcode erzeugt wurde, inzwischen nichts geändert haben. Für unseren Beispiel-Durchgang gehen wir davon aus, dass Sie die Anweisungen befolgt haben und daher beide Voraussetzungen erfüllt sind.

▶ Um die POE online anzusehen oder zu ändern: Markieren Sie den POE-Namen „POSITION“ im linken Feld und wählen Sie den Menüpunkt ‹ Programm → POE anzeigen ändern› oder die entsprechende Schaltfläche an.

 *POE anzeigen/ändern*

Es öffnet sich der POE-EDITOR im Online-Modus, in dem Ihre POE „POSITION“ geöffnet und in den Vordergrund geschaltet ist.

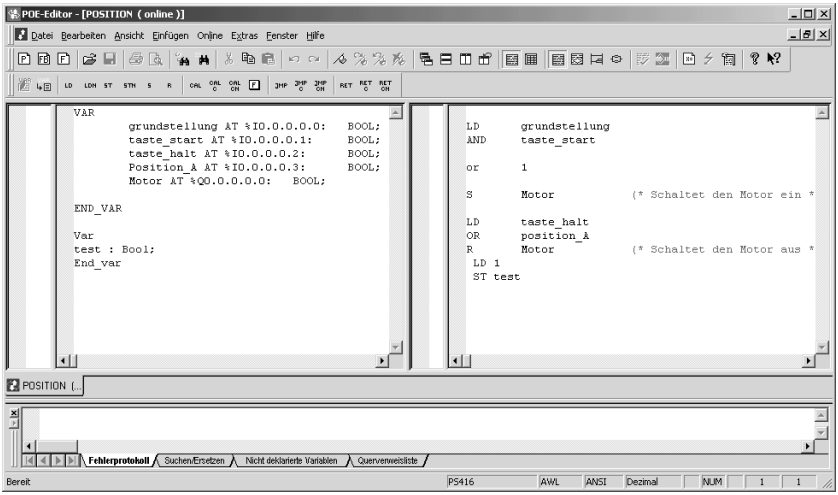


Abbildung 36: POE-EDITOR im Online-Modus

Das Menü „Online“ und die Schaltflächen für folgende Aktionen werden verfügbar:

Zustandsanzeige von Variablen,

Aktivierung vorgenommener Änderungen im Anweisungsteil

Übergabe von Variablen an das Variablenfenster.

POE prüfen - Variablenzustände anzeigen

▶ Wählen Sie im POE-EDITOR-Fenster ‹ Online → Zustandsanzeige› oder klicken Sie auf die Schaltfläche „Zustandsanzeige“, um die Zustandsanzeige ein- oder auszuschalten.

 *„Zustandsanzeige“*

Es werden die Variablenzustände in Deklarations- und Anweisungsteil eingeblendet. In unserem AWL-Programm werden die Variablenzustände in der Zustandsspalte jeweils links von Deklarations- und Anweisungsteil dargestellt.

▶ Betätigen Sie nochmals das Symbol „Zustandsanzeige“ um das Aktualisieren der Variablenzustände zu beenden.

Auf dem Bildschirm werden die zuletzt übernommenen Zustände „eingefroren“ angezeigt. Diese Möglichkeit können Sie zur Fehlersuche verwenden – für unser Beispielprogramm ist dies nicht notwendig.

Möchten Sie sich in einem umfangreichen Programm die Zustände von bestimmten Variablen anzeigen lassen, die im Programm weiter auseinander liegen und daher nicht gleichzeitig im Fenster angezeigt werden, oder möchten Sie Variablen aus verschiedenen POEs anzeigen lassen, haben Sie die Möglichkeit, Variablen für die Anzeige auszuwählen und sie im Variablenfenster anzuzeigen. Sie können sich die Zustände von Variablen unterschiedlicher POEs zur gleichen Zeit im Variablenfenster und durch den POE-EDITOR im Online-Modus ansehen. Diese Eigenschaft der Sucosoft ist besonders hilfreich beim Test von Programmen, die Funktionsbausteine aufrufen. Bei unserem Mini-Programm ist dies natürlich nicht relevant.

Probieren Sie die Funktion aber trotzdem einmal an den beiden Variablen „grundstellung“ und „motor“ aus:

▶ Positionieren Sie den Cursor auf der Variablen „grundstellung“ und wählen Sie „Online → Variable übergeben“ oder die entsprechende Schaltfläche.

 *„Variable übergeben“*

▶ Wiederholen Sie dies mit der Variable „motor“.

Das Variablenfenster öffnet sich im Hintergrund und kann nach Wechsel zum Hauptfenster der T & I eingesehen werden.

▶ Aktivieren Sie das Fenster „Verbindungsliste" und klicken Sie auf die Schaltfläche „Variablenfenster“.

Das Variablenfenster wird in den Vordergrund geschaltet.

▶ Klicken Sie in der Variablenliste (linkes Fenster) doppelt auf „POSITION“, um die beiden übergebenen Variablen im Infoteil (rechtes Fenster) anzuzeigen.

▶ Schalten Sie die Zustandsanzeige für die Variablen im Fenster über den Menüpunkt ‹ Variablen Status anzeigen› oder die entsprechende Schaltfläche ein, damit die aktuellen Zustände der ausgewählten Variablen angezeigt werden.

 *„Status anzeigen“*

02/02 AWB27-1307-D

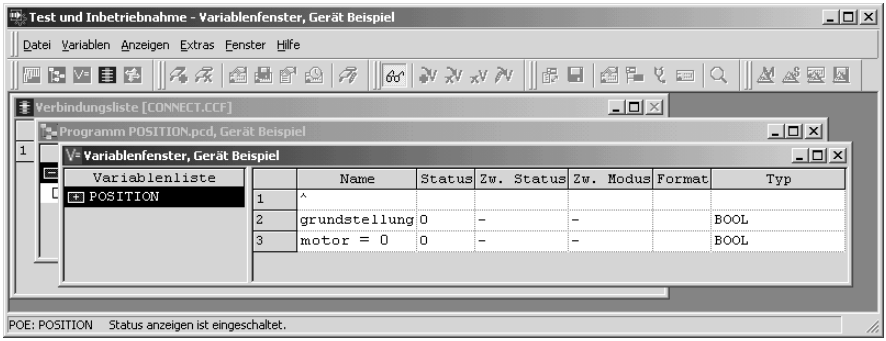


Abbildung 37: Variablenfenster mit Zustandsanzeige

Wenn Sie gezielt eine Variable (z. B. „grundstellung") im linken oder rechten Teilfenster markieren, wird bei einer PS4-300 oder PS416 die Schaltfläche zum Zwangssetzen einer Variablen im Zustand RUN verfügbar. Nähere Informationen zum Zwangssetzen (Forcen) finden Sie im Handbuch „S40-Benutzeroberfläche" (AWB2700-1305-D).

- ▶ Beenden Sie die Statusanzeige durch erneutes Anklicken auf die Schaltfläche „Status anzeigen".
- ▶ Schalten Sie mit der Tastenkombination ALT+TAB zum Fenster „Online-Editor" zurück.

POE online ändern

Wenn Sie die POE verändert haben, ist die Aktion ‹ Online → Aktivieren› und die Schaltfläche „Aktivieren“ in der Symbolleiste verfügbar.

 *„Aktivieren“*

▶ Wählen Sie ‹ Online → Aktivieren› oder klicken Sie auf die Schaltfläche.

Die durchgeführten Änderungen werden in der POE, im Programm und in der Steuerung aktualisiert.

▶ Wählen Sie ‹ Datei → Schließen› , um den POE-EDITOR im Online-Modus zu beenden.

Sie haben jetzt alle Werkzeuge der Sucosoft S40 zum Erstellen und Testen eines SPS-Programms an einem Beispielprogramm in der Sprache AWL kennengelernt. Im folgenden Kapitel wird das Beispielprogramm um die Programmiersprachen KOP und FBS erweitert.

02/02 AWB27-1307-D

10 Programmeingabe in KOP/FBS

Wie für die textuelle Programmiersprache AWL beschrieben, können Sie POEs auch in den grafischen Programmiersprachen KOP und FBS erstellen und in Betrieb nehmen. Der Anweisungsteil einer POE wird in KOP und FBS durch grafische Symbole dargestellt. Der Deklarationsteil wird in KOP und FBS genauso dargestellt wie in AWL oder ST.

Die Programmiersprachen AWL, KOP und FBS sind untereinander übersetzbar. Dadurch können Sie beispielsweise ein Programm in KOP erstellen und es in AWL oder FBS darstellen und weiter verarbeiten. Die Umschaltung zwischen ST und den grafischen Sprachen ist nicht möglich.

Zur Erstellung einer POE in der Programmiersprache KOP aktivieren Sie den Editor über den Menüpunkt ‹ Extras → Programmiersprache → KOP› oder die Symbol-Schaltfläche.

„POE-EDITOR KOP“

Anschließend beinhaltet die Sprachelementeleiste KOP-spezifische Operatoren, über die alle Aktionen für die Programmeingabe im POE-EDITOR KOP möglich sind.

Abbildung 38: Toolbar KOP

Zur Erstellung einer POE in der Programmiersprache FBS aktivieren Sie den Editor über den Menüpunkt ‹ Extras → Programmiersprache → FBS› oder die Symbol-Schaltfläche.

„POE-EDITOR FBS“

Anschließend beinhaltet die Sprachelementeleiste FBS-spezifische Operatoren, über die alle Aktionen für die Programmeingabe im POE-EDITOR FBS möglich sind.

Abbildung 39: Toolbar FBS

EinTooltip (erscheint, wenn Sie mit dem Mauszeiger auf die Schaltfläche gehen) gibt Ihnen eine kurze Beschreibung des Operators.

Beispielprogramm in KOP/FBS anzeigen

Das Beispielprogramm, das Sie im Offline-Modus des POE-EDITORs in AWL erstellt haben, können Sie sich in einer der grafischen Programmiersprachen anzeigen lassen und ändern.

POE in KOP anzeigen

▶ Wählen Sie die Programmiersprache KOP wie zuvor beschrieben.

Die POE „position“ wird im Anweisungsteil als Kontaktplan dargestellt:

02/02 AWB27-1307-D

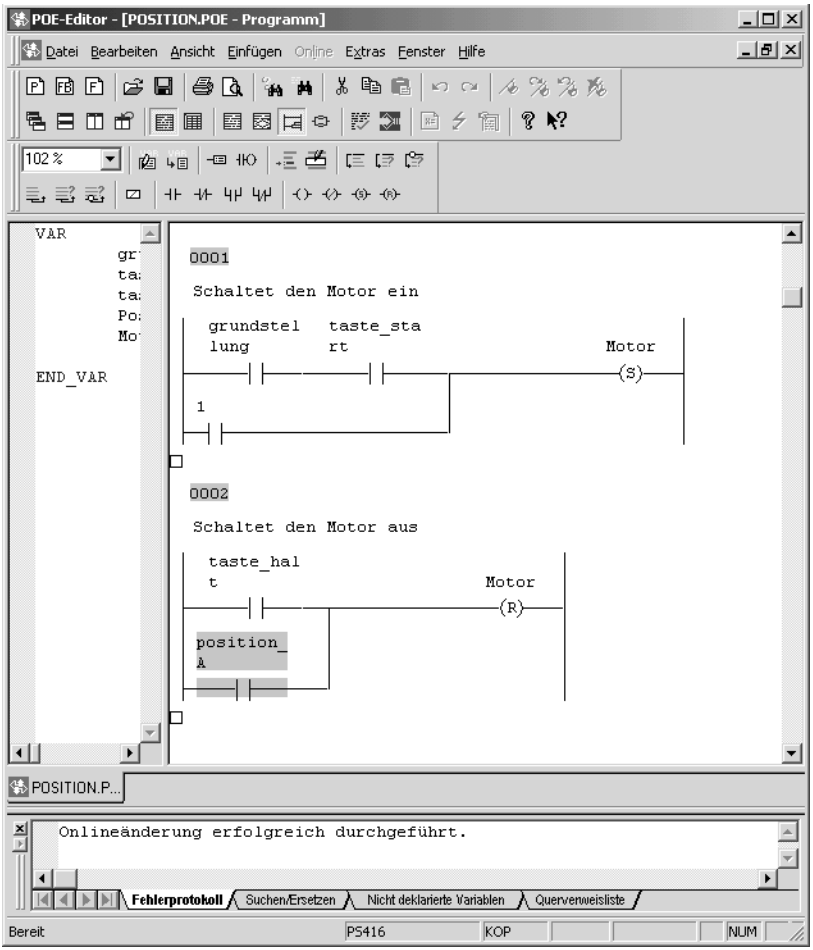


Abbildung 40: POE „position“ in der KOP-Darstellung

POE in FBS anzeigen

▶ Wählen Sie die Programmiersprache FBS wie zuvor beschrieben.

Die POE „position“ wird im Anweisungsteil durch Schaltsymbole dargestellt:

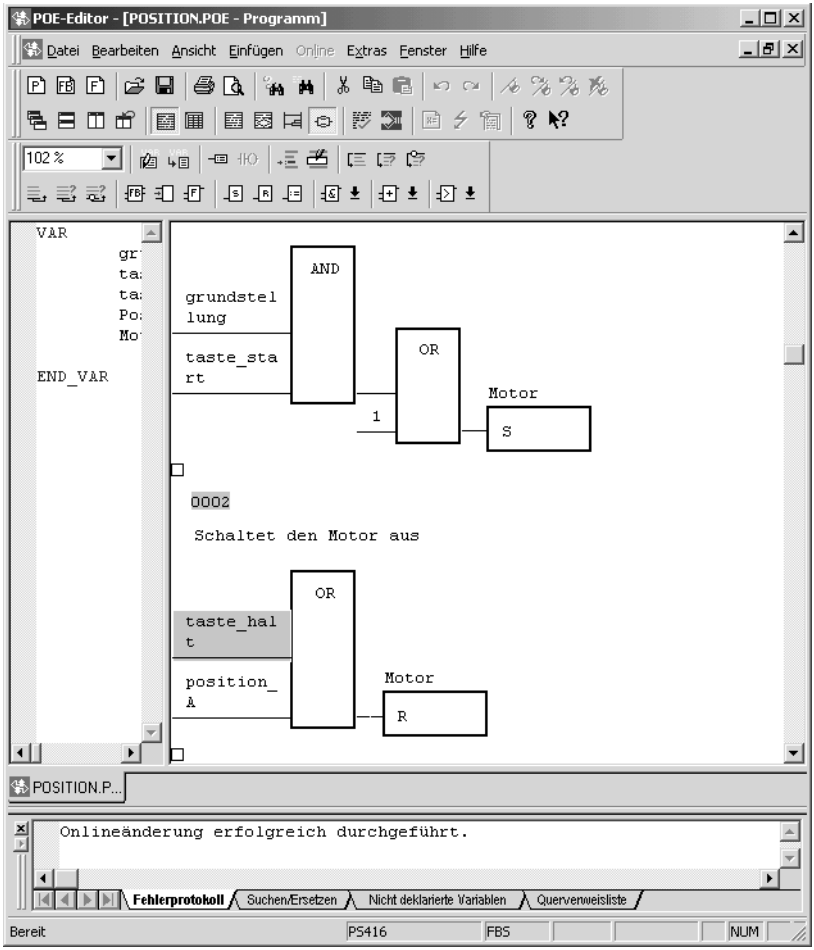


Abbildung 41: POE „position“ in der FBS-Darstellung

02/02 AWB27-1307-D

Der logische Inhalt des in KOP oder FBS dargestellten Programms entspricht der Aufgabenstellung des Beispiel-Programms.

Die Anweisungskommentare werden nicht den einzelnen grafischen Symbolen zugeordnet, sondern erscheinen jeweils als Netzwerk-Kommentar.

Programmeingabe in KOP

Voraussetzungen:

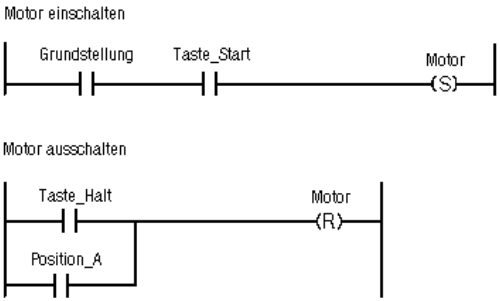
Sie kennen die Aufgabenstellung des Programmierbeispiels 1.

Das Projekt „beispiel“ ist angelegt.

Eine neue POE vom Typ „Programm“ ist angelegt.

Die benötigten Variablen sind deklariert.

Programmentwurf für das Beispielprogramm in KOP



Das Programm ist in zwei Netzwerke aufgeteilt.

Die im Netzwerk 0001 seriell geschalteten Kontakte „Grundstellung“ und „Taste_Start“ bilden die erforderliche UND-Verknüpfung, um den Motor einzuschalten.

Die im Netzwerk 0002 parallel geschalteten Kontakte „Taste_Halt“ und „Position_A“ bilden eine ODER-Verknüpfung, die als Ausschaltbedingung dient.

Im Netzwerkkopf steht ein Kommentar zur Kurzbeschreibung des Netzwerkprogramms.

Geben Sie jetzt das oben dargestellte Beispiel-Programm selbst in KOP ein. Benutzen Sie dazu die Schaltflächen der KOP-Sprachelementeleiste.

- ▶ Wählen Sie die Programmiersprache KOP über den Menüpunkt ‹ Extras → Programmiersprache → KOP› , oder die entsprechende Schaltfläche der Standard-Symbolleiste.
- ▶ Klicken Sie auf die Schaltfläche „Anfangs-KOP-Netzwerk einfügen“, um ein Anfangs-KOP-Netzwerk zu erstellen.

Das erstellte Netzwerk 0001 enthält ein Kontakt- und ein Ausgangssymbol. Beide Symbole erhalten automatisch die Bezeichnung „undef_opd“ (undefinierter Operand).

 *„Anfangs-KOP-Netzwerk“*

- ▶ Positionieren Sie den Cursor mit Mausklick zwischen dem ersten und letzten Symbol, um anschließend eine UND-Verknüpfung einzufügen.
- ▶ Klicken Sie auf die Schaltfläche AND der Sprachelementeleiste.

Seriell zum bestehenden Kontakt wird ein neuer Kontakt eingefügt.

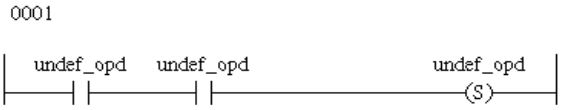
- ▶ Markieren Sie das Ausgangssymbol, um anschließend als Netzwerkabschluss einen Spulenkontakt mit Set-Bedingung einzufügen.

Parallel zu dem bestehenden Ausgang wird der Setz-Ausgang angelegt.

- ▶ Markieren Sie den „alten“ Ausgang und löschen Sie ihn.

Die Grafik für das Netzwerk 0001 ist fertig.

02/02 AWB27-1307-D



▶ Klicken Sie auf die Schaltfläche „Anfangs-KOP-Netzwerk“, um ein neues Netzwerk zu erstellen.

Das erstellte Netzwerk 0002 enthält ein Kontakt- und ein Ausgangssymbol.

▶ Markieren Sie das Kontaktsymbol, um anschließend eine ODER-Verknüpfung einzufügen.

▶ Klicken Sie auf die Schaltfläche „Einfügen eines Parallelkontaktes“.

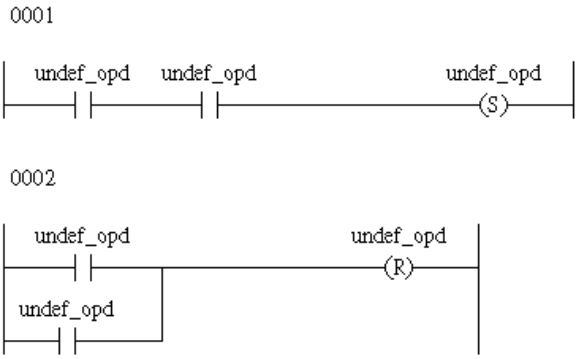
Parallel zu dem bestehenden Kontakt wird ein neuer Kontakt eingefügt.

▶ Markieren Sie das Ausgangssymbol per Mausklick, um anschließend als Netzwerkabschluss einen Spulenkontakt mit Reset-Bedingung einzufügen.

Parallel zu dem bestehenden Ausgang wird der Rücksetz-Ausgang angelegt.

▶ Markieren Sie den „alten“ Ausgang und löschen Sie ihn.

Die Netzwerk-Grafik der beiden Netzwerke ist fertig.



Die vorbelegten Variablennamen „undef_opd“ müssen durch die deklarierten Variablennamen ersetzt werden; dafür gibt es zwei grundsätzlich verschiedene Möglichkeiten:

Variablenname aus der Liste der deklarierten Variablen übernehmen (siehe Kapitel Programmeingabe im AWL-Editor auf Seite 32).

Variablenname über das Dialogfenster „Element benennen“ eingeben.

▶ Doppelklicken Sie auf das zu benennende Element.

Es wird das entsprechende Dialogfenster gestartet.

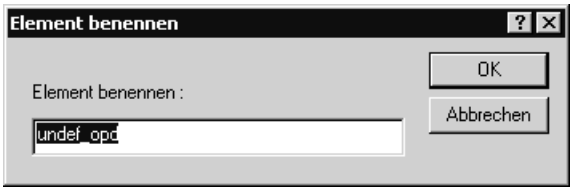


Abbildung 42: Dialogfenster „Element benennen“

▶ Geben Sie „grundstellung“ ein und bestätigen Sie mit der Schaltfläche OK.

02/02 AWB27-1307-D

Der Variablenname erscheint oberhalb des Kontaktsymbols.

- ▶ Geben Sie auf die gleiche Weise die restlichen Variablennamen in den beiden Netzwerken ein.
- ▶ Zur Eingabe des Kommentares platzieren Sie den Cursor einfach im zu kommentierenden Netzwerk und rufen ‹ Bearbeiten → Netzwerk Kommentar› auf.

 *„Netzwerk-Kommentar“*

Es öffnet sich das Dialogfenster „Netzwerk-Kommentar“:

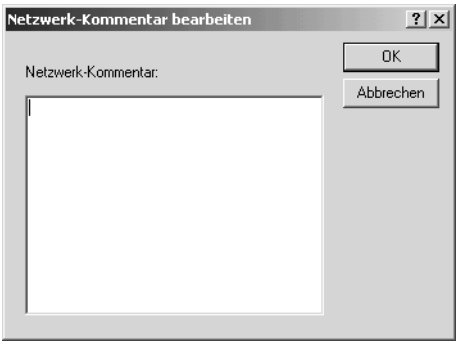


Abbildung 43: „Netzwerk-Kommentar“

- ▶ Geben Sie „Motor einschalten“ ein und bestätigen Sie mit der Schaltfläche OK.

Der eingegebene Kommentar erscheint im Netzwerkkopf unterhalb der Netzwerknummer.

- ▶ Geben Sie auf die gleiche Weise den Kommentar für das Netzwerk 0002 ein.

Die Programmeingabe in KOP ist damit beendet.

Programmeingabe in FBS

Voraussetzungen:

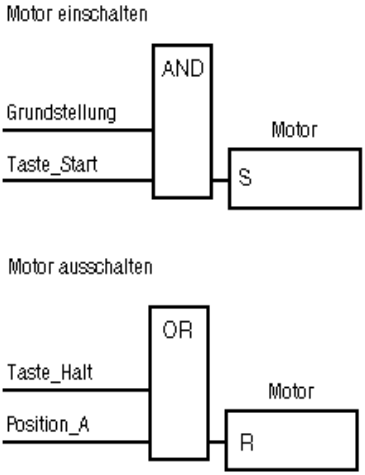
Sie kennen die Aufgabenstellung des Programmierbeispiels 1.

Das Projekt „beispiel“ ist angelegt.

Eine neue POE vom Typ „Programm“ ist angelegt.

Die benötigten Variablen sind deklariert.

Programmentwurf für das Beispielprogramm in FBS



Das Programm ist in zwei Netzwerke aufgeteilt.

Im ersten Netzwerk sind die Variablen „Grundstellung“ und „Taste_Start“ nach der UND-Funktion verknüpft. Sie bilden die erforderliche Bedingung, damit der Motor eingeschaltet wird.

Im zweiten Netzwerk stellt die ODER-Verknüpfung der Variablen „Taste_Halt“ und „Position_A“ die Ausschaltbedingung dar.

02/02 AWB27-1307-D

Im Netzwerkkopf steht ein Kommentar zur Kurzbeschreibung des Netzwerkprogramms.

Geben Sie jetzt das oben dargestellte Beispiel-Programm selbst in FBS ein. Benutzen Sie dazu die Schaltflächen der Sprachelementeleiste FBS.

- ▶ Wählen Sie die Programmiersprache FBS über den Menüpunkt ‹ Extras → Programmiersprache → FBS › oder die entsprechende Schaltfläche der Standard-Symbolleiste.
- ▶ Klicken Sie auf die Schaltfläche „Anfangs-FBS-Netzwerk einfügen“, um ein Anfangs-FBS-Netzwerk zu erstellen.

Das erstellte Netzwerk 0001 enthält ein Eingangs- und ein Ausgangssymbol. Beide Symbole erhalten automatisch die Bezeichnung „undef_opd“.

 *„Anfangs-FBS-Netzwerk“*

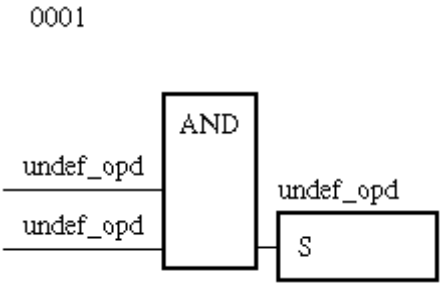
- ▶ Markieren Sie die Eingangs-Anschlusslinie, um anschließend per Schaltfläche eine UND-Verknüpfung einzufügen.
- ▶ Markieren Sie das Ausgangssymbol mit Mausklick, um anschließend per Schaltfläche als Netzwerkabschluss einen Setzkontakt „S“ bzw. eine Set-Funktion einzufügen.

Das Ausgangssymbol des Setzbefehls wird unter dem bestehenden Zuweisungssymbol hinzugefügt. Das obere Zuweisungssymbol hat jetzt keine Funktion mehr und kann gelöscht werden.

- ▶ Markieren Sie das obere Zuweisungssymbol und betätigen Sie die Taste „ENTF“.

Das obere Zuweisungssymbol wird gelöscht.

Die Grafik für das Netzwerk 0001 ist fertig.



▶ Klicken Sie auf die Schaltfläche „Anfangs-FBS-Netzwerk“, um ein neues Netzwerk zu erstellen.

Das erstellte Netzwerk 0002 enthält ein Kontakt- und ein Ausgangssymbol.

▶ Markieren Sie die Eingangs-Anschlusslinie mit Mausklick, um anschließend per Schaltfläche eine ODER-Verknüpfung einzufügen.

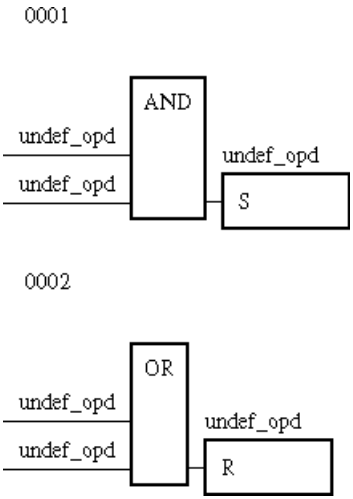
▶ Markieren Sie das Ausgangssymbol mit Mausklick, um anschließend per Schaltfläche als Netzwerkabschluss einen Rücksetzkontakt „R“ bzw. eine Reset-Funktion einzufügen.

Das Ausgangssymbol des Rücksetzbefehls wird unter dem bestehenden Zuweisungssymbol hinzugefügt.

▶ Löschen Sie das überflüssige Zuweisungssymbol.

Die Netzwerk-Grafik der beiden Netzwerke ist fertig.

02/02 AWB27-1307-D



Die vorbelegten Variablennamen „undef_opd“ müssen durch die deklarierten Variablennamen ersetzt werden, dafür gibt es zwei grundsätzlich verschiedene Möglichkeiten:

Variablenname aus der Liste der deklarierten Variablen übernehmen (siehe Kapitel Programmeingabe im AWL-Editor auf Seite 32),

Variablenname über das Dialogfenster „Element benennen“ eingeben.

▶ Doppelklicken Sie auf das zu benennende Element.

Es wird das entspreche Dialogfenster gestartet.

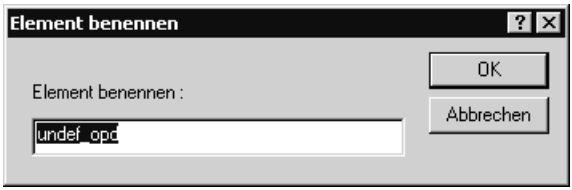


Abbildung 44: „Element benennen“

▶ Geben Sie „grundstellung“ ein, und bestätigen Sie mit der Schaltfläche OK.

Der Variablenname erscheint oberhalb der markierten Anschlusslinie.

▶ Geben Sie auf die gleiche Weise die restlichen Variablennamen in den beiden Netzwerken ein.

Anschließend müssen Sie noch die Netzwerk-Kommentare eingeben, wozu das betreffende Netzwerk markiert werden muss.

▶ Zur Eingabe des Kommentares platzieren Sie den Cursor einfach im zu kommentierenden Netzwerk und rufen Sie im Menü ‹ Bearbeiten → Netzwerk Kommentar› auf.

 *„Netzwerk-Kommentar“*

▶ Geben Sie „Motor einschalten“ ein, und bestätigen Sie mit der Schaltfläche OK.

Der eingegebene Kommentar erscheint im Netzwerkkopf unterhalb der Netzwerknummer.

▶ Geben Sie auf die gleiche Weise den Kommentar für das Netzwerk 0002 ein.

Die Programmeingabe in FBS ist damit beendet.

02/02 AWB27-1307-D

Programm in KOP oder FBS online anzeigen

Das Programm, das in der Steuerung ausgeführt wird, kann im POE-EDITOR im Online-Modus in einer der Programmiersprachen AWL, KOP oder FBS dargestellt werden.

Wenn Sie das Programm im POE-EDITOR KOP/FBS erstellt haben, werden die Kommentare in der AWL-Darstellung nicht den einzelnen Zeilen zugeordnet, sondern erscheinen am Anfang der zugehörigen Programmsequenz:

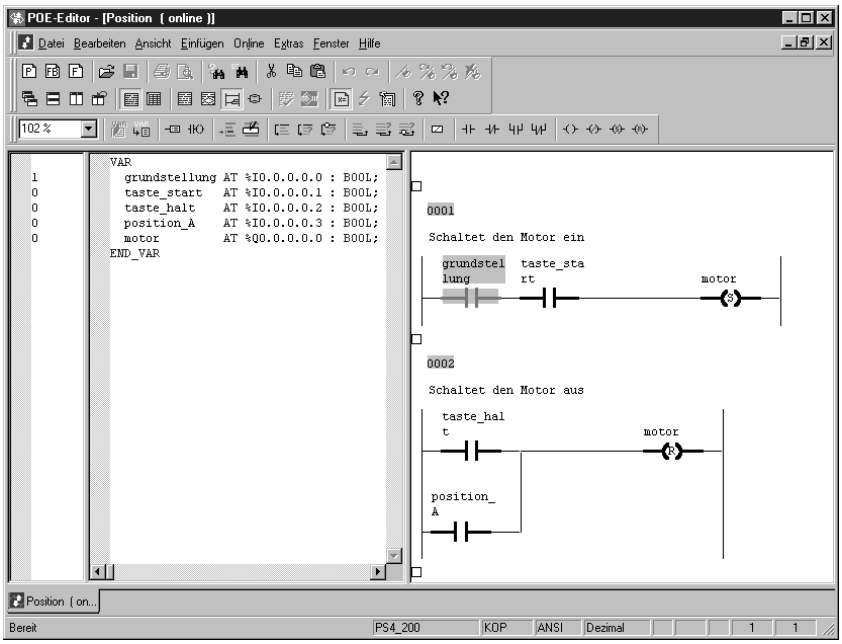


Abbildung 45: POE „position“ in der KOP-Darstellung (online)

KOP-Darstellung

Sie können zwischen den Programmiersprachen umschalten. Dies ist sowohl bei ausgeschalteter als auch bei aktiver Zustandsanzeige möglich.

▶ Wählen Sie ‹ Extras → Programmiersprache → KOP› oder die entsprechende Schaltfläche.

 *„POE-EDITOR KOP“*

▶ Führen Sie Änderungen auf die gleiche Weise durch wie im POE-EDITOR AWL.

Genaue Angaben hierzu finden Sie im Handbuch AWB 27-1305-D „Sucosoft S40, Programmiersoftware, Benutzeroberfläche“.

Wenn Sie die POE geändert haben, ist der Befehl „Aktivieren“ im Menü „Online“ oder die Schaltfläche in der Symbolleiste verfügbar.

 *„Aktivieren“*

▶ Führen Sie den Befehl „Aktivieren“ aus.

Die durchgeführten Änderungen werden in der POE, im Programm und in der Steuerung aktualisiert.

02/02 AWB27-1307-D

FBS-Darstellung

Sie können zwischen den Programmiersprachen umschalten. Dies ist sowohl bei ausgeschalteter Zustandsanzeige als auch bei aktiver Zustandsanzeige möglich.

▶ Wählen Sie ‹ Extras → Programmiersprache → FBS› oder die entsprechende Schaltfläche.

 *„ FBS-Editor“*

▶ Führen Sie Änderungen auf die gleiche Weise durch wie im POE-EDITOR KOP beschrieben.

Programmeingabe in ST

Eine weitere Programmiersprache der Sucosoft ist ST, Strukturierter Text. Diese Sprache ist ähnlich PASCAL.

Der Deklarationsteil ist identisch mit AWL, KOP und FBS, der Anweisungsteil – wie AWL – textbasierend. Es kann jedoch NICHT zwischen den grafischen Sprachen KOP/FBS und ST umgeschaltet werden.

Zur einfachen Eingabe der ST-Befehle steht ihnen eine Toolbar zur Verfügung.

Abbildung 46: Toolbar ST

Beispielprogramm in ST erstellen

Das Beispiel, das Sie bisher erstellt haben, können Sie nun in ST nachbauen.

Voraussetzungen:

Sie kennen die Aufgabenstellung des Programmierbeispiels 1.

Das Projekt „beispiel“ ist angelegt.

Eine neue POE vom Typ „Programm“ ist angelegt.

Die benötigten Variablen sind deklariert.

Erinnern wir uns noch einmal an die Aufgabenstellung:

Wenn der Motor in der Grundstellung ist und die Taste „Taste_Start“ gedrückt wird, wird der Motor eingeschaltet.

Wird die Taste „Taste_Halt“ gedrückt oder hat der Motor die Position_A erreicht, wird der Motor ausgeschaltet.

Nun Schritt für Schritt die Vorgehensweise:

▶ Wählen Sie die Programmiersprache ST über den Menüpunkt ‹ Extras → Programmiersprache → ST› oder die Schaltfläche

 *„POE-EDITOR ST“*

Geben Sie das ST-Beispiel ein:

▶ Benutzen Sie zur Eingabe der IF...THEN-Konstrukte die entsprechende Schaltfläche der Werkzeugleiste.

02/02 AWB27-1307-D

Sie sehen, dass das leere Konstrukt aufgebaut wird:

```
IF ...  THEN...
...;
ELSE
...;
END_IF;
```

▶ Vervollständigen Sie das Konstrukt wie folgt:

```
(* Schaltet den Motor ein *)

IF grundstellung = 1 AND taste_start=1 THEN
  Motor:=1;
END_IF;

(* Schaltet den Motor aus *)

IF taste_halt = 1 OR position_A = 1 then
  Motor:=0;
END_IF;
```

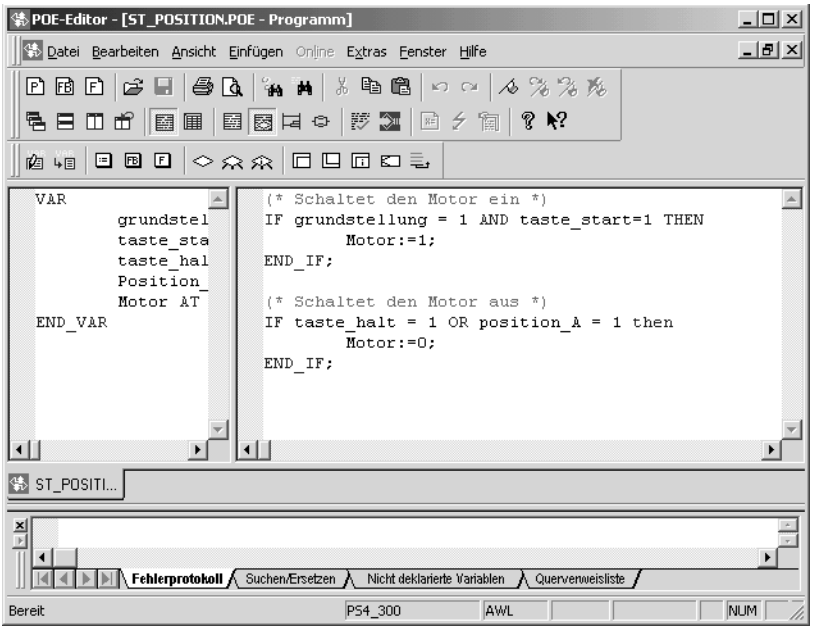


Abbildung 47: POE „position“ in der ST-Darstellung

02/02 AWB27-1307-D

11 Programmieraufgabe 2

An einer Digital-Ausgabebaugruppe soll ein Lauflichteffekt für acht LED-Anzeigen programmiert werden, das heißt, dass die einzelnen Anzeigen der Reihe nach kurz aufleuchten. Der Wechsel soll im 2-Hz-Takt erfolgen.

Der Lauflichteffekt soll selbstständig starten, wenn die Steuerung eingeschaltet wird, und so lange laufen, bis die Programmausführung von außen gestoppt wird. Eine Beeinflussung des Ablaufs durch Bedienelemente ist nicht vorgesehen. Damit entfällt die Notwendigkeit, Eingänge abzufragen.

Nach der Inbetriebnahme soll das Programm online geändert werden: Die Laufrichtung soll gewechselt, die Laufgeschwindigkeit auf 1 Hz gesetzt werden.

Umsetzung der Aufgabe
Lesen Sie zuerst in diesem Kapitel durch, wie die Aufgabe in ein AWL-Programm umgesetzt wird. Sie erhalten alle Informationen zum Programmaufbau und zu den verwendeten Sprachelementen. Im Abschnitt „Eingabe der Programmieraufgabe 2“ ab Seite 105 ist dann beschrieben, wie Sie das Programm in der Sucosoft S40 eingeben.

Die zwei wesentlichen Größen, die zur Umsetzung der Aufgabe programmiert werden müssen, sind die **Bestimmung der Leuchtdauer** und die **Ansteuerung der Anzeigen**.

Zur Bestimmung der **Leuchtdauer** kann ein beliebiger Hersteller-Funktionsbaustein aus der Gruppe der Zeitglieder verwendet werden. Für die Lösung der gestellten Aufgabe wird der Funktionsbaustein „TimePulse“ (Impuls) eingesetzt, der von den Zeitgliedern die einfachste Lösung bietet, da keine Rücksetz-Schritte oder dergleichen eingebaut werden müssen.

Zur **Ansteuerung der Anzeigen** soll ein „neutraler“ Funktionsbaustein geschrieben werden, damit dieser für ähnliche Aufgaben wieder verwendet werden kann. Deshalb sollen die Ausgangsbyte-Adresse sowie der konkrete Zeitwert für die Leuchtdauer von außen, also von der aufrufenden POE, an den Funktionsbaustein übergeben werden. Dieser neutrale Funktionsbaustein kann dann mehrfach im gleichen Programm aufgerufen werden, sodass beispielsweise mehrere Ausgangsbytes mit einem Lauflicht von verschiedener Leuchtdauer angesteuert werden könnten.

Es muss daher ein Funktionsbaustein erstellt werden, in dem

- die Anzeigen angesteuert werden und
- der Hersteller-Funktionsbaustein TP aufgerufen wird zur Bestimmung der Leuchtdauer.

Dieser Funktionsbaustein soll den Namen „LICHT“ erhalten.

Im eigentlichen Programm erfolgt der Aufruf des Funktionsbausteins „LICHT“. Das Programm soll den Namen „BSP_PS4“ erhalten. In der Programm-POE werden die Ausgangsbyte-Adresse sowie der Wert für die Leuchtdauer übergeben.

02/02 AWB27-1307-D

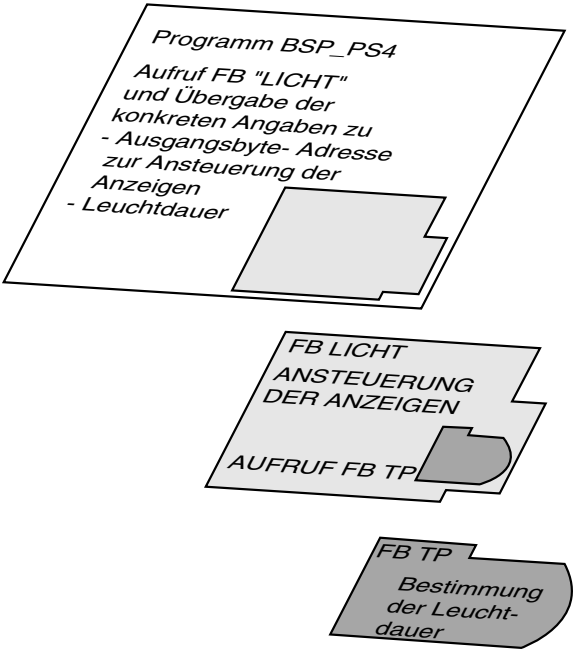


Abbildung 48: Programmkomponenten zur Lösung der Aufgabe

Für die Aufgabe sind also zwei POEs zu schreiben:

eine POE vom Typ „FUNCTION_BLOCK“ mit dem Namen „LICHT“

eine zweite POE vom Typ „PROGRAM“ mit dem Namen „BSP_PS4“.

Entwurf des Funktionsbausteins „LICHT“

Ansteuern der Anzeigen

Der Lauflichteffekt kann erreicht werden, indem ein Bitmuster mit einem gesetzten Bit in gleichmäßigen Zeitabständen in das Arbeitsregister geladen, um eine Bitstelle geschoben und danach zurückübertragen wird. Das Bitmuster wird in Form einer Variable übergeben. Da die Sucosoft die Möglichkeit bietet, eine Variable als Eingangsvariable und gleichzeitig Ausgangsvariable unter dem gleichen Namen einzulesen, zu verarbeiten und wieder auszugeben, soll die benötigte Variable als eine solche Ein-/Ausgangsvariable definiert werden, und zwar mit dem Namen „Lichtleiste“ und dem Datentyp „Byte“. Das Schlüsselwort für Ein-/Ausgangsvariablen heißt VAR_IN_OUT, das Ende eines Deklarationsblocks kennzeichnet bei allen Gültigkeitsbereichen das Schlüsselwort END_VAR.

Die konkrete Adresse wird erst beim Aufruf durch die übergeordnete Programm-POE übergeben und bleibt daher zunächst unberücksichtigt. Daraus ergibt sich der Deklarationsblock:

```
VAR_IN_OUT
  Lichtleiste : BYTE ;
END_VAR
```

Zum Verschieben des Bitmusters kann entweder ein Schiebebefehl oder ein Rotationsbefehl verwendet werden, beide entweder nach links oder nach rechts. Die Verwendung eines Schiebebefehls würde jedoch das Nachladen einer „1“ nach spätestens acht Schiebeschritten erfordern, deshalb soll hier ein Rotationsbefehl eingesetzt werden, und zwar nach rechts. Die Anweisungszeilen lauten daher:

```
LD  Lichtleiste
ROR 1
ST  Lichtleiste
```


02/02 AWB27-1307-D

Bestimmung der Leuchtdauer

Zur Bestimmung der Leuchtdauer wird der Hersteller-Funktionsbaustein TP (Impuls) verwendet.

Der Funktionsbaustein TP (Impuls)

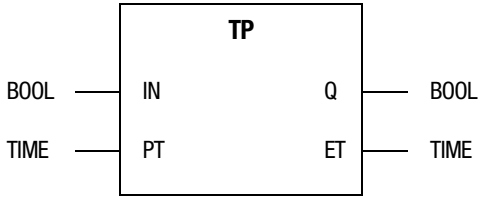


Abbildung 49: Prototyp des Funktionsbausteins TP

IN Startbedingung
PT Zeitwertvorgabe
Q binärer Zustand des Zeitglieds
ET aktueller Zeitwert

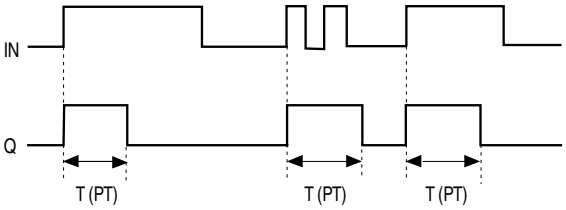


Abbildung 50: Zeitdiagramm von TP

Das Zeitglied startet mit einer steigenden Flanke am Eingang IN und hält den Zustand „1“ am booleschen Ausgang Q für die Dauer des vorgegebenen Zeitwertes PT. Der Ausgang ET zeigt den aktuellen Zeitwert an. Solange die Zeit läuft, hält der Zustand „1“ am Ausgang Q an, unabhängig vom Zustand des Eingangs IN.

Die Operanden IN und Q sind boolesche Operanden, die Operanden PT und ET sind vom Standard-Datentyp TIME (Defaultwert T#0s).

Verwendung von Funktionsbausteinen

Für jede Anwendung eines vorhandenen Funktionsbausteins muss im **Deklarationsteil** der aufrufenden POE eine Instanz, d. h. eine Kopie des Funktionsbausteins, angelegt werden. Dazu wird dem Funktionsbaustein ein frei wählbarer Instanzname gegeben, der als **lokale** Variable deklariert wird.

Der Funktionsbaustein TP soll im Funktionsbaustein „LICHT“ aufgerufen werden, TP muss daher im Deklarationsteil von „LICHT“ deklariert werden. In diesem Fall wird für TP der Instanzname „Takt“ vergeben. Das Schlüsselwort für lokale Variablen heißt VAR:

```
VAR
  Takt : TP ;
END_VAR
```

Beim Aufruf eines Funktionsbausteins werden seinen Eingangsoperanden die Werte übergeben, die zu verarbeiten sind. Die Ergebnisse liefert der Funktionsbaustein über die Ausgangsvariablen. In der aufrufenden POE werden daher für die Übergabeparameter und die Übernahme der Ergebnisse weitere Variablen benötigt. Die Variablennamen sind frei wählbar.

In unserem Beispiel legen wir für den Einsatz des Funktionsbausteins TP im Funktionsbaustein „LICHT“ folgende Variablen fest:

Start	Startbedingung, wird an den Operanden IN übergeben
Taktdauer	Zeitwertvorgabe an den Operanden PT
Zeit_laeuft	boolescher Zustand des Zeitglieds, wird über den Operanden Q ausgegeben
aktuelle_Zeit	aktueller Zeitwert, wird über den Operanden ET ausgegeben

02/02 AWB27-1307-D

Die Variable „Taktdauer“ wird als Eingangsvariable deklariert, damit beim Aufruf des Funktionsbausteins „LICHT“ dem Operanden PT des Funktionsbausteins TP ein beliebiger Zeitwert übergeben werden kann. Die restlichen Variablen können als lokale Variablen deklariert werden, da sie nur im Funktionsbaustein „LICHT“ und nicht in der aufrufenden POE „BSP_PS4“ Bedeutung haben. Das Schlüsselwort für Eingangsvariablen heißt VAR_INPUT, für lokale Variablen VAR. Daraus leiten sich für das Beispiel folgende Deklarationszeilen ab:

```
VAR_INPUT
  Taktdauer : TIME ;
END_VAR

VAR
  Start : BOOL ;
  Zeit_laeuft : BOOL ;
  aktuelle_Zeit : TIME ;
END_VAR
```

Der **Aufruf** eines Funktionsbausteins erfolgt mit dem Befehl CAL, wobei der Funktionsbaustein unter dem vergebenen Instanznamen angesprochen wird, in unserem Beispiel: CAL Takt.

Für die **Parameterübergabe** gibt es drei verschiedene Verfahren. Hier sollen zwei davon vorgestellt werden:

Beim **ersten Verfahren** erfolgt die Parameterübergabe direkt beim Aufruf des Funktionsbausteins. Die Parameter werden in runden Klammern angegeben, getrennt durch ein Komma, zwischen die Eingangsparameter und die Ausgangsparameter wird das Zeichen „|“ gesetzt:

```
  CAL Takt (IN := Start,
    PT := Taktdauer |
    Zeit_laeuft := Q,
    aktuelle_Zeit := ET)
```

Beim **zweiten Verfahren** werden vor dem Funktionsbausteinaufruf die Eingangsparameter einzeln mit dem Befehl LD geladen und mit ST den Funktionsbausteinoperanden übergeben. Nach dem Aufruf werden die Ausgangsoperanden abgefragt. Die Schreibweise für die Funktionsbausteinoperanden lautet:

Instanzname.Operand

Der Aufruf mit Parameterübergabe nach dem zweiten Verfahren hat in unserem Beispiel daher folgende Form:

```
LD  Start
ST  Takt.IN
LD  Taktdauer
ST  Takt.PT
CAL Takt
LD  Takt.Q
ST  Zeit_laeuft
LD  Takt.ET
ST  aktuelle_Zeit
```

Welches der beiden Verfahren Sie verwenden, ist Geschmackssache. Das zweite Verfahren hat lediglich den Vorteil, dass es einer der standardisierten Varianten des FB-Aufrufs nach IEC/EN 61131-3 entspricht und daher evtl. von anderen IEC-Systemen wieder verwendet werden kann; beim kompakteren ersten Verfahren ist die Parametrierung der Ausgangsparameter nach dem Zeichen „I“ Moeller-spezifisch. Für das Beispiel soll der Aufruf von TP nach dem zweiten Verfahren erfolgen.

02/02 AWB27-1307-D

Vervollständigung des Programmabschnitts „Leuchtdauer“

Durch Zurückführung des negierten Ausgangs Q an den Eingang IN erhält man einen Taktgeber:

```
LDN Zeit_laeuft
ST  Start
```

Der Ausgang Q führt den Zustand „1“, solange die Zeit läuft. Danach steht für die Dauer eines Zyklus der Zustand „0“ an, der das wiederholte Starten des Zeitglieds bewirkt. Die Periodendauer entspricht annähernd dem Wert PT, der Fehler von einer Zyklusdauer kann in diesem Fall vernachlässigt werden.

Die Anweisungszeilen zur Programmierung der Leuchtdauer lauten also:

```
LD  Start
ST  Takt.IN
LD  Taktdauer
ST  Takt.PT
CAL Takt
LD  Takt.Q
ST  Zeit_laeuft
LD  Takt.ET
ST  aktuelle_Zeit
LDN Zeit_laeuft
ST  Start
```

Das Programm kann jetzt vereinfacht werden: Das Zeitglied kann direkt mit dem negierten Zustand des Ausgangsoperanden Q gestartet werden – dadurch werden die Variablen „Start“ und „Zeit_laeuft“ nicht mehr benötigt. Die Abfrage des aktuellen Zeitwertes ist zwar in der Aufgabenstellung nicht gefordert, die Variable „aktuelle_Zeit“ wird aber dennoch eingesetzt, damit beim Online-Test die ablaufende Zeit angezeigt wird.

Das Programm hat dadurch folgende Form:

```
LDN Takt.Q
ST  Takt.IN
LD  Taktdauer
ST  Takt.PT
CAL Takt
LD  Takt.ET
ST  aktuelle_Zeit
```

Zusammensetzung des Funktionsbausteins „LICHT“

Nach dem Ablaufen der Zeit PT soll das Bitmuster um eine Stelle geschoben werden. Dafür bietet sich der Einsatz einer Sprungmarke an, die es ermöglicht, das Rotieren zu überspringen, solange die Taktgeber-Zeit läuft. Das Bitmuster der Variable „Lichtleiste“ soll verschoben werden, wenn die Zeit gerade abgelaufen ist, d. h. im Zustand „0“ am Ausgang Q. Ein bedingter Sprung zu einer Sprungmarke kann mit dem Befehl JMPC programmiert werden. Der Sprung erfolgt, wenn im Arbeitsregister „1“ steht, also wenn das Verknüpfungsergebnis VKE = 1 ist. Der Name der Sprungmarke ist frei wählbar, in diesem Fall „nicht_schieben“:

```
LD      Takt.Q
JMPC    nicht_schieben

LD      Lichtleiste
ROR     1
ST      Lichtleiste

nicht_schieben:    (*Sprungmarke*)
LDN     Takt.Q
ST      Takt.IN
LD      Taktdauer
ST      Takt.PT
CAL     Takt
LD      Takt.ET
ST      aktuelle_Zeit
```


02/02 AWB27-1307-D

Das Programm kann jetzt optimiert werden, d. h. durch geringfügige Änderungen in eine kürzere Form gebracht werden.

Die Anweisung „LDN Takt.Q“, die die Startbedingung für den Taktgeber liefert, kann gleichzeitig als Bedingung für die Sprungmarke genutzt werden - der Sprungbefehl JMPC muss in den Sprungbefehl JMPCN, d. h. Sprung bei VKE = 0, geändert werden.

```
FUNCTION_BLOCK LICHT

VAR_INPUT
  Taktdauer : TIME ;                    (* Zeitwertvorgabe *)
END_VAR

VAR_IN_OUT
  Lichtleiste : BYTE ;                  (* Lauflicht *)
END_VAR

VAR
  aktuelle_Zeit : TIME ;                (* Aktueller Zeitwert*)
  Takt : TP ;                           (* Instanz des Funktionsbausteins TP *)
END_VAR

  LDN     Takt.Q                        (* VKE=1, wenn Zeitglied abgelaufen*)
  ST      Takt.IN                       (* Starten des Zeitglieds*)
  JMPCN   nicht_schieben                (* Überspringen der Bitmusterver-
                                        schiebung, solange die Zeit läuft *)

  LD      Lichtleiste
  ROR     1                             (* Rotieren des Bitmusters *)
  ST      Lichtleiste

nicht_schieben :
  LD      Taktdauer                     (* Übergabe des Zeitwerts *)
  ST      Takt.PT
  CAL     Takt                          (* Aufruf des Funktionsbausteins *)
  LD      Takt.ET
  ST      aktuelle_Zeit                 (* Anzeige des aktuellen Wertes für
                                        den Online-Test*)

END_FUNCTION_BLOCK
```

Der Programmabschnitt, in dem die Variable „Lichtleiste“ rotiert wird, wird nur bearbeitet, wenn die Zeit im Zeitglied TP gerade abgelaufen ist. Danach wird die Zeit wieder gestartet, und die Rotier-Anweisungen werden übersprungen, solange die Zeit läuft.

Entwurf des Programms „BSP_PS4“

Der Funktionsbaustein „LICHT“ soll in der Programm-POE „BSP_PS4“ aufgerufen werden. Beim Aufruf müssen zwei Parameter übergeben werden:

der Zeitwert zur Bestimmung der Laufgeschwindigkeit

die physikalische Adresse, an der das Lauflicht angezeigt werden soll.

Zuerst muss der Funktionsbaustein wieder deklariert werden. Im Deklarationsteil der POE „BSP_PS4“ wird der Instanzname „Lichtkette“ vergeben:

```
VAR
  Lichtkette : LICHT ;
END_VAR
```

Die Laufgeschwindigkeit soll durch eine Variable vom Datentyp TIME festgelegt werden, die mit dem Wert 500 ms initialisiert wird und damit dem Takt von 2 Hz entspricht. Die Variable wird als lokale Variable deklariert, da sie nur innerhalb dieser POE benötigt wird. Sie erhält den Namen „Lauftempo“ :

```
VAR
  Lauftempo : TIME := T#500ms ;
END_VAR
```

Die Ansteuerung der Anzeigen soll über eine Variable „Anzeige“ an einen Ausgang gegeben werden. Mit dieser Variable kann die Adresse festgelegt werden, in diesem Fall das Ausgangsbyte 0. Bei einem Kaltstart soll die Variable „Anzeige“ mit dem Wert „1“

02/02 AWB27-1307-D

initialisiert werden. Sie wird als lokale Variable festgelegt.

```
VAR
  Anzeige AT %QB0.0.0.0 : BYTE := 1 ;
END_VAR
```

Das gleiche Ergebnis würde das Initialisieren mit dem Bitmuster 00000001 liefern:

```
Anzeige AT %QB0.0.0.0 : BYTE := 2#00000001 ;
```

Im Anweisungsteil der POE „BSP_PS4“ muss lediglich der Funktionsbaustein „LICHT“ unter dem deklarierten Instanznamen aufgerufen werden. In diesem Fall wird das erste beschriebene Verfahren für die Parameterübergabe verwendet:

```
CAL Lichtkette (Taktdauer := Lauftempo,
    Lichtleiste := Anzeige)
```

Die Programm-POE „BSP_PS4“ hat also folgende Form:

```
PROGRAM BSP_PS4

VAR
  Lichtkette : LICHT ;
  Lauftempo : TIME := T#500ms ;
END_VAR

VAR
  Anzeige AT %QB0.0.0.0 :BYTE := 1 ;
END_VAR

  CAL Lichtkette (Taktdauer := Lauftempo,
    Lichtleiste := Anzeige)

END_PROGRAM
```

Die folgende Grafik fasst den Ablauf von der Programmeingabe bis zum Testlauf des fertigen Programms zusammen.

Eingabe FB LICHT	
	Variablendeklaration
	Programmeingabe
Eingabe Programm BSP_PS4	
	Variablendeklaration
	Programmeingabe
Topologie-Konfiguration der Systemkomponenten	
Programmcode-Erzeugung	
Test und Inbetriebnahme	
	Programm in SPS übertragen
	Programm starten
	Programm online prüfen und ändern

Die Topologie-Konfiguration, die Programmcode-Erzeugung und die T & I haben Sie mit der Programmieraufgabe 1 in den vorherigen Kapiteln kennengelernt. Für die Programmieraufgabe 2 wird der Gebrauch des Werkzeugs T & I vertieft.

Im folgenden Abschnitt werden Sie zunächst den Funktionsbaustein „LICHT“ eingeben, danach das Programm BSP_PS4.

02/02 AWB27-1307-D

Eingabe der Programmieraufgabe 2

In diesem Abschnitt werden Sie angeleitet, wie Sie die beiden POEs, die zur Umsetzung der Aufgabe erforderlich sind und deren Aufbau im vorhergehenden Abschnitt beschrieben wurde, konkret im POE-EDITOR eingeben.

Zuerst erstellen Sie den Funktionsbaustein „LICHT“, anschließend die POE „BSP_PS4“. Prinzipiell kann die Reihenfolge auch umgedreht werden, es ist jedoch sinnvoll, mit den POEs zu beginnen, die in der untersten Strukturebene stehen, hier der Funktionsbaustein „LICHT“. Er wird von der POE „BSP_PS4“ aufgerufen und sollte deshalb vor dem Aufruf fertig erstellt und seine Syntax geprüft sein.

Für beide POEs wird jeweils die Variablen-Deklaration und anschließend die Programmeingabe in AWL beschrieben.

Legen Sie zunächst mit dem NAVIGATOR im Ordner „PROJEKTE“ Ihren neuen Projektordner (ein neues Projekt) mit dem Namen „lern_PS4“ an:

- Benutzen Sie dazu das Menü ‹ Projekt → Neu...› oder die entsprechende Schaltfläche.

 *„Neues Projekt anlegen“*

Es öffnet sich das Dialogfenster „Neues Projekt anlegen“:

- Wählen Sie zuerst das Festplattenlaufwerk „C“ und dann den Ordner „PROJEKTE“ aus.
- Tragen Sie anschließend in dem Eingabefeld „Neuer Projektordner“ als Namen des Ordners für das neue Projekt „lern_PS4“ ein.
- Bestätigen Sie die Eingabe mit der Schaltfläche OK.

Eingabe des Funktionsbausteins „LICHT“

Benutzen Sie zur Erstellung des Funktionsbausteins das NAVIGATOR-Menü:

▶ Wählen Sie ‹ Werkzeuge → POE-Editor› oder die entsprechende Schaltfläche in der Werkzeugleiste.

 *„POE-EDITOR“*

Es öffnet sich der POE-EDITOR.

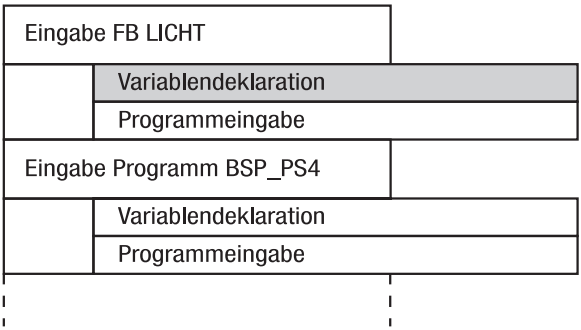
▶ Drücken Sie, dem benötigten POE-Typ „Programm“ entsprechend, die Schaltfläche „FB“ in der Standard-Symbolleiste oder

▶ Wählen Sie ‹ Datei → POE neu› und markieren Sie anschließend den POE-Typ „Funktionsbaustein“.

Es öffnen sich die beiden Fenster „Deklarations- und Anweisungsteil“, die entsprechend der Grundeinstellung des POE-EDITORs auf dem Bildschirm angeordnet sind.

02/02 AWB27-1307-D

Deklaration der Variablen



Die Variablen werden im syntaxgesteuerten Variableneditor (Syntax-Modus) deklariert. In diesem Modus brauchen Sie sich nicht um die Schlüsselwörter und die Syntax zu kümmern.

Im Programmentwurf haben wir folgende Variablen festgelegt:

```
VAR_INPUT
  Taktdauer : TIME ;          (* Zeitwertvorgabe *)
END_VAR

VAR_IN_OUT
  Lichtleiste : BYTE ;        (* Lauflicht *)
END_VAR

VAR
  aktuelle_Zeit : TIME ;      (* Aktueller Zeitwert*)
  Takt : TP ;                 (* Instanz des Funk-
                              tionsbausteins *)
END_VAR
```

▶ Schalten Sie – wenn nötig – mit ‹ Extras → Variableneditor → Syntax-Modus› oder der entsprechenden Schaltfläche in den Syntax-Modus.

 *„Syntax-Modus“*

Der Gültigkeitsbereich „Lokal“ ist beim Öffnen des Variableneditors zum Erstellen von Funktionsbausteinen voreingestellt. Der Karteikarten-Reiter für den gewählten Gültigkeitsbereich wird im Vordergrund dargestellt.

▶ Zur Eingabe der Variablen wählen Sie unter dem Menüpunkt „Einfügen“ die „Variablendeklaration...“.

Es öffnet sich folgender Dialog:

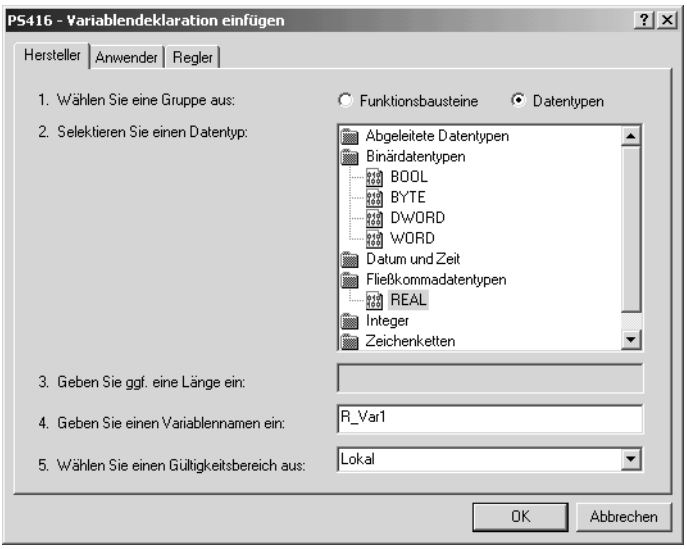


Abbildung 51: Variablendeklaration

Beginnen Sie mit der Variablen „aktuelleZeit“.

▶ Der Variablen müssen Sie als erstes einen Datentyp zuordnen. Wählen Sie daher als Gruppe „Datentypen“.

Im Fenster darunter sind nun alle Datentypen der SPS, in verschiedenen Gruppen zusammengefasst, dargestellt.

▶ Öffnen Sie die Gruppe „Datum und Zeit“ und wählen Sie „TIME“ aus.

Eine Länge brauchen Sie an dieser Stellen nicht eintragen, diese Angabe ist z. B. bei Strings sinnvoll.

▶ Geben Sie nun den Variablennamen „aktuelleZeit“ ein.

▶ Bestimmen Sie den Gültigkeitsbereich der Variablen, in diesem Fall also „LOKAL“.

Nach OK wird die Variable in den syntaxgesteuerten Variableneditor eingetragen.

Gehen Sie bei der Deklaration von „Takt“ entsprechend vor:

▶ „Takt“ soll einem Hersteller-Funktionsbaustein zugeordnet werden. Wählen Sie also „Funktionsbausteine“ als Gruppe aus.

▶ Suchen Sie unter den Funktionsbausteinen nach der Gruppe „Zeitbausteine“ und wählen Sie hier den Baustein „TP“ aus.

Auch hier brauchen Sie keine Länge anzugeben.

▶ Geben Sie den Instanznamen Ihres Bausteins an: „Takt“.

▶ Auch hier ist der Gültigkeitsbereich „Lokal“.

Natürlich können Sie auch alle Variablen komplett über die Tastatur eingeben, wenn Ihnen die Namen bekannt sind.

▶ Wiederholen Sie die Schritte entsprechend für die Variable „Lichtleiste“ und „Taktdauer“.

„Lichtleiste“ soll den Gültigkeitsbereich „In_Out“ haben, Taktdauer den Gültigkeitsbereich „Input“.

▶ Schalten Sie in den freien Variableneditor um, um Ihre deklarierten Variablen komplett zu sehen.

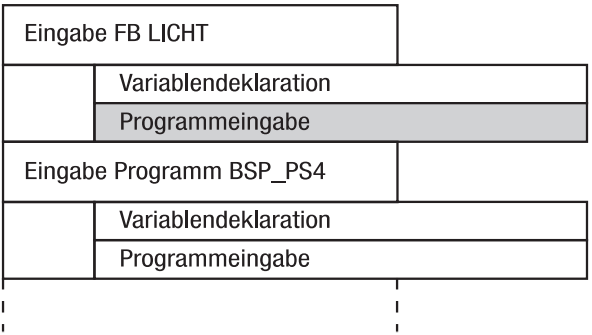
```
VAR_INPUT
        Taktdauer : TIME ;       (* Zeitwertvorgabe *)
END_VAR

VAR_IN_OUT
        Lichtleiste : BYTE ;     (* Lauflicht *)
END_VAR

VAR
        aktuelle_Zeit : TIME ;   (* Aktueller Zeitwert *)
        Takt : TP ;              (* Instanz des Funktionsbausteins TP *)
END_VAR
```

Abbildung 52: Deklarierte Variablen des Funktionsbausteins „LICHT“

Programmeingabe im Anweisungsteil



▶ Verwenden Sie die Programmiersprache AWL, schalten Sie ggf. um mit ‹ Extras → Programmiersprache → AWL› oder der entsprechenden Schaltfläche.

„POE-EDITOR AWL“

- Geben Sie das Programm entsprechend der Abbildung „Anweisungsteil des Funktionsbausteins LICHT“ ein. Beachten Sie dabei folgende Punkte:

  Groß- oder Kleinschreibung ist beliebig.

  Setzen Sie den Kommentar wie angegeben in Klammern mit Stern.

  Trennen Sie Operatoren, Operanden und Kommentare zur besseren Übersicht jeweils durch einen Tabulator.

  Trennen Sie die Anweisungssequenzen, die eine zusammenhängende Einheit bilden, durch eine Leerzeile von anderen Anweisungen.

- Speichern Sie die POE mit ‹ Datei → Speichern unter...› oder durch die entsprechende Schaltfläche.

 *„Speichern“*

Im sich öffnenden Dialogfenster „Speichern unter“ ist das aktuelle Projekt bereits voreingestellt.

- Geben Sie den Namen LICHT ein. Bestätigen Sie die Eingabe mit OK.
- Führen Sie mit ‹ Datei → Syntaxprüfung› , oder durch Betätigung der entsprechenden Schaltfläche die Syntaxprüfung durch.

 *„Syntaxprüfung“*

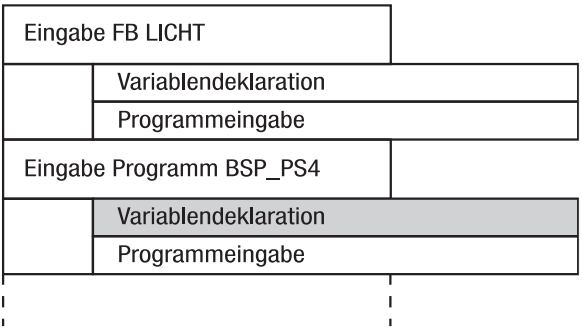
Wenn Sie die POE fehlerfrei nach dem vorliegenden Entwurf eingegeben haben, wird die Syntaxprüfung erfolgreich abgeschlossen. Sollte die Syntaxprüfung Fehler melden, überprüfen Sie alle Eingaben, korrigieren Sie die gefundenen Fehler und starten Sie die Syntaxprüfung erneut; dabei wird die POE automatisch gespeichert.

Eingabe des Programms „BSP_PS4“

Verbleiben Sie weiter im POE-EDITOR bei aktiviertem Syntax-Modus und gewählter AWL-Programmiersprache.

▶ Legen Sie mit ‹ Datei → POE neu› oder der Schaltfläche „P“ in der Standard-Symbolleiste eine neue POE vom Typ „Programm“ an.

Deklaration der Variablen



Im Programmentwurf haben wir folgende Variablen festgelegt:

```
VAR
  Lichtkette : LICHT ;
  Lauftempo : TIME := T#500ms ;
END_VAR

VAR
  Anzeige AT %QB0.0.0.0 : BYTE := 1 ;
END_VAR
```

In POEs vom Typ „Programm“ sind lokale Variablen voreingestellt.

Deklarieren Sie zunächst den Funktionsbaustein LICHT, den Sie soeben erstellt haben. Gehen Sie dabei vor, wie bei der Deklaration des Hersteller-Funktionsbausteins „Takt“.

02/02 AWB27-1307-D

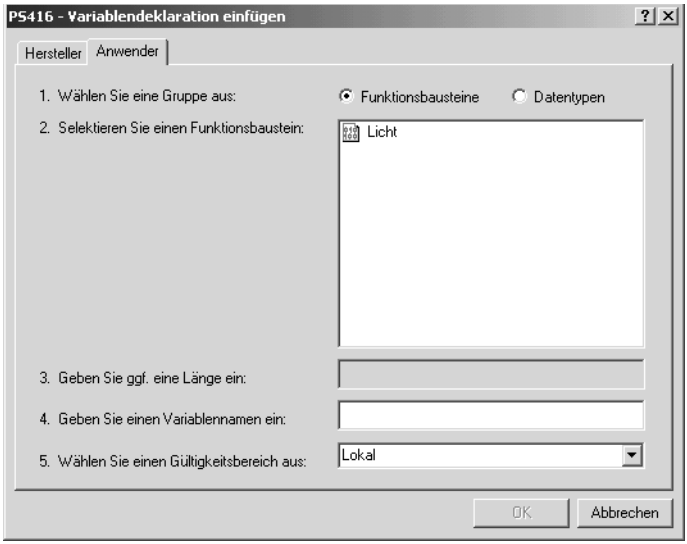
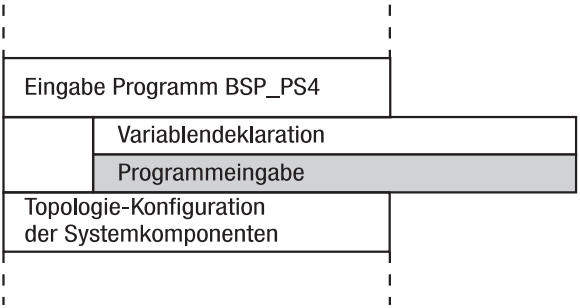


Abbildung 53: Variablendeklaration für Funktionsbaustein „LICHT“

- ▶ Wählen Sie die Karteikarte „Anwender“. Hier sehen Sie Ihren Funktionsbaustein „Licht“.
- ▶ Markieren Sie ihn und tragen Sie den Namen „Lichtkette“ ein.
- ▶ Deklarieren Sie die lokale Variable „Lauftempo“. Geben Sie außer Variablenname und Datentyp im Feld „Initialwert“ die Zeitvorgabe t#500ms ein.
- ▶ Deklarieren Sie die lokale Variable „Anzeige“. Geben Sie zusätzlich zu Variablenname und Datentyp als Initialwert „1“ ein und geben Sie als Adresse QB0.0.0.0 an.

Der Deklarationsteil der POE „Programm“ ist nun vollständig.

Programmeingabe im Anweisungsteil



▶ Schalten Sie vom Deklarationsteil zum Anweisungsteil um, verwenden Sie weiter die Programmiersprache AWL.

Im Anweisungsteil der POE muss der Funktionsbaustein „LICHT“ unter dem deklarierten Instanznamen aufgerufen werden. Der Aufruf und die Parameterübergabe können von Hand eingegeben werden.

Die Sucosoft S40 bietet auch die Möglichkeit, einen vorgefertigten Aufruf automatisch einbinden zu lassen. Dazu ist es notwendig, dass der Funktionsbaustein im aktuellen Projekt liegt und die Syntaxprüfung des Funktionsbausteins fehlerfrei durchgeführt wurde. Beide Voraussetzungen sind in diesem Fall erfüllt.

Hersteller-Funktionsbausteine sind projektunabhängig und können ohne weitere Maßnahmen in jedem Projekt verwendet werden.

▶ Wählen Sie ‹ Einfügen → Variable einfügen› .

02/02 AWB27-1307-D

Sie erhalten folgenden Dialog:

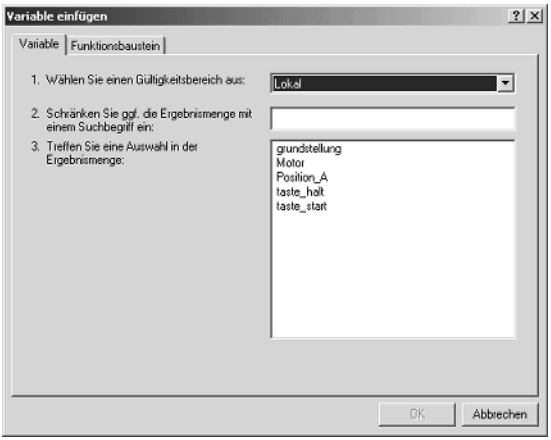


Abbildung 54: Variable einfügen

Oben können Sie den Gültigkeitsbereich der Funktionsbausteine einstellen. Sie sehen nur die Funktionsbausteine, die dem entsprechenden Bereich bei der Deklaration zugeordnet wurden. Sie haben Ihren Funktionsbaustein „Lichtkette“ LOKAL angelegt.

▶ Wählen Sie „Lichtkette“ aus und achten Sie darauf, dass „FB-Instanzen mit Aufrufschablone“ angewählt ist.

Sie erhalten im Anweisungsteil folgende Schablone:

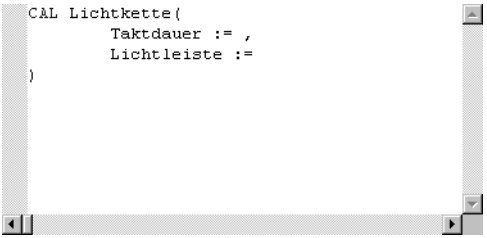


Abbildung 55: Schablone

▶ Vervollständigen Sie diese Schablone wie folgt:

```
CAL Lichtkette(Taktdauer := Lauftempo,

  Lichtleiste := Anzeige )
```

Der Input-Variablen „Taktdauer“ wird die Variable „Lauftempo“ zugewiesen, der IN_OUT-Variablen „Lichtleiste“ wird „Anzeige“ zugeordnet.

Sollten Sie den Aufruf des Bausteins von Hand eingeben, beachten Sie bitte, dass Parameter an Variablen vom Typ IN_OUT nur innerhalb der Klammern übergeben werden dürfen und dass IN_OUT-Parameter immer mit einem Wert versorgt werden müssen.

▶ Speichern Sie die POE unter dem Namen „BSP_PS4“.

▶ Führen Sie die Syntaxprüfung durch.

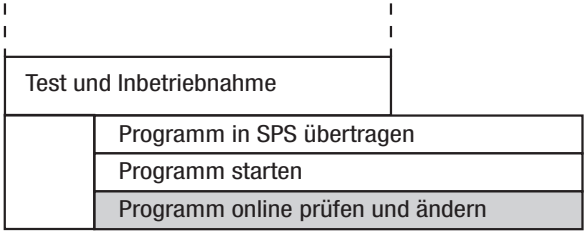
Wenn Sie das Programm fehlerfrei nach dem vorliegenden Entwurf eingegeben haben, wird die Syntaxprüfung erfolgreich abgeschlossen. Sollte die Syntaxprüfung Fehler melden, überprüfen Sie alle Eingaben, korrigieren Sie die gefundenen Fehler, speichern Sie die POE, und starten Sie die Syntaxprüfung erneut.

Bevor das Programm online geprüft und geändert werden kann, muss Ihr syntaxgeprüftes Programmbeispiel in ein ablauffähiges SPS-Programm übersetzt, in die Steuerung transferiert und gestartet werden. Dazu müssen Sie die Topologie definieren und den Programmcode generieren. Beides wurde für das Programmbeispiel 1 ausführlich in Kapitel 7 und Kapitel 8 erläutert. Für das Programmbeispiel 2 gestaltet sich der Arbeitsablauf gleich.

Im Folgenden wird deshalb nur noch auf den Programmtest und auf Online-Änderungen für das Programmbeispiel 2 eingegangen.

02/02 AWB27-1307-D

Beispielprogramm online prüfen und ändern



Die einzelnen POEs des bearbeiteten Programms können während der Programmausführung angezeigt und bei Bedarf geändert werden. Die Aufgabe für diesen Abschnitt ist es, die Laufrichtung und die Laufgeschwindigkeit zu ändern. Dazu muss das Fenster „Programm“ geöffnet, die POE „LICHT“ ausgewählt und der dazugehörige Anweisungsteil eingeblendet werden.

▶ Wählen Sie in der T & I, bei geöffnetem Fenster Verbindungsliste, die Schaltfläche „Programm“.

 *„Programm“*

Es öffnet sich das Fenster „Programm“.

Zunächst ist nur „RESSOURCE“ als oberste Ebene im linken Fensterteil zu sehen; im rechten steht der Name der Programm-POE. Durch Doppelklicks im linken Fenster können Sie die untergeordneten Ebenen des Gesamtprogramms einblenden. Wird im linken Fenster ein Programm markiert, so erscheinen im rechten Fenster die Programmkomponenten der darunter liegenden Ebene; d. h. Funktionsbaustein-POEs.

▶ Klicken Sie doppelt auf „Ressource“.

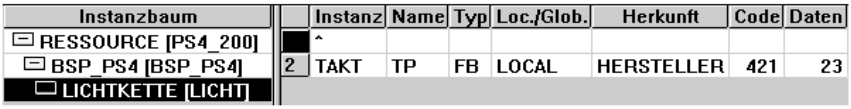
Im linken Fenster wird der Name der Programm-POE „BSP_PS4“ eingeblendet.

▶ Klicken Sie im linken Fenster doppelt auf „BSP_PS4 [BSP_PS4]“.

Es wird der untergeordnete Funktionsbaustein „LICHT“ eingeblendet.

▶ Klicken Sie im linken Fenster doppelt auf „Lichtkette [LICHT]“.

Im rechten Fenster wird als unterste Ebene der Funktionsbaustein „TP“ angezeigt.



▶ Markieren Sie den Funktionsbaustein „LICHTKETTE [LICHT]“ und wählen Sie die Schaltfläche „POE anzeigen/ändern“.

 *POE anzeigen/ändern*

Es öffnet sich der POE-EDITOR im Online-Modus mit der angewählten POE.

In diesem Fenster können Sie sich die Variablenzustände anzeigen lassen, eine Möglichkeit, die Ihnen bei der Fehlersuche nützlich sein kann.

▶ Wählen Sie im Menü Online den Befehl „Zustandsanzeige“.

 *Zustandsanzeige*

Im Anweisungs- und Deklarationsteil werden die Zustände der einzelnen Variablen eingeblendet, deren Werte ständig aktualisiert werden:

02/02 AWB27-1307-D

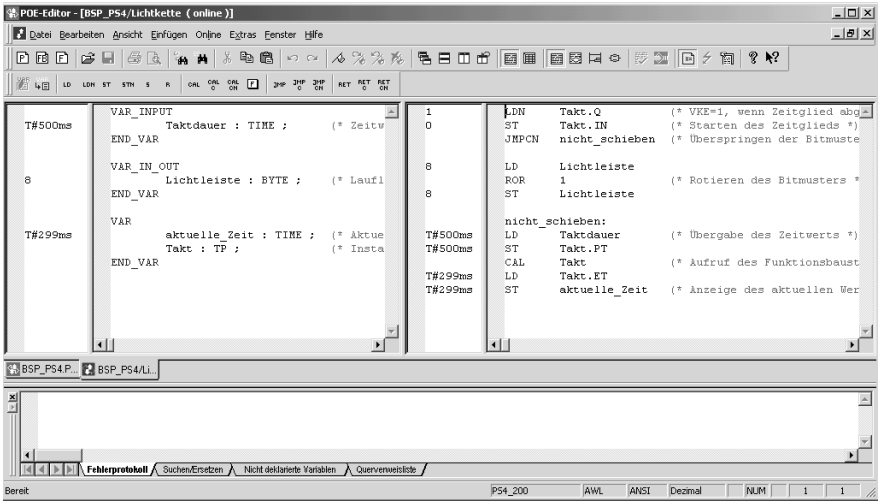


Abbildung 56: AWL-EDITOR im Online-Modus mit Zustandsanzeige der Variablen

- Betätigen sie nochmals den Befehl oder das Symbol „Zustandsanzeige“, um das Aktualisieren der Variablenzustände zu beenden. Auf dem Bildschirm werden die zuletzt aktualisierten Werte „eingefroren“ dargestellt.

Änderungen im POE-EDITOR (Online-Modus)

Mit dem POE-EDITOR im Online-Modus können Sie Ihr Programm online ändern. Dies hat den Vorteil, dass Sie sich von allen Nebenarbeiten wie POE speichern, Programmcode erzeugen und neues Programm in die Steuerung transferieren, entlastet werden. Wenn sie die Schaltfläche „Aktualisieren“ drücken, wird jede Änderung automatisch in der dazugehörigen POE-Quelldatei und im ablauffähigen Programm nachgezogen.

Laufrichtung ändern

Die Änderung der Laufrichtung ist möglich, indem der Befehl „ROR“ (Rotieren rechts) durch den Befehl „ROL“ (Rotieren links) ersetzt wird.

Durch die Anordnung der Digital-Ausgänge bei PS4-Steuerungen mit dem niederwertigen Bit auf der linken und dem höchstwertigen Bit auf der rechten Seite bewegt sich das Lauflicht mit dem Befehl ROR nach links statt rechts und umgekehrt bei dem Befehl ROL nach rechts statt links. Der Grund dieser umgekehrten Laufrichtung liegt darin, dass sich die Rotierbefehle auf die in der Digitaltechnik übliche Darstellung beziehen, bei der das niederwertige Bit auf der rechten Seite steht.

Die Laufgeschwindigkeit kann auf 1 Hz geändert werden, indem die Zeitwertvorgabe für den Funktionsbaustein TP auf eine Sekunde definiert wird.

▶ Klicken Sie in die Programmzeile, in der sich der Befehl „ROR“ befindet, und ändern Sie ihn in „ROL“.

Der Befehl und die Schaltfläche „Aktivieren“ sind nun verfügbar.

▶ Wählen Sie die Schaltfläche „Aktivieren“.

 *„Aktivieren“*

Die Änderung wird automatisch im Funktionsbaustein gespeichert, und es wird ein neuer Programmcode erzeugt. Danach wird der geänderte Programmcode automatisch in die Steuerung übertragen und ist sofort bei der Programmausführung wirksam – die Laufrichtung an den LEDs der angesteuerten Ausgabebaugruppe ändert sich.

02/02 AWB27-1307-D

Laufgeschwindigkeit ändern

Die Laufgeschwindigkeit wurde festgelegt, indem dem Eingangsoperanden PT im Funktionsbaustein „TP“ die Eingangsvariable „Taktdauer“ zugeordnet wurde. Der konkrete Zeitwert wird erst beim Aufruf des Funktionsbausteins „LICHT“ übergeben, wobei die Variable „Taktdauer“ den Wert der im Programm deklarierten Variable „Lauftempo“ übernimmt. Dieser Wert wurde im Deklarationsteil der Programm-POE mit 500 ms initialisiert.

Da mit dem POE-EDITOR im Online-Modus nur der Anweisungsteil, nicht aber der Deklarationsteil der Programm-POE geändert werden kann, muss hier der neue Zeitwert beim Aufruf des Funktionsbausteins in Form einer Zeitkonstanten übergeben werden. Diese Zeitkonstante könnte bereits im Funktionsbaustein „LICHT“ festgelegt und an den Standard-Baustein „TP“ übergeben werden, jedoch wäre in diesem Fall der Funktionsbaustein „LICHT“ nicht mehr „neutral“. Der neue Zeitwert soll daher beim Aufruf des Funktionsbausteins „LICHT“ in der Programm-POE „BSP_PS4“ eingegeben werden. Anstelle der Variable Lauftempo – die mit 500 ms initialisiert wurde – soll jetzt der Variablen „Taktdauer“ die Zeitkonstante von 1 Sekunde übergeben werden:

```
CAL Lichtkette(
  Taktdauer := T#1s,
  Lichtleiste :=Anzeige
)
```

- ▶ Schließen Sie die online geöffnete POE.
- ▶ Wählen Sie die „POE“ „BSP_PS4“ aus dem Instanzbaum aus: Markieren Sie die POE auf der linken Seite des Programmfensters im Instanzbaum, und betätigen Sie die Schaltfläche „POE anzeigen/ändern“.

- ▶ Klicken Sie mit der Maus an die zu ändernde Stelle.
- ▶ Ändern Sie die Zeitwertvorgabe in die Konstante T#1s.
- ▶ Wählen Sie die Schaltfläche „Aktivieren“.

 *„Aktivieren“*

Die Änderung wird im Programmiergerät und in der Steuerung gespeichert.

Die geforderte Änderung ist vollständig: Das Lauflicht hat die Laufrichtung und die Laufgeschwindigkeit geändert.

- ▶ Schließen Sie die online geöffnete POE.
- ▶ Beenden Sie T & I.

Mehrfachinstanzierung des FB Licht

Um den Vorteil der adressunabhängigen Funktionsbausteinprogrammierung mit lokalen Variablen zu zeigen, finden Sie hier eine einfache Erweiterung der Aufgabe 2.

Der einmal erstellte Funktionsbaustein LICHT, der nun als abgeschlossenes und getestetes Modul betrachtet wird, soll ein zweites Mal instanziert werden. Wenn Sie das Beispiel mit einer Kompaktsteuerung aus der Familie PS4-200 nachvollziehen, können Sie damit ein weiteres Lauflicht auf dem Ausgabebyte des Erweiterungsmoduls EM4-101-DD1 steuern. Bei der Modularsteuerung PS416 soll die Ausgabe des zweiten Lauflichts auf dem Ausgangsbyte 1 der digitalen Ausgabegruppe PS416-OUT-400 erfolgen. Hierfür ist keine Änderung in der Topologie-Konfiguration für die PS416 notwendig.

Bei der PS4 muss zunächst die Topologie-Konfiguration um das Erweiterungsmodul ergänzt werden.

02/02 AWB27-1307-D

- ▶ Öffnen Sie im TOPOLOGIE-KONFIGURATOR die Topologie-Konfiguration „GERAET“.
- ▶ Markieren Sie die PS4 und wählen Sie ‹ Bearbeiten → Dezentral erweitern› .
- ▶ Wählen Sie aus der Liste den Eintrag „EM4-101-DD1/88“ und bestätigen Sie mit OK.

Dieser Eintrag konfiguriert das Erweiterungsmodul für 8 Eingänge und 8 Ausgänge.

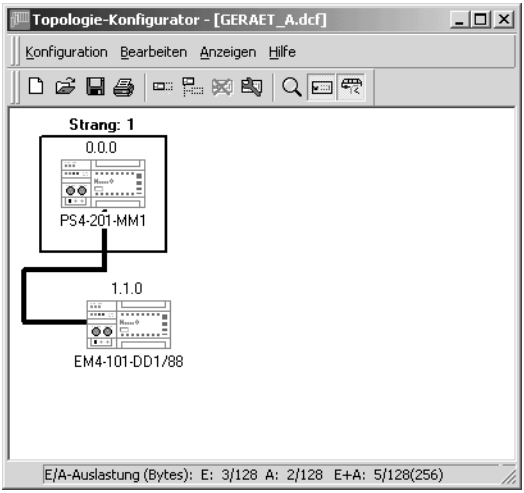


Abbildung 57: TOPOLOGIE-KONFIGURATOR

- ▶ Speichern Sie die Konfiguration.
- ▶ Öffnen Sie das Programm „BSP_PS4“ im POE-EDITOR und erweitern Sie die Variablendeklaration im Variableneditor um die Variablen für eine zweite Lichtkette.

Um eine zweite Instanz des Funktionsbausteins LICHT anzulegen, wird der Funktionsbaustein ein zweites Mal, diesmal unter dem Instanznamen „Lichtkette2“, deklariert.

Um zu sehen, dass die zweite Instanz ohne weiteren Aufwand völlig unabhängig von der ersten Instanz „Lichtkette“ läuft, deklarieren Sie noch zwei Variablen für die Versorgung der Parameter von „Lichtkette2“.

Die Variable „Lauftempo2“ wird mit dem Wert T#250ms initialisiert, was für das zweite Lauflicht eine doppelt so hohe Taktfrequenz ergibt wie für das erste Lauflicht. Die Ausgabe des zweiten Lauflichts erfolgt beim Beispielaufbau mit einer PS4 auf dem Ausgangsbyte des EM4-101-DD1. Die erste Stelle der Adresse kennzeichnet den Netzwerkstrang, an den der anzusprechende Teilnehmer angeschlossen ist, also in diesem Fall Strang 1. Das EM4-101-DD1 ist in unserem Beispiel der erste Teilnehmer in diesem Strang, also lautet die zweite Stelle der Adresse ebenfalls „1“. Da das Erweiterungsmodul jenes Modul ist, welches direkt als Teilnehmer am Netzwerkstrang angeschlossen ist, hat es die Kennzeichnung „Modul 0“. Daraus ergibt sich die dritte Stelle der Adresse. Die vierte Stelle der Adresse schließlich kennzeichnet das anzusprechende Byte, also das Byte 0. Die komplette Adresse für die Ausgabe des zweiten Lauflichts lautet also %QB1.1.0.0 für das Ausgangsbyte des EM4-101-DD1.

02/02 AWB27-1307-D

Die erweiterte Variablendeklaration für den Aufbau mit der Kompaktsteuerung lautet damit:

```
VAR

  Lichtkette: LICHT ;
  Lauftempo : TIME := T#500ms ;
  Anzeige AT %QB0.0.0.0 : BYTE := 1 ;
  Lichtkette2 : LICHT ;    (* Zweite Instanz des
                              Bausteins LICHT   *)

  Lauftempo2 : TIME := T#250ms ; (* Taktzeit für zweite
                                    Lichtkette => 4 Hz *)

  Anzeige2 AT %QB1.1.0.0 : BYTE := 1;(* Ausgabe auf dem
                                        EM4-101-DD1       *)

END_VAR
```

Die erweiterte Variablendeklaration für den Aufbau mit der PS416 lautet dementsprechend:

```
VAR

  Lichtkette : LICHT ;
  Lauftempo : TIME := T#500ms ;
  Anzeige AT %QB0.0.0.0 : BYTE := 1 ;
  Lichtkette2 : LICHT ;    (*Zweite Instanz des
                             Bausteins LICHT    *)

  Lauftempo 2: TIME := T#250ms ; (*Taktzeit für zweite
                                    Lichtkette => 4 Hz *)

  Anzeige2 AT %QB0.0.0.1 : BYTE := 1; (*Ausgabe auf dem
                                        HIGH Byte der OUT-400
                                        *)

END_VAR
```

Nun muss noch im Anweisungsteil der Aufruf der zweiten Bausteininstanz eingebunden werden.

- ▶ Positionieren Sie den Cursor am bisherigen Programmende.
- ▶ Wählen Sie ‹ Einfügen → Variable einfügen...› verwenden oder die entsprechende Schaltfläche.

„Variable einfügen“

Im Dialog wählen Sie nun ggf. den Gültigkeitsbereich „LOKAL“ aus und klicken Sie doppelt auf „Lichtkette2“.

Der neue Baustein erscheint als Aufrufschablone mit allen Operanden, denen von Ihnen die Variablen „Lauftempo2“ und „Anzeige2“ zugewiesen werden.

```
CAL Lichtkette(
        Taktdauer := Lauftempo,
        Lichtleiste := Anzeige
)

CAL Lichtkette2(
        Taktdauer := Lauftempo2,
        Lichtleiste := Anzeige2
)
```

Abbildung 58: Aufruf von „Lichtkette“ und „Lichtkette2“

- ▶ Speichern Sie die POE und erstellen Sie in der Programmcode-Erzeugung eine neue Generierliste.
- ▶ Erzeugen Sie mit ‹ Generierung → Programmcode generieren› oder der entsprechenden Schaltfläche ein neues Programm und transferieren Sie es in die Steuerung.

02/02 AWB27-1307-D

Nach dem Start sehen Sie, dass das Lauflicht auf dem EM4-101-DD1 doppelt so schnell läuft wie auf der PS4. Auf der PS416-OUT-400 läuft das Lauflicht auf dem HIGH Byte doppelt so schnell wie auf dem LOW Byte. Sie haben also einen selbst geschriebenen Funktionsbaustein auf einfache Art und Weise zweimal instanziert und verwendet, ohne dass Sie sich darum kümmern mussten, wo die im FB LICHT intern verwendeten lokalen Variablen in der Steuerung abgelegt werden. Auch der im Baustein LICHT verwendete Hersteller-Baustein TP wird automatisch mehrfach instanziert, ohne dass Sie sich um die Zuordnung von Datenbereichen oder Merkeradressen kümmern müssen.

In KOP oder FBS sieht das erweiterte Programm „BSP_PS4“ dann so aus:

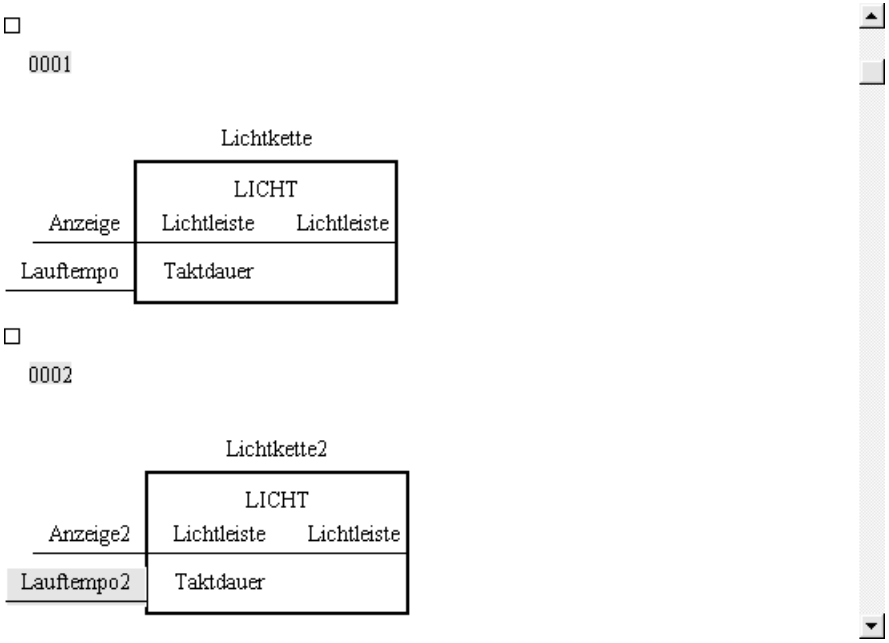


Abbildung 59: Erweitertes Programm in KOP/FBS

Ein-/Ausgänge auf der PS4 anzeigen/zwangssetzen

Sie können die Zustände der Ein- und Ausgänge und die Diagnosedaten der Steuerung und des Erweiterungsmoduls beobachten.

- ▶ Starten Sie die T & I.
- ▶ Klicken Sie auf die Schaltfläche „Topologie“. Die aktuelle Topologie-Konfiguration wird aus der Steuerung gelesen und dargestellt.
- ▶ Markieren Sie bei gedrückter „Strg“-Taste beide Geräte PS4-141-MM1 und EM4-101-DD1/88 mit der linken Maustaste, und klicken Sie auf die Schaltfläche „Ein-/Ausgänge anzeigen/zwangssetzen...“.

Die Ein- und Ausgänge der gewählten Komponenten werden in einem Fenster dargestellt. Bei einer CPU vom Typ PS4 werden die Ein-und Ausgänge sowohl im RUN als auch im HALT angezeigt; bei einer CPU vom Typ PS416 werden die Ausgänge nur im Zustand HALT angezeigt.

02/02 AWB27-1307-D

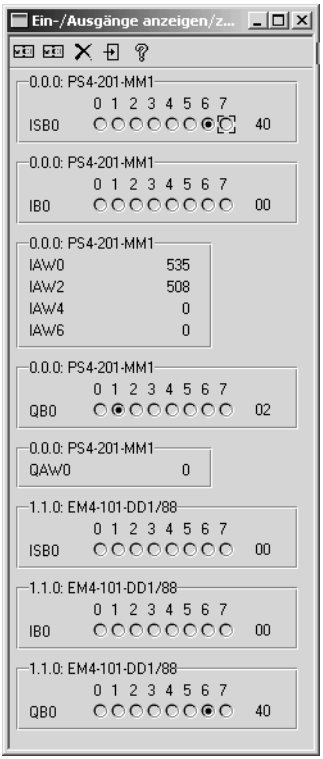


Abbildung 60: Zustandsanzeige der Ein-/Ausgänge

Um den Status einzelner Geräte in einem vernetzten System abzufragen, kann in der T & I eine Überwachungsfunktion über die Schaltfläche „Netzwerkdiagnose“ eingeschaltet werden. Gestörte Geräte oder Baugruppen werden mit einer Schraffur hinterlegt. Der Master eines Suconet-K-Stranges, in dem ein Teilnehmer gestört ist, wird zusätzlich durch einen Blitz gekennzeichnet. Sie können dies ausprobieren, indem Sie bei eingeschalteter Funktion „Netzwerkdiagnose“ das Verbindungskabel zwischen der Steuerung und dem EM4-101-DD1 ziehen.

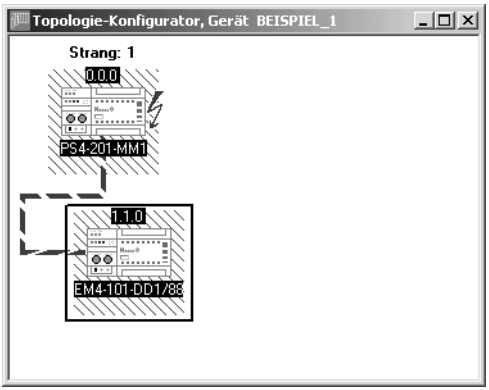


Abbildung 61: Netzwerkdiagnose mit gestörtem Gerät

Genauere Informationen über die Fehlerursache erhalten Sie, wenn Sie ein Gerät oder eine Baugruppe markieren und die Schaltfläche „Diagnosestatus...“ anklicken.

Schaltfläche „Diagnosestatus“

Es erscheint folgender Dialog:

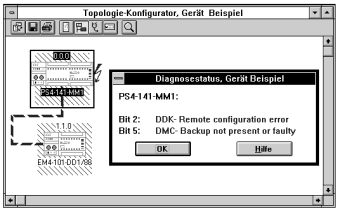


Abbildung 62: Diagnosestatus des gestörten Teilnehmers

Sie haben jetzt alle Programmteile der Sucosoft S40 kennengelernt. Natürlich konnten in diesem „Kurzeinstieg“ nicht alle Funktionen berücksichtigt werden, die die Sucosoft S40 bietet. Wir hoffen aber, Sie haben einen Überblick gewonnen und freuen sich darauf, weitere Möglichkeiten auszuprobieren und zu erforschen. Die einzelnen Arbeitsschritte

02/02 AWB27-1307-D

finden Sie in den Nachschlagewerken zur Sucosoft Benutzeroberfläche (AWB2700-1305-D) sowie zu den Sprachelementen für PS4-200, PS4-300 und PS416 (AWB2700-1306-D). Dort sind alle zur Verfügung stehenden Möglichkeiten erläutert.

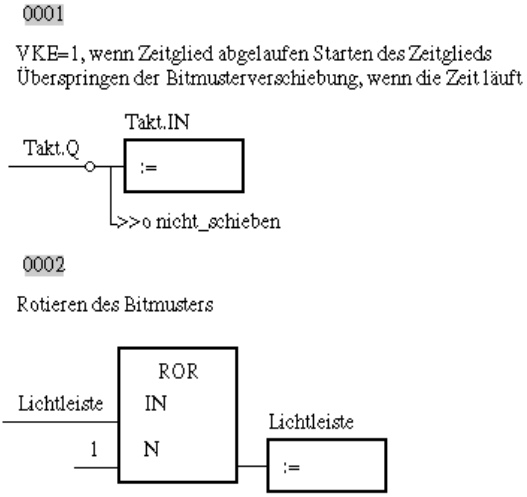
KOP/FBS-Darstellung Programmieraufgabe 2

Abschließend finden Sie hier die Anweisungsteile der Programmieraufgabe 2 für das Programm „BSP_PS4“ in den grafischen Programmiersprachen KOP und FBS dargestellt.

Das Vorgehen zur Erzeugung eines ablauffähigen Programms, zum Transferieren und zum Starten des Programms entspricht dem Vorgehen, wie es für die Programmiersprache AWL beschrieben wurde.

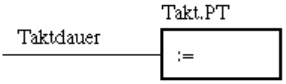
Funktionsbaustein LICHT

FBS-Darstellung:



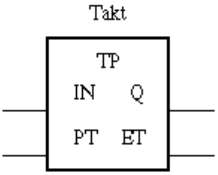
0003 nicht_schieben

Übergabe des Zeitwerts



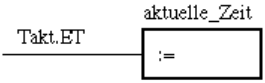
0004

Aufruf des Funktionsbausteins



0005

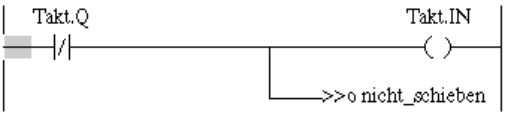
Anzeige des aktuellen Wertes für den Online-Test



KOP-Darstellung:

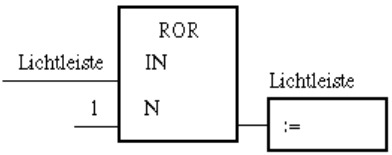
0001

VKE=1, wenn Zeitglied abgelaufen Starten des Zeitglieds
Überspringen der Bitmusterverschiebung, wenn die Zeit läuft



0002

Rotieren des Bitmusters

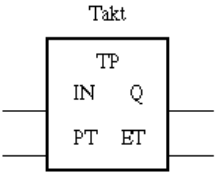


0003 nicht_schieben

Übergabe des Zeitwerts

0004

Aufruf des Funktionsbausteins



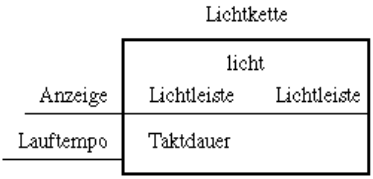
0005

Anzeige des aktuellen Wertes für den Online-Test

Programm BSP_PS4

KOP und FBS-Darstellung:

0001



Stichwortverzeichnis

02/02 AWB27-1307-D

S

T

U