SIEMENS

SIMATIC

Erste Schritte und Übungen mit STEP 7

Getting Started

Diese Dokumentation ist Bestandteil des Dokumentationspaketes mit der Bestellnummer:
6ES7810-4CA08-8AW0

Ausgabe 03/2006
C79000-P7000-C48-01

Sicherheitshinweise

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

Gefahr
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Warnung
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht
mit Warndreieck bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht
ohne Warndreieck bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Achtung
bedeutet, dass ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zugehörige Gerät/System darf nur in Verbindung mit dieser Dokumentation eingerichtet und betrieben werden. Inbetriebsetzung und Betrieb eines Gerätes/Systems dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieser Dokumentation sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

Beachten Sie Folgendes:

Warnung
Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden. Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Automation and Drives
Postfach 4848
90437 NÜRNBERG
DEUTSCHLAND

C79000-P7000-C48-01
03/2006

Willkommen zu STEP 7 ...

... der SIMATIC Basissoftware für die Erstellung von SPS-Programmen in KOP, FUP oder AWL für SIMATIC S7-300/400 Stationen.

Informationen zu diesem Getting Started

In diesem Buch lernen Sie die Grundlagen von SIMATIC STEP 7 kennen. Wir zeigen Ihnen die wichtigsten Bildschirmdialoge und Vorgehensweisen anhand von praktischen Übungen, die so aufbereitet sind, dass Sie fast mit jedem beliebigen Kapitel loslegen können.

Sie finden in jedem Unterkapitel einen grau gekennzeichneten erklärenden Teil und einen ablauforientierten, grünen Teil. Die Arbeitsanweisungen beginnen mit einem Pfeil im grünen Balken, erstrecken sich teilweise über mehrere Seiten und schließen mit einem Punkt plus weiterführender Information ab.

Hilfreich ist es, wenn Sie bereits mit Maus, Fenstertechnik, Pulldown-Menüs usw. arbeiten können und SPS-Grundkenntnisse haben.

In STEP 7 Trainingskursen können Sie über das Getting Started hinaus Ihr Know-how vertiefen und lernen, wie komplette Automatisierungslösungen mit STEP 7 erstellt werden.

Voraussetzungen zum Arbeiten mit dem Getting Started

Um die praktischen Übungen zu STEP 7 in diesem Getting Started durchführen zu können, benötigen Sie

- ein Siemens Programmiergerät oder einen PC
- das STEP 7 Softwarepaket und den entsprechenden License Key
- ein Automatisierungssystem SIMATIC S7-300 oder S7-400
  (für das Kapitel 7 „Laden und Testen des Programms“)

Weitere Dokumentation zu STEP 7

- STEP 7 Grundwissen
- STEP 7 Referenzwissen

Die elektronischen Handbücher finden Sie nach der Installation von STEP 7 im Startmenü unter **Simatic > Dokumentation** oder sie sind über jede Siemens Verkaufsniederlassung bestellbar. Sämtliche Informationen aus den Handbüchern sind in STEP 7 über die Online-Hilfe abrufbar.

Wir wünschen Ihnen viel Spaß und Erfolg!

Ihre SIEMENS AG

Inhaltsverzeichnis

In den Kapiteln 6 und 7 bauen Sie die Hardware auf und testen Ihr Programm.

In den Kapiteln 8 bis 11 vertiefen Sie Ihr Wissen durch neue Funktionen.

1 Einführung in STEP 7

1.1 Was werden Sie lernen

Anhand praktischer Übungen vermitteln wir Ihnen, wie einfach das Programmieren in KOP, FUP oder AWL mit STEP 7 ist.

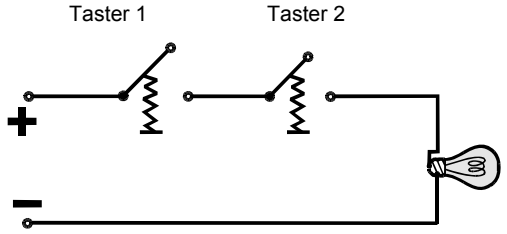
Detaillierte Arbeitsanweisungen in den einzelnen Kapiteln zeigen Ihnen schrittweise die umfangreichen Verwendungsmöglichkeiten von STEP 7.

Programm mit binären Verknüpfungen erstellen

In den Kapiteln 2 bis 7 erstellen Sie ein Programm mit binären Verknüpfungen. Über die programmierten Verknüpfungen sprechen Sie die Ein- und Ausgänge Ihrer CPU (falls vorhanden) an.

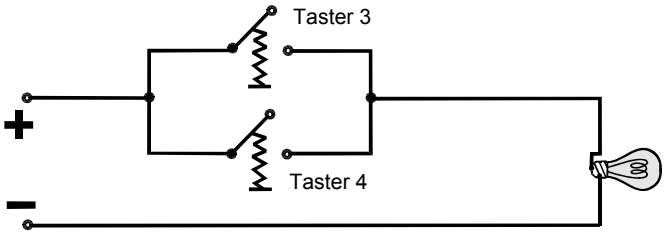
Die Programmierbeispiele im "Getting Started" bauen unter anderem auf drei fundamentalen binären Verknüpfungen auf.

Die erste binäre Verknüpfung, die Sie später programmieren, ist die UND-Funktion. Die UND-Funktion kann in einer elektrischen Schaltung mit zwei Tastern verdeutlicht werden.



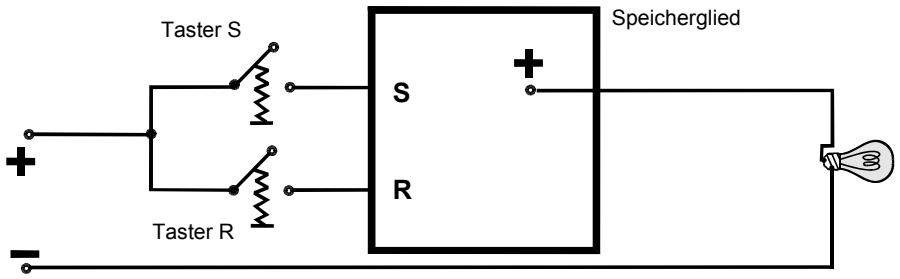
Wird Taster 1 **und** Taster 2 gedrückt, brennt die Lampe.

Die zweite binäre Verknüpfung ist die ODER-Funktion. Die ODER-Funktion kann ebenfalls in einer elektrischen Schaltung dargestellt werden.



Wird Taster 3 **oder** Taster 4 gedrückt, brennt die Lampe.

Die dritte binäre Verknüpfung ist das Speicherglied. Die SR-Funktion reagiert in einer elektrischen Schaltung auf bestimmte Spannungszustände und gibt diese entsprechend weiter.

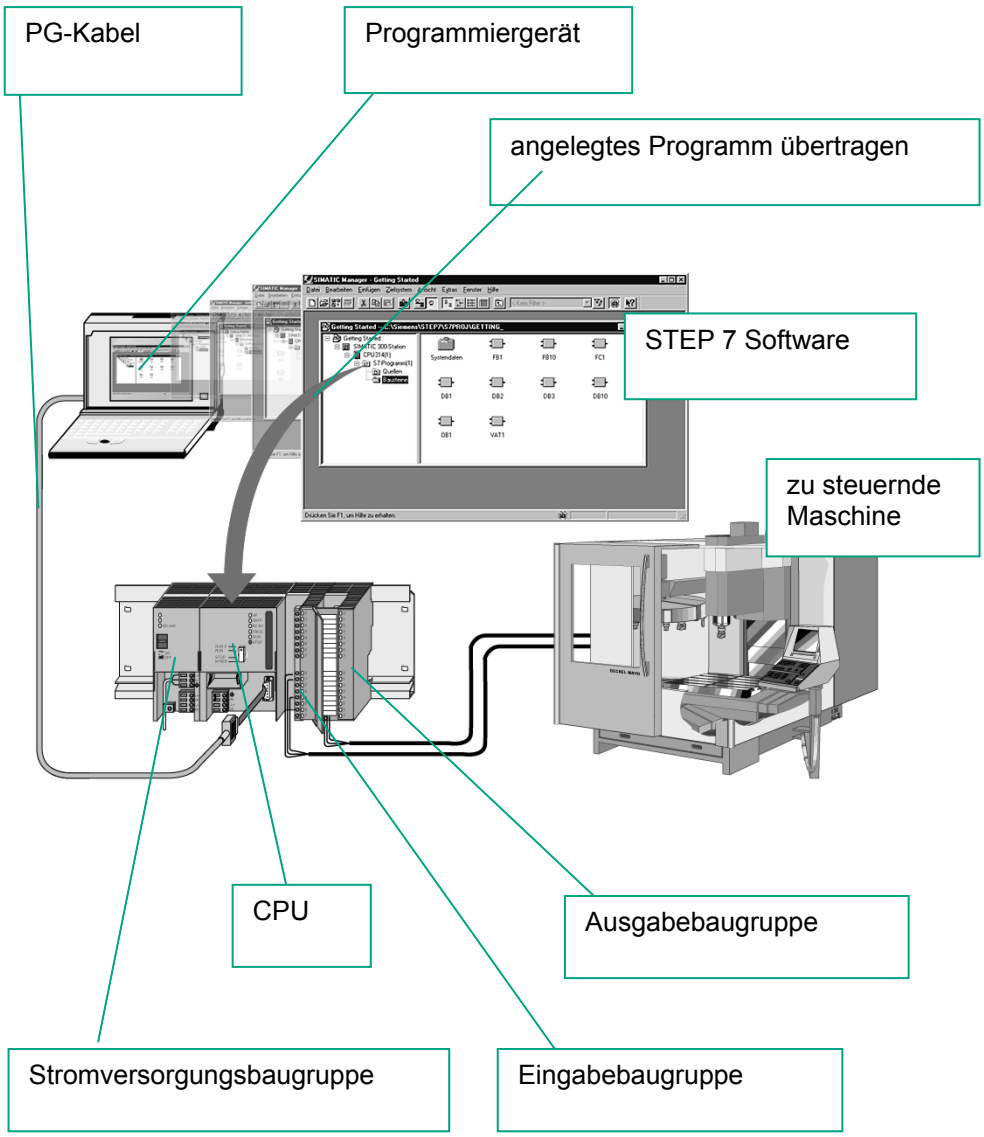


Wird Taster S gedrückt, brennt die Lampe solange, bis Taster R gedrückt wird.

1.2 Zusammenspiel von Software und Hardware

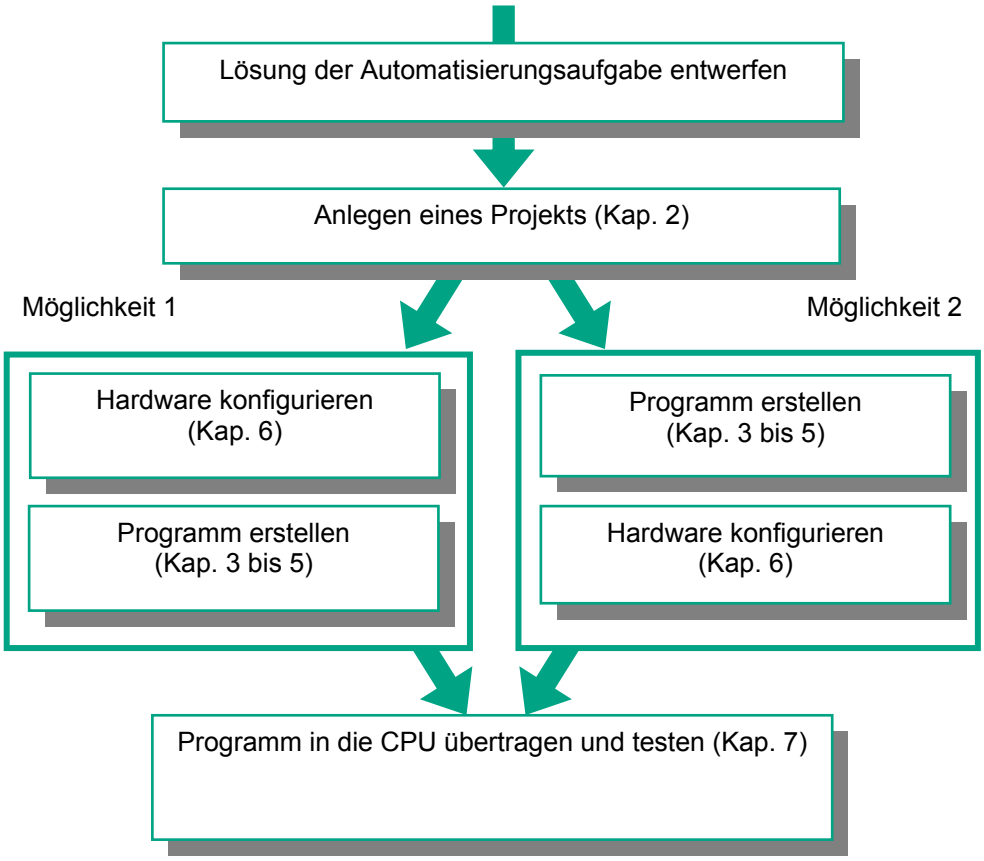
Mit der STEP 7 Software erstellen Sie innerhalb eines Projekts Ihr S7-Programm. Die S7-Steuerung besteht aus einer Stromversorgung, einer CPU und Ein- bzw. Ausgangsbaugruppen (E/A-Baugruppen).

Die Speicher-Programmierbare-Steuerung (SPS) überwacht und steuert mit dem S7-Programm Ihre Maschine. Die E/A-Baugruppen werden im S7-Programm über die Adressen angesprochen.



1.3 Grundsätzliche Vorgehensweise mit STEP 7

Bevor Sie ein Projekt anlegen, sollten Sie wissen, daß sich STEP 7 Projekte in unterschiedlicher Reihenfolge erstellen lassen.



Bei umfangreichen Programmen mit vielen Ein- und Ausgängen empfehlen wir zunächst, die Hardware zu konfigurieren. Mit dem Vorteil, daß STEP 7 mögliche Adressen im HW-Konfig Editor anzeigt.

Bei der zweiten Möglichkeit müssen Sie in Abhängigkeit Ihrer gewählten Komponenten die jeweiligen Adressen selbst ermitteln und können diese nicht über STEP 7 abrufen.

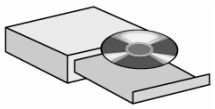
Mit der Hardwarekonfiguration können Sie nicht nur Adressen festlegen, sondern auch die Parameter und Eigenschaften von Baugruppen ändern. Für den Betrieb von mehreren CPUs ist es beispielsweise notwendig die MPI-Adressen der CPUs anzupassen.

Da im "Getting Started" nur wenig Ein- und Ausgänge notwendig sind, überspringen wir zunächst die Hardwarekonfiguration und beginnen sofort mit dem Programmieren.

1.4 Installieren von STEP 7

Unabhängig davon, ob Sie mit dem Programmieren oder Hardware Konfigurieren beginnen wollen, müssen Sie zunächst STEP 7 installieren. Falls Sie ein SIMATIC PG benutzen, ist STEP 7 bereits installiert.

Zur Installation der STEP 7 Software auf einem PG/PC ohne vorinstallierter STEP 7 Software beachten Sie bitte die Software- und Hardware-voraussetzungen. Sie finden diese in der Liesmich.wri auf der STEP 7 CD unter **<Laufwerk>:\STEP 7\Disk1**.



Falls Sie STEP 7 erst installieren müssen, legen Sie nun die STEP 7 CD ein. Das Installationsprogramm wird automatisch gestartet. Folgen Sie den Installationsanweisungen.

Falls der automatische Installationsstart nicht gelingt, finden Sie das Installationsprogramm auch auf der CD-ROM unter **<Laufwerk>:\STEP 7\Disk1\setup.exe**.

Nach Beendigung der Installation und Neustart des Rechners erscheint auf Ihrem Windows Desktop das Symbol "SIMATIC Manager".

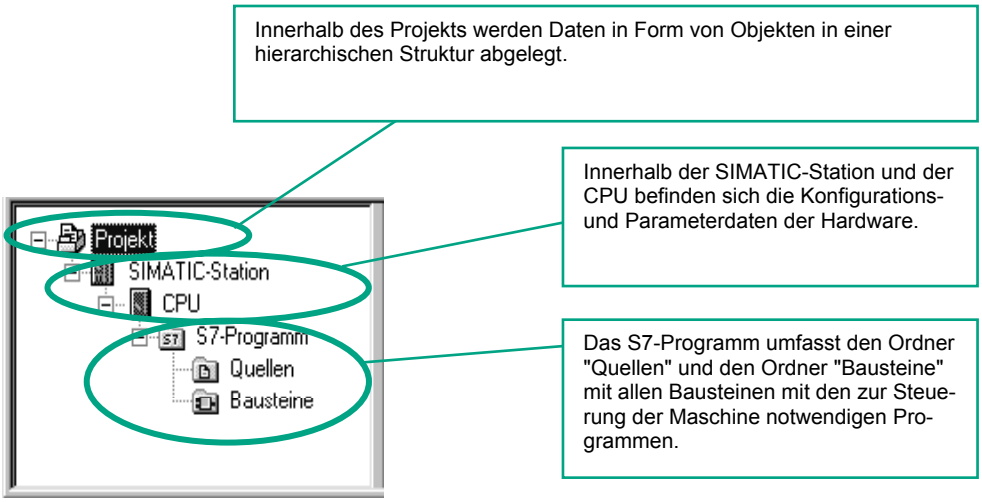
Wenn Sie nach der Installation auf das Symbol „SIMATIC Manager“ doppelklicken, wird automatisch der STEP 7 Assistent gestartet.

Weitere Hinweise zur Installation finden Sie in der Liesmich.wri Datei auf der STEP 7 CD unter **<Laufwerk>:\STEP 7\Disk1\Liesmich.wri**.

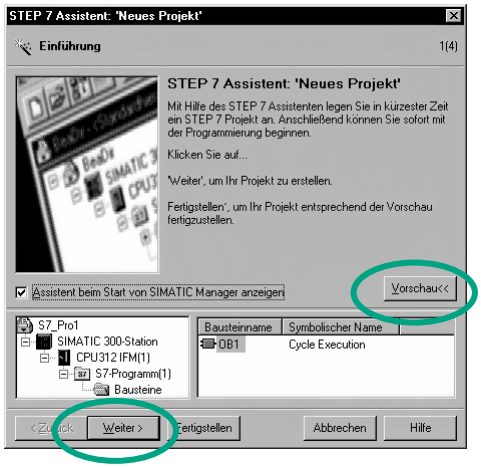
2 SIMATIC Manager

2.1 SIMATIC Manager starten und Projekt anlegen

Der SIMATIC Manager wird als zentrales Fenster nach dem Start von STEP 7 aktiv. In der Voreinstellung wird gleichzeitig der STEP 7 Assistent gestartet, der Sie beim Anlegen eines STEP 7 Projekts unterstützt. Die Projektstruktur dient dazu, alle anfallenden Daten und Programme geordnet abzulegen.

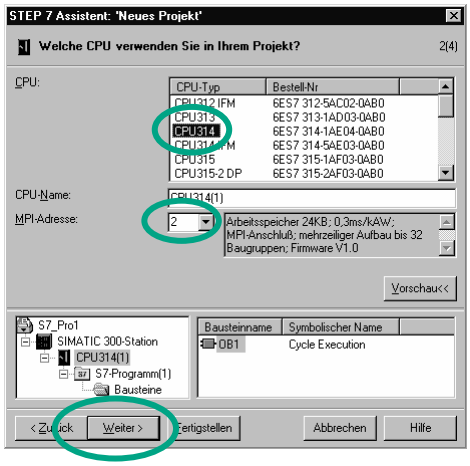


Doppelklicken Sie auf das Symbol **SIMATIC Manager** auf dem Windows Desktop. Wählen Sie den Menübefehl **Datei > Assistent "Neues Projekt"**, falls der Assistent nicht selbständig aktiviert wird.



Mit **Vorschau** lässt sich die Projektstruktur, die angelegt wird, ein- und ausblenden.

Zum zweiten Dialogfeld gelangen Sie mit **Weiter**.



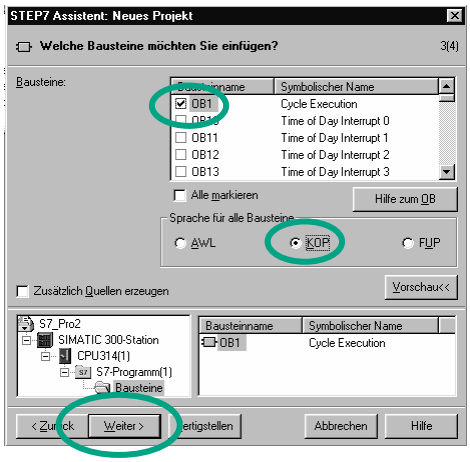
Wählen Sie für das Beispielprojekt „Getting Started“ die CPU 314 aus. Das Beispiel wurde so angelegt, dass Sie auch jederzeit die Ihnen gelieferte CPU auswählen können.

Belassen Sie die MPI-Adresse 2 in ihrer Voreinstellung.

Mit **Weiter** bestätigen Sie die Einstellungen und gelangen zum nächsten Dialogfeld.

Jede CPU hat bestimmte Eigenschaften, z.B. bezüglich Speicherausbau oder Operandenbereiche. Deshalb muss die CPU vor einer Programmierung ausgewählt werden.

Die MPI-Adresse (Multi Point Interface) wird für die Kommunikation Ihrer CPU mit dem PG/PC benötigt.



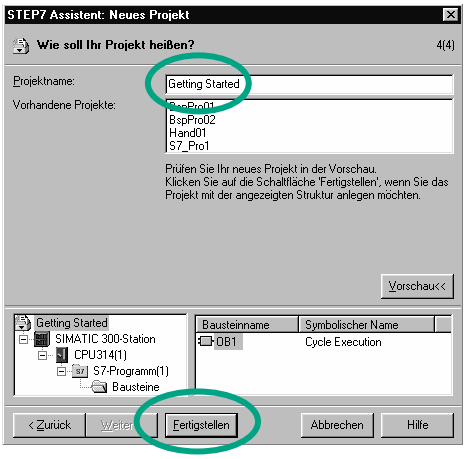
Selektieren Sie den Organisationsbaustein **OB1** (falls dieser nicht bereits angewählt ist).

Wählen Sie die Programmiersprache **KOP**, **FUP** oder **AWL**.

Bestätigen Sie Ihre Einstellungen mit **Weiter**.

Der OB1 repräsentiert dabei die oberste Programmebene und organisiert die anderen Bausteine des S7-Programms.

Die Auswahl der Programmiersprache können Sie zu einem späteren Zeitpunkt wieder ändern.



Wählen Sie im Schriftfeld „Projektname“ mit Doppelklick den vorgeschlagenen Namen an, und überschreiben Sie diesen mit „Getting Started“.

Mit **Fertigstellen** wird Ihr neues Projekt gemäß der Vorschau erzeugt und angelegt.

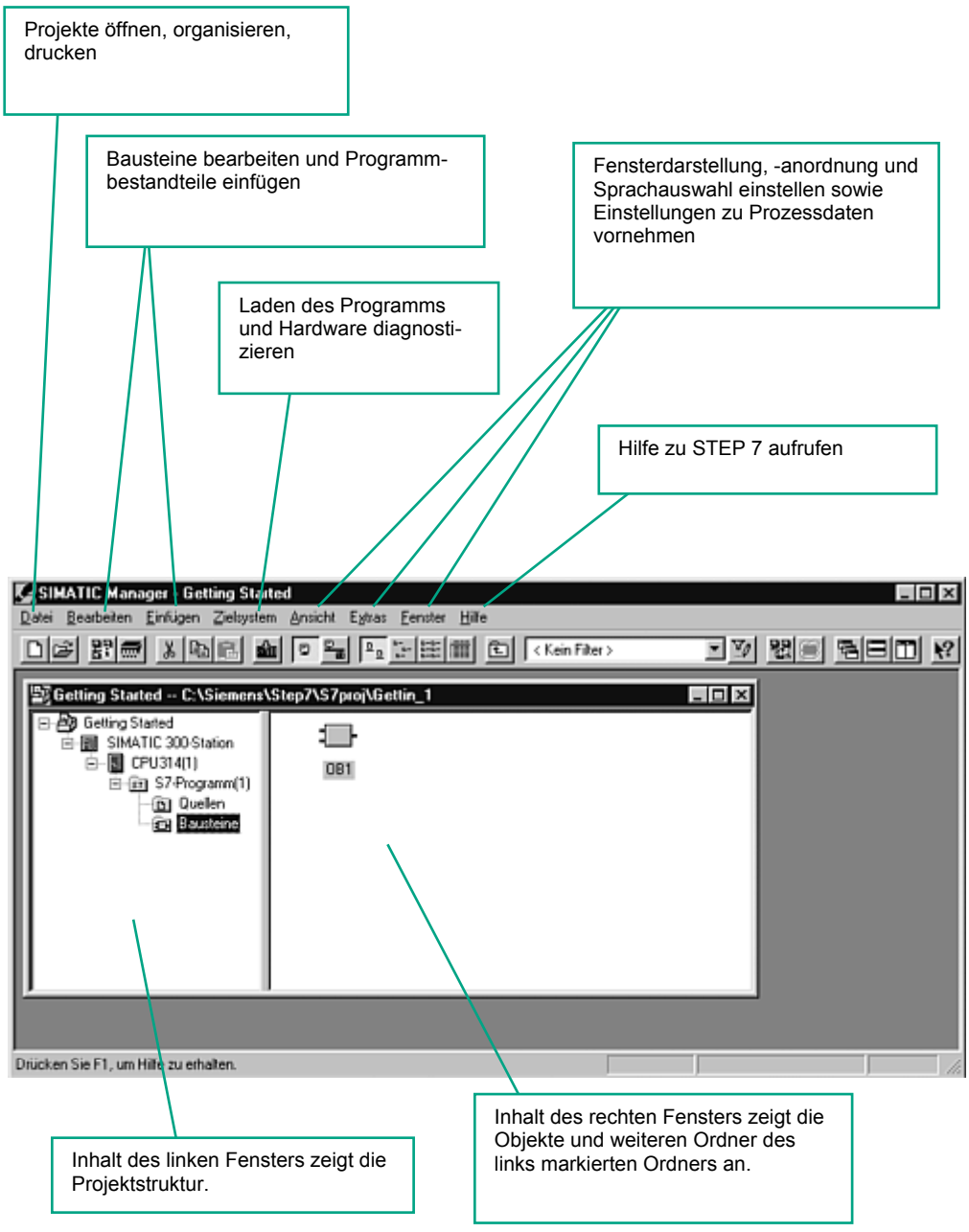
Nach dem Kommando **Fertigstellen** wird der SIMATIC Manager mit dem Fenster des angelegten Projekts „Getting Started“ geöffnet. Auf den nächsten Seiten zeigen wir Ihnen, welche Bedeutung die angelegten Dateien und Ordner haben und wie Sie damit effektiv arbeiten können.

Der STEP 7 Assistent kann bei jedem Programmstart aktiviert werden. Diese Voreinstellung können Sie im ersten Dialogfeld des Assistenten aktivieren. Erstellen Sie Projekte ohne den STEP 7 Assistenten, müssen Sie jedoch jedes Verzeichnis innerhalb des Projekts selbst anlegen.

Mehr Informationen über **Hilfe > Hilfethemen** „Einrichten und Bearbeiten des Projekts“.

2.2 Projektstruktur im SIMATIC Manager und Aufrufen der Hilfe zu STEP 7

Sobald der STEP 7 Assistent geschlossen ist, erscheint der SIMATIC Manager mit dem geöffneten Projektfenster „Getting Started“. Von ihm aus rufen Sie alle STEP 7 Funktionen und Fenster auf.

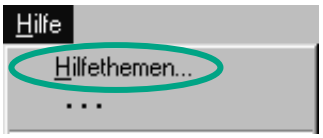


Aufrufen der Hilfe zu STEP 7

F1

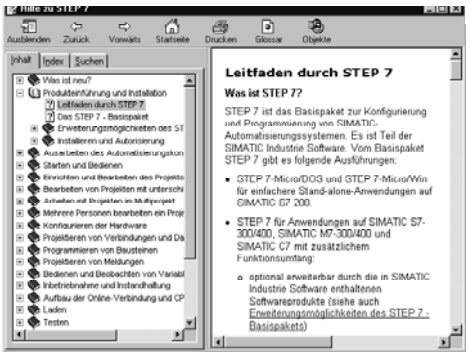
Möglichkeit 1:

Markieren Sie einen beliebigen Menübefehl und drücken Sie die Funktionstaste **F1**. Sie erhalten die kontextsensitive Hilfe zum markierten Menübefehl.



Möglichkeit 2:

Sie gelangen über das Menü zur Hilfe zu STEP 7.



Im linken Teilfenster erscheint das Inhaltsverzeichnis mit verschiedenen Hilfethemen und im rechten wird das angewählte Topic angezeigt.

Navigieren Sie zum gesuchten Thema, indem Sie im **Inhalt** auf das **+** klicken. Im rechten Fenster wird parallel hierzu der Inhalt des ausgewählten Topics dargestellt.

Mit **Index** und **Suchen** können Sie Suchbegriffe eingeben und gezielt nach Ihrem gewünschten Thema suchen.

Möglichkeit 3:

Klicken Sie in der Hilfe zu STEP 7 auf das Symbol "Startseite".

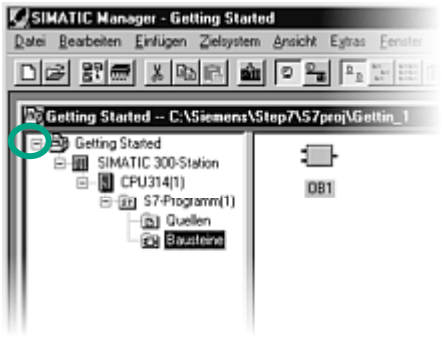
In der Startseite wird ein Infoportal eingeblendet, über das Sie einen kompakten Zugang zu den zentralen Themen der Onlinehilfe wie:

- Einsteigen in STEP 7
- Projektieren & Programmieren
- Testen & Fehler suchen
- SIMATIC im Internet

Möglichkeit 4:

Klicken Sie auf den Hilfezeiger. Der nächste Klick auf ein bestimmtes Objekt aktiviert die Hilfe.

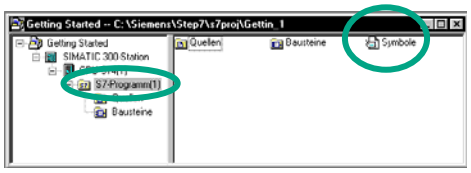
In der Projektstruktur navigieren



Das eben angelegte Projekt mit der ausgewählten S7-Station und der CPU wird Ihnen angezeigt.

Klicken Sie auf das **+** bzw. **–**, um die jeweiligen Ordner zu öffnen oder zu schließen.

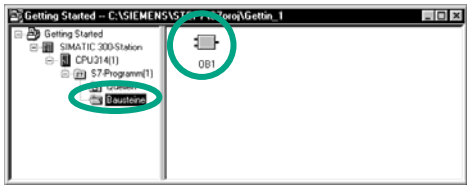
Über die angezeigten Symbole im rechten Fenster rufen Sie später weitere Funktionen auf.



Klicken Sie auf den Ordner **S7-Programm (1)**. Er enthält alle notwendigen Programmbestandteile.

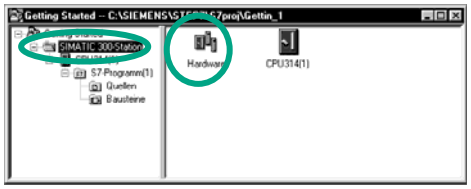
Über Symbole geben Sie im Kapitel 3 den Adressen symbolische Namen.

Der Ordner Quellen dient zur Ablage von Quellprogrammen. Quellprogramme werden im „Getting Started“ nicht behandelt.



Klicken Sie auf den Ordner **Bausteine**. Er enthält den bisher angelegten **OB1** und später alle weiteren Bausteine.

Über die Bausteine gelangen Sie in Kapitel 4 und 5 zur Programmeingabe in KOP, FUP oder AWL.



Klicken Sie auf den Ordner **SIMATIC 300 Station**. Hier werden alle hardwarebezogenen Projektdaten abgelegt.

Über **Hardware** spezifizieren Sie im Kapitel 6 die Parameter Ihres Automatisierungssystems.

Falls Sie für Ihre Automatisierungsaufgabe SIMATIC Erweiterungssoftware benötigen, wie beispielsweise die Optionspakete PLC-SIM (Hardware Simulationsprogramm) oder S7-GRAPH (grafische Programmiersprache), werden auch diese in STEP 7 integriert. Über den SIMATIC Manager können Sie die zugehörigen Objekte, z.B. einen S7-GRAPH Funktionsbaustein, direkt aus dem SIMATIC Manager heraus öffnen.

Mehr Informationen über **Hilfe > Hilfethemen** "Ausarbeiten des Automatisierungskonzept“ und "Grundlagen zum Entwerfen der Programmstruktur“.

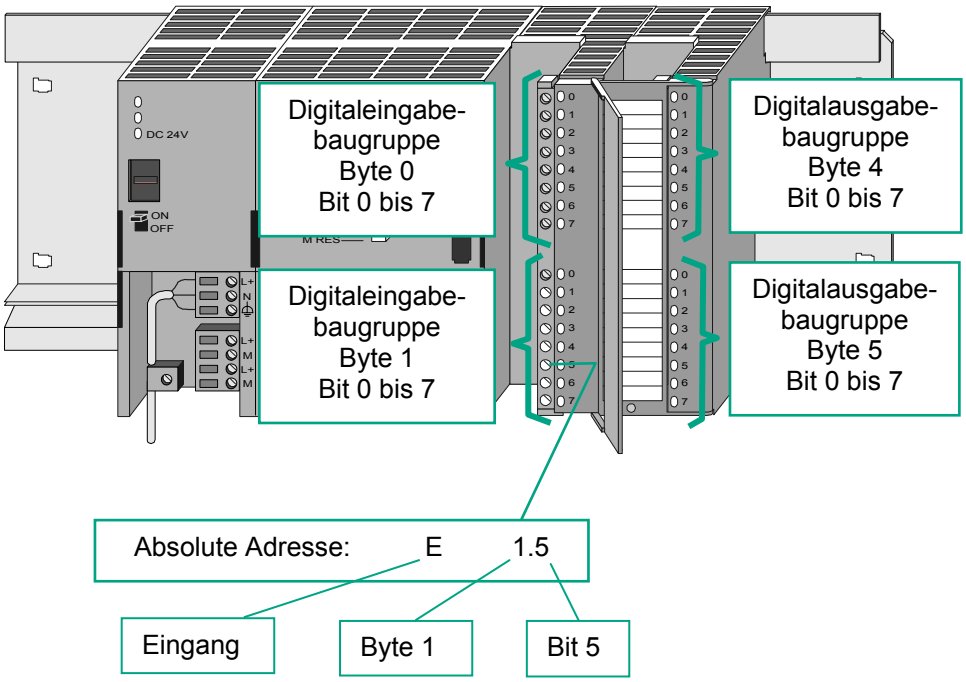
Mehr Informationen zu Optionspaketen im SIMATIC Katalog "Komponenten für die Vollintegrierte Automation“ ST 70.

3 Symbolische Programmierung

3.1 Absolute Adresse

Jeder Ein- und Ausgang hat durch den Hardwareaufbau eine vorgegebene absolute Adresse. Diese wird direkt, d. h. absolut angegeben.

Die absolute Adresse kann durch frei wählbare symbolische Namen ersetzt werden.



Die absolute Programmierung sollten Sie nur dann nutzen, wenn Sie in Ihrem S7-Programm nur wenige Ein- und Ausgänge ansprechen müssen.

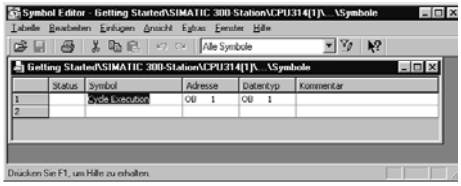
3.2 Symbolisch programmieren

In der Symboltabelle weisen Sie allen absoluten Adressen, die Sie in Ihrem späteren Programm ansprechen, einen symbolischen Namen und den Datentyp zu, z.B. für den Eingang E0.1 das Symbol Taster 1. Diese Namen gelten für alle Programmteile und werden als globale Variablen bezeichnet.

Mit Hilfe der symbolischen Programmierung können Sie somit die Lesbarkeit Ihres erstellten S7-Programms deutlich verbessern.

Mit dem Symboleditor arbeiten

Navigieren Sie im Projektfenster "Getting Started" bis **S7-Programme (1)** und öffnen Sie **Symbole** mit einem Doppelklick.



Ihre Symboltabelle besteht momentan nur aus dem vordefinierten Organisationsbaustein OB1.

	Status	Symbol	Adresse	Datentyp
1		Cycle Execution	OB 1	OB 1
2				

Klicken Sie auf **Cycle Execution** und überschreiben Sie es für unser Beispiel mit "Hauptprogramm".

	Status	Symbol	Adresse	Datentyp
1		Hauptprogramm	OB 1	OB 1
2		Lampe Grün	A 4.0	BOOL

Tragen Sie in der Zeile 2 "Lampe Grün" und "A 4.0" ein. Der Datentyp wird automatisch hinzugefügt.

Klicken Sie in die Kommentarspalte der Zeile 1 oder 2, um einen Kommentar zum jeweiligen Symbol einzutragen. Einträge einer Zeile werden mit **Return** abgeschlossen, um eine neue Zeile einzufügen.

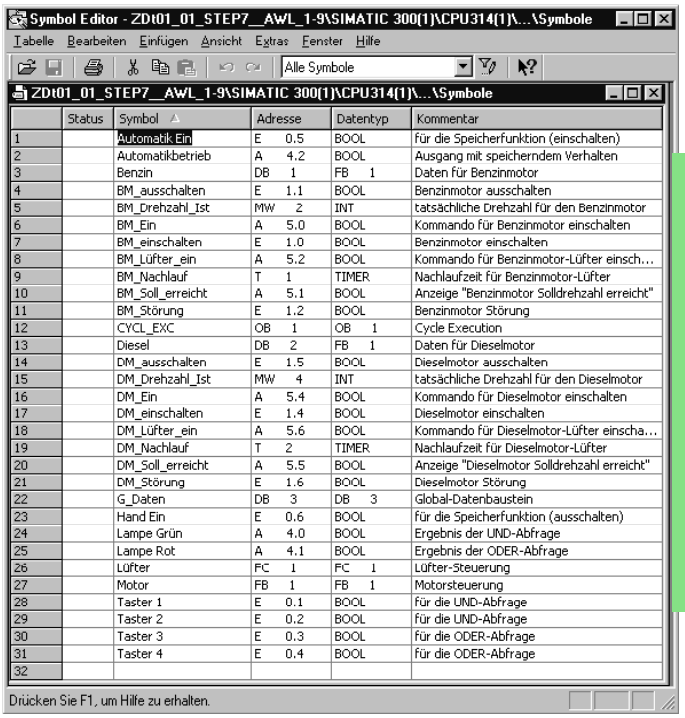
	Status	Symbol	Adresse	Datentyp
1		Hauptprogramm	OB 1	OB 1
2		Lampe Grün	A 4.0	BOOL
3		Lampe Rot	A 4.1	BOOL

Tragen Sie in der Zeile 3 "Lampe Rot" und "A 4.1" ein und schließen Sie die Eingabe mit **Return** ab.

Auf diesem Weg weisen Sie allen Ein- und Ausgängen, die Ihr Programm benötigt, einen symbolischen Namen zu.

Speichern Sie die Eintragungen oder Änderungen in der Symboltabelle, und schließen Sie das Fenster.

Da es für das gesamte Projekt "Getting Started" recht viele Namen sind, können Sie im Kapitel 4.1 die Symboltabelle in Ihr Projekt "Getting Started" kopieren.



	Status	Symbol	Adresse	Datentyp	Kommentar
1		Automatik Ein	E 0.5	BOOL	für die Speicherfunktion (einschalten)
2		Automatikbetrieb	A 4.2	BOOL	Ausgang mit speicherndem Verhalten
3		Benzin	DB 1	FB 1	Daten für Benzinmotor
4		BM_ausschalten	E 1.1	BOOL	Benzinmotor ausschalten
5		BM_Drehzahl_Ist	MW 2	INT	tatsächliche Drehzahl für den Benzinmotor
6		BM_Ein	A 5.0	BOOL	Kommando für Benzinmotor einschalten
7		BM_einschalten	E 1.0	BOOL	Benzinmotor einschalten
8		BM_Lüfter_ein	A 5.2	BOOL	Kommando für Benzinmotor-Lüfter einsch...
9		BM_Nachlauf	T 1	TIMER	Nachlaufzeit für Benzinmotor-Lüfter
10		BM_Soll_erreicht	A 5.1	BOOL	Anzeige "Benzinmotor Solldrehzahl erreicht"
11		BM_Störung	E 1.2	BOOL	Benzinmotor Störung
12		CYCL_EXC	OB 1	OB 1	Cycle Execution
13		Diesel	DB 2	FB 1	Daten für Dieselmotor
14		DM_ausschalten	E 1.5	BOOL	Dieselmotor ausschalten
15		DM_Drehzahl_Ist	MW 4	INT	tatsächliche Drehzahl für den Dieselmotor
16		DM_Ein	A 5.4	BOOL	Kommando für Dieselmotor einschalten
17		DM_einschalten	E 1.4	BOOL	Dieselmotor einschalten
18		DM_Lüfter_ein	A 5.6	BOOL	Kommando für Dieselmotor-Lüfter einscha...
19		DM_Nachlauf	T 2	TIMER	Nachlaufzeit für Dieselmotor-Lüfter
20		DM_Soll_erreicht	A 5.5	BOOL	Anzeige "Dieselmotor Solldrehzahl erreicht"
21		DM_Störung	E 1.6	BOOL	Dieselmotor Störung
22		G_Daten	DB 3	DB 3	Global-Datenbaustein
23		Hand Ein	E 0.6	BOOL	für die Speicherfunktion (ausschalten)
24		Lampe Grün	A 4.0	BOOL	Ergebnis der UND-Abfrage
25		Lampe Rot	A 4.1	BOOL	Ergebnis der ODER-Abfrage
26		Lüfter	FC 1	FC 1	Lüfter-Steuerung
27		Motor	FB 1	FB 1	Motorsteuerung
28		Taster 1	E 0.1	BOOL	für die UND-Abfrage
29		Taster 2	E 0.2	BOOL	für die UND-Abfrage
30		Taster 3	E 0.3	BOOL	für die ODER-Abfrage
31		Taster 4	E 0.4	BOOL	für die ODER-Abfrage
32					

Hier ist stellvertretend die Symboltabelle für das S7-Programm zum "Getting Started" Beispiel für AWL abgebildet.

Generell wird pro S7-Programm eine Symboltabelle angelegt, unabhängig davon, welche Programmiersprache Sie gewählt haben.

In der Symboltabelle sind alle druckbaren Zeichen (z. B. Umlaute, Leerzeichen) erlaubt.

Der Datentyp, der vorher automatisch in der Symboltabelle eingefügt wurde, legt die Art des zu verarbeitenden Signals für die CPU fest. STEP 7 verwendet u. a. folgende Datentypen:

BOOL BYTE WORD DWORD	Daten diesen Typs sind Bitkombinationen. Bit (Typ BOOL) bis 32 Bit (DWORD).
CHAR	Daten dieses Typs belegen genau 1 Zeichen des ASCII-Zeichensatzes.
INT DINT REAL	Sie stehen für die Verarbeitung numerischer Werte zur Verfügung (z. B. zum Berechnen von arithmetischen Ausdrücken).
S5TIME TIME DATE TIME_OF_DAY	Daten dieses Typs repräsentieren die unterschiedlichen Zeit- und Datumswerte innerhalb von STEP 7 (z. B. zum Einstellen des Datums oder zum Eingeben des Zeitwerts).

Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen" und "Festlegen von Symbolen".

4 Erstellen eines Programms im OB1

4.1 KOP/AWL/FUP-Programmfenster und OB1 öffnen

Entscheiden Sie sich für KOP, AWL oder FUP

Mit STEP 7 erstellen Sie S7-Programme in den Standardsprachen KOP, AWL oder FUP. In der Praxis und auch für dieses Kapitel müssen Sie sich für eine Sprache entscheiden.

KOP (Kontaktplan)
eignet sich z. B. für Anwender aus dem gewerblichen Elektrobereich.

AWL (Anweisungsliste)
eignet sich z. B. für Anwender aus dem Umfeld der Informatik.

```
U      "Taster 1"
U      "Taster 2"
=      "Lampe Grün"
```

FUP (Funktionsplan)
eignet sich z. B. für Anwender aus dem Umfeld der Schaltungstechnik.

Abhängig davon, in welcher Sprache Sie den Baustein OB1 mit dem Projekt-Assistenten angelegt haben, wird der Baustein nun geöffnet. Sie können jedoch die voreingestellte Programmiersprache zu jedem späteren Zeitpunkt wieder ändern.

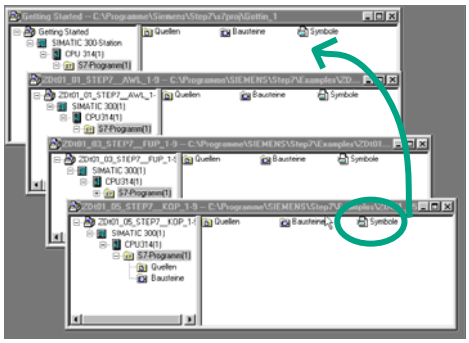
Symboltabelle kopieren und OB1 öffnen

Falls notwendig öffnen Sie Ihr "Getting Started". Klicken Sie hierzu auf das Symbol **Öffnen**, wählen Sie Ihr angelegtes "Getting Started" aus, und bestätigen Sie mit **OK**.

Je nachdem für welche Programmiersprache Sie sich entschieden haben, öffnen Sie bitte im Register "Beispielprojekte" zusätzlich das Projekt:

- ZDt01_05_STEP7__**KOP**_1-9,
- ZDt01_01_STEP7__**AWL**_1-9 oder
- ZDt01_03_STEP7__**FUP**_1-9

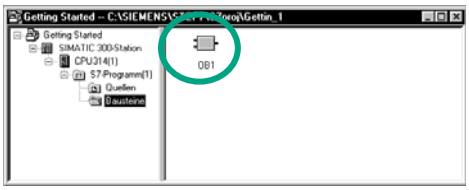
Hier sind exemplarisch alle drei Beispielprojekte dargestellt.



Navigieren Sie im "ZDt01_XXX" zu **Symbole** und kopieren Sie diese per Drag and Drop in Ihr Projektfenster "Getting Started", Ordner **S7-Programm**.

Schließen Sie daraufhin das Fenster "ZDt01_XXX".

Drag and Drop bedeutet, dass Sie ein beliebiges Objekt mit der Maus anklicken und es mit gedrückter Maustaste verschieben. Indem Sie die Maustaste loslassen, wird das Objekt abgelegt.



Doppelklicken Sie im Projekt "Getting Started" auf **OB1**. Das KOP/AWL/FUP-Programmfenster wird geöffnet.

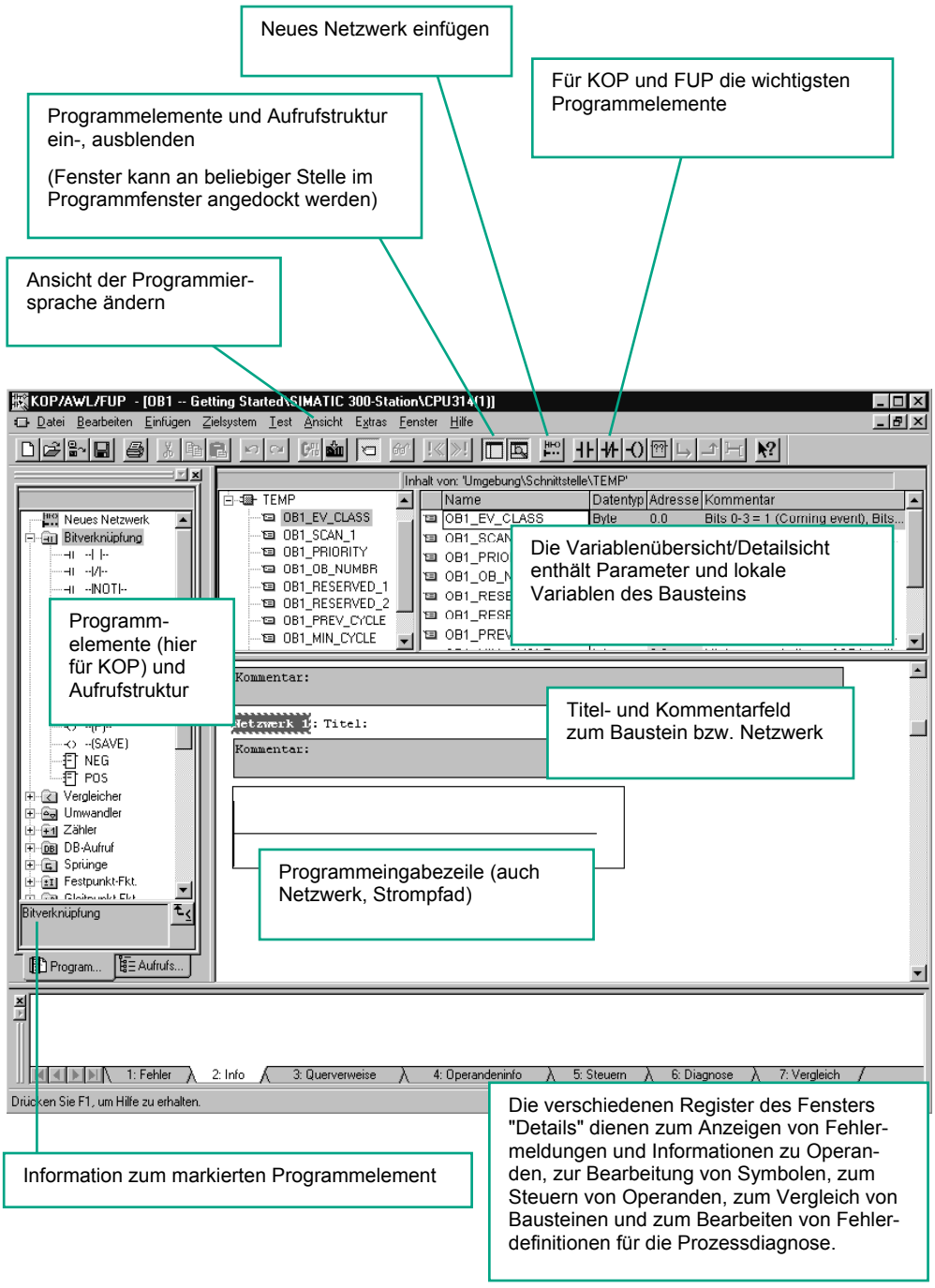
In STEP 7 wird der OB1 zyklisch von der CPU abgearbeitet. Dabei liest die CPU Zeile für Zeile und führt die Programmbefehle aus. Beginnt die CPU wieder in der ersten Programmzeile, hat sie genau einen Zyklus durchlaufen. Die hierfür benötigte Zeit wird als Zykluszeit bezeichnet.

Entsprechend Ihrer gewählten Programmiersprache lesen Sie bitte zum Programmieren mit KOP im Kapitel 4.2, mit AWL im Kapitel 4.3 und mit FUP im Kapitel 4.4 weiter.

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen“ und "Anlegen von Bausteinen und Bibliotheken“.

Das KOP/AWL/FUP-Programmfenster

Im KOP/AWL/FUP-Programmfenster werden alle Bausteine programmiert. Stellvertretend für die Programmiersprachen haben wir hier die Ansicht für KOP abgebildet.

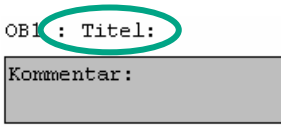


4.2 Programmieren des OB1 in KOP

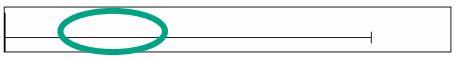
Im folgenden werden Sie eine Reihenschaltung, eine Parallelschaltung und die Speicherfunktion Setzen und Rücksetzen in KOP (Kontaktplan) programmieren.

Reihenschaltung in KOP programmieren

Falls notwendig stellen Sie über das Menü **Ansicht** die Programmiersprache **KOP** ein.



Klicken Sie in den Bereich **Titel** des OB1 und tragen Sie beispielsweise "Zyklisch bearbeitetes Hauptprogramm" ein.



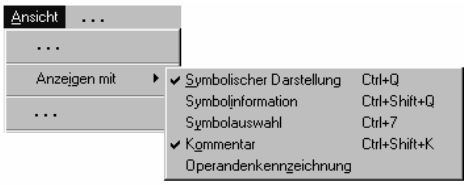
Markieren Sie den Strompfad für Ihr erstes Element.

Klicken Sie auf das Symbol in der Funktionsleiste und fügen Sie einen Schließer ein.

Fügen Sie analog einen zweiten Schließer ein.

Fügen Sie am rechten Ende des Strompfads eine Spule ein.

In der Reihenschaltung fehlt noch die Adressierung von Schließern und Spule.

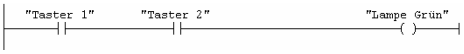


Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.

Klicken Sie auf **??.?** und tragen Sie den symbolischen Namen "Taster 1" ein (mit Anführungszeichen!). Sie können alternativ auch den Namen in der eingeblendeten Klappliste anwählen. Bestätigen Sie mit **Return**.

Tragen Sie für den zweiten Schließer den symbolischen Namen "Taster 2" ein.

Tragen Sie für die Spule den Namen "Lampe Grün" ein.

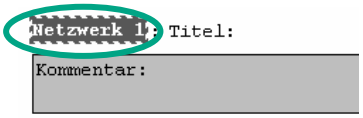


Ihre Reihenschaltung ist vollständig programmiert.

Speichern Sie den Baustein, falls keine Symbole mehr rot gekennzeichnet sind.

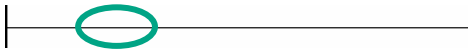
Symbole werden rot gekennzeichnet, wenn z. B. das Symbol nicht in der Symboltabelle enthalten ist oder ein Syntaxfehler vorliegt.

Parallelschaltung in KOP programmieren



Markieren Sie das **Netzwerk 1**.

Fügen Sie ein neues Netzwerk ein.

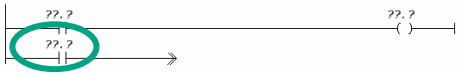


Markieren Sie wieder den Strompfad.

Fügen Sie einen Schließer und eine Spule ein.

Markieren Sie den senkrechten Strang des Strompfads.

Fügen Sie einen parallelen Strang ein.



In den parallelen Strang kommt ein weiterer Schließer.

Schließen Sie die Verzweigung (falls nötig untere Pfeilspitze markieren).

In der Parallelschaltung fehlt nur noch die Adressierung.

Für die symbolische Adressierung gehen Sie analog zur Reihenschaltung vor.

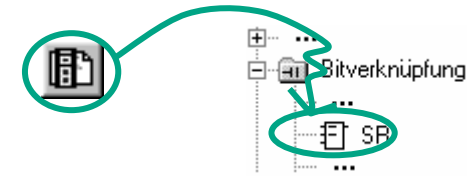
Überschreiben Sie den oberen Schließer mit "Taster 3", den unteren Schließer mit "Taster 4" und die Spule mit "Lampe Rot".

Speichern Sie den Baustein.

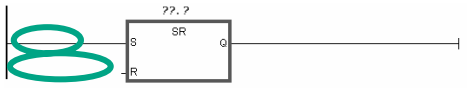
Speicherfunktion in KOP programmieren

Markieren Sie das Netzwerk 2, und fügen Sie ein weiteres Netzwerk ein.

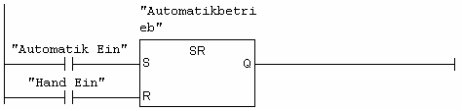
Markieren Sie wieder den Strompfad.



Navigieren Sie im Register "Programmelemente" über **Bitverknüpfungen** zum **SR**-Element. Mit Doppelklick wird es eingefügt.



Fügen Sie vor den Eingängen S und R je einen Schließer ein.



Tragen Sie für das SR-Element folgende symbolische Namen ein: oberer Schließer "Automatik Ein", unterer Schließer "Hand Ein", SR-Element "Automatikbetrieb".

Speichern Sie den Baustein, und schließen Sie das Fenster.

Wenn Sie den Unterschied zwischen absoluter und symbolischer Adressierung sehen möchten, deaktivieren Sie im Menü **Ansicht > Anzeigen mit > Symbolischer Darstellung**.

Beispiel:
Symbolische Adressierung in KOP

Beispiel:
Absolute Adressierung in KOP

Den Zeilenumbruch der symbolischen Adressierung ändern Sie im KOP/AWL/FUP-Programmfenster mit **Extras > Einstellungen > KOP/FUP > Operandenfeldbreite**. Sie können dort den Zeilenumbruch zwischen dem 10. und 26. Zeichen einstellen.

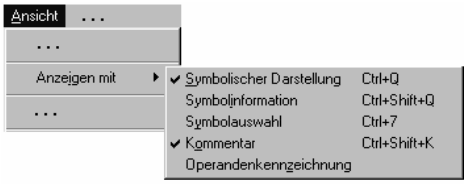
Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen“, "Erstellen von Codebausteinen“ und "Editieren von KOP-Anweisungen“.

4.3 Programmieren des OB1 in AWL

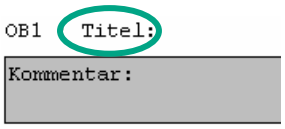
Im folgenden werden Sie eine UND-Anweisung, eine ODER-Anweisung und die Speicheranweisungen Setzen bzw. Rücksetzen in AWL (Anweisungsliste) programmieren.

UND-Anweisung in AWL programmieren

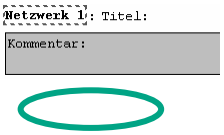
Falls notwendig stellen Sie über das Menü **Ansicht** die Programmiersprache **AWL** ein.



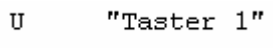
Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.



Klicken Sie in den Bereich **Titel** des OB1 und tragen Sie beispielsweise "Zyklisch bearbeitetes Hauptprogramm" ein.



Markieren Sie den Bereich für Ihre erste Anweisung.



Schreiben Sie in die erste Programmzeile ein U (UND) mit Leerzeichen und das Symbol "Taster 1" (mit Anführungszeichen).

Schließen Sie die Zeile mit **Return** ab. Der Cursor springt in die neue Zeile.

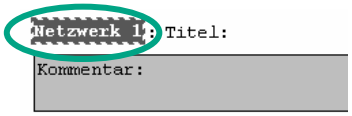
```
U     "Taster 1"
U     "Taster 2"
=     "Lampe Grün"
```

Vervollständigen Sie analog die UND-Anweisung.

Ihr UND ist vollständig programmiert. Speichern Sie den Baustein, falls keine Symbole rot gekennzeichnet sind.

Symbole werden rot gekennzeichnet, wenn z. B. das Symbol nicht in der Symboltabelle enthalten ist oder ein Syntaxfehler vorliegt.

ODER-Anweisung in AWL programmieren



Markieren Sie das **Netzwerk 1**.

Fügen Sie ein neues Netzwerk ein und markieren Sie wieder den Eingabebereich.

```
O     "Taster 3"
```

Tragen Sie ein O (ODER) und das Symbol "Taster 3" (analog zum UND) ein.

```
O     "Taster 3"
O     "Taster 4"
=     "Lampe Rot"
```

Vervollständigen Sie die ODER-Anweisung und speichern Sie diese.

Speicheranweisung in AWL programmieren

Markieren Sie das Netzwerk 2 und fügen Sie ein weiteres Netzwerk ein.

```
U     "Automatik Ein"
```

In die erste Zeile schreiben Sie die Anweisung U mit dem symbolischen Namen "Automatik Ein".

```
U     "Automatik Ein"
S     "Automatikbetrieb"
U     "Hand Ein"
R     "Automatikbetrieb"
```

Vervollständigen Sie die Speicheranweisung und speichern Sie diese. Schließen Sie den Baustein.

Wenn Sie den Unterschied zwischen absoluter und symbolischer Adressierung sehen möchten, deaktivieren Sie im Menü **Ansicht > Anzeigen mit > Symbolischer Darstellung**.

```
U     "Taster 1"
U     "Taster 2"
=     "Lampe Grün"
```

Beispiel:
Symbolische Adressierung in AWL

```
U     E     0.1
U     E     0.2
=     A     4.0
```

Beispiel:
Absolute Adressierung in AWL

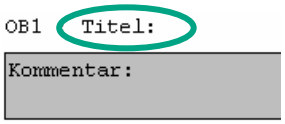
Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen", "Erstellen von Codebausteinen" und "Editieren von AWL-Anweisungen".

4.4 Programmieren des OB1 mit FUP

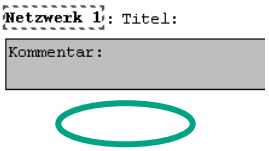
Im folgenden werden Sie eine UND-Funktion, eine ODER-Funktion und eine Speicherfunktion in FUP (Funktionsplan) programmieren.

Eine UND-Funktion in FUP programmieren

Falls notwendig stellen Sie über das Menü **Ansicht** die Programmiersprache **FUP** ein.

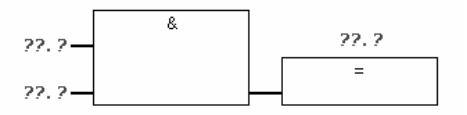


Klicken Sie in den Bereich **Titel** des OB1 und tragen Sie beispielsweise "Zyklisch bearbeitetes Hauptprogramm" ein.

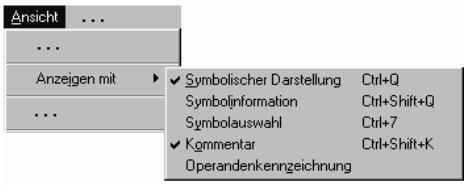


Markieren Sie den Eingabebereich für die UND-Funktion (unterhalb des Kommentarfeldes).

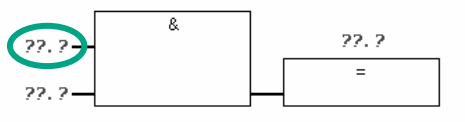
Fügen Sie eine UND-Box (**&**) und eine Zuweisung (**=**) ein.



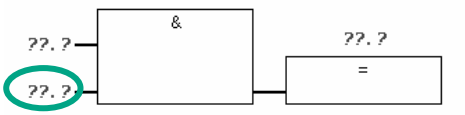
In der UND-Funktion fehlt noch die Adressierung der Elemente.



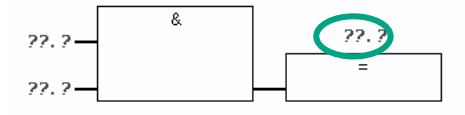
Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.



Klicken Sie auf **??.?** und tragen Sie den symbolischen Namen "Taster 1" ein (mit Anführungszeichen!). Sie können alternativ auch den Namen in der eingeblendeten Klappliste anwählen.
Bestätigen Sie mit **Return**.



Tragen Sie für den zweiten Eingang den symbolischen Namen "Taster 2" ein.



Tragen Sie für die Zuweisung den Namen "Lampe Grün" ein.

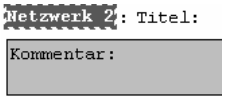
Ihre UND-Funktion ist vollständig programmiert.

Wenn keine Symbole mehr rot gekennzeichnet sind, können Sie speichern.

Symbole werden rot gekennzeichnet, wenn z. B. das Symbol nicht in der Symboltabelle enthalten ist oder ein Syntaxfehler vorliegt.

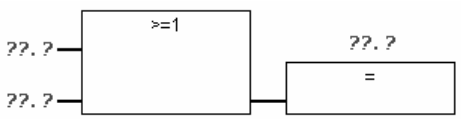
Eine ODER-Funktion in FUP programmieren

Fügen Sie ein neues Netzwerk ein.

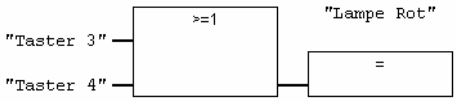


Markieren Sie wieder den Eingabebereich für die ODER-Funktion.


Fügen Sie eine ODER-Box (≥**1**) und eine Zuweisung (**=**) ein.



In der ODER-Funktion fehlt noch die Adressierung. Gehen Sie analog zur UND-Funktion vor.

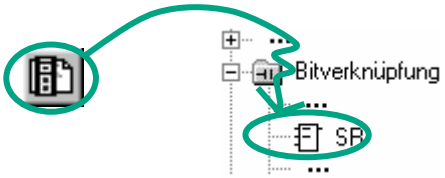


Tragen Sie
für den oberen Eingang "Taster 3",
für den unteren Eingang "Taster 4" und
für die Zuweisung "Lampe Rot" ein.

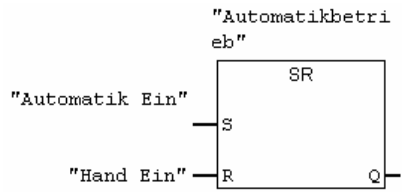
Speichern Sie den Baustein.

Eine Speicherfunktion in FUP programmieren

Markieren Sie das Netzwerk 2 und fügen Sie ein weiteres Netzwerk ein. Markieren Sie wieder den Eingabebereich (unterhalb des Kommentarfeldes).



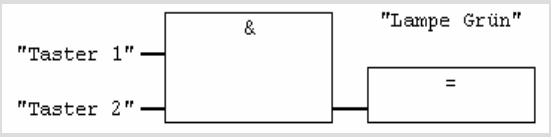
Navigieren Sie im Register "Programmelemente" über **Bitverknüpfungen** zum **SR-**Element. Mit Doppelklick wird es eingefügt.



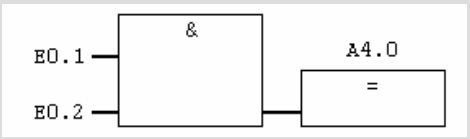
Tragen Sie für das SR-Element folgende symbolische Namen ein: Setzen "Automatik Ein", Rücksetzen "Hand Ein", Merker "Automatikbetrieb".

Speichern Sie den Baustein, und schließen Sie das Fenster.

Wenn Sie den Unterschied zwischen absoluter und symbolischer Adressierung sehen möchten, deaktivieren Sie im Menü **Ansicht > Anzeigen mit > Symbolischer Darstellung**.



Beispiel:
Symbolische Adressierung in FUP



Beispiel:
Absolute Adressierung in FUP

Den Zeilenumbruch der symbolischen Adressierung ändern Sie im KOP/AWL/FUP-Programmfenster mit **Extras > Einstellungen > KOP/FUP > Operandenfeldbreite**. Sie können dort den Zeilenumbruch zwischen dem 10. und 26. Zeichen einstellen.

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen“, "Erstellen von Codebausteinen“ und "Editieren von FUP-Anweisungen“.

5 Erstellen eines Programms mit FBs und DBs

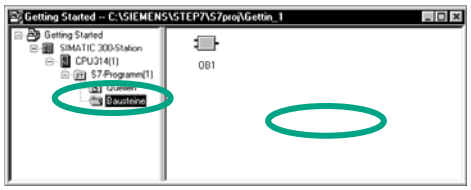
5.1 Funktionsbaustein anlegen und öffnen

Der Funktionsbaustein (FB) ist dem Organisationsbaustein untergeordnet. Er beinhaltet einen Programmteil, der innerhalb des OB1 beliebig oft aufgerufen werden kann. Alle Formalparameter und statischen Daten des Funktionsbausteins werden dabei in einem separaten Datenbaustein DB gespeichert, der dem Funktionsbaustein zugeordnet ist.

Sie programmieren den Funktionsbaustein (FB1, symbolischer Name "Motor“, vgl. Symboltabelle Seite 3-3) in dem bereits bekannten KOP/AWL/FUP-Programmfenster. Hierzu sollten Sie die gleiche Programmiersprache wie im Kapitel 4 (Programmieren des OB1) benutzen.

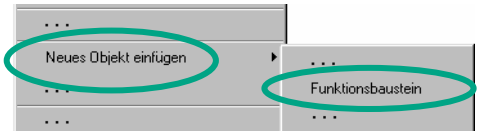
Sie sollten die Symboltabelle in Ihr Projekt "Getting Started" kopiert haben. Falls nicht, lesen Sie bitte hierzu auf Seite 4-2, Symboltabelle kopieren, und kehren Sie anschließend hierher zurück.

Falls notwendig öffnen Sie das Projekt "Getting Started".

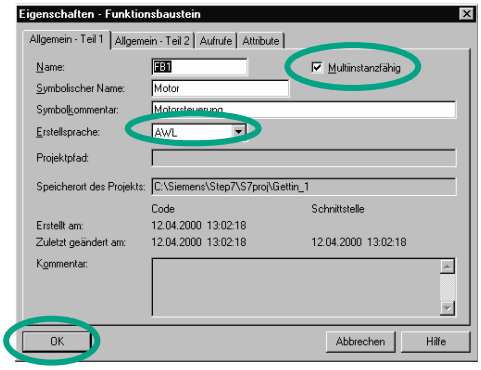


Navigieren Sie zum Ordner **Bausteine** und öffnen Sie ihn.

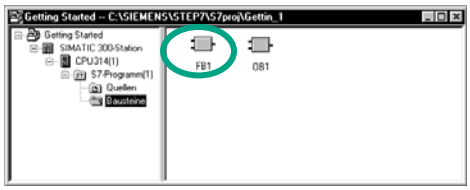
Klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte.



Das Kontext-Menü der rechten Maustaste enthält die wichtigsten Befehle aus der Menüleiste. Fügen Sie als neues Objekt einen **Funktions-baustein** ein.



Im Dialogfeld "Eigenschaften - Funktionsbaustein" wählen Sie die Erstellsprache, aktivieren Sie die **Multiinstanzfähigkeit**, und übernehmen Sie die restlichen Voreinstellungen mit **OK**.



Der Funktionsbaustein **FB1** wurde in das Verzeichnis Bausteine eingefügt.

Mit Doppelklick auf den FB1 gelangen Sie zum KOP/AWL/FUP-Programmfenster.

Je nachdem, welche Programmiersprache Sie gewählt haben, lesen Sie bitte im Kapitel 5.2 für KOP, im Kapitel 5.3 für AWL und im Kapitel 5.4 für FUP weiter.

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen" und "Anlegen von Bausteinen und Bibliotheken".

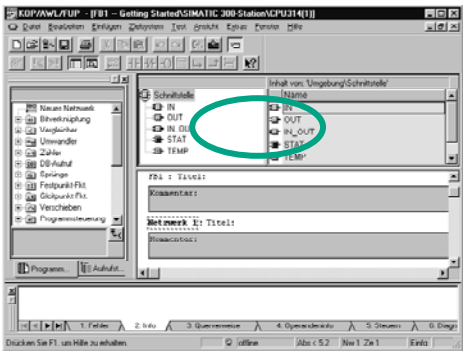
5.2 Programmieren des FB1 in KOP

Wir zeigen Ihnen, wie Sie einen Funktionsbaustein programmieren, der beispielsweise einen Benzin- und Dieselmotor mit je einem Datenbaustein steuert und überwacht.

Alle "motorspezifischen" Signale werden dabei als Bausteinparameter vom Organisationsbaustein an den Funktionsbaustein übergeben und müssen deshalb in der Variablendeklarationstabelle als Ein- und Ausgangsparameter (Deklaration "in" und "out") aufgeführt sein.

Sie sollten bereits wissen, wie Sie eine Reihenschaltung, eine Parallelschaltung und eine Speicherfunktion mit STEP 7 eingeben.

Zuerst Variablen deklarieren / festlegen



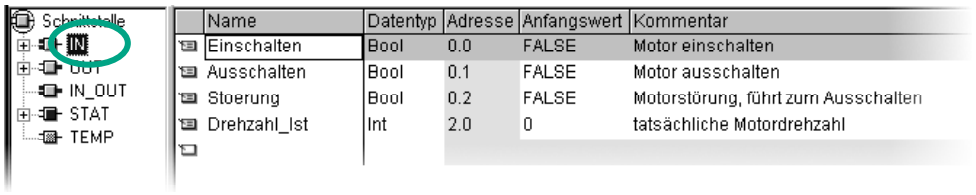
Ihr KOP/AWL/FUP-Programmfenster ist geöffnet und **Ansicht > KOP** (Programmiersprache) ist aktiviert.

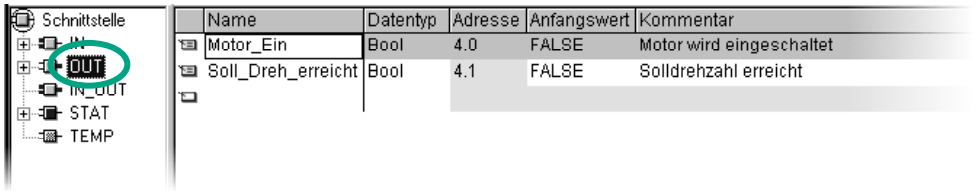
Achten Sie auf die Kopfzeile, dort steht nun FB1, da Sie das Programmfenster mit einem Doppelklick auf den FB1 geöffnet haben.

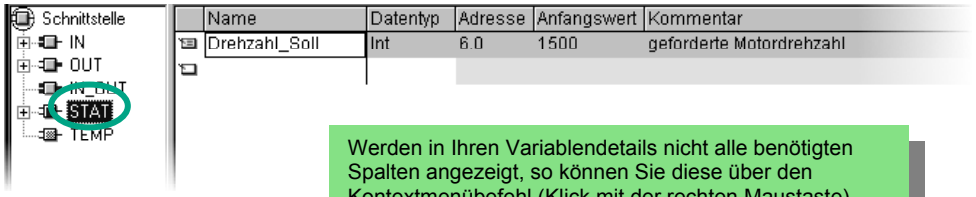
Der Variablendeklarationsbereich besteht aus einer Variablenübersicht (linkes Teilfenster) und der Variablendetailsicht (rechtes Teilfenster).

Markieren Sie in der Variablenübersicht nacheinander die Deklarationstypen "IN", "OUT" und "STAT" und geben Sie in den entsprechenden Variablendetails folgende Deklarationen ein.

Klicken Sie in der Variablendetailsicht in die entsprechenden Zellen, und übernehmen Sie die Eintragungen aus den nachfolgenden Abbildungen. Den Datentyp können Sie aus der eingeblendeten Klappliste auswählen.



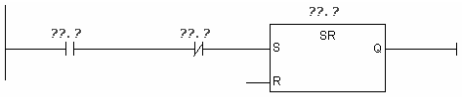




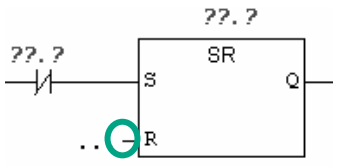
Werden in Ihren Variablendetails nicht alle benötigten Spalten angezeigt, so können Sie diese über den Kontextmenübefehl (Klick mit der rechten Maustaste) einblenden.

Für die Namen der Bausteinparameter in der Variablendetails sind nur Buchstaben (ohne Umlaute), Ziffern und der Unterstrich zugelassen.

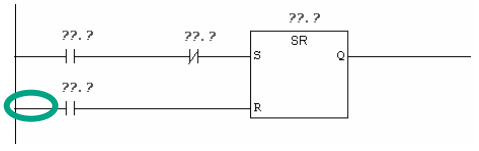
Ein- und Ausschaltvorgang eines Motors programmieren



Fügen Sie im Netzwerk 1 über die entsprechenden Symbole bzw. den Programmelemente-Katalog einen Schließer, einen Öffner und ein SR-Element in Reihe ein.

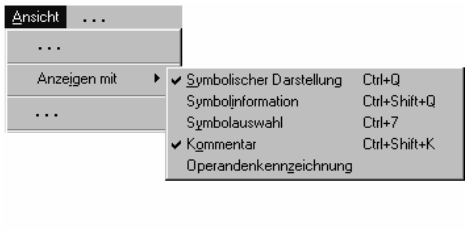


Markieren Sie anschließend den Strompfad unmittelbar vor dem Eingang R.



Fügen Sie einen weiteren Schließer ein. Markieren Sie den Strompfad unmittelbar vor dem Schließer.

Fügen Sie parallel zum Schließer einen Öffner ein.

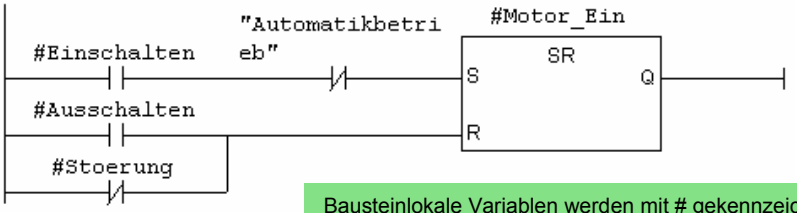


Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.

Markieren Sie die Fragezeichen und tragen Sie die entsprechenden Namen der Deklarationstabelle ein (# wird automatisch vergeben).

Tragen Sie für den Öffner der Reihenschaltung den symbolischen Namen "Automatikbetrieb" ein.

Speichern Sie anschließend Ihr Programm.

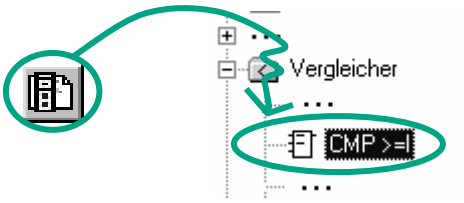


Bausteinlokale Variablen werden mit # gekennzeichnet und sind nur im Baustein gültig.

Globale Variablen stehen in Anführungszeichen. Sie werden in der Symboltabelle definiert und sind im gesamten Programm gültig.

Der Signalzustand „Automatikbetrieb“ wird im OB1 (Netzwerk 3, vgl. Seite 4-7) durch ein anderes SR-Element festgelegt und jetzt im FB1 abgefragt.

Drehzahlüberwachung programmieren



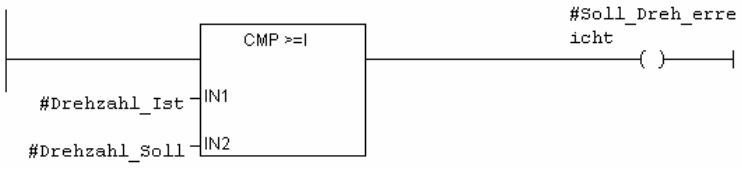
Fügen Sie ein neues Netzwerk ein und markieren Sie den Strompfad.

Navigieren Sie anschließend im Programmelemente-Katalog zum **Vergleicher**, und fügen Sie ein **CMP>=I** ein.

Fügen Sie außerdem im Strompfad eine Spule ein.

Markieren Sie wieder die Fragezeichen und beschriften Sie die Spule und den Vergleicher mit den Namen aus der Variablendeklarationstabelle.

Speichern Sie anschließend Ihr Programm.



Wann wird der Motor ein- und ausgeschaltet?

Wenn die Variable #Einschalten den Signalzustand "1" und die Variable "Automatikbetrieb" den Signalzustand "0" führen, wird der Motor eingeschaltet. Erst das Negieren (Öffnerkontakt) von "Automatikbetrieb" ermöglicht diese Funktionalität.

Wenn die Variable #Ausschalten den Signalzustand "1" oder die Variable #Stoerung den Signalzustand "0" führen, wird der Motor ausgeschaltet. Die gewünschte Funktionalität wird wieder durch das Negieren von #Stoerung erreicht (#Stoerung ist ein "nullaktives" Signal und hat im Normalfall "1", im Störfall "0").

Wie überwacht der Vergleicher die Motordrehzahl?

Über den Vergleicher werden die Variablen #Drehzahl_Ist und #Drehzahl_Soll verglichen und das Ergebnis der Variablen #Soll_Drehzahl_erreicht zugewiesen (Signalzustand 1).

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen", "Erstellen von Codebausteinen" und "Editieren der Variablendeklaration" bzw. "Editieren von KOP-Anweisungen".

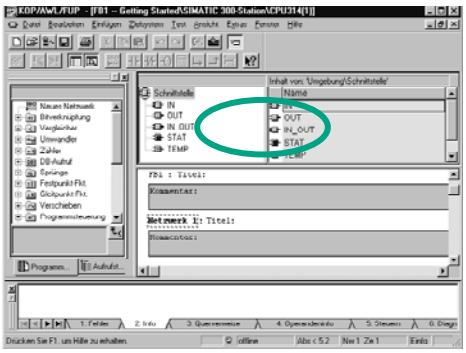
5.3 Programmieren des FB1 in AWL

Wir zeigen Ihnen, wie Sie einen Funktionsbaustein programmieren, der beispielsweise einen Benzin- und Dieselmotor mit je einem Datenbaustein steuert und überwacht.

Alle "motorspezifischen" Signale werden dabei als Bausteinparameter vom Organisationsbaustein an den Funktionsbaustein übergeben und müssen deshalb in der Variablendeklarationstabelle als Ein- und Ausgangsparameter (Deklaration "in" und "out") aufgeführt sein.

Sie sollten bereits wissen, wie Sie eine UND-Anweisung, eine ODER-Anweisung und die Speicheranweisungen Setzen und Rücksetzen mit STEP 7 eingeben.

Zuerst Variablen deklarieren / festlegen



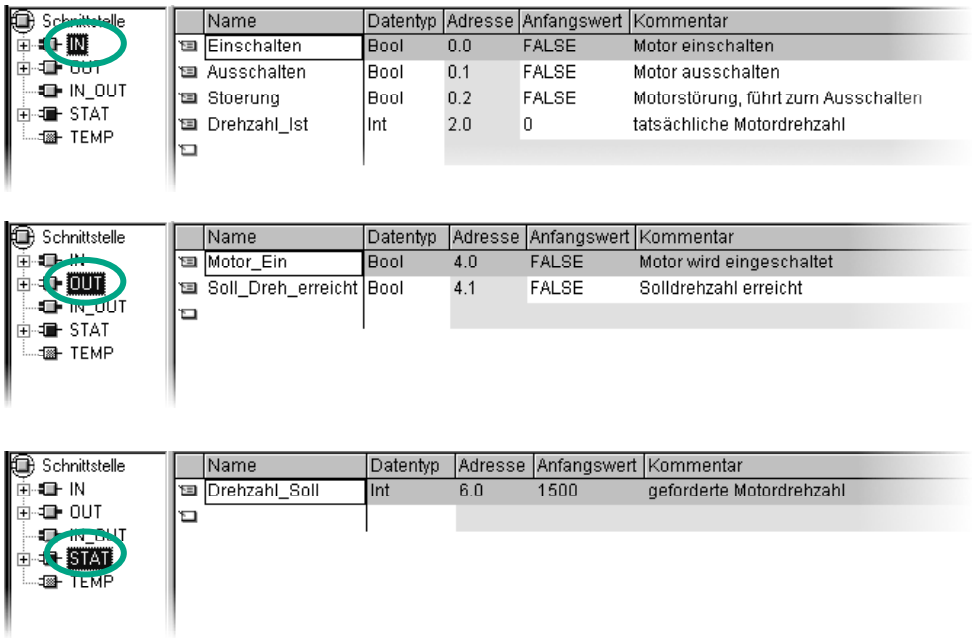
Ihr KOP/AWL/FUP-Programmfenster ist geöffnet und **Ansicht > AWL** (Programmiersprache) ist aktiviert.

Achten Sie auf die Kopfzeile, dort steht nun FB1, da Sie das Programmfenster mit einem Doppelklick auf den FB1 geöffnet haben.

Der Variablendeklarationsbereich besteht aus einer Variablenübersicht (linkes Teilfenster) und der Variablendetailsicht (rechtes Teilfenster).

Markieren Sie in der Variablenübersicht nacheinander die Deklarationstypen "IN", "OUT" und "STAT" und geben Sie in den entsprechenden Variablendetails folgende Deklarationen ein.

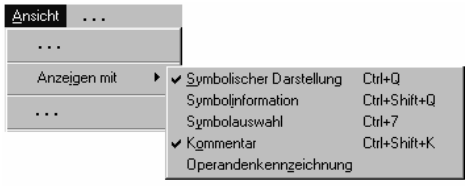
Klicken Sie in der Variablendetailsicht in die entsprechenden Zellen, und übernehmen Sie die Eintragungen aus den nachfolgenden Abbildungen. Den Datentyp können Sie aus der eingeblendeten Klappliste auswählen.



Bausteinlokale Variablen werden mit # gekennzeichnet und sind nur im Baustein gültig.

Globale Variablen stehen in Anführungszeichen. Sie werden in der Symboltabelle definiert und sind im gesamten Programm gültig.

Ein- und Ausschaltvorgang eines Motors programmieren



Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.

```
U     #Einschalten
UN    "Automatikbetrieb"
S     #Motor_Ein
O     #Ausschalten
ON    #Stoerung
R     #Motor_Ein
```

Tragen Sie im Netzwerk 1 die entsprechenden Anweisungen ein.

Bausteinlokale Variablen werden mit # gekennzeichnet und sind nur im Baustein gültig.

Globale Variablen stehen in Anführungszeichen. Sie werden in der Symboltabelle definiert und sind im gesamten Programm gültig.

Der Signalzustand "Automatikbetrieb" wird im OB1 (Netzwerk 3, vgl. Seite 4-10) durch ein anderes SR-Element festgelegt und jetzt im FB1 abgefragt.

Drehzahlüberwachung programmieren

```
L     #Drehzahl_Ist
L     #Drehzahl_Soll
>=I
=     #Soll_Dreh_erreicht
```

Fügen Sie ein neues Netzwerk ein, und tragen Sie die entsprechenden Anweisungen ein. Speichern Sie anschließend Ihr Programm.

Wann wird der Motor ein- und ausgeschaltet?

Wenn die Variable #Einschalten den Signalzustand "1" und die Variable "Automatikbetrieb" den Signalzustand "0" führen, wird der Motor eingeschaltet. Erst das Negieren (Öffnerkontakt) von "Automatikbetrieb" ermöglicht diese Funktionalität.

Wenn die Variable #Ausschalten den Signalzustand "1" oder die Variable #Stoerung den Signalzustand "0" führen, wird der Motor ausgeschaltet. Die gewünschte Funktionalität wird wieder durch das Negieren von #Stoerung erreicht (#Stoerung ist ein "nullaktives" Signal und hat im Normalfall "1", im Störfall "0").

Wie überwacht der Vergleicher die Motordrehzahl?

Über den Vergleicher werden die Variablen #Drehzahl_Ist und #Drehzahl_Soll verglichen und das Ergebnis der Variablen #Soll_Drehzahl_erreicht zugewiesen (Signalzustand 1).

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen", "Erstellen von Codebausteinen" und "Editieren der Variablendeklaration" bzw. "Editieren von AWL-Anweisungen".

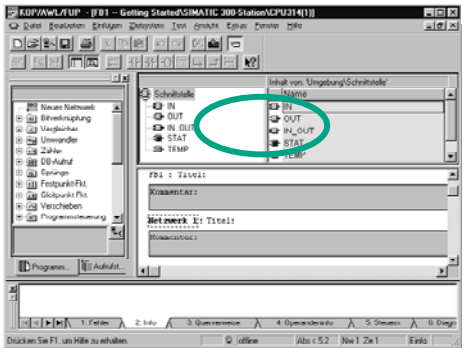
5.4 Programmieren des FB1 in FUP

Wir zeigen Ihnen, wie Sie einen Funktionsbaustein programmieren, der beispielsweise einen Benzin- und Dieselmotor mit je einem Datenbaustein steuert und überwacht.

Alle "motorspezifischen" Signale werden dabei als Bausteinparameter vom Organisationsbaustein an den Funktionsbaustein übergeben und müssen deshalb in der Variablendeklarationstabelle als Ein- und Ausgangsparameter (Deklaration "in" und "out") aufgeführt sein.

Sie sollten bereits wissen, wie Sie eine UND-Funktion, eine ODER-Funktion und eine Speicherfunktion mit STEP 7 eingeben.

Zuerst Variablen deklarieren / festlegen



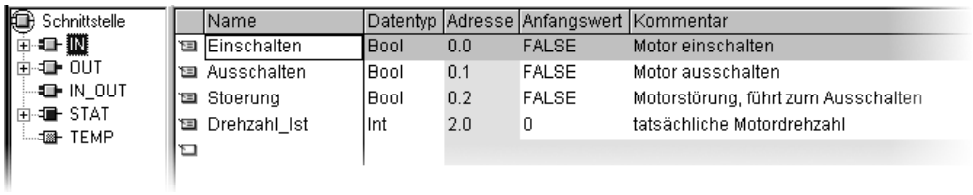
Ihr KOP/AWL/FUP-Programmfenster ist geöffnet und **Ansicht > FUP** (Programmiersprache) ist aktiviert.

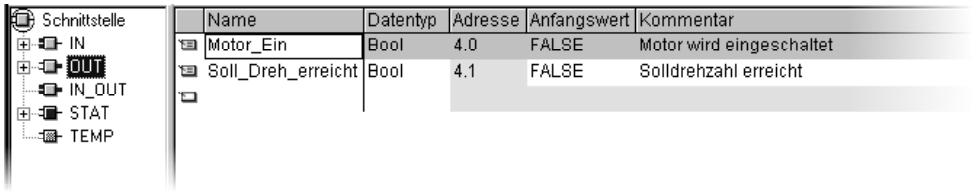
Achten Sie auf die Kopfzeile, dort steht nun FB1, da Sie das Programmfenster mit einem Doppelklick auf den FB1 geöffnet haben.

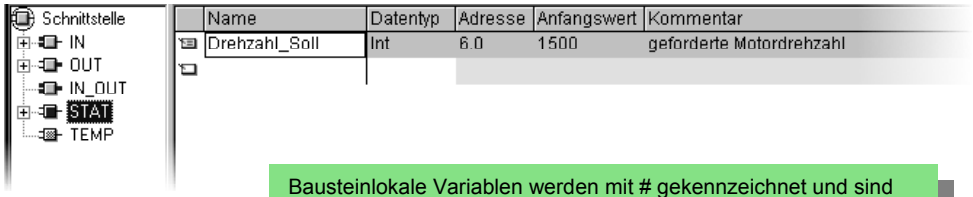
Der Variablendeklarationsbereich besteht aus einer Variablenübersicht (linkes Teilfenster) und der Variablendetailsicht (rechtes Teilfenster).

Markieren Sie in der Variablenübersicht nacheinander die Deklarationstypen "IN", "OUT" und "STAT" und geben Sie in den entsprechenden Variablendetails folgende Deklarationen ein.

Klicken Sie in der Variablendetailsicht in die entsprechenden Zellen, und übernehmen Sie die Eintragungen aus den nachfolgenden Abbildungen. Den Datentyp können Sie aus der eingeblendeten Klappliste auswählen.



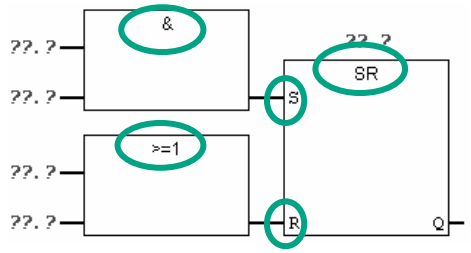




Bausteinlokale Variablen werden mit # gekennzeichnet und sind nur im Baustein gültig.

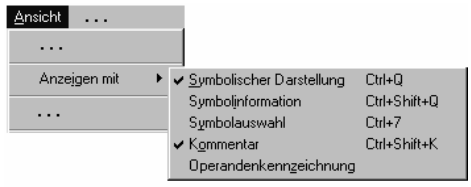
Globale Variablen stehen in Anführungszeichen. Sie werden in der Symboltabelle definiert und sind im gesamten Programm gültig.

Ein- und Ausschaltvorgang eines Motors programmieren



Fügen Sie im Netzwerk 1 über den Programmelemente-Katalog eine SR-Funktion (Verzeichnis Bitverknüpfung) ein.

Den Eingang S (Setzen) belegen Sie mit einer UND-Box. Den Eingang R (Rücksetzen) mit einer ODER-Box.



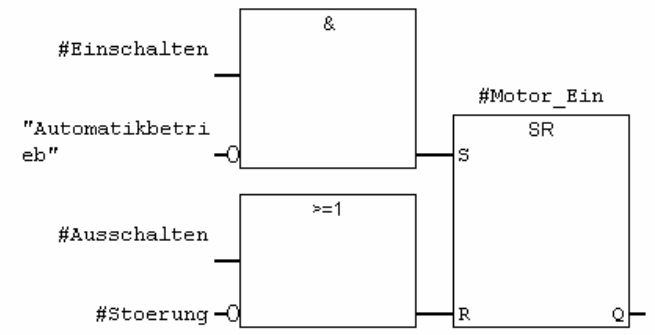
Prüfen Sie, ob die Symbolische Darstellung aktiviert ist.

Klicken Sie auf **??.?** und tragen Sie die entsprechenden Namen der Deklarationstabelle ein (# wird automatisch vergeben).

Achten Sie darauf, daß ein Eingang der UND-Funktion mit dem symbolischen Namen "Automatikbetrieb" adressiert wird.

Negieren Sie noch die Eingänge "Automatikbetrieb" und #Stoerung mit dem entsprechenden Symbol (Button aus der Funktionsleiste).

Speichern Sie anschließend Ihr Programm.

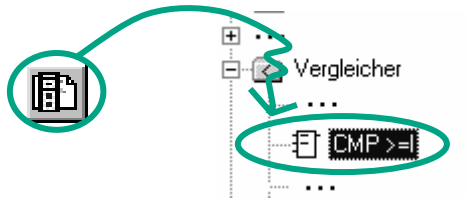


Bausteinlokale Variablen werden mit # gekennzeichnet und sind nur im Baustein gültig.

Globale Variablen stehen in Anführungszeichen. Sie werden in der Symboltabelle definiert und sind im gesamten Programm gültig.

Der Signalzustand "Automatikbetrieb" wird im OB1 (Netzwerk 3, vgl. Seite 4-14) durch eine andere SR-Funktion festgelegt und jetzt im FB1 abgefragt.

Drehzahlüberwachung programmieren

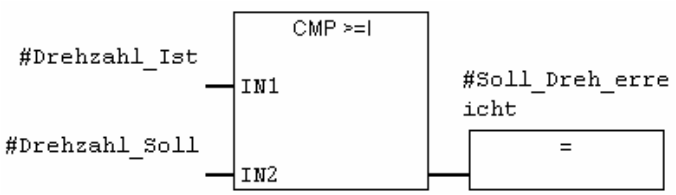


Fügen Sie ein neues Netzwerk ein und markieren Sie den Eingabebereich.

Navigieren Sie anschließend im Programmelemente-Katalog zum **Vergleicher**, und fügen Sie ein **CMP>=I** ein.

Fügen Sie eine Ausgangszuweisung an den Vergleicher an, und adressieren Sie die Eingänge mit den Namen aus der Variablendeklarationstabelle.

Speichern Sie anschließend Ihr Programm.



Wann wird der Motor ein- und ausgeschaltet?

Wenn die Variable #Einschalten den Signalzustand "1" und die Variable "Automatikbetrieb" den Signalzustand "0" führen, wird der Motor eingeschaltet. Erst das Negieren (Öffnerkontakt) von "Automatikbetrieb" ermöglicht diese Funktionalität.

Wenn die Variable #Ausschalten den Signalzustand "1" oder die Variable #Stoerung den Signalzustand "0" führen, wird der Motor ausgeschaltet. Die gewünschte Funktionalität wird wieder durch das Negieren von #Stoerung erreicht (#Stoerung ist ein "nullaktives" Signal und hat im Normalfall "1", im Störfall "0").

Wie überwacht der Vergleicher die Motordrehzahl?

Über den Vergleicher werden die Variablen #Drehzahl_Ist und #Drehzahl_Soll verglichen und das Ergebnis der Variablen #Soll_Drehzahl_erreicht zugewiesen (Signalzustand 1).

Mehr Informationen über **Hilfe > Hilfethemen** in "Programmieren von Bausteinen", "Erstellen von Codebausteinen" und "Editieren der Variablendeklaration" bzw. "Editieren von AWL-Anweisungen".

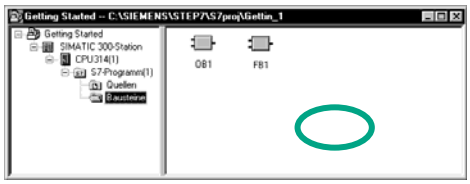
5.5 Instanz-Datenbausteine erzeugen und Aktualwerte ändern

Sie haben gerade den Funktionsbaustein FB1 ("Motor") programmiert und dabei u. a. die motorspezifischen Parameter in der Variablendetailsicht definiert.

Um später den Aufruf (CALL) des FBs im OB1 programmieren zu können, müssen Sie den zugehörigen Datenbaustein erzeugen. Einem FB ist grundsätzlich ein Instanz-Datenbaustein (DB) zugeordnet.

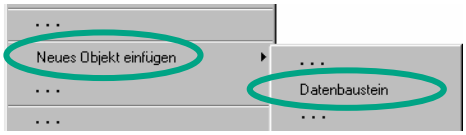
Der FB soll einen Benzin- bzw. Dieselmotor steuern und überwachen. Die unterschiedlichen Soll-Drehzahlen der Motoren werden dabei in zwei separaten DBs hinterlegt, indem jeweils der Aktualwert (#Drehzahl_ Soll) geändert wird.

Indem Sie den Funktionsbaustein nur einmal zentral programmieren, sparen Sie Programmieraufwand.

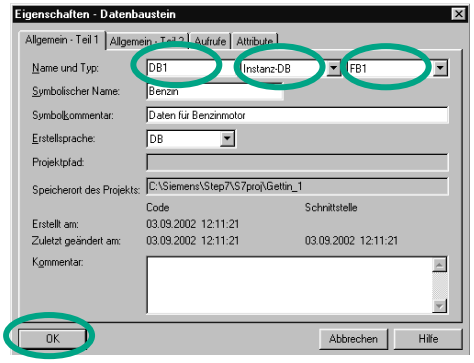


Im SIMATIC Manager ist das Projekt "Getting Started" geöffnet.

Navigieren Sie zum Ordner **Bausteine** und klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte.

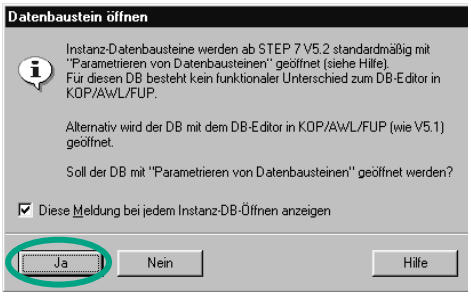


Fügen Sie mit dem Kontextmenü der rechten Maustaste einen **Datenbaustein** ein.



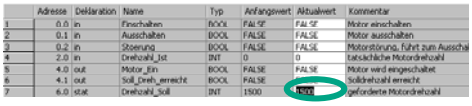
Übernehmen Sie im Dialogfeld "Eigenschaften - Datenbaustein" den Namen DB1, wählen Sie in der benachbarten Klappliste den Verwendungszweck "Instanz-DB" und übernehmen Sie den Namen des zugeordneten Funktionsbausteins "FB1". Bestätigen Sie alle Voreinstellungen mit **OK**.

Der Datenbaustein DB1 wird im Projekt "Getting Started" hinzugefügt.



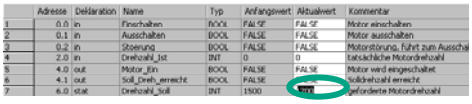
Öffnen Sie den **DB1** mit Doppelklick.

Bestätigen Sie den nachfolgenden Dialog mit **JA**, um den Instanz-Datenbaustein zu parametrieren.



Tragen Sie jetzt für den Benzinmotor in der Spalte Aktualwert den Wert "1500" ein (zur Zeile "Drehzahl_Soll"). Damit haben Sie die maximale Drehzahl für diesen Motor festgelegt.

Speichern Sie den DB1, und schließen Sie das Programmfenster.



Erzeugen Sie nun analog zu DB1 einen weiteren Datenbaustein DB2 zum FB1.

Tragen Sie jetzt für den Dieselmotor den Aktualwert von "1200" ein.

Speichern Sie den DB2, und schließen Sie das Programmfenster.

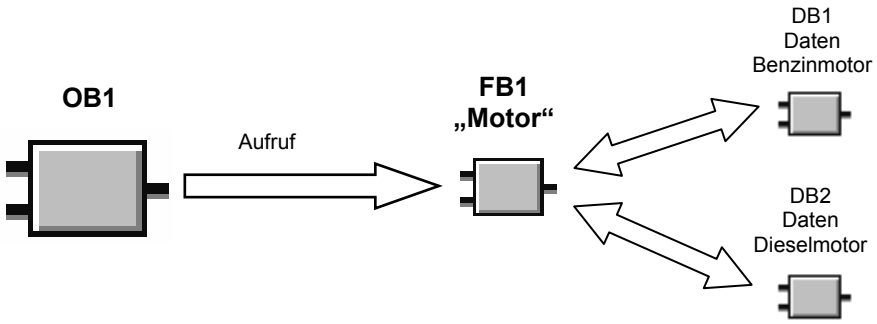
Mit der Änderung der Aktualwerte sind die Vorbereitungen abgeschlossen, mit nur einem Funktionsbaustein zwei Motoren zu steuern. Um weitere Motoren zu steuern, müßten lediglich weitere Datenbausteine erzeugt werden.

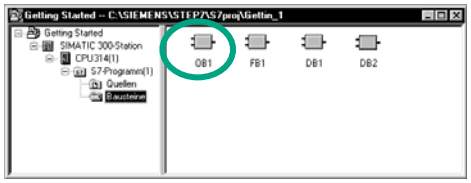
Um nun den Aufruf des FBs im OB1 zu programmieren, lesen Sie bitte entsprechend Ihrer Programmiersprache im Kapitel 5.6 für KOP, Kapitel 5.7 für AWL und Kapitel 5.8 für FUP weiter.

Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen" und "Erstellen von Datenbausteinen".

5.6 Bausteinaufruf in KOP programmieren

Die gesamte Programmierung eines FBs ist ohne Aufruf im OB1 unwirksam. Pro Aufruf des FBs wird je ein Datenbaustein benutzt und somit beide Motoren gesteuert.

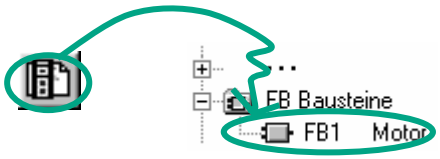




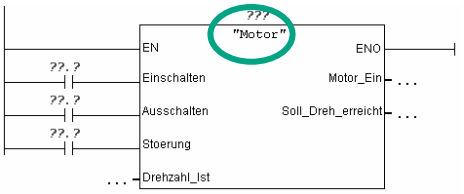
Der SIMATIC Manager ist mit Ihrem Projekt "Getting Started" geöffnet.

Navigieren Sie zum Ordner **Bausteine**, und öffnen Sie den **OB1**.

Markieren Sie im KOP/AWL/FUP-Programmfenster das Netzwerk 3 und fügen Sie das Netzwerk 4 ein.

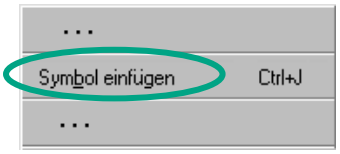


Navigieren Sie anschließend im Programmelemente-Katalog zum **FB1** und fügen ihn mit Doppelklick ein.

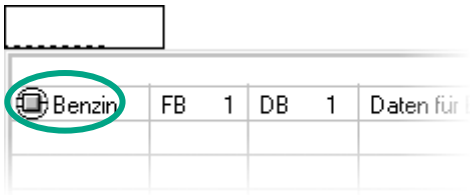


Fügen Sie vor Einschalten, Ausschalten und Störung je einen Schließer ein.

Klicken Sie auf die **???** über "Motor", gleich darauf klicken Sie mit der rechten Maustaste in den Eingaberahmen.

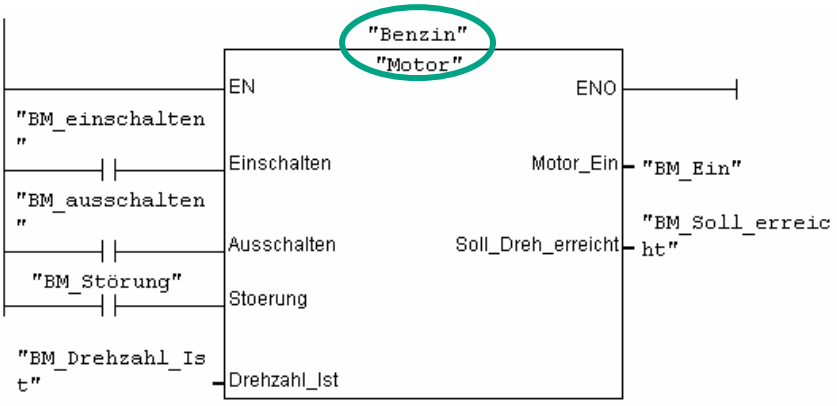


Klicken Sie im Kontextmenü der rechten Maustaste auf **Symbole einfügen**. Eine Klappliste öffnet sich.



Doppelklicken Sie auf das Symbol **Benzin**. Es wird automatisch mit Anführungszeichen in das Eingabefeld übernommen.

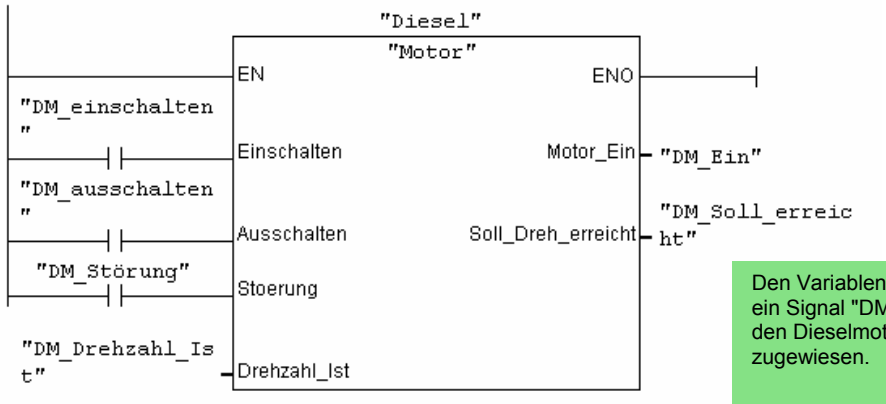
Klicken Sie auf die Fragezeichen, und adressieren Sie nach Eingabe eines Anführungszeichens mit der Klappliste alle weiteren Parameter des Funktionsbausteins mit den entsprechenden symbolischen Namen.



Die motorspezifischen Ein- und Ausgangsvariablen (Deklaration "in" und "out") werden im FB "Motor" angezeigt.

Den Variablen wird je ein Signal "BM_xxx" für den Benzinmotor zugewiesen.

Programmieren Sie in einem neuen Netzwerk den Aufruf des Funktionsbausteins "Motor" (FB1) mit dem Datenbaustein "Diesel" (DB2) und übernehmen Sie die entsprechenden Adressen aus der Klappliste.



Den Variablen wird je ein Signal "DM_xxx" für den Dieselmotor zugewiesen.

Speichern Sie Ihr Programm, und schließen Sie den Baustein.

Wenn Sie Programmstrukturen mit OBs, FBs und DBs anlegen, so müssen Sie den Aufruf eines untergeordneten Bausteins (z. B. FB1) im übergeordneten Baustein (z. B. OB1) programmieren. Die Vorgehensweise ist dabei immer identisch.

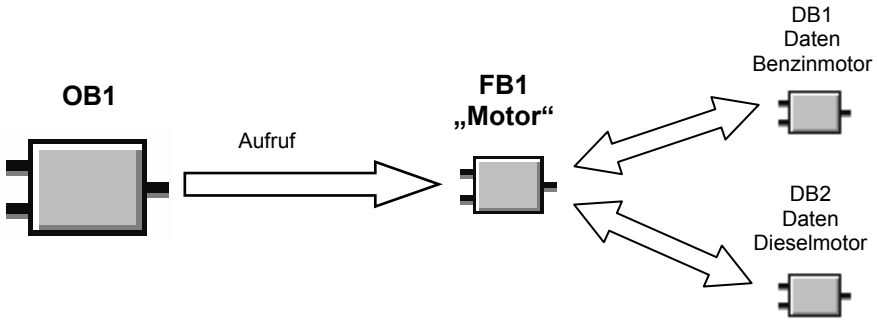
Sie können in der Symboltabelle auch den verschiedenen Bausteinen symbolische Namen geben (z. B. FB1 hat den Namen "Motor" und DB1 den Namen "Benzin").

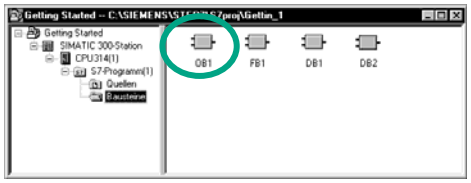
Sie können jederzeit die programmierten Bausteine archivieren oder ausdrucken. Die entsprechenden Funktionen finden Sie im SIMATIC Manager unter den Menüpunkten **Datei > Archivieren** bzw. **Datei > Drucken**.

Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Aufruf von Referenzhilfen", über "Sprachbeschreibung KOP" unter "KOP-Operationen", "Programmsteuerung".

5.7 Bausteinaufruf in AWL programmieren

Die gesamte Programmierung eines Funktionsbausteins ist ohne Aufruf im OB1 unwirksam. Pro Aufruf des Funktionsbausteins wird je ein Datenbaustein benutzt und somit beide Motoren gesteuert.





Der SIMATIC Manager ist mit Ihrem Projekt "Getting Started" geöffnet.

Navigieren Sie zum Ordner **Bausteine**, und öffnen Sie den OB1.

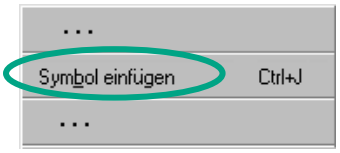
Markieren Sie im KOP/AWL/FUP-Programmfenster das Netzwerk 3 und fügen Sie das Netzwerk 4 ein.

```
CALL "Motor" , "Benzin"
   Einschalten       :=
   Ausschalten       :=
   Stoerung          :=
   Drehzahl_Ist      :=
   Motor_Ein         :=
   Soll_Dreh_erreicht:=
```

Schreiben Sie in den Anweisungsteil **CALL "Motor", "Benzin"** und drücken Sie anschließend **Return**.

Alle Parameter des Funktionsbausteins "Benzin" werden angezeigt.

Positionieren Sie den Cursor hinter dem Gleichheitszeichen von Einschalten und drücken Sie die rechte Maustaste.



Klicken Sie im Kontextmenü der rechten Maustaste auf **Symbole einfügen**. Eine Klappliste öffnet sich.

Doppelklicken Sie auf den Namen **BM_einschalten**. Er wird automatisch mit Anführungszeichen aus der Klappliste übernommen.

```
CALL  "Motor" , "Benzin"
 Einschalten        :="BM_einschalten"
 Ausschalten        :="BM_ausschalten"
 Stoerung           :="BM_Störung"
 Drehzahl_Ist       :="BM_Drehzahl_Ist"
 Motor_Ein          :="BM_Ein"
 Soll_Dreh_erreicht:="BM_Soll_erreicht"
```

Weisen Sie mit der Klappliste den Variablen des Funktionsbausteins alle notwendigen Adressen zu.

Den Variablen wird je ein Signal „BM_xxx“ für den Benzinmotor zugewiesen.

```
CALL  "Motor" , "Diesel"
 Einschalten        :="DM_einschalten"
 Ausschalten        :="DM_ausschalten"
 Stoerung           :="DM_ausschalten"
 Drehzahl_Ist       :="DM_Drehzahl_Ist"
 Motor_Ein          :="DM_Ein"
 Soll_Dreh_erreicht:="DM_Soll_erreicht"
```

Programmieren Sie in einem neuen Netzwerk den Aufruf des Funktionsbausteins "Motor" (FB1) mit dem Datenbaustein "Diesel" (DB2). Gehen Sie dabei analog zum oberen Aufruf vor.

Speichern Sie Ihr Programm, und schließen Sie den Baustein.

Wenn Sie Programmstrukturen mit OBs, FBs und DBs anlegen, so müssen Sie den Aufruf eines untergeordneten Bausteins (z. B. FB1) im übergeordneten Baustein (z. B. OB1) programmieren. Die Vorgehensweise ist dabei immer identisch.

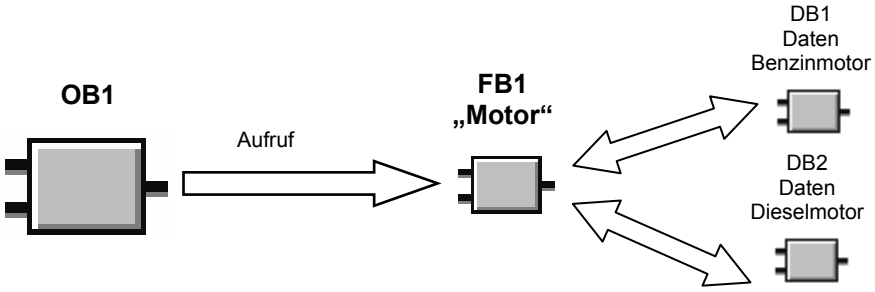
Sie können in der Symboltabelle auch den verschiedenen Bausteinen symbolische Namen geben (z. B. FB1 hat den Namen "Motor“ und DB1 den Namen "Benzin“).

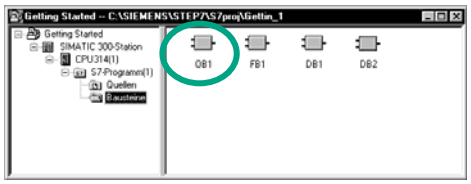
Sie können jederzeit die programmierten Bausteine archivieren oder ausdrucken. Die entsprechenden Funktionen finden Sie im SIMATIC Manager unter den Menüpunkten **Datei > Archivieren** bzw. **Datei > Drucken**.

Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Aufruf von Referenzhilfen", über "Sprachbeschreibung AWL" unter "Anweisungsliste", "Programmsteuerung".

5.8 Bausteinaufruf in FUP programmieren

Die gesamte Programmierung eines Funktionsbausteins ist ohne Aufruf im OB1 unwirksam. Pro Aufruf des Funktionsbausteins wird je ein Datenbaustein benutzt und somit beide Motoren gesteuert.

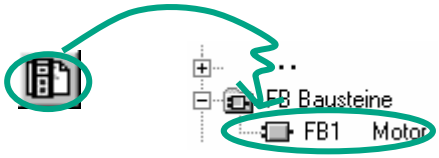




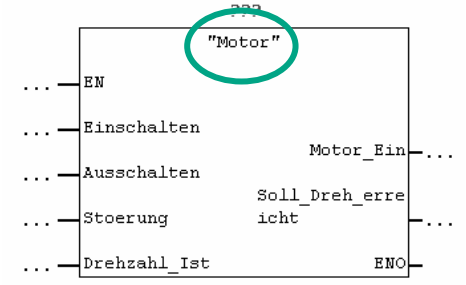
Der SIMATIC Manager ist mit Ihrem Projekt "Getting Started" geöffnet.

Navigieren Sie zum Ordner **Bausteine**, und öffnen Sie den **OB1**.

Markieren Sie im KOP/AWL/FUP-Programmfenster das Netzwerk 3 und fügen Sie das Netzwerk 4 ein.

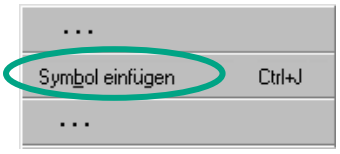


Navigieren Sie anschließend im Programmelemente-Katalog zum **FB1** und fügen ihn ein.

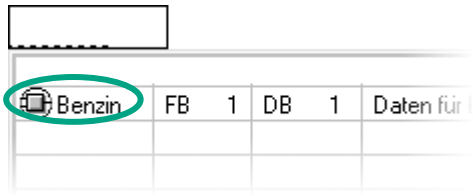


Alle motorspezifischen Ein- und Ausgangsvariablen werden angezeigt.

Klicken Sie auf die **???** über "Motor", gleich darauf klicken Sie mit der rechten Maustaste in den Eingaberahmen.

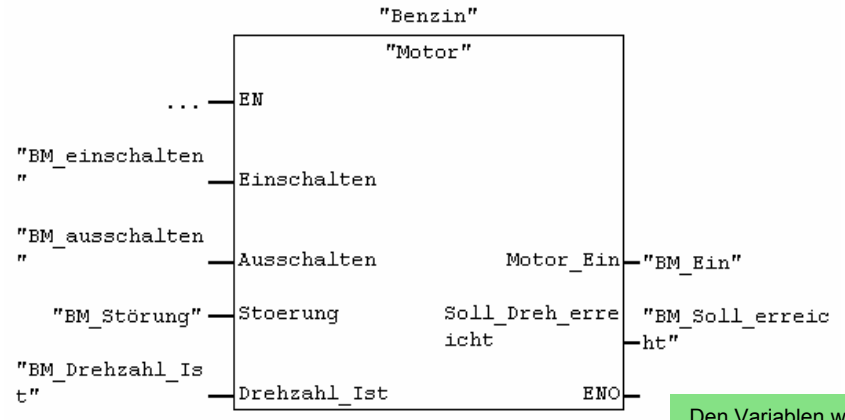


Klicken Sie im Kontextmenü der rechten Maustaste auf **Symbole einfügen**. Eine Klappliste öffnet sich, dieser Vorgang dauert das erste Mal einige Zeit.



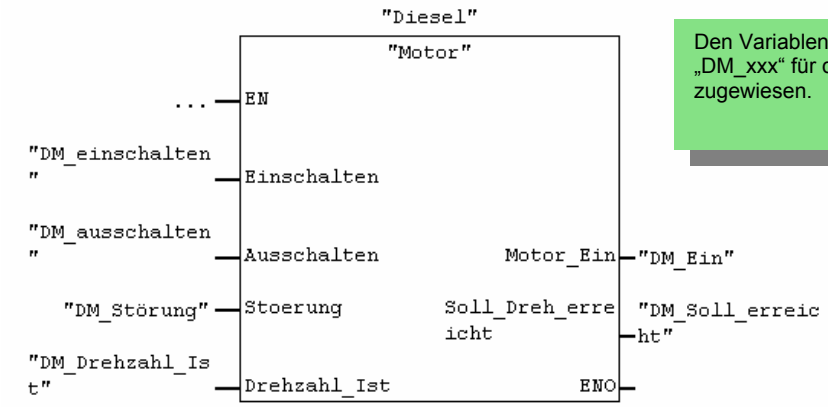
Doppelklicken Sie auf das Symbol **Benzin**. Es wird automatisch mit Anführungszeichen in das Eingabefeld übernommen.

Adressieren Sie mit der Klappliste alle weiteren Parameter des Funktionsbausteins mit den entsprechenden symbolischen Namen.



Den Variablen wird je ein Signal "BM_xxx" für den Benzinmotor zugewiesen.

Programmieren Sie in einem neuen Netzwerk den Aufruf des Funktionsbausteins "Motor" (FB1) mit dem Datenbaustein "Diesel" (DB2) und übernehmen Sie die entsprechenden Adressen aus der Klappliste.



Den Variablen wird je ein Signal „DM_xxx“ für den Dieselmotor zugewiesen.

Speichern Sie Ihr Programm und schließen Sie den Baustein.

Wenn Sie Programmstrukturen mit OBs, FBs und DBs anlegen, so müssen Sie den Aufruf eines untergeordneten Bausteins (z. B. FB1) im übergeordneten Baustein (z. B. OB1) programmieren. Die Vorgehensweise ist dabei immer identisch.

Sie können in der Symboltabelle auch den verschiedenen Bausteinen symbolische Namen geben (z. B. FB1 hat den Namen "Motor" und DB1 den Namen "Benzin").

Sie können jederzeit die programmierten Bausteine archivieren oder ausdrucken. Die entsprechenden Funktionen finden Sie im SIMATIC Manager unter den Menüpunkten **Datei > Archivieren** bzw. **Datei > Drucken**.

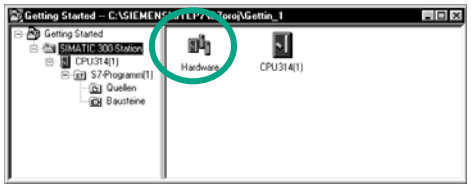
Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Aufruf von Referenzhilfen", über "Sprachbeschreibung FUP" unter "Funktionsplanoperationen", "Programmsteuerung".

6 Konfigurieren der zentralen Baugruppen

6.1 Hardware konfigurieren

Die Hardware können Sie konfigurieren, wenn Sie ein Projekt mit einer SIMATIC-Station angelegt haben. Die Projektstruktur, die mit dem "STEP 7 Assistenten" im Kapitel 2.1 angelegt wurde, verfügt über alle Voraussetzungen.

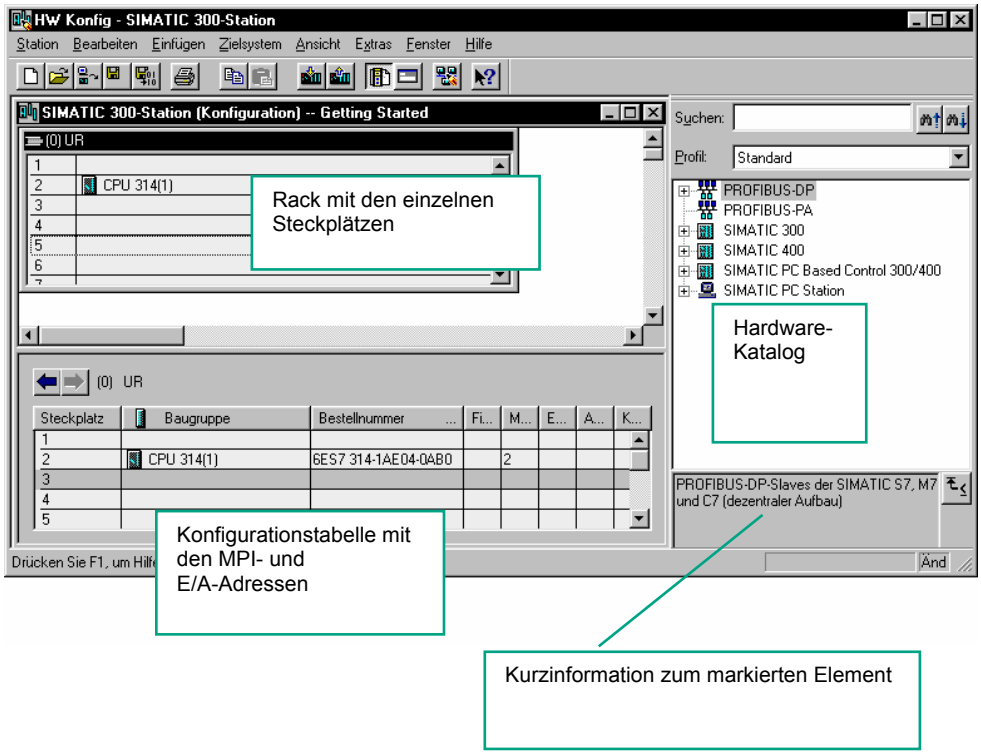
Mit STEP 7 wird die Hardware konfiguriert. Diese Konfigurationsdaten werden später beim "Laden" (vgl. Kapitel 7) in das Automatisierungssystem übertragen.

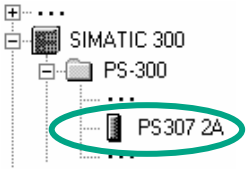


Ausgangspunkt ist der geöffnete SIMATIC Manager zusammen mit dem Projekt "Getting Started".

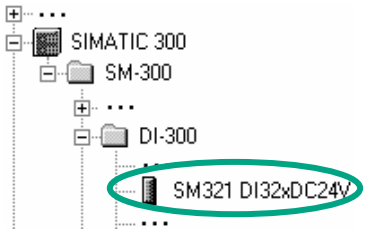
Öffnen Sie den Ordner **SIMATIC 300-Station**, und doppelklicken Sie auf das Symbol **Hardware**.

Das Fenster "HW Konfig" öffnet sich. Die bereits beim Anlegen des Projekts ausgesuchte CPU wird angezeigt. Für das "Getting Started" ist dies die CPU314.

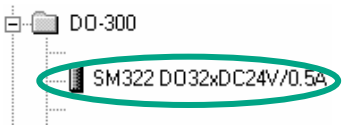




Sie benötigen zunächst eine Stromversorgung. Navigieren Sie im Katalog zur **PS307 2A** und fügen Sie diese per Drag and Drop auf Steckplatz 1 ein.



Navigieren Sie für eine Eingabebaugruppe (DI, Digital Input) zu **SM321 DI32xDC24V** und fügen Sie diese auf Steckplatz 4 ein. Der Steckplatz 3 bleibt frei.

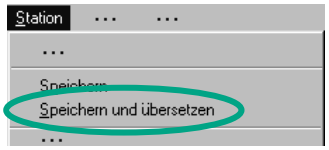


Analog hierzu fügen Sie auf Steckplatz 5 die Ausgabebaugruppe **SM322 DO32xDC24V/0.5A** ein.

Um die Parameter (z.B. Adresse) einer Baugruppe innerhalb des Projekts zu ändern, öffnen Sie diese mit Doppelklick. Parameter sollten Sie jedoch nur ändern, wenn Sie sich sicher sind, welche Auswirkungen die Änderungen auf Ihre SPS haben.

Für das Projekt "Getting Started" sind keine Änderungen nötig.

Steckplatz	Baugruppe	Bestellnummer	MPI-Adresse	E-Adresse	A-Adre...	Kommentar
1	PS307 5A	6ES7 307-1EA00-0AA0				
2	CPU314(1)	6ES7 314-1AE02-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4...7	
6						
7						
8						
9						
10						
11						



Die Daten werden mit **Speichern und übersetzen** gleich für das Übertragen in die CPU vorbereitet.

Nach dem Beenden von "HW Konfig" wird im Ordner Bausteine das Symbol Systemdaten angezeigt.

Sie können zusätzlich mit dem Menübefehl **Station > Konsistenz prüfen** die Konfiguration auf Fehler überprüfen. Bei eventuellen Fehlern bietet STEP 7 Lösungsmöglichkeiten.

Mehr Informationen **Hilfe > Hilfethemen** "Konfigurieren der Hardware" und "Konfigurieren der zentralen Baugruppen".

7 Laden und Testen des Programms

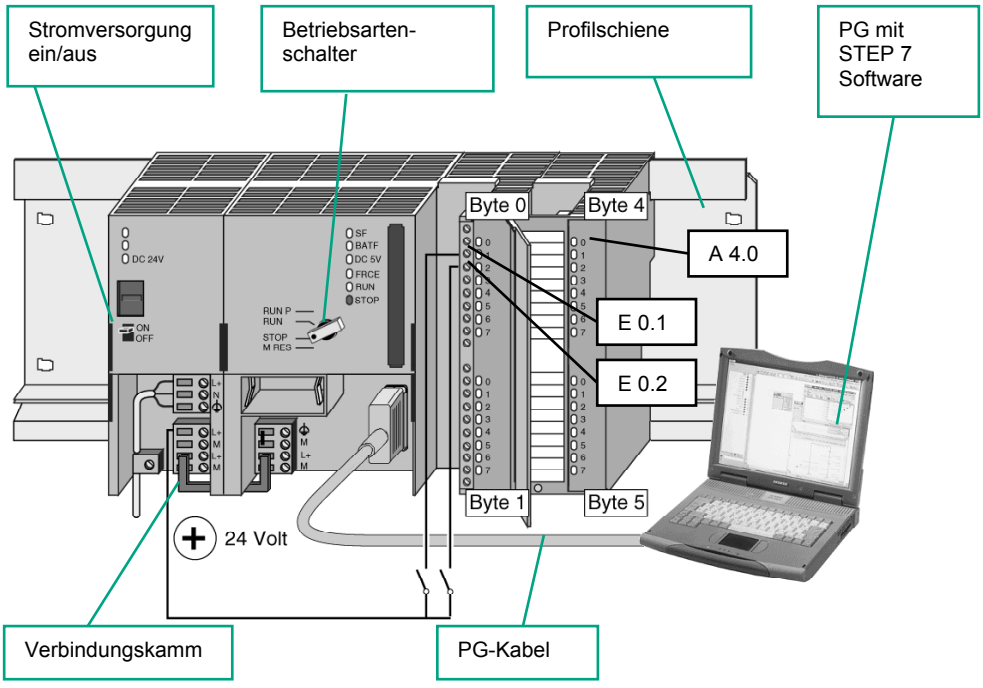
7.1 Online-Verbindung aufbauen

An einem der mitgelieferten Beispielprojekte oder dem bisher erstellten "Getting Started" und einem einfachen Testaufbau zeigen wir, wie Sie das Programm in das Automatisierungssystem (AS) laden können und es anschließend testen.

Sie sollten:

- die Hardware für das "Getting Started" konfiguriert haben (vgl. Kapitel 6)
- die Hardware gemäß Handbuch aufbauen

Beispiel für eine Reihenschaltung (UND-Funktion):
Ausgang A 4.0 darf erst leuchten (an der Digitalausgabebaugruppe leuchtet Diode A 4.0), wenn Taster E 0.1 **und** Taster E 0.2 gedrückt sind. Bauen Sie den Testaufbau mit Hilfe von Drähten und Ihrer CPU nach.



Hardware aufbauen

Um eine Baugruppe auf die Profilschiene zu montieren, gehen Sie in folgender Reihenfolge vor:

- Baugruppe auf den Busverbinder aufstecken
- Baugruppe einhängen und nach unten schwenken
- Baugruppe festschrauben
- Verbleibende Baugruppen montieren
- Schlüssel in die CPU stecken, nachdem Sie alle Baugruppen montiert haben

Der Test ist auch durchführbar, wenn Sie eine andere Hardware verwenden als abgebildet. Halten Sie lediglich die Adressierung der Ein- und Ausgänge ein.

STEP 7 bietet Ihnen zum Testen verschiedene Möglichkeiten an z. B. der Test über Programmstatus oder über die Variablentabelle.

Mehr Informationen zum Aufbauen der zentralen Baugruppen in den Handbüchern "S7-300 - Aufbauen, CPU-Daten" bzw. "S7-400/M7-400 - Aufbauen".

7.2 Laden des Programms in das Zielsystem

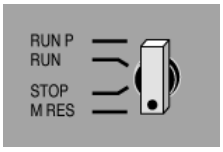
Das Laden des Programms setzt voraus, daß Sie die Online-Verbindung hergestellt haben.

Spannung anlegen

Schalten Sie das Netz am ON/OFF-Schalter ein. Diode "DC 5V" an der CPU leuchtet.

Drehen Sie den Betriebsartenschalter auf STOP (falls nicht bereits auf STOP). Die LED "STOP" leuchtet rot.

CPU urlöschen und in RUN setzen



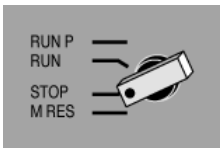
Drehen Sie den Betriebsartenschalter in die Stellung **MRES** und halten diesen für mindestens 3 sec. gedrückt bis die LED "STOP" langsam rot blinkt.

Betriebsartenschalter wieder loslassen und nach spätestens 3 sec. wieder in Stellung **MRES** drehen. Wenn die LED "STOP" schnell blinkt wird die CPU urgelöscht.

Falls "STOP" nicht entsprechend schnell blinkt, wiederholen Sie bitte den Vorgang.

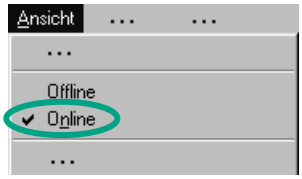
Urlöschen löscht alle Daten auf der CPU. Die CPU befindet sich nun im Grundzustand.

Programm in CPU laden

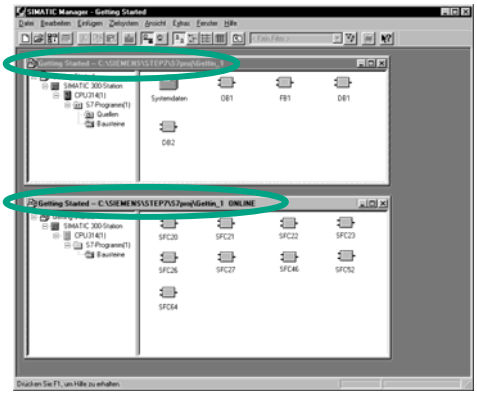


Drehen Sie nun zum Laden des Programms den Betriebsartenschalter wieder auf "STOP".

Starten Sie den SIMATIC Manager, und öffnen Sie im SIMATIC Manager über das Dialogfeld "Öffnen" das Projekt "Getting Started" (falls nicht bereits erfolgt).



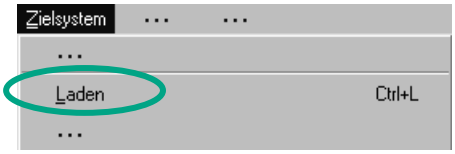
Rufen Sie zusätzlich zum Fenster "Getting Started Offline" das Fenster "Getting Started Online" auf. Der Status Offline/Online ist in der Kopfzeile durch die Farbumschaltung gekennzeichnet.



Navigieren Sie in beiden Fenstern zum Ordner **Bausteine**.

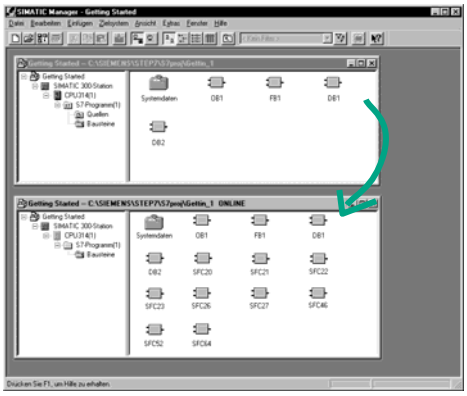
Fenster "Offline" zeigt die Situation auf dem PG, Fenster "Online" zeigt die Situation auf der CPU.

Trotz Urlöschen befinden sich in der CPU die Systemfunktionen (SFCs). Diese Funktionen des Betriebssytems stellt die CPU bereit. Sie müssen nicht geladen werden, können jedoch auch nicht gelöscht werden.



Markieren Sie im Fenster "Offline" den Ordner **Bausteine** und laden Sie anschließend das Programm über **Zielsystem > Laden** in die CPU.

Bestätigen Sie die Abfrage mit **OK**.



Im Fenster "Online" werden nach dem Laden die Bausteine des Programms angezeigt.

Den Menübefehl **Zielsystem > Laden** können Sie auch über das entsprechende Symbol in der Funktionsleiste oder über das Kontextmenü der rechten Maustaste aufrufen.

CPU einschalten und Betriebszustand überprüfen

Drehen Sie den Betriebsartenschalter auf **RUN-P**. Die LED "RUN" leuchtet grün und die LED "STOP" rot erlischt. Die CPU ist betriebsbereit.

Wenn die LED grün leuchtet, dann können Sie mit dem Testen des Programms beginnen.

Wenn die LED weiterhin rot leuchtet liegt ein Fehler vor. Zur Fehlerdiagnose würden Sie dann den Diagnosepuffer auswerten.

Laden einzelner Bausteine

Um in der Praxis schnell auf Fehler reagieren zu können, lassen sich Bausteine einzeln per Drag and Drop auf die CPU übertragen.

Beim Laden von Bausteinen muss der Betriebsartenschalter an der CPU entweder auf "RUN-P" oder "STOP" stehen. Im Betriebszustand "RUN" geladene Bausteine werden sofort aktiviert. Sie sollten dabei bedenken:

- Werden fehlerfreie Bausteine durch fehlerhafte Bausteine überschrieben, hat dies eine Fehlfunktion Ihrer Anlage zur Folge. Dies vermeiden Sie, indem Sie vor dem Laden Ihre Bausteine testen.
- Wurde die Reihenfolge für das Laden der Bausteine nicht beachtet – zuerst untere, dann obere Bausteinebenen laden – geht die CPU in den Betriebszustand "STOP" über. Dies vermeiden Sie, indem Sie das gesamte Programm auf die CPU laden.

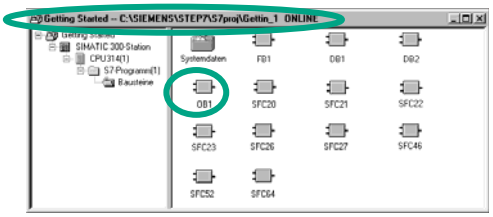
Online programmieren

Für Testzwecke kann es in der Praxis notwendig sein, bereits auf die CPU geladene Bausteine zu ändern. Hierzu doppelklicken Sie auf den gewünschten Baustein im Fenster "Online", um das KOP/AWL/FUP-Programmfenster zu öffnen. Programmieren Sie anschließend den Baustein wie gewohnt. Beachten Sie bitte, daß der programmierte Baustein sofort in Ihrer CPU aktiv wird.

Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Laden" und im Buch "Aufbau der Online-Verbindung und CPU-Einstellung".

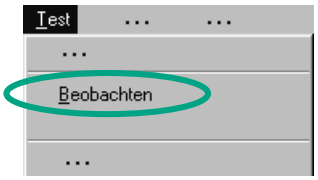
7.3 Programm mit Programmstatus testen

Über die Funktion Programmstatus testen Sie das Programm eines Bausteins. Voraussetzung ist, daß eine Online-Verbindung zur CPU besteht, die CPU sich in RUN bzw. RUN-P befindet und das Programm auf die CPU geladen wurde.



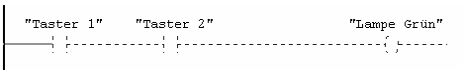
Öffnen Sie den **OB1** im Projektfenster "Getting Started Online".

Das KOP/AWL/FUP-Programmfenster wird geöffnet.



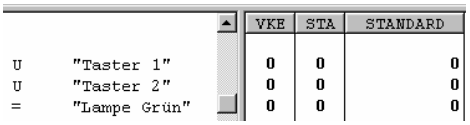
Aktivieren Sie die Funktion **Test > Beobachten**.

Testen mit KOP



Die Reihenschaltung im Netzwerk 1 in KOP wird angezeigt. Bis zum Taster 1 (E 0.1) wird der Strompfad durchgezogen dargestellt, d. h. hier liegt bereits Spannung an.

Testen mit AWL

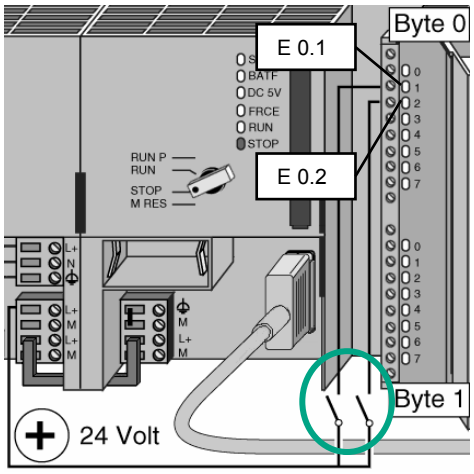


Für AWL werden
– Verknüpfungsergebnis (VKE)
– Statusbit (STA)
– Standardstatus (STANDARD)
in Tabellenform angezeigt.

Testen mit FUP

Der Signalzustand wird mit "0" und "1" gekennzeichnet. Gestrichelte Linie bedeutet, dass kein Verknüpfungsergebnis vorliegt.

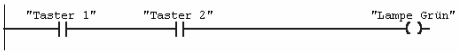
Über **Extras > Einstellungen** können Sie die Darstellungsart der Programmiersprache beim Testen ändern.



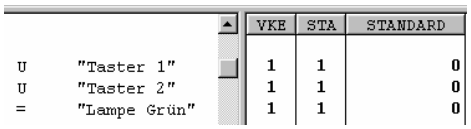
Schließen Sie jetzt an Ihrem Testaufbau beide Taster.

An der Eingabebaugruppe leuchten die Dioden für die Eingänge E 0.1 und E 0.2.

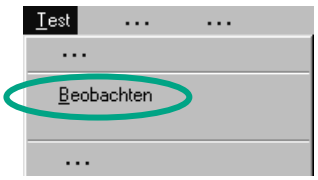
An der Ausgabebaugruppe leuchtet die Diode für den Ausgang A 4.0.



In den grafischen Programmiersprachen KOP / FUP können Sie das Testergebnis am Farbumschlag im programmierten Netzwerk mitverfolgen. Der Farbumschlag symbolisiert, dass das Verknüpfungsergebnis bis zu dieser Stelle erfüllt ist.



Bei der Programmiersprache AWL ändert sich bei erfülltem Verknüpfungsergebnis die Anzeige in der Spalte STA und in der Spalte VKE.



Deaktivieren Sie **Test > Beobachten** und schließen Sie das Fenster.

Schließen Sie daraufhin im SIMATIC Manager das "Online" Fenster.

Wir empfehlen umfangreiche Programme nie komplett auf die CPU zu laden und dort ablaufen zu lassen, da eine Fehlerdiagnose durch die Vielzahl der möglichen Fehlerquellen schwieriger ist. Vielmehr sollten Sie zur besseren Übersicht einzelne Bausteine separat laden und anschließend testen.

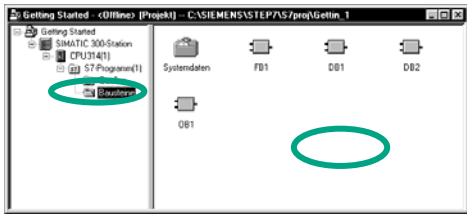
Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Testen" unter "Testen mit Programmstatus".

7.4 Programm mit Variablentabelle testen

Sie testen einzelne Programmvariablen, indem Sie diese beobachten und steuern. Voraussetzung ist, dass eine Online-Verbindung zur CPU besteht, diese sich in RUN-P befindet und das Programm geladen wurde.

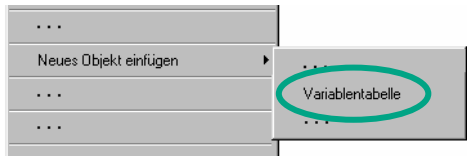
Wie beim Testen mit Programmstatus können Sie die Ein- und Ausgänge des Netzwerks 1 (Reihenschaltung bzw. UND-Funktion) in der Variablentabelle beobachten. Zusätzlich können Sie durch Vorgabe einer IST-Drehzahl den Vergleicher für die Motordrehzahl im FB1 testen.

Variablentabelle erstellen

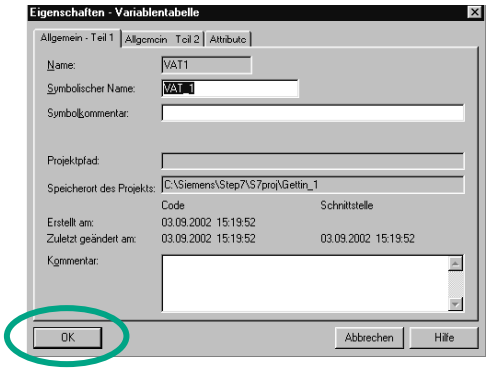


Ausgangspunkt ist wieder der SIMATIC Manager mit dem geöffneten Projekt "Getting Started Offline".

Navigieren Sie zum Ordner **Bausteine**, und klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte.

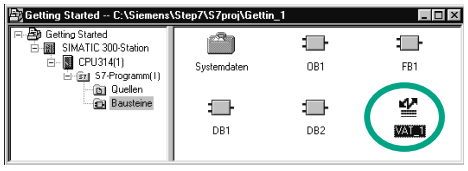


Fügen Sie mit dem Kontextmenü der rechten Maustaste die **Variablen-tabelle** ein.



Übernehmen Sie die Voreinstellungen, indem Sie das Dialogfeld "Eigenschaften" mit **OK** schließen.

Sie können alternativ der Variablen-tabelle einen Symbolnamen geben und mit einem Symbolkommentar versehen.

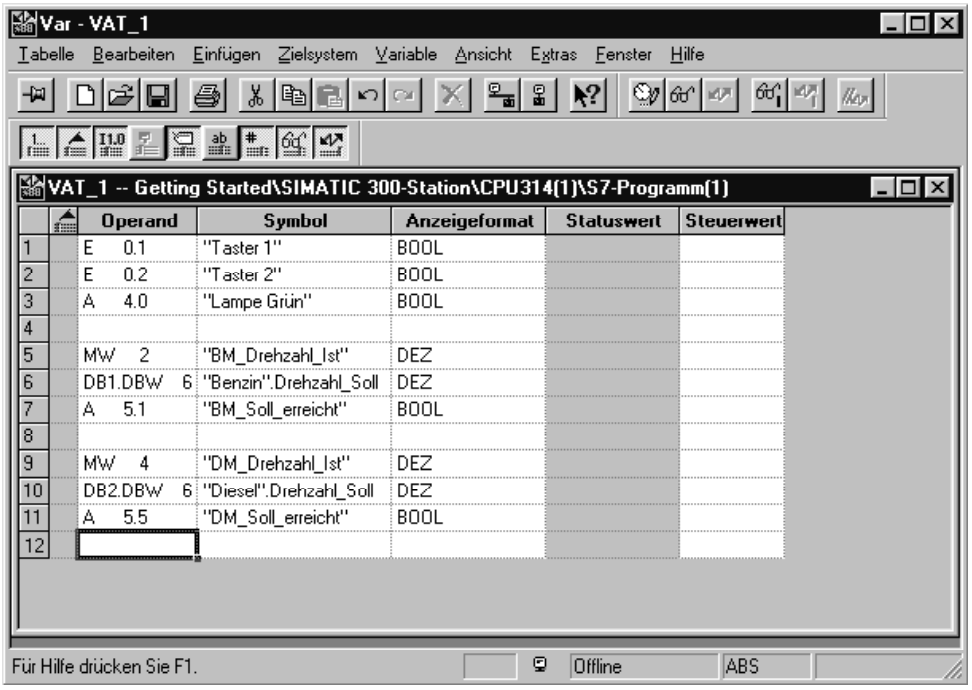


Eine VAT1 (Variablentabelle) wird im Ordner Bausteine angelegt.

Öffnen Sie **VAT1** (Doppelklick), Sie gelangen zum Fenster "Variablen steuern und beobachten".

Die Variablentabelle ist zunächst leer. Tragen Sie für das Beispiel "Getting Started" die Symbolnamen oder den Operanden entsprechend der Abbildung ein. Die restlichen Angaben werden vervollständigt, wenn Sie in die nächste Zelle wechseln.

Ändern Sie das Anzeigeformat aller Drehzahlwerte auf das Format DEZ. Markieren Sie die entsprechende Zelle und wählen Sie im Kontextmenü (Klick mit der rechten Maustaste) das gewünschte Format aus.

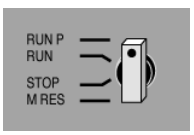


	Operand	Symbol	Anzeigeformat	Statuswert	Steuerwert
1	E 0.1	"Taster 1"	BOOL		
2	E 0.2	"Taster 2"	BOOL		
3	A 4.0	"Lampe Grün"	BOOL		
4					
5	MW 2	"BM_Drehzahl_Ist"	DEZ		
6	DB1.DBW 6	"Benzin".Drehzahl_Soll	DEZ		
7	A 5.1	"BM_Soll_erreicht"	BOOL		
8					
9	MW 4	"DM_Drehzahl_Ist"	DEZ		
10	DB2.DBW 6	"Diesel".Drehzahl_Soll	DEZ		
11	A 5.5	"DM_Soll_erreicht"	BOOL		
12					

Speichern Sie Ihre Variablentabelle.

Variablentabelle online schalten

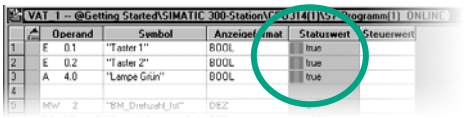
Stellen Sie die Verbindung zu der projektierten CPU her. In der Statuszeile wird der Betriebszustand der CPU angezeigt.



Stellen Sie den Schlüsselschalter der CPU auf **RUN-P** (falls noch nicht erfolgt).

Variablen beobachten

Klicken Sie auf **Variable beobachten**.

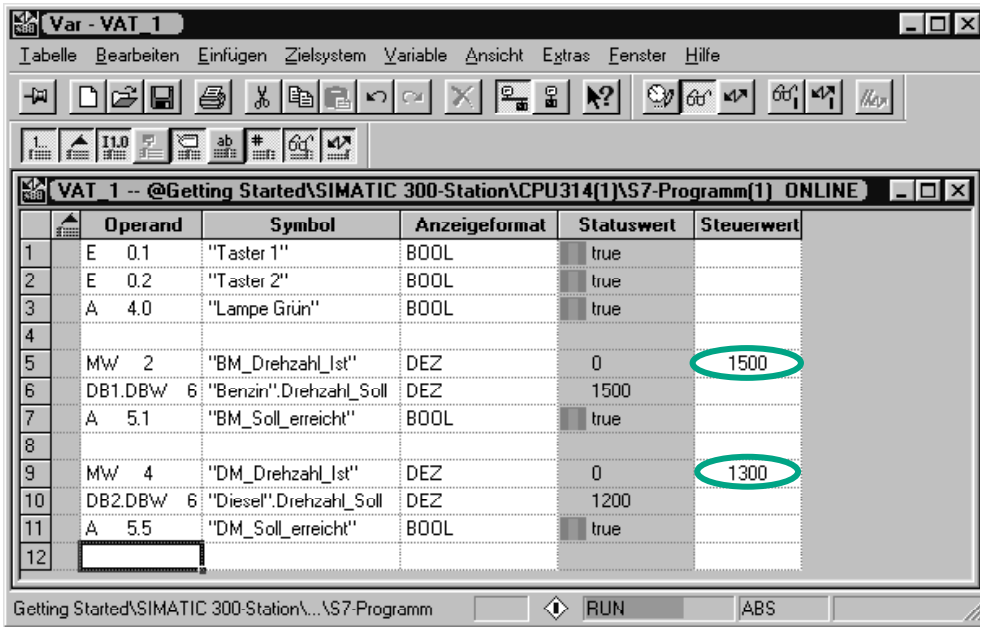


Schließen Sie die Taster 1 und Taster 2 in Ihrem Versuchsaufbau und beobachten Sie das Ergebnis in der Variablentabelle.

Die Statuswerte in der Variablentabelle ändern sich von false auf true.

Variablen steuern

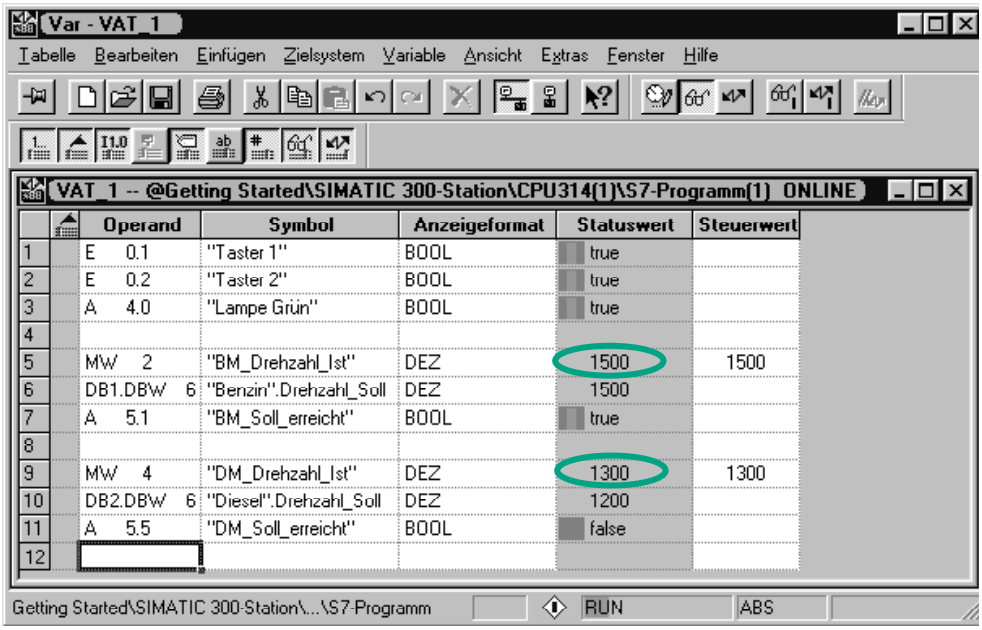
Tragen Sie in der Spalte Steuerwert für den Operanden MW2 den Wert "1500" und für den Operanden MW4 den Wert "1300" ein.



Übertragen Sie die Steuerwerte auf Ihre CPU.

Nach dem Übertragen werden diese Werte in Ihrer CPU verarbeitet. Das Ergebnis des Vergleichs wird sichtbar.

Beenden Sie Variablen beobachten und schließen Sie das Fenster. Eine eventuelle Abfrage beantworten Sie mit **Ja** bzw. mit **OK**.



	Operand	Symbol	Anzeigeformat	Statuswert	Steuerwert
1	E 0.1	"Taster 1"	BOOL	true	
2	E 0.2	"Taster 2"	BOOL	true	
3	A 4.0	"Lampe Grün"	BOOL	true	
4					
5	MW 2	"BM_Drehzahl_Ist"	DEZ	1500	1500
6	DB1.DBW 6	"Benzin".Drehzahl_Soll	DEZ	1500	
7	A 5.1	"BM_Soll_erreicht"	BOOL	true	
8					
9	MW 4	"DM_Drehzahl_Ist"	DEZ	1300	1300
10	DB2.DBW 6	"Diesel".Drehzahl_Soll	DEZ	1200	
11	A 5.5	"DM_Soll_erreicht"	BOOL	false	
12					

Eine sehr umfangreiche Variablentabelle kann häufig aufgrund der Bildschirmbegrenzung nicht vollständig angezeigt werden.

Sollten Sie große Variablentabellen haben, empfehlen wir mit STEP 7 mehrere Variablentabellen zu einem S7-Programm anzulegen. Die Variablentabellen können Sie genau auf Ihre Testbedürfnisse abstimmen.

Analog zu den Bausteinen können Sie Variablentabellen individuelle Namen zuweisen (z. B. anstelle von VAT1 den Namen OB1_Netzwerk1). Die Zuweisung der Namen erfolgt über die Symboltabelle.

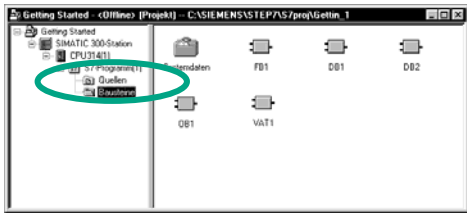
Mehr Informationen über **Hilfe > Hilfethemen** im Buch "Testen" unter "Testen mit der Variablentabelle".

7.5 Diagnosepuffer auswerten

Für den Fall der Fälle, dass die CPU beim Abarbeiten eines S7-Programms in STOP geht oder sich die CPU nach dem Laden des Programms nicht mehr in RUN schalten lässt, können Sie aus den im Diagnosepuffer aufgelisteten Ereignissen auf die Fehlerursache schließen.

Voraussetzung ist, dass eine Online-Verbindung zur CPU besteht und sich die CPU im Betriebszustand "STOP" befindet.

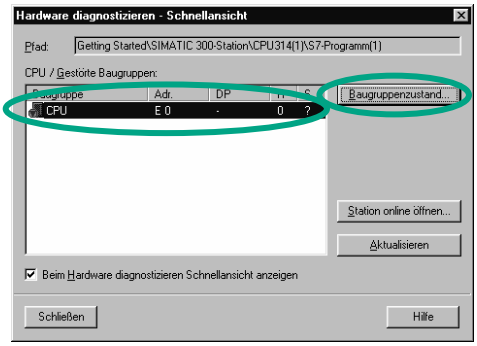
Drehen Sie zuerst den Betriebsartenschalter an der CPU auf "STOP".



Ausgangspunkt ist wieder der SIMATIC Manager mit dem geöffneten Projekt "Getting Started Offline".

Markieren Sie den Ordner **Bausteine**.

Befinden sich mehrere CPUs in Ihrem Projekt, stellen Sie zunächst fest, welche CPU in STOP gegangen ist.



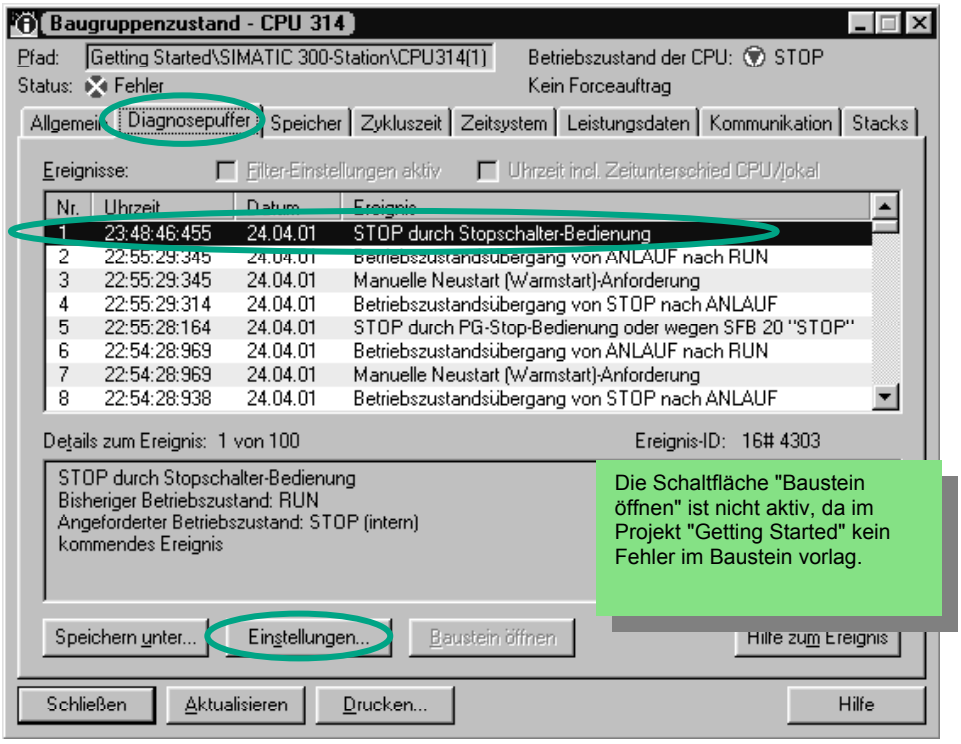
Im Dialogfeld "Hardware diagnostizieren" werden alle erreichbaren CPUs aufgeführt. Die CPU mit dem Betriebszustand STOP ist markiert.

Das Projekt "Getting Started" verfügt nur über eine CPU, die angezeigt wird.

Klicken Sie auf **Baugruppenzustand**, um den Diagnosepuffer dieser CPU auszuwerten.

Ist nur eine CPU angeschlossen, können Sie den Baugruppenzustand dieser CPU auch direkt abfragen über den Menübefehl **Zielsystem > Diagnose/Einstellung > Baugruppenzustand**.

Im Fenster "Baugruppenzustand" können Sie sich über Eigenschaften und Parameter Ihrer CPU informieren. Wählen Sie nun das Register **Diagnosepuffer**, um die Ursache des Betriebszustands STOP zu ermitteln.



Die Schaltfläche "Baustein öffnen" ist nicht aktiv, da im Projekt "Getting Started" kein Fehler im Baustein vorlag.

Das jüngste Ereignis (Nr. 1) steht dabei an oberster Stelle. Die Ursache von STOP wird angezeigt. Schließen Sie alle Fenster bis auf den SIMATIC Manager.

Ist ein Programmierfehler die Ursache von STOP, markieren Sie das Ereignis und klicken Sie auf die Schaltfläche **Baustein öffnen.**

Der Baustein wird dann im bekannten KOP/AWL/FUP-Programmfenster geöffnet und das fehlerhafte Netzwerk wird markiert.

Mit diesem Kapitel haben Sie das Projektbeispiel "Getting Started" vom Anlegen eines Projektes bis zum Testen des fertigen Programms erfolgreich abgeschlossen. In den nächsten Kapiteln können Sie Ihr Wissen durch ausgewählte Übungen weiter vertiefen.

Mehr Informationen über **Hilfe > Hilfethemen** "Diagnose" und "Baugruppenzustand".

8 Programmieren einer Funktion (FC)

8.1 Funktion anlegen und öffnen

Die Funktion ist wie der Funktionsbaustein dem Organisationsbaustein untergeordnet. Damit die Funktion von der CPU bearbeitet werden kann, muss diese ebenfalls im übergeordneten Baustein aufgerufen werden. Dabei ist im Gegensatz zum Funktionsbaustein kein Datenbaustein notwendig.

Bei einer Funktion werden die Parameter ebenfalls in der Variablendeklarationstabelle aufgeführt, jedoch sind keine statischen Lokaldaten zugelassen.

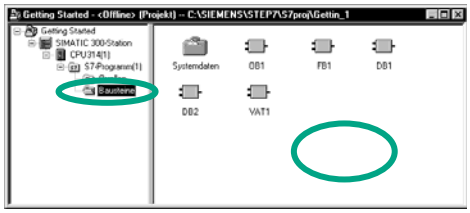
Die Funktion programmieren Sie analog zum Funktionsbaustein im KOP/FUP/AWL-Programmfenster.

Sie sollten bereits mit der Programmierung in KOP, FUP oder AWL (vgl. Kapitel 4 und 5) sowie der symbolischen Programmierung (vgl. Kapitel 3) vertraut sein.

Falls Sie das Beispielprojekt "Getting Started" Kapitel 1 bis 7 durchgeführt haben, öffnen Sie es nun.

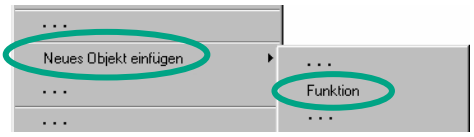
Falls nicht, legen Sie bitte ein neues Projekt im SIMATIC Manager mit **Datei > Assistent "Neues Projekt"** an. Gehen Sie dabei analog zum Kapitel 2.1 vor und benennen Sie das Projekt mit "Getting Started Funktion".

Wir gehen im weiteren Verlauf vom Projekt "Getting Started" aus.
Sie können jedoch jeden Schritt auch anhand eines neu angelegten Projekts nachvollziehen.

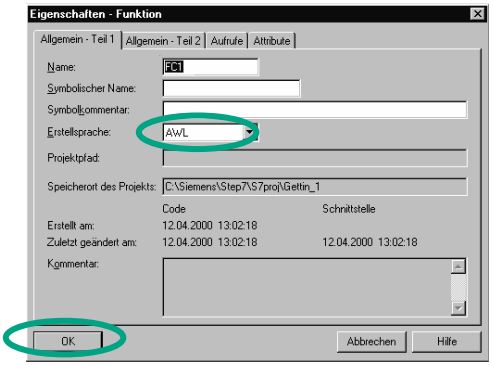


Navigieren Sie zum Ordner **Bausteine** und öffnen Sie ihn.

Klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte.

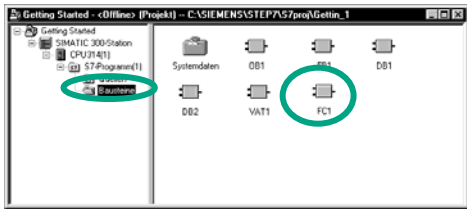


Fügen Sie über das Kontextmenü eine **Funktion** (FC) ein.



Im Dialogfenster "Eigenschaften - Funktion" übernehmen Sie den Namen FC1, und wählen Sie die gewünschte Erstellsprache aus.

Bestätigen Sie die restlichen Voreinstellungen mit **OK**.



Die Funktion FC1 wurde dem Ordner Bausteine hinzugefügt.

Öffnen Sie **FC1** durch Doppelklick.

Innerhalb einer Funktion können im Gegensatz zum Funktionsbaustein keine statischen Daten in der Variablendeklarationstabelle definiert werden.

Die in einem Funktionsbaustein definierten statischen Daten bleiben nach dem Abarbeiten des Bausteins bestehen. Statische Daten sind beispielsweise die verwendeten Merker für die Grenzwerte "Drehzahl" (vgl. Kapitel 5).

Für das Programmieren der Funktion können Sie wie gewohnt auf die symbolischen Namen aus der Symboltabelle zurückgreifen.

Mehr Information über **Hilfe > Hilfethemen** "Ausarbeiten des Automatisierungskonzepts“, "Grundlagen zum Entwerfen einer Programmstruktur“ und "Bausteine im Anwenderprogramm“.

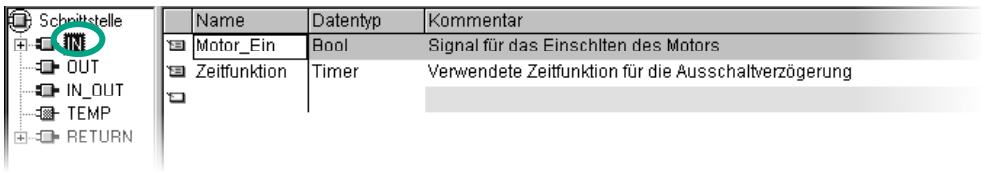
8.2 Funktion programmieren

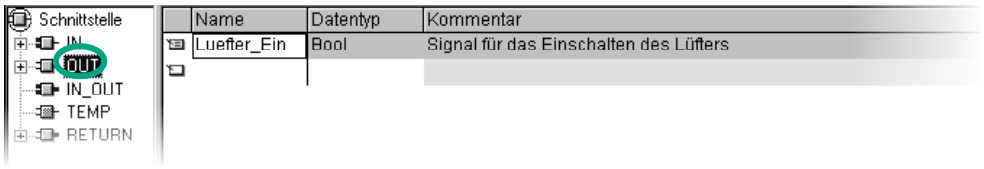
In unserem Beispiel programmieren Sie nachfolgend eine Zeitfunktion. Die Zeitfunktion bewirkt, dass sich beim Einschalten eines Motors (vgl. Kapitel 5) gleichzeitig ein Lüfter einschaltet, der nach dem Ausschalten des Motors noch vier Sekunden nachläuft (Ausschaltverzögerung).

Wie bereits erwähnt, müssen Sie die Ein- und Ausgangsparameter der Funktion (Deklaration "in" und "out") in der Variablendetailsicht eingeben.

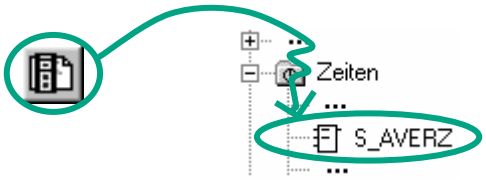
Das KOP/AWL/FUP-Programmfenster wurde geöffnet. Sie arbeiten mit dieser Variablendetailsicht wie mit der Variablendetailsicht des Funktionsbausteins (vgl. Kapitel 5).

Tragen Sie folgende Deklarationen ein.



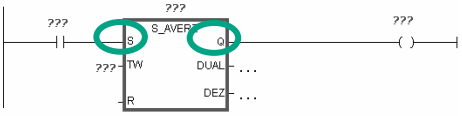


Zeitfunktion programmieren in KOP



Markieren Sie den Strompfad zur Eingabe der KOP-Anweisung.

Navigieren Sie im Register "Programmelemente" zum Element **S_AVERZ** (Zeit als Ausschaltverzögerung starten), und fügen Sie das Element ein.

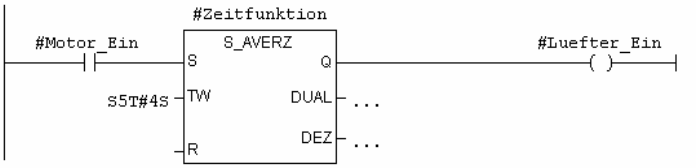


Fügen Sie vor den Eingang **S** einen Schließer ein.
Fügen Sie nach dem Ausgang **Q** eine Spule ein.

Markieren Sie die Fragezeichen, geben Sie ein "#" ein und wählen Sie die entsprechenden Namen aus.

Am Eingang TW des S_AVERZ stellen Sie die Dauer der Verzögerung ein. Dabei bedeutet S5T#4s, dass eine Konstante mit dem Datentyp S5Time#(S5T#) mit einer Dauer von vier Sekunden (4s) definiert wird.

Speichern Sie anschließend die Funktion, und schließen Sie das Fenster.



Mit dem Eingangsparameter "#Motor_Ein" wird die "#Zeitfunktion" gestartet. Sie wird später beim Aufruf im OB1 einmal mit den Parametern für den Benzinmotor und einmal mit den Parametern für den Dieselmotor (z.B. T1 für "BM_Nachlauf") versorgt. Die symbolischen Namen dieser Parameter werden Sie später in die Symboltabelle eintragen.

Zeitfunktion programmieren in AWL

```
U     #Motor_Ein
L     S5T#4S
SA    #Zeitfunktion
U     #Zeitfunktion
=     #Luefter_Ein
```

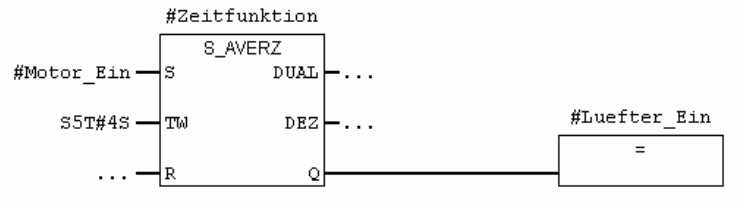
Falls Sie in AWL programmieren, markieren Sie den Eingabebereich unterhalb des Netzwerks, und geben Sie die nebenstehende Anweisung ein.

Speichern Sie anschließend die Funktion, und schließen Sie das Fenster.

Zeitfunktion programmieren in FUP

Falls Sie in FUP programmieren, markieren Sie den Eingabebereich unterhalb des Netzwerks, und geben Sie das untenstehende FUP-Programm der Zeitfunktion ein.

Speichern Sie anschließend die Funktion, und schließen Sie das Fenster.



Damit die Zeitfunktion abgearbeitet wird, ist ein Aufruf der Funktion in einem übergeordneten Baustein notwendig (in unserem Beispiel im OB1).

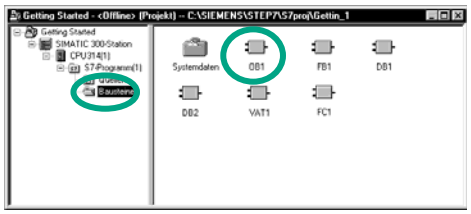
Mehr Informationen über **Hilfe > Hilfethemen** "Aufruf von Referenzhilfen", "Sprachbeschreibung KOP/FUP bzw. AWL" und "Zeiten".

8.3 Aufrufen der Funktion im OB1

Der Aufruf der Funktion FC1 erfolgt ähnlich wie der Aufruf des Funktionsbausteins im OB1. Alle Parameter der Funktion werden im OB1 mit den entsprechenden Operanden des Benzin- bzw. Dieselmotors versorgt.

Da diese Operanden noch nicht in der Symboltabelle definiert sind, werden die symbolischen Namen der Operanden noch in der Symboltabelle nachgetragen.

Ein Operand ist der Teil einer STEP 7-Anweisung, der aussagt, womit der Prozessor etwas tun soll. Er kann absolut oder symbolisch adressiert werden.



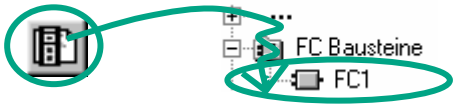
Der SIMATIC Manager ist mit dem Projekt "Getting Started" oder Ihrem neu angelegten Projekt geöffnet.

Navigieren Sie zum Ordner **Bausteine** und öffnen Sie den **OB1**.

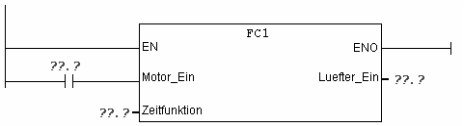
Das KOP/AWL/FUP-Programmfenster öffnet sich.

Aufruf programmieren in KOP

Sie befinden sich in der Ansicht **KOP**. Markieren Sie Netzwerk Nr. 5 und fügen Sie ein neues Netzwerk (Nr. 6) ein.



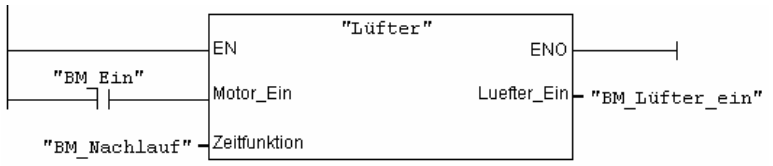
Navigieren Sie anschließend im Register "Programmelemente" zu FC1, und fügen Sie die Funktion ein.



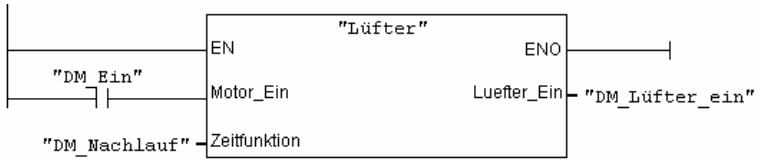
Fügen Sie vor "Motor_Ein" einen Schließer ein.

Über das Menü **Ansicht > Anzeigen mit > Symbolischer Darstellung** können Sie zwischen symbolischer und absoluter Darstellung wechseln.

Klicken Sie auf die Fragezeichen des FC1-Aufrufs, und tragen Sie die symbolischen Namen ein.



Programmieren Sie im Netzwerk 7 den Aufruf der Funktion FC1 mit den Operanden des Dieselmotors. Gehen Sie dabei analog zum vorherigen Netzwerk vor (die Operanden für den Dieselmotor haben Sie bereits in der Symboltabelle aufgenommen).



Speichern Sie den Baustein, und schließen Sie das Fenster.

Aktivieren Sie **Ansicht > Anzeigen mit > Symbolinformation**, um in jedem Netzwerk Informationen einzelner Adressen zu erhalten.

Um mehrere Netzwerke auf einem Bildschirm darzustellen, deaktivieren Sie **Ansicht > Anzeigen mit > Kommentar** und gegebenenfalls **Ansicht > Anzeigen mit > Symbolinformation**.

Mit **Ansicht > Zoomfaktor** können Sie die Darstellungsgröße der Netzwerke verstellen.

Aufruf programmieren in AWL

```
Netzwerk 6 : Lüftersteuerung für Benzinmotor
      CALL  "Lüfter"
       Motor_Ein    :="BM_Ein"
       Zeitfunktion:="BM_Nachlauf"
       Luefter_Ein :="BM_Lüfter_ein"

Netzwerk 7 : Lüftersteuerung für Dieselmotor
      CALL  "Lüfter"
       Motor_Ein    :="DM_Ein"
       Zeitfunktion:="DM_Nachlauf"
       Luefter_Ein :="DM_Lüfter_ein"
```

Falls Sie in AWL programmieren, markieren Sie jeweils den Eingabebereich unterhalb eines neuen Netzwerks, und geben Sie die nebenstehenden AWL-Anweisungen ein.

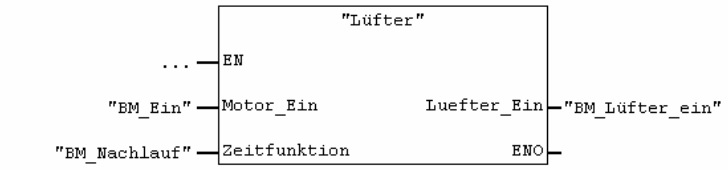
Speichern Sie anschließend den Aufruf, und schließen Sie das Fenster.

Aufruf programmieren in FUP

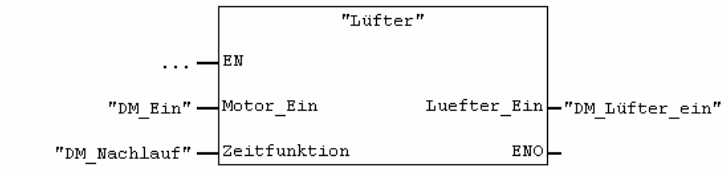
Falls Sie in FUP programmieren, markieren Sie jeweils den Eingabebereich unterhalb eines neuen Netzwerks, und geben Sie die untenstehenden FUP-Anweisungen ein.

Speichern Sie anschließend den Aufruf, und schließen Sie das Fenster.

Netzwerk 6 : Lüftersteuerung für Benzinmotor



Netzwerk 7 : Lüftersteuerung für Dieselmotor



Der Aufruf der Funktionen wurde in unserem Beispiel als unbedingter Aufruf programmiert, d. h. die Funktion wird immer bearbeitet.

Abhängig von den Anforderungen Ihrer Automatisierungsaufgabe können Sie den Aufruf von FCs oder FBs auch mit gewissen Bedingungen verknüpfen: z. B. an einen Eingang oder an eine Vorverschaltung. Zur Programmierung von Bedingungen steht der EN-Eingang bzw. der ENO-Ausgang der Box zur Verfügung.

Mehr Informationen über **Hilfe > Hilfethemen** "Aufruf von Referenzhilfen", "Sprachbeschreibung KOP/FUP" bzw. "AWL".

9 Programmieren eines Global-Datenbausteins

9.1 Global-Datenbaustein anlegen und öffnen

Falls die Anzahl der internen Merker einer CPU (Speicherzellen) nicht mehr ausreicht, den Datenbestand aufzunehmen, können ausgesuchte Daten in einem globalen Datenbaustein abgelegt werden.

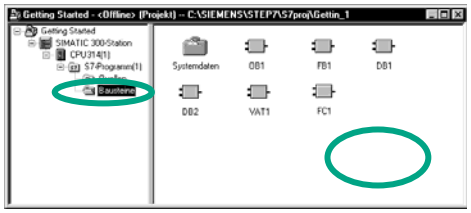
Die Daten des globalen Datenbausteins stehen jedem anderen Baustein zur Verfügung. Ein Instanz-Datenbaustein hingegen ist einem bestimmten Funktionsbaustein zugeordnet, seine Daten sind nur lokal in diesem Funktionsbaustein verfügbar (vgl. Kapitel 5.5).

Sie sollten bereits mit der Programmierung in KOP, FUP oder AWL (vgl. Kapitel 4 und 5) sowie der symbolischen Programmierung (vgl. Kapitel 3) vertraut sein.

Falls Sie das Beispielprojekt "Getting Started" Kapitel 1 bis 7 durchgeführt haben, öffnen Sie es nun.

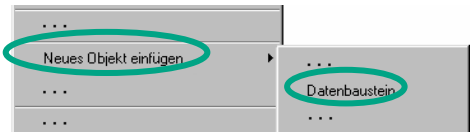
Falls nicht, legen Sie bitte ein neues Projekt im SIMATIC Manager mit **Datei > Assistent "Neues Projekt"** an. Gehen Sie dabei analog zum Kapitel 2.1 vor und benennen Sie das Projekt mit "Getting Started Global-DB".

Wir gehen im weiteren Verlauf vom Projekt "Getting Started" aus.
Sie können jedoch jeden Schritt auch anhand eines neu angelegten Projekts nachvollziehen.

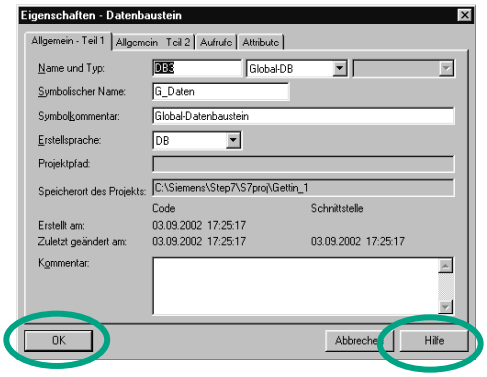


Navigieren Sie zum Ordner **Bausteine** und öffnen Sie ihn.

Klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte.



Fügen Sie über das Kontextmenü einen **Datenbaustein** ein.



Im Dialogfeld "Eigenschaften - Datenbaustein" übernehmen Sie alle Voreinstellungen mit **OK**.

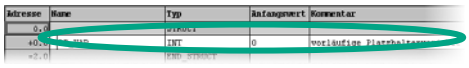
Benutzen Sie die **Hilfe** für weitere Informationen.

Der Datenbaustein DB3 wurde dem Ordner **Bausteine** hinzugefügt.

Öffnen Sie den **DB3** durch Doppelklick.

Zur Erinnerung: Im Kapitel 5.5 haben Sie einen "Instanz-DB" erzeugt. Über "Global-DB" legen Sie dagegen einen Global-Datenbaustein an.

Variablen im Datenbaustein programmieren



Tragen Sie in der Spalte Name "BM_Ist_Drehzahl" ein.

Wählen Sie für Typ mit der rechten Maustaste über das Kontextmenü **Elementare Typen > INT** aus.

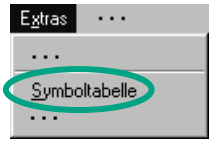
Exemplarisch sind drei globale Daten im DB3 definiert. Tragen Sie die Daten in der Variablendeklarationstabelle entsprechend nach.

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	BM_Ist_Drehzahl	INT	0	Ist-Drehzahl für Benzinmotor
+2.0	DM_Ist_Drehzahl	INT	0	Ist-Drehzahl für Dieselmotor
+4.0	Soll_erreicht	BOOL	FALSE	beide Motoren haben Soll-Drehzahl erreicht
=6.0		END_STRUCT		

Die Variablen für die Ist-Drehzahlen im Datenbaustein "BM_Ist_Drehzahl" und "DM_Ist_Drehzahl" werden genauso behandelt wie die Merkerwörter MW2 (BM_Drehzahl_Ist) und MW4 (DM_Drehzahl_Ist). Dies wird im nächsten Kapitel gezeigt.

Speichern Sie den globalen Datenbaustein.

Symbole zuordnen



Einem Datenbaustein können Sie ebenfalls einen symbolischen Namen zuordnen.

Öffnen Sie die **Symboltabelle** und tragen Sie für den Datenbaustein DB3 das Symbol "G_Daten" ein.

Falls Sie im Kapitel 4 die Symboltabelle aus einem Beispielprojekt (zDt01_02_STEP7__AWL_1-10, zDt01_06_STEP7__KOP_1-10 oder zDt01_04_STEP7__FUP_1-10) in Ihr Projekt "Getting Started" kopiert haben, müssen Sie keine Symbole nachträglich einfügen.

Symbol	Adresse	Datentyp	Kommentar
...	...	...	...
G_Daten	DB 3	DB 3	Global-Datenbaustein

Speichern Sie die Symboltabelle und schließen Sie das Fenster "Symbol Editor".

Schließen Sie außerdem den Global-Datenbaustein.

Global-DB in der Variablendeklarationstabelle:

Mit **Ansicht > Datensicht** können Sie in der Tabelle des Global-Datenbausteins die Aktualwerte des Datentyps INT ändern (vgl. Kapitel 5.5).

Global-DB in der Symboltabelle:

Im Gegensatz zum Instanz-Datenbaustein ist in der Symboltabelle der Datentyp zum Global-Datenbaustein immer die absolute Adresse. In unserem Beispiel ist der Datentyp "DB3". Beim Instanz-Datenbaustein ist als Datentyp stets der zugehörige FB angegeben.

Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen" und "Erstellen von Datenbausteinen".

10 Programmieren einer Multiinstanz

10.1 Übergeordneten Funktionsbaustein anlegen und öffnen

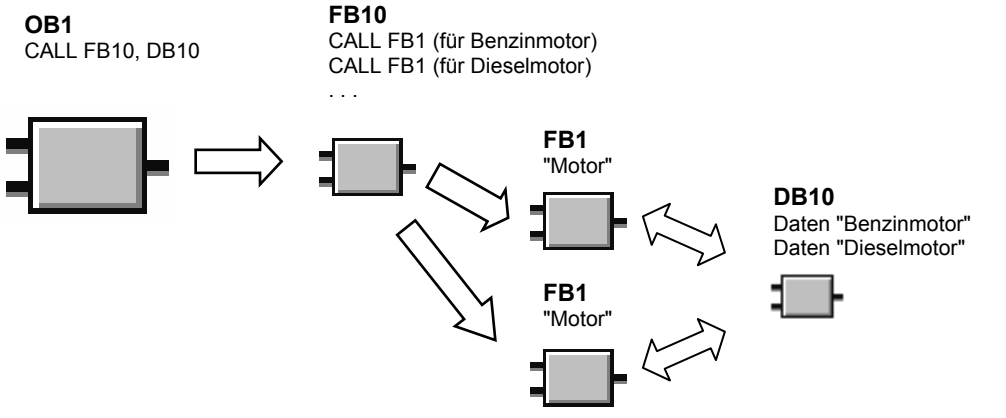
Im Kapitel 5 haben Sie eine Motorsteuerung mit dem Funktionsbaustein "Motor" (FB1) programmiert. Beim Aufruf des Funktionsbausteins FB1 im Organisationsbaustein OB1 benutzte der FB1 hier die Datenbausteine "Benzin" (DB1) und "Diesel" (DB2). Die Datenbausteine enthielten jeweils die unterschiedlichen Daten (z. B. #Drehzahl_Soll) der Motoren.

Stellen Sie sich nun vor, Sie benötigen für Ihre Automatisierungsaufgaben weitere Motorsteuerungen, z. B. für die Steuerung eines Rapsölmotors, eines Wasserstoffmotors usw.

Bei dem bisher kennengelernten Vorgehen würden Sie nun für jede zusätzliche Motorsteuerung den FB1 verwenden und diesem jeweils einen neuen DB mit den jeweiligen Daten des Motors zuordnen. Für die Steuerung des Rapsölmotors den FB1 mit DB3, für den Wasserstoffmotor den FB1 mit DB4 usw. Die Anzahl der Bausteine erhöht sich deutlich mit den Motorsteuerungen.

Sie können die Anzahl der Bausteine reduzieren, indem Sie mit Multiinstanzen arbeiten. Legen Sie hierzu einen übergeordneten neuen FB an (in unserem Beispiel den FB10), und rufen Sie in ihm den unveränderten FB1 als "Lokale Instanz" auf. Für jeden Aufruf legt der untergeordnete FB1 seine Daten im Datenbaustein DB10 des übergeordneten FB10 ab. Dem FB1 müssen somit keine DBs zugeordnet werden. Alle FBs greifen auf einen einzigen Datenbaustein zurück (hier den DB10).

Im DB10 werden die Datenbausteine DB1 und DB2 integriert. Hierzu ist es notwendig, den FB1 in den statischen Lokaldaten des FB10 zu deklarieren.

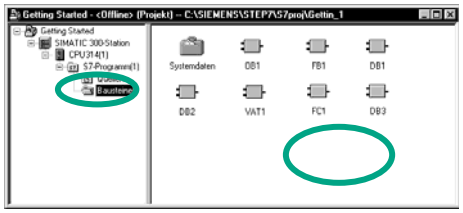


Sie sollten bereits mit der Programmierung in KOP, FUP oder AWL (vgl. Kapitel 4 und 5) sowie der symbolischen Programmierung (vgl. Kapitel 3) vertraut sein.

Falls Sie das Beispiel "Getting Started" Kapitel 1 bis 7 durchgeführt haben, öffnen Sie das Projekt "Getting Started".

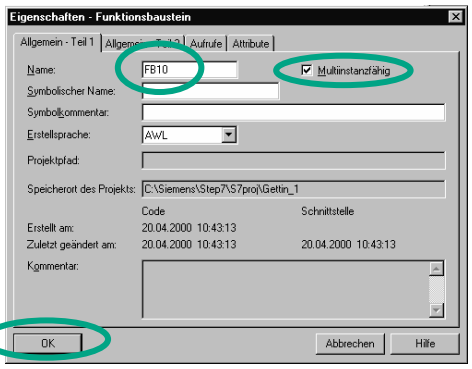
Falls nicht, öffnen Sie bitte über den SIMATIC Manager das Projekt

- ZDt01_01_STEP7__**AWL**_1-9,
- ZDt01_05_STEP7__**KOP**_1-9,
- ZDt01_03_STEP7__**FUP**_1-9



Navigieren Sie zum Ordner **Bausteine**, und öffnen Sie ihn.

Klicken Sie mit der rechten Maustaste in die rechte Fensterhälfte, und fügen Sie über das Kontextmenü einen Funktionsbaustein ein.



Ändern Sie den Namen des Bausteins auf "FB10", und wählen Sie die gewünschte Erstellsprache aus.

Aktivieren Sie, wenn erforderlich, die **Multiinstanzfähigkeit**, und übernehmen Sie die restlichen Voreinstellungen mit **OK**.

Der FB10 wurde dem Ordner Bausteine hinzugefügt. Öffnen Sie den **FB10** durch Doppelklick.

Multiinstanzen können Sie für beliebige Funktionsbausteine anlegen, z. B. auch für Ventilsteuerungen. Wenn Sie mit Multiinstanzen arbeiten wollen, beachten Sie bitte, dass sowohl der aufrufende als auch die aufgerufenen Funktionsbausteine multiinstanzfähig sind.

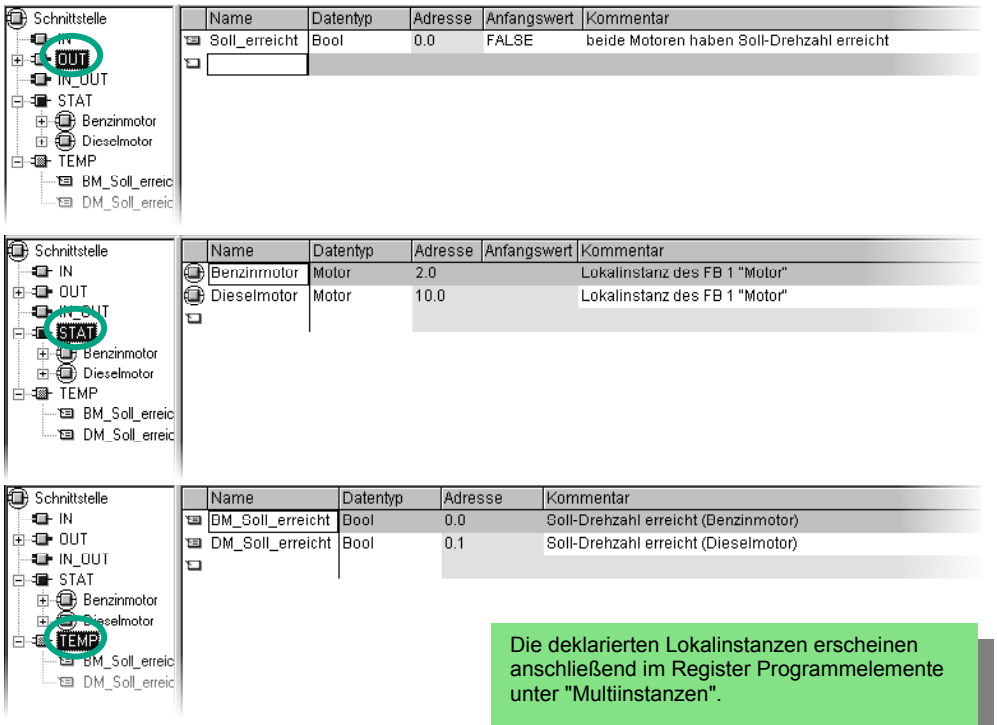
Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen" und "Anlegen von Bausteinen und Bibliotheken".

10.2 FB10 programmieren

Um den FB1 als "Lokale-Instanz" des FB10 aufzurufen, wird in der Variablendetailsicht für jeden geplanten Aufruf des FB1 eine statische Variable deklariert und ein anderer Name vorgegeben. Der Datentyp ist hierbei der FB1 ("Motor").

Variablen deklarieren / festlegen

Der FB 10 ist im KOP/AWL/FUP-Programmfenster geöffnet. Übertragen Sie die Deklarationen der nachfolgenden Abbildung in Ihre Variablendetailsicht. Wählen Sie hierzu nacheinander in der Variablenübersicht die Deklarationstypen "OUT", "STAT" und "TEMP" und tätigen Sie in der Variablendetailsicht Ihre Eintragungen. Wählen Sie beim Deklarationstyp "STAT" als Datentyp "FB <nr>" aus der Klappliste aus und ersetzen Sie die Zeichenfolge "<nr>" mit der Ziffer "1".

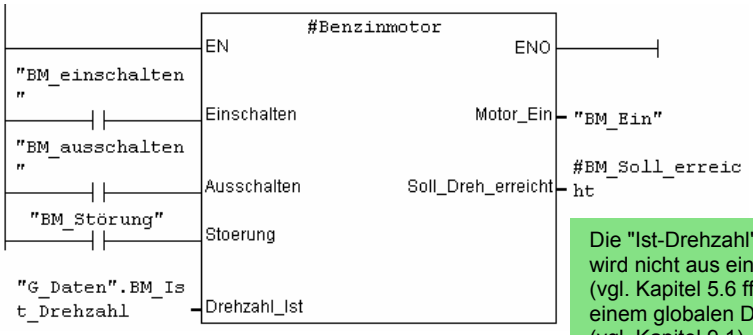


Die deklarierten Lokalinstanzen erscheinen anschließend im Register Programmelemente unter "Multiinstanzen".

FB10 programmieren in KOP

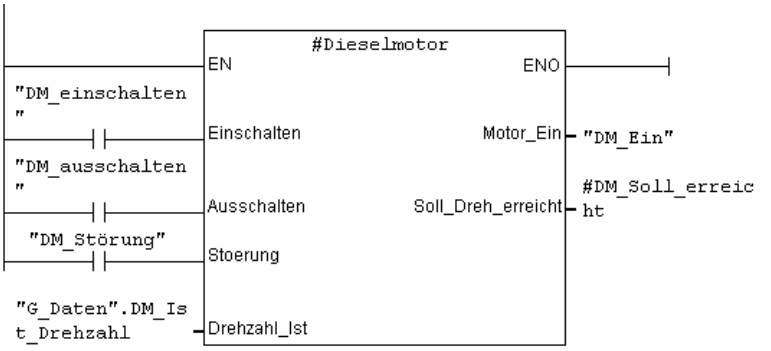
Fügen Sie den Aufruf "Benzinmotor" als Multiinstanzbaustein "Benzinmotor" im Netzwerk 1 ein.

Fügen Sie anschließend die benötigten Schließer ein, und vervollständigen Sie den Aufruf mit den symbolischen Namen.



Die "Ist-Drehzahl" für die Motoren wird nicht aus einem Merker geholt (vgl. Kapitel 5.6 ff.), sondern aus einem globalen Datenbaustein (vgl. Kapitel 9.1). Die allgemeine Adressierung lautet "Datenbaustein".Operand, z. B. "G_Daten".BM_Ist_Drehzahl.

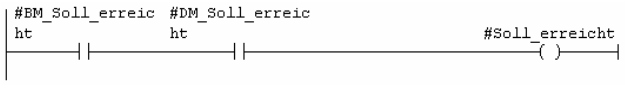
Fügen Sie ein neues Netzwerk ein, und programmieren Sie den Aufruf des Dieselmotors. Gehen Sie dabei analog zum Netzwerk 1 vor.



Fügen Sie ein neues Netzwerk ein, und programmieren Sie eine Reihenschaltung mit der entsprechenden Adressierung.

Verwenden Sie jeweils die temporären Variablen. Die temporären Variablen erkennen Sie in der Klappliste am links abgebildeten Symbol.

Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.



Die temporären Variablen ("BM_Soll_erreicht" und "DM_Soll_erreicht") werden an den Ausgangsparameter "Soll_erreicht" übergeben, der dann im OB1 weiterverarbeitet wird.

FB10 programmieren mit AWL

```
CALL  #Benzinmotor
 Einschalten        :="BM_einschalten"
 Ausschalten        :="BM_ausschalten"
 Stoerung           :="BM_Störung"
 Drehzahl_Ist       :="G_Daten".BM_Ist_Drehzahl
 Motor_Ein          :="BM_Ein"
 Soll_Dreh_erreicht:=#BM_Soll_erreicht

CALL  #Dieselmotor
 Einschalten        :="DM_einschalten"
 Ausschalten        :="DM_ausschalten"
 Stoerung           :="DM_Störung"
 Drehzahl_Ist       :="G_Daten".DM_Ist_Drehzahl
 Motor_Ein          :="DM_Ein"
 Soll_Dreh_erreicht:=#DM_Soll_erreicht

U     #BM_Soll_erreicht
U     #DM_Soll_erreicht
=     #Soll_erreicht
```

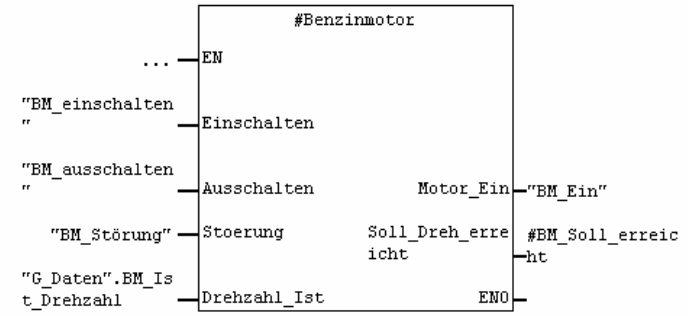
Falls Sie in AWL programmieren, markieren Sie jeweils den Eingabebereich unterhalb eines neuen Netzwerks, und geben Sie die nebenstehenden AWL-Anweisungen ein.

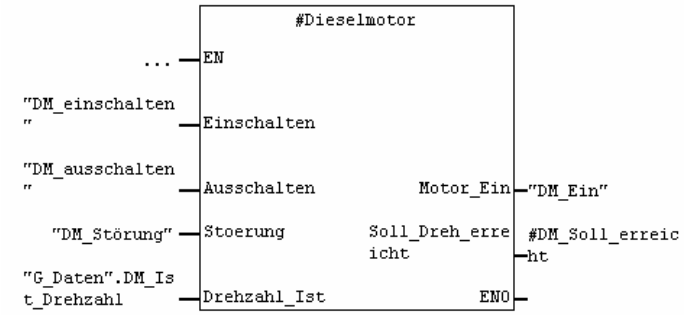
Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.

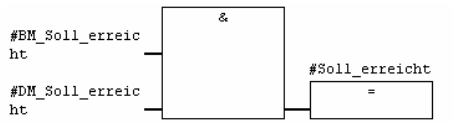
FB10 programmieren in FUP

Falls Sie in FUP programmieren, markieren Sie jeweils den Eingabebereich unterhalb eines neuen Netzwerks, und geben Sie die untenstehenden FUP-Anweisungen ein.

Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.







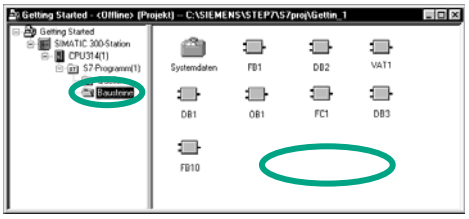
Um die beiden Aufrufe des FB1 im FB10 zu bearbeiten, muss der FB10 selbst aufgerufen werden.

Multiinstanzen können ausschließlich für Funktionsbausteine programmiert werden. Das Anlegen von Multiinstanzen für Funktionen (FCs) ist nicht möglich.

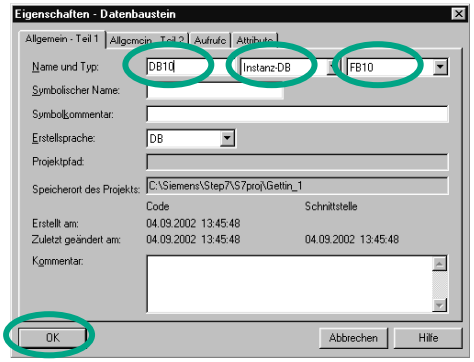
Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen", "Erstellen von Codebausteinen" und "Multiinstanzen in der Variablendeklaration".

10.3 DB10 erzeugen und Aktualwert anpassen

Der neue Datenbaustein DB10 wird die Datenbausteine DB1 und DB2 ersetzen. Im DB10 werden die Daten des Benzin- und Dieselmotors abgelegt, die später beim Aufruf des FB10 im OB1 benötigt werden (vgl. Aufruf des FB1 im OB1 in Kapitel 5.6 ff.).



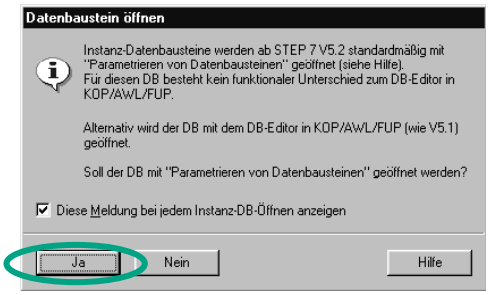
Erzeugen Sie im SIMATIC Manager Projekt "Getting Started", Ordner **Bausteine** mit dem Kontextmenü der rechten Maustaste den Datenbaustein DB10.



Ändern Sie hierzu im Dialogfeld "Eigenschaften - Datenbaustein" den Namen des Datenbausteins auf DB10, wählen Sie in der benachbarten Klappliste den Verwendungszweck "Instanz-DB". Wählen Sie in der rechten Klappliste den zuzuordnenden Funktionsbausteins "FB10" und bestätigen Sie alle Einstellungen mit **OK**.

Der Datenbaustein DB10 wird im Projekt "Getting Started" hinzugefügt.

Öffnen Sie den **DB10** mit Doppelklick.



Bestätigen Sie den nachfolgenden Dialog mit **Ja**, um den Instanz-Datenbaustein zu parametrieren. Wählen Sie den Menübefehl **Ansicht > Datensicht.**

Die Datensicht zeigt jede einzelne Variable im DB10 an, auch die "internen" Variablen der beiden Aufrufe des FB1 ("Lokalinstanzen").

Die Deklarationsicht zeigt die Variablen, wie Sie im FB10 deklariert sind.

Ändern Sie den Aktualwert des Dieselmotors auf "1300", speichern Sie den Datenbaustein, und schließen Sie ihn anschließend.

	Adresse	Deklaration	Name	Typ	Anfangswert	Aktualwert	Kommentar
1	0.0	out	Soll_erreicht	BOOL	FALSE	FALSE	beide Motoren haben Soll-Drehzahl erreicht
2	2.0	stat:in	Benzinmotor.Einschalten	BOOL	FALSE	FALSE	Motor einschalten
3	2.1	stat:in	Benzinmotor.Ausschalten	BOOL	FALSE	FALSE	Motor ausschalten
4	2.2	stat:in	Benzinmotor.Stoerung	BOOL	FALSE	FALSE	Motorstörung, führt zum Ausschalten
5	4.0	stat:in	Benzinmotor.Drehzahl_Ist	INT	0	0	tatsächliche Motordrehzahl
6	6.0	stat:out	Benzinmotor.Motor_Ein	BOOL	FALSE	FALSE	Motor wird eingeschaltet
7	6.1	stat:out	Benzinmotor.Soll_Dreh_erreicht	BOOL	FALSE	FALSE	Solldrehzahl erreicht
8	8.0	stat	Benzinmotor.Drehzahl_Soll	INT	1500	1500	geforderte Motordrehzahl
9	10.0	stat:in	Dieselmotor.Einschalten	BOOL	FALSE	FALSE	Motor einschalten
10	10.1	stat:in	Dieselmotor.Ausschalten	BOOL	FALSE	FALSE	Motor ausschalten
11	10.2	stat:in	Dieselmotor.Stoerung	BOOL	FALSE	FALSE	Motorstörung, führt zum Ausschalten
12	12.0	stat:in	Dieselmotor.Drehzahl_Ist	INT	0	0	tatsächliche Motordrehzahl
13	14.0	stat:out	Dieselmotor.Motor_Ein	BOOL	FALSE	FALSE	Motor wird eingeschaltet
14	14.1	stat:out	Dieselmotor.Soll_Dreh_erreicht	BOOL	FALSE	FALSE	Solldrehzahl erreicht
15	16.0	stat	Dieselmotor.Drehzahl_Soll	INT	1500	1300	geforderte Motordrehzahl

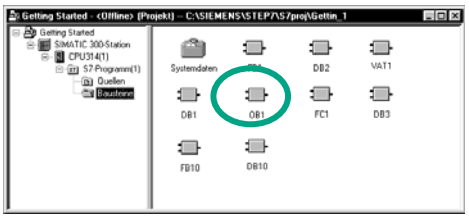
In der Variablendeklarationstabelle des DB10 sind nun sämtliche Variablen enthalten. Im ersten Teil der Tabelle sehen Sie die Variablen für den Aufruf des Funktionsbausteins "Benzinmotor" und im unteren Teil den Aufruf für den Funktionsbaustein "Dieselmotor" (vgl. Kapitel 5.5).

Die "internen" Variablen des FB1 behalten ihre symbolischen Namen, z. B. "Einschalten". Diesen wird nun der Name der Lokalinstanz vorangestellt, z. B. "Benzinmotor.Einschalten".

Mehr Informationen über **Hilfe > Hilfethemen** "Programmieren von Bausteinen", "Erstellen von Datenbausteinen".

10.4 Aufruf des FB10 im OB1

Der Aufruf des FB10 erfolgt in unserem Beispiel im OB1. Dieser Aufruf stellt die gleiche Funktionalität dar, wie Sie sie beim Programmieren und Aufrufen des FB1 im OB1 kennengelernt haben (vgl. Kapitel 5.6 ff.). Durch die Verwendung der Multiinstanz werden die im Kapitel 5.6 ff. programmierten Netzwerke 4 und 5 ersetzt.



Öffnen Sie den **OB1** in dem Projekt, in dem Sie gerade den FB10 programmiert haben.

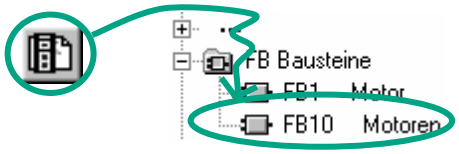
Symbolischen Namen definieren

Das KOP/AWL/FUP-Programmfenster ist geöffnet. Öffnen Sie die Symboltabelle mit **Extras > Symboltabelle**, und tragen Sie die symbolischen Namen für den Funktionsbaustein FB10 und den Datenbaustein DB10 in die Symboltabelle ein.

Speichern Sie anschließend die Symboltabelle, und schließen Sie das Fenster.

Symbol	Adresse	Datentyp	Kommentar
...	...	...	...
Motoren	FB 10	FB 10	Beispiel für Multiinstanzen
Motoren_Daten	DB 10	FB 10	Instanz-Datenbaustein für FB 10
...	...	...	...

Aufruf programmieren in KOP

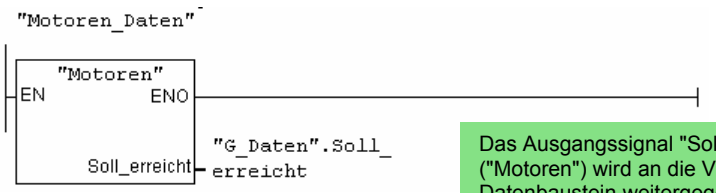


Fügen Sie am Ende des OB1 ein neues Netzwerk ein und ergänzen Sie dieses um den Aufruf des **FB10** ("Motoren").

Vervollständigen Sie den untenstehenden Aufruf mit den entsprechenden symbolischen Namen.

Löschen Sie den Aufruf des FB1 im OB1 (Netzwerke 4 und 5 aus Kapitel 5.6 ff), da nun der FB1 zentral über den FB10 aufgerufen wird.

Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.



Das Ausgangssignal "Soll_erreicht" des FB10 ("Motoren") wird an die Variable im globalen Datenbaustein weitergegeben.

Aufruf in AWL

Falls Sie in AWL programmieren, markieren Sie den Eingabebereich unterhalb des neuen Netzwerks, und geben Sie die untenstehenden AWL-Anweisungen ein. Benutzen Sie hierfür im Programmelemente-Katalog den **FB Bausteine > FB10 Motoren.**

Löschen Sie den Aufruf des FB1 im OB1 (Netzwerke 4 und 5 aus Kapitel 5.6 ff), da nun der FB1 zentral über den FB10 aufgerufen wird.

Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.

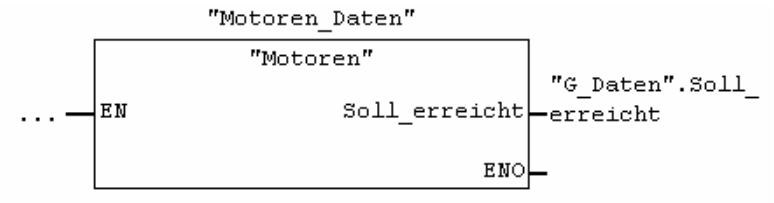
```
CALL  "Motoren" , "Motoren_Daten"
 Soll_erreicht:="G_Daten".Soll_erreicht
```

Aufruf in FUP

Falls Sie in FUP programmieren, markieren Sie den Eingabebereich unterhalb des neuen Netzwerks, und geben Sie die untenstehenden FUP-Anweisungen ein. Benutzen Sie hierfür im Programmelemente-Katalog den **FB Bausteine > FB10 Motoren.**

Löschen Sie den Aufruf des FB1 im OB1 (Netzwerke 4 und 5 aus Kapitel 5.6 ff), da nun der FB1 zentral über den FB10 aufgerufen wird.

Speichern Sie anschließend Ihr Programm, und schließen Sie den Baustein.



Benötigen Sie für Ihre Automatisierungslösung weitere Motorsteuerungen, z. B. Erdgasmotoren, Biogasmotoren usw., sind diese analog als Multiinstanz zu programmieren und über den FB10 aufzurufen.

Hierzu deklarieren Sie, wie dargestellt, in der Variablendeklarationstabelle des FB10 ("Motoren") die weiteren Motoren und programmieren im FB10 den Aufruf des FB1 (Multiinstanz im Programmelemente-Katalog). Für die symbolische Programmierung sind anschließend die neuen symbolischen Namen z. B. für die Ein-/Ausschaltvorgänge in der Symboltabelle zu definieren.

Mehr Informationen über **Hilfe > Hilfethemen** "Aufruf von Referenzhilfen", "Sprachbeschreibung KOP/FUP" bzw. "AWL".

11 Konfigurieren der Dezentralen Peripherie

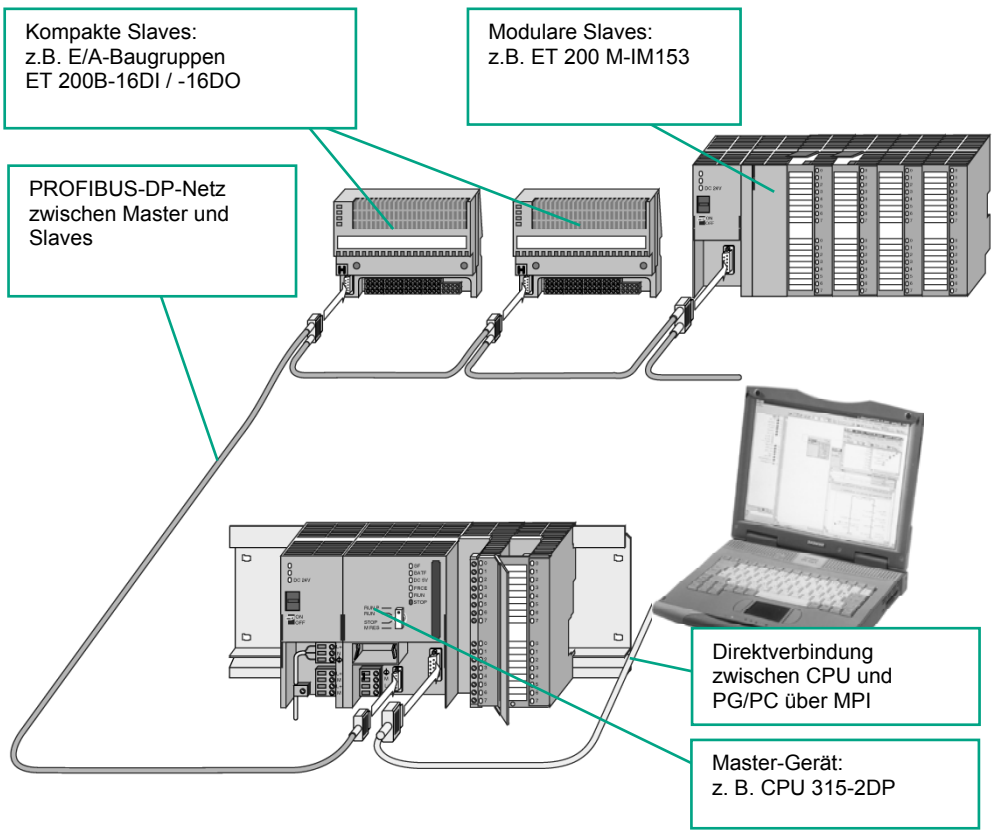
11.1 Dezentrale Peripherie mit PROFIBUS-DP aufbauen

Beim konventionellen Aufbau von Automatisierungsanlagen werden die Kabelverbindungen der Sensoren und Aktoren direkt in die Ein-/Ausgabebaugruppen des zentralen Automatisierungsgeräts gesteckt. Das führt häufig zu einem hohen Verdrahtungsaufwand.

Mit einem dezentralen Aufbau verringern Sie den Verdrahtungsaufwand erheblich indem Sie die Ein-/Ausgabebaugruppen in Nähe der Sensoren und Aktoren platzieren. Die Verbindung zwischen dem Automatisierungssystem, den Peripheriebaugruppen und Feldgeräten stellen Sie über den Feldbus PROFIBUS-DP her.

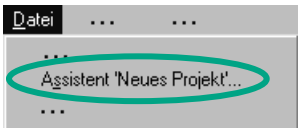
Die Programmierung des konventionellen Aufbaus konnten Sie im Kapitel 6 kennenlernen. Für das Konfigurieren des dezentralen Aufbaus ergeben sich keine Unterschiede im Vergleich zum Konfigurieren eines zentralen Aufbaus. Sie wählen die teilnehmenden Baugruppen aus dem Hardware-Katalog aus, ordnen diese an und passen deren Eigenschaften Ihren Anforderungen an.

Vorteilhaft wäre, wenn Sie bereits wissen, wie Sie ein Projekt anlegen und einen zentralen Aufbau konfigurieren (vgl. Kapitel 2.1 und 6).

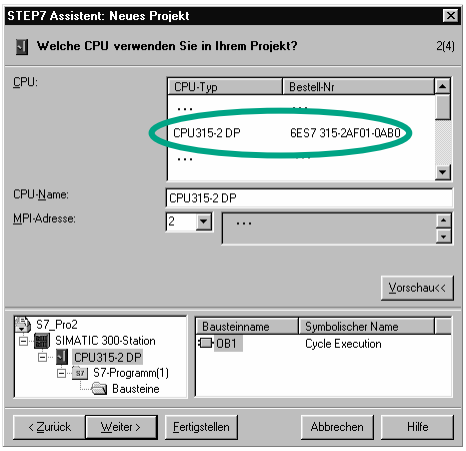


Ein neues Projekt anlegen

Ausgangspunkt ist der SIMATIC Manager. Schließen Sie wegen der besseren Übersichtlichkeit eventuell noch offene Projekte.



Legen Sie ein **neues Projekt** an.

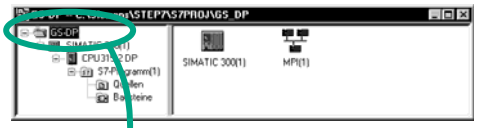


Wählen Sie im entsprechenden Dialogfeld die **CPU 315-2DP** aus (CPU mit PROFIBUS-DP-Netz).

Gehen Sie ansonsten analog zu Kapitel 2.1 vor, und geben Sie dem Projekt den Namen "GS-DP" (Getting Started - Dezentrale Peripherie).

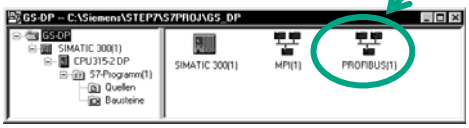
Falls Sie Ihre eigene Konfiguration gleich an dieser Stelle anlegen wollen, dann geben Sie bitte jetzt Ihre CPU an. Achten Sie darauf, dass diese DP-fähig ist.

Einfügen des PROFIBUS Netz

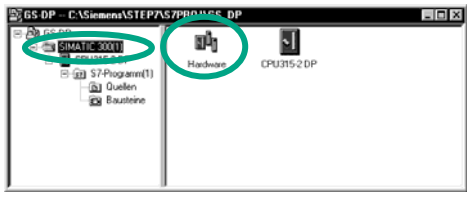


Markieren Sie den Ordner **GS-DP**.

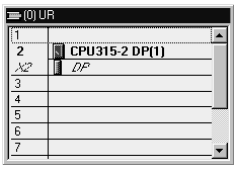
Fügen Sie über das Kontextmenü der rechten Maustaste das **PROFIBUS-Netz** ein.



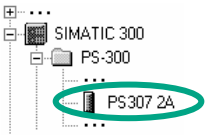
Station konfigurieren



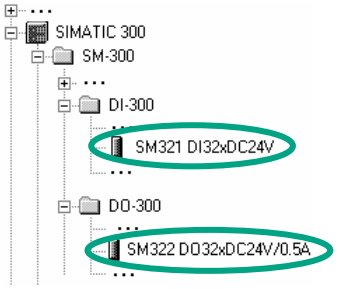
Markieren Sie den Ordner **SIMATIC 300-Station**, und doppelklicken Sie auf **Hardware**.
Das "HW Konfig"-Fenster wird geöffnet (vgl. Kapitel 6.1).



Die CPU 315-2DP ist bereits im Rack vorhanden. Falls notwendig, aktivieren Sie Hardware-Katalog über **Ansicht > Hardware-Katalog** oder den Button.



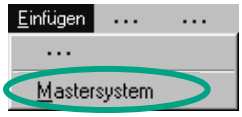
Fügen Sie auf dem Steckplatz 1 die Stromversorgungsbaugruppe **PS307 2A** mit Drag and Drop ein.



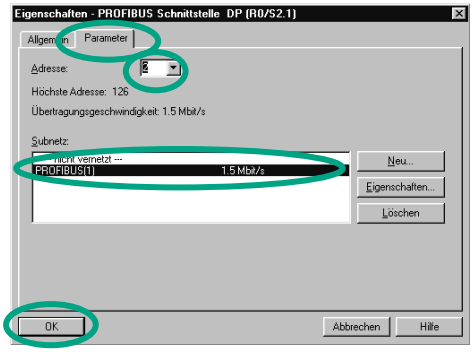
Fügen Sie auf den Steckplätzen 4 und 5 die Ein-/Ausgabebaugruppen **DI32xDC24V** und **DO32xDC24V/0.5A** ein.

Zusätzlich zur DP-fähigen CPU können Sie auch zentrale Baugruppen auf dem gleichen Rack platzieren (wird hier nicht durchgeführt).

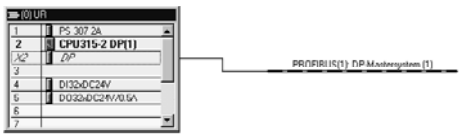
DP-Mastersystem konfigurieren



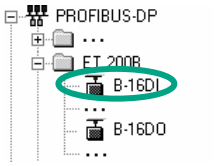
Markieren Sie den DP-Master auf Steckplatz X2, und fügen Sie ein **Mastersystem** ein.



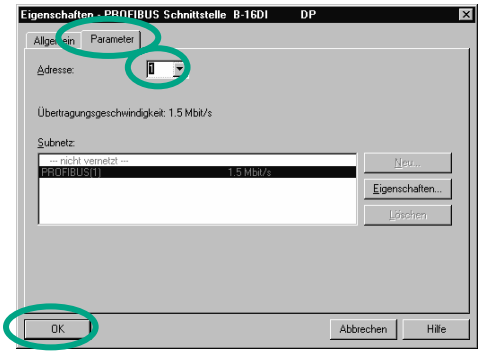
Übernehmen Sie im nun eingeblendeten Dialog die vorgeschlagene Adresse. Markieren Sie im Feld "Subnetz" den Eintrag "PROFIBUS(1)" und übernehmen Sie die Einstellungen mit **OK**.



Alle Objekte auf dem Mastersystem können Sie verschieben, indem Sie diese markieren und mit gedrückter Maustaste bewegen.

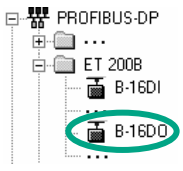


Navigieren Sie im Hardware-Katalog zur Baugruppe **B-16DI** und fügen Sie diese in das Mastersystem ein (mit Drag and Drop direkt auf das Mastersystem ziehen, Mauszeiger ändert sich, loslassen).



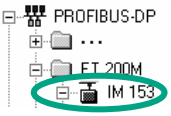
Im Dialogfeld "Objekteigenschaften" können Sie unter **Parameter** die Teilnehmer-Adresse der eingefügten Baugruppe ändern.

Bestätigen Sie die vorgeschlagene Adresse mit **OK**.



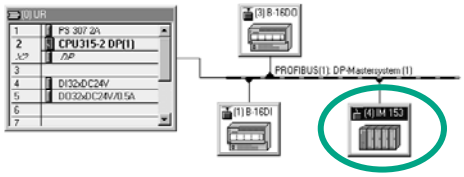
Ziehen Sie analog dazu die Baugruppe **B-16DO** auf das Mastersystem.

Im Dialogfeld wird die Teilnehmer-Adresse automatisch angepasst. Bestätigen Sie diese mit **OK**.

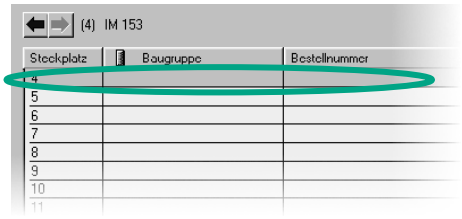


Ziehen Sie die Anschaltungsbaugruppe **IM153** auf das Mastersystem, und bestätigen Sie wieder die Teilnehmer-Adresse mit **OK**.

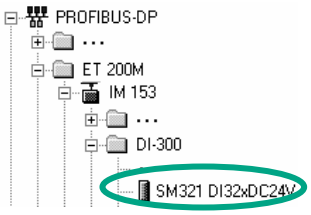
In unserem Beispiel übernehmen wir die voreingestellten Teilnehmer-Adressen. Diese Adressen können Sie jedoch jederzeit ändern und Ihren Anforderungen anpassen.



Markieren Sie das **ET200M** im Netz. In der unteren Konfigurationstabelle werden nun die freien Steckplätze der ET200M angezeigt.



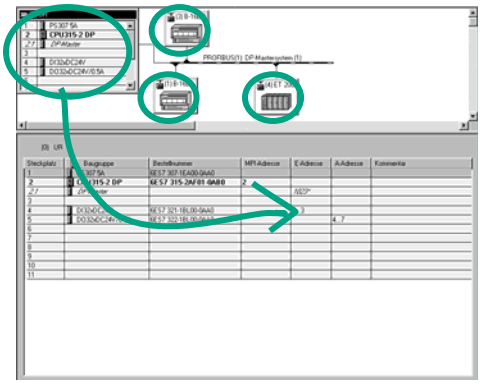
Markieren Sie dort den Steckplatz **4**.



Die ET200M selbst kann weitere Ein-/Ausgabebaugruppen aufnehmen. Wählen Sie beispielsweise die Baugruppe **DI32xDC24V** für den Steckplatz 4 aus und fügen Sie die Baugruppe mit Doppelklick ein.

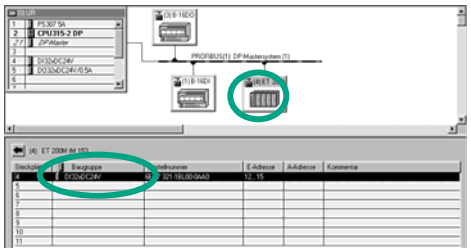
Achten Sie stets bei der Auswahl von Baugruppen darauf, dass Sie sich im Hardware-Katalog auch im richtigen Ordner befinden. Z.B. für die Auswahl von Baugruppen zur ET200M im Ordner ET200M.

Teilnehmer-Adresse ändern



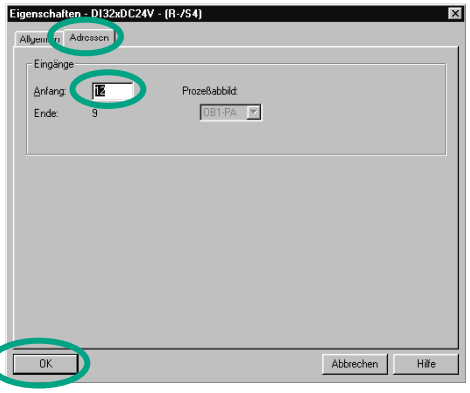
In unserem Beispiel ist das Ändern der Teilnehmer-Adresse nicht notwendig. In der Praxis wird dies jedoch häufig gebraucht.

Markieren Sie die anderen Teilnehmer der Reihe nach und überprüfen Sie die Ein- und Ausgangsadressen.
Die Hardware-Konfiguration hat alle Adressen angepasst, es gibt keine Doppelbelegung.

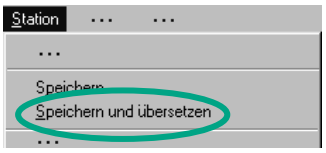


Angenommen Sie wollen die Adresse der ET200M ändern:

Markieren Sie die **ET200M** und doppelklicken Sie auf **DI32xDC24V** (Steckplatz 4).



Ändern Sie im Dialogfeld "Eigenschaften" unter **Adressen** nun die Eingangsadressen von 6 auf **12**.
Schließen Sie das Dialogfeld mit **OK**.



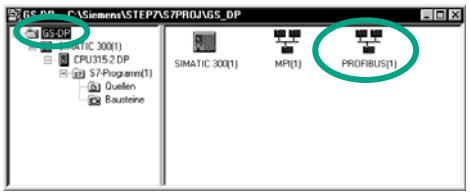
Zum Abschluss **Speichern und Übersetzen** Sie die Konfiguration der Dezentralen Peripherie.

Schließen Sie das Fenster.

Mit Speichern und Übersetzen wird die Konfiguration automatisch einer Konsistenzprüfung unterzogen. Bei fehlerfreier Konfiguration werden dann die Systemdaten erzeugt und können ins Zielsystem geladen werden.

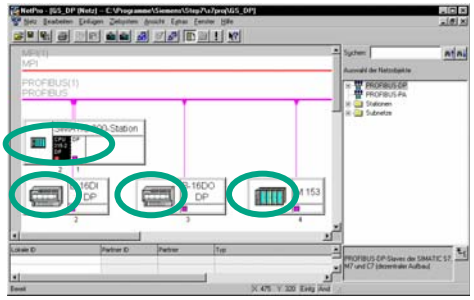
Mit Speichern kann die Konfiguration auch im fehlerhaften Zustand abgespeichert werden. Ein Laden ins Zielsystem ist dann nicht möglich.

Alternative: Netzprojektierung



Die Konfiguration der Dezentralen Peripherie können Sie auch mit der Netzprojektierung durchführen.

Doppelklicken Sie im SIMATIC Manager auf das Netz **PROFIBUS (1)**.



Das Fenster "NetPro" wird geöffnet.

Aus dem Katalog der Netzobjekte können Sie weitere DP-Slaves auf den PROFIBUS-DP ziehen.

Doppelklicken Sie auf ein beliebiges Element, um es zu konfigurieren. Das Fenster "Hardware konfigurieren" wird geöffnet.

Mit **Station > Konsistenz prüfen** (Fenster Hardware-Konfiguration) und **Netz > Konsistenz prüfen** (Fenster Netzprojektierung) können Sie die Konfiguration auf Fehler überprüfen, bevor Sie diese speichern. Bei eventuellen Fehlern werden diese von STEP 7 angezeigt und Lösungsmöglichkeiten angeboten.

Mehr Information über **Hilfe > Hilfethemen** "Konfigurieren der Hardware" und "Konfigurieren der Dezentralen Peripherie".

Gratulation! Sie haben das "Getting Started" durchgearbeitet und dabei wichtige Begriffe, Vorgehensweisen und Funktionalitäten von STEP 7 kennengelernt. Damit können Sie sich bereits jetzt an Ihr erstes Projekt wagen.

Für den Fall, dass Sie in zukünftigen Projekten nach bestimmten Funktionen suchen oder Bedienfolgen von STEP 7 vergessen haben, nutzen Sie unsere umfangreiche Hilfe zu STEP 7.

Wenn Sie Ihre STEP 7 Kenntnisse erweitern wollen, bieten wir hierfür spezielle Schulungen an. Ihr Siemens-Vertriebsberater hilft Ihnen gern weiter.

Bei Ihren Projekten wünschen wir Ihnen viel Erfolg!

Ihre Siemens AG

A. Anhang A

A.1 Übersicht der Beispielprojekte zum Getting Started

- **ZDt01_02_STEP7__AWL_1-10:**
  Die programmierten Kapitel 1 bis 10 inklusive Symboltabelle in der Programmiersprache AWL.
- **ZDt01_01_STEP7__AWL_1-9:**
  Die programmierten Kapitel 1 bis 9 inklusive Symboltabelle in der Programmiersprache AWL.
- **ZDt01_06_STEP7__KOP_1-10:**
  Die programmierten Kapitel 1 bis 10 inklusive Symboltabelle in der Programmiersprache KOP.
- **ZDt01_05_STEP7__KOP_1-9:**
  Die programmierten Kapitel 1 bis 9 inklusive Symboltabelle in der Programmiersprache KOP.
- **ZDt01_04_STEP7__FUP_1-10:**
  Die programmierten Kapitel 1 bis 10 inklusive Symboltabelle in der Programmiersprache FUP.
- **ZDt01_03_STEP7__FUP_1-9:**
  Die programmierten Kapitel 1 bis 9 inklusive Symboltabelle in der Programmiersprache FUP.
- **ZDt01_07_STEP7__DezP_11:**
  Das programmierte Kapitel 11 mit der Dezentralen Peripherie.

Stichwortverzeichnis