Ausbildungsunterlage für die durchgängige Automatisierungslösung Totally Integrated Automation (T I A)

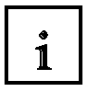
MODUL B5

Strukturierte Programmierung mit Funktionsbausteinen

Wir danken der Fa. Michael Dziallas Engineering und den Lehrkräften von beruflichen Schulen sowie weiteren Personen für die Unterstützung bei der Erstellung der Unterlage

SEITE:

Die folgenden Symbole führen durch dieses Modul:

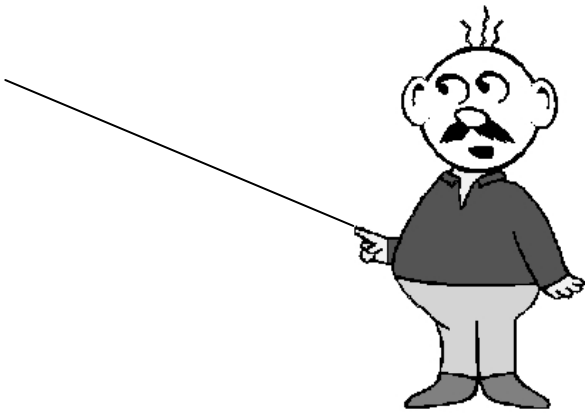


Information

Programmierung

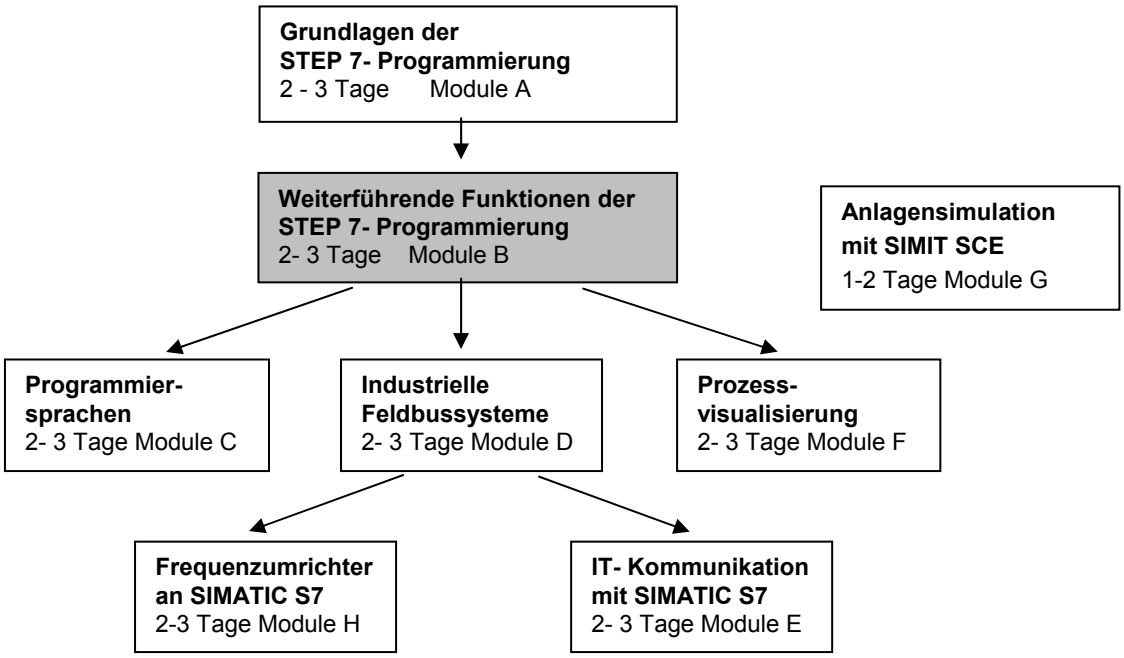
Beispielaufgabe

Hinweise



1. VORWORT

Das Modul B5 ist inhaltlich der Lehreinheit **‚Weiterführende Funktionen der STEP 7-Programmierung'** zugeordnet.



Lernziel:

Der Leser soll in den folgenden Schritten lernen wie ein Funktionsbaustein mit internen Variablen für die strukturierte Programmierung erstellt wird.

- Funktionsbaustein erstellen
- Interne Variablen definieren
- Programmieren mit internen Variablen im Funktionsbaustein
- Aufruf und Parametrisierung des Funktionsbausteins im OB1

Voraussetzungen:

Für die erfolgreiche Bearbeitung dieses Moduls wird folgendes Wissen vorausgesetzt:

- Kenntnisse in der Handhabung von Windows
- Grundlagen der SPS- Programmierung mit STEP 7 (z.B. Modul A3 - ‚Startup' SPS- Programmierung mit STEP 7)
- Grundlagen zur strukturierten Programmierung (z.B. Anhang I - Grundlagen zur SPS – Programmierung mit SIMATIC S7-300)

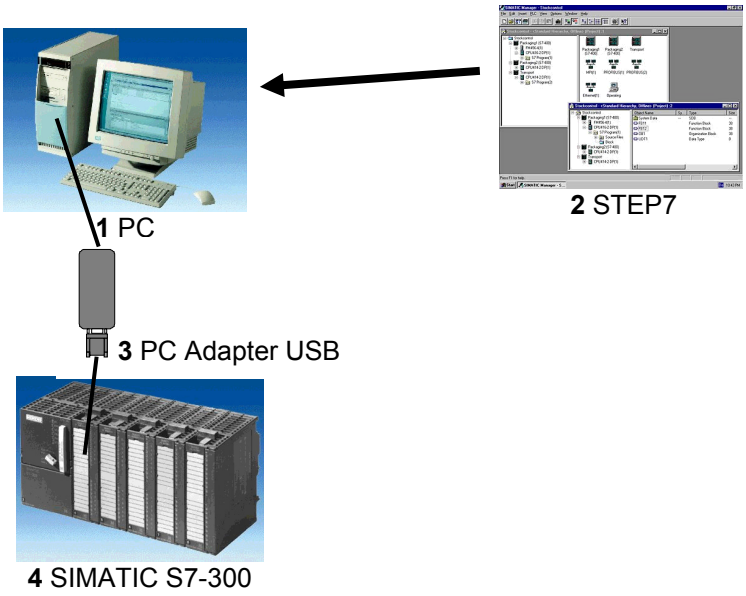
Benötigte Hardware und Software

1 PC, Betriebssystem Windows XP Professional mit SP2 oder SP3 / Vista 32 Bit Ultimate und Business / Server 2003 SP2 mit 600MHz (nur XP) / 1 GHz und 512MB (nur XP) / 1 GB RAM, freier Plattenspeicher ca. 650 - 900 MB, MS-Internet-Explorer 6.0 und Netzwerkkarte

2 Software STEP7 V 5.4

3 MPI- Schnittstelle für den PC (z.B. PC Adapter USB)

4 SPS SIMATIC S7-300 mit mindestens einer digitalen Ein- und Ausgabebaugruppe. Die Eingänge müssen auf ein Schaltfeld herausgeführt sein.
Beispielkonfiguration:
- Netzteil: PS 307 2A
- CPU: CPU 314
- Digitale Eingänge: DI 16x DC24V
- Digitale Ausgänge: DO 16x DC24V / 0,5 A



2. HINWEISE ZUR STRUKTURIERTEN PROGRAMMIERUNG MIT FCS UND FBS

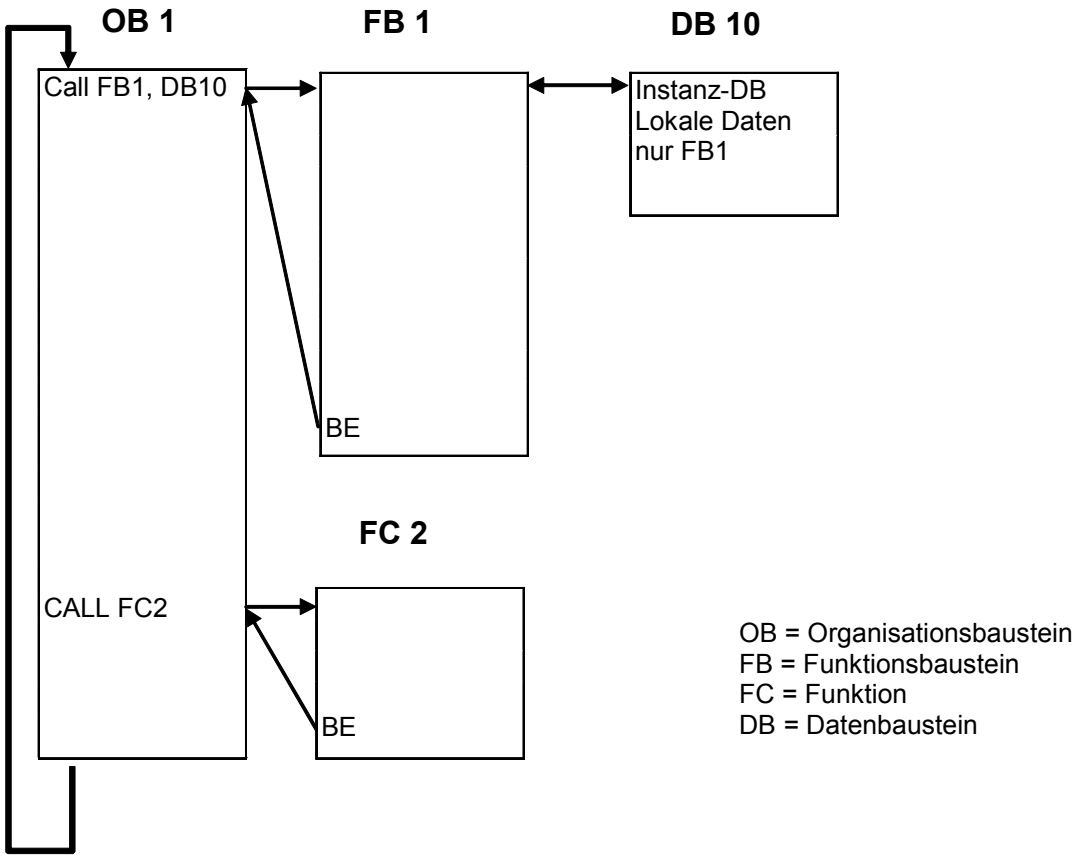
Der Programmablauf wird in STEP 7 in sogenannten Bausteinen geschrieben. Standardmäßig ist bereits der Organisationsbaustein OB1 vorhanden.
Dieser stellt die Schnittstelle zum Betriebssystem der CPU dar und wird automatisch von diesem aufgerufen und zyklisch bearbeitet.
Bei umfangreichen Steuerungsaufgaben unterteilt man das Programm in kleine, überschaubare und nach Funktionen geordnete Programmbausteine.
Diese Bausteine werden dann vom Organisationsbaustein aus über Bausteinaufrufbefehle (Call xx / UC xx / CC xx) aufgerufen. Ist Bausteinende erkannt worden, wird das Programm in dem aufrufenden Baustein hinter dem Aufruf weiterbearbeitet.

STEP 7 bietet für diese strukturierte Programmierung folgende Anwenderbausteine an:

- FB (Funktionsbaustein):
  der FB verfügt über einen zugeordneten Speicherbereich. Wenn ein FB aufgerufen wird kann ihm ein Datenbaustein (DB) zugeordnet werden. Auf die Daten in diesem lokalen Instanz- DB kann über die Aufrufe des FBs zugegriffen werden. Ein FB kann verschiedenen DBs zugeordnet werden. Weitere FBs und FCs können über Bausteinaufrufbefehle in einem Funktionsbaustein aufgerufen werden.

- FC (Funktion):
  eine FC besitzt keinen zugeordneten Speicherbereich. Die lokalen Daten einer Funktion sind nach der Bearbeitung der Funktion verloren. Weitere FBs und FCs können über Bausteinaufrufbefehle in einer Funktion aufgerufen werden.

Die Struktur eines solchen Programms könnte folgendermaßen aussehen:



Hinweis: Um solche Anwenderbausteine, die einmal erstellt wurden, mehrmals zu nutzen gibt es noch die Möglichkeit diese FCs und FBs in Form von Standardbausteinen unter Verwendung von internen Variablen zu Programmieren. Dann können diese beliebig oft aufgerufen werden, wobei den FBs jedes Mal noch ein anderer lokaler Instanz- DB zugewiesen werden muss.

3. FUNKTIONSBAUSTEIN MIT VARIABLENDEKLARATION ERSTELLEN

Wenn mit STEP 7 Bausteine erstellt werden sollen die quasi als „Black-Box“ in beliebigen Programmen funktionieren, so müssen diese unter Verwendung von Variablen programmiert werden. Dabei gilt die Regel, dass in diesen Bausteinen keine absolut adressierten Ein-/Ausgänge, Merker, Zeiten, Zähler etc. verwendet werden dürfen. Hier kommen lediglich Variablen und Konstanten zum Einsatz.

In dem folgenden Beispiel soll ein Funktionsbaustein mit Variablendeklaration erstellt werden der zum einen eine betriebsartenabhängige Bandansteuerung und zusätzlich noch einen Zykluszähler enthält.

Dabei soll mit dem Taster 'S0' die Betriebsart ‚Manuell’ und mit dem Taster 'S1' die Betriebsart ‚Automatik’ angewählt werden können.
In der Betriebsart ‚Manuell’ wird der Motor solange eingeschaltet solange der Taster ‚S2’ betätigt ist wobei der Taster ‚S3’ nicht betätigt sein darf.
In der Betriebsart ‚Automatik’ soll mit dem Taster 'S2' der Bandmotor eingeschaltet und mit dem Taster 'S3' der Bandmotor ausgeschaltet werden können.

In einem Merkerdoppelwort sollen die durchlaufenen Programmzyklen gezählt werden.
Das Beispiel bezieht sich auf die unten dargestellten Adressen:

Eingänge:
- Taster Betriebsart Manuell S0 = E 0.0
- Taster Betriebsart Automatik S1 = E 0.1
- Ein-Taster S2 = E 0.2
- Aus-Taster S3 = E 0.3

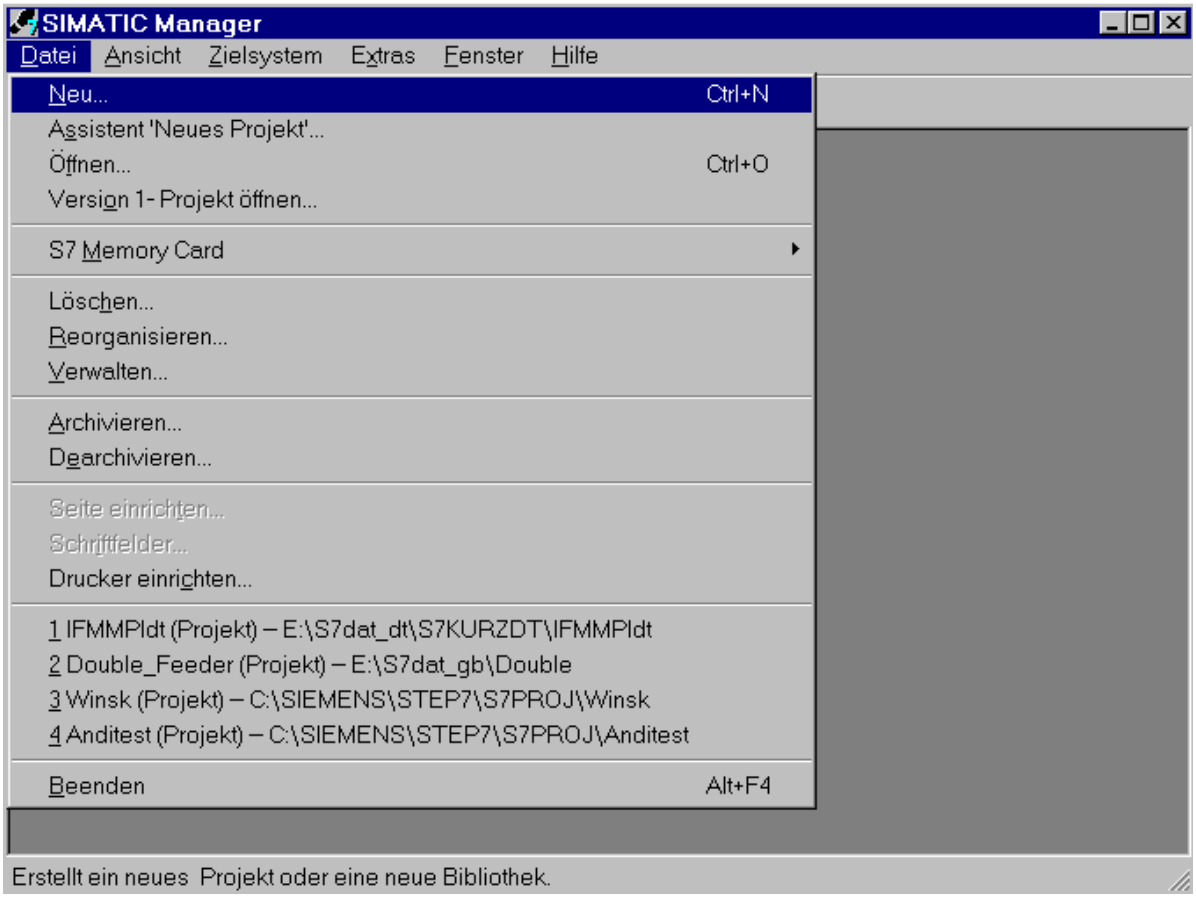
Ausgänge:
- Bandmotor = A 4.0

Merker:
- Zykluszähler = MD20

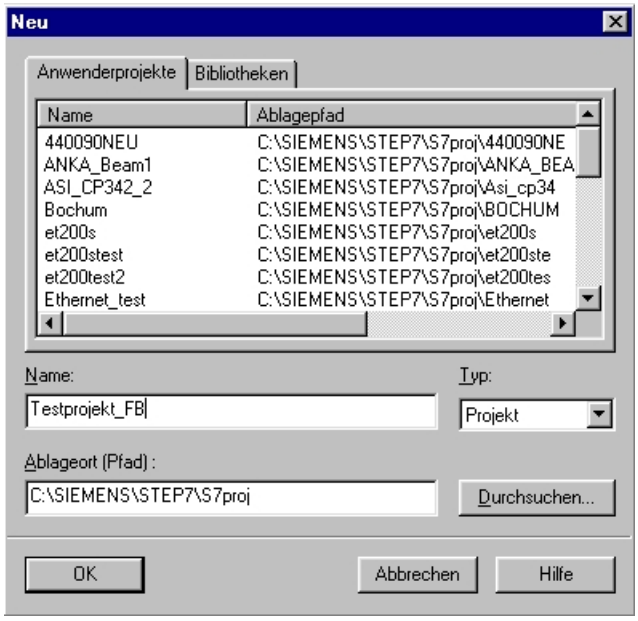
Zum Erstellen dieses Programmbeispiels müssen die folgenden Schritte durchgeführt werden (Dabei wird auf die Erstellung einer Hardwarekonfiguration verzichtet.):

1. **‚SIMATIC Manager'** durch einen Doppelklick aufrufen (→ SIMATIC Manager)

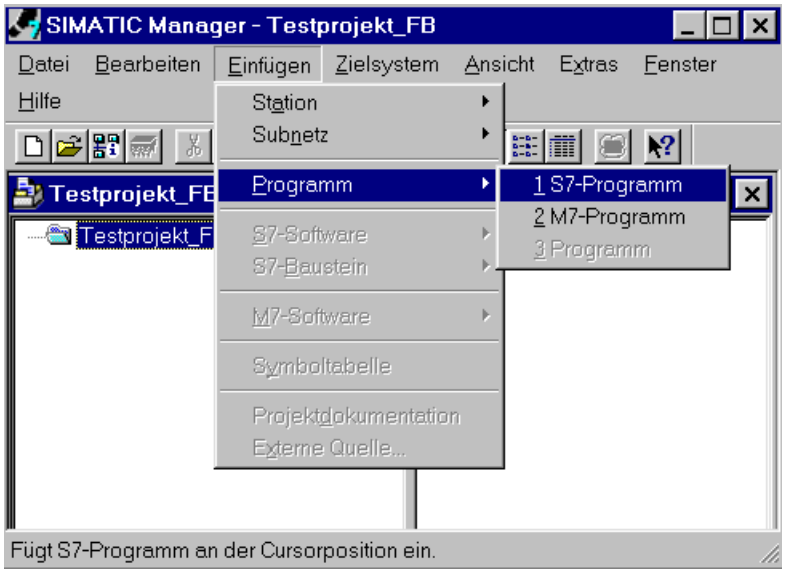
2. Neues Projekt anlegen (→ Datei → Neu)



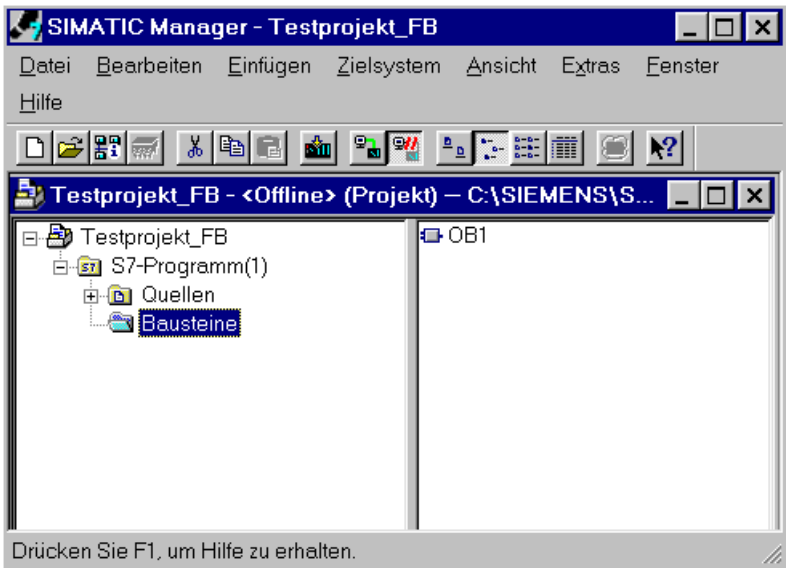
3. Neues Projekt erstellen, Pfad wählen und Projektname **‚Testprojekt_FB'** vergeben (→ Testprojekt_FB)



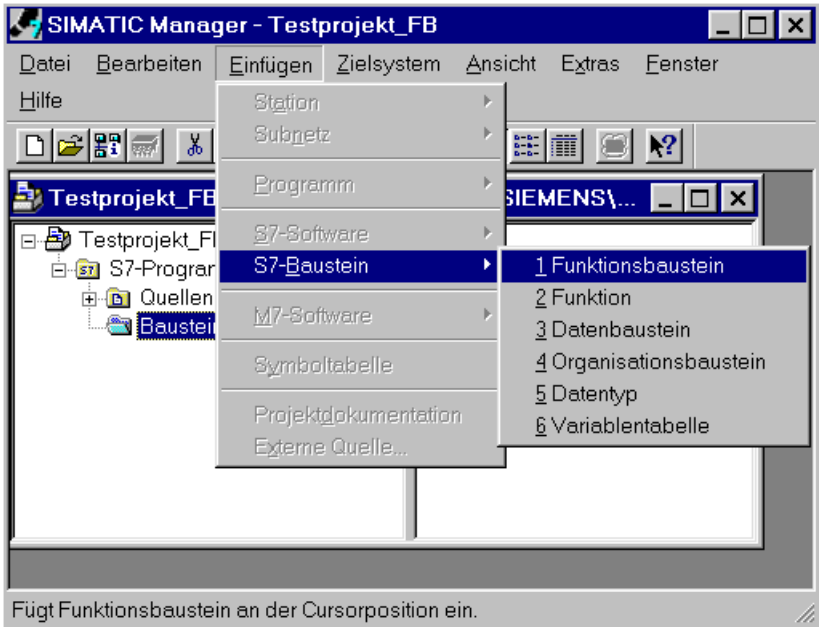
4. Neues **‚S7-Programm'** einfügen (→ Einfügen → Programm → S7-Programm).



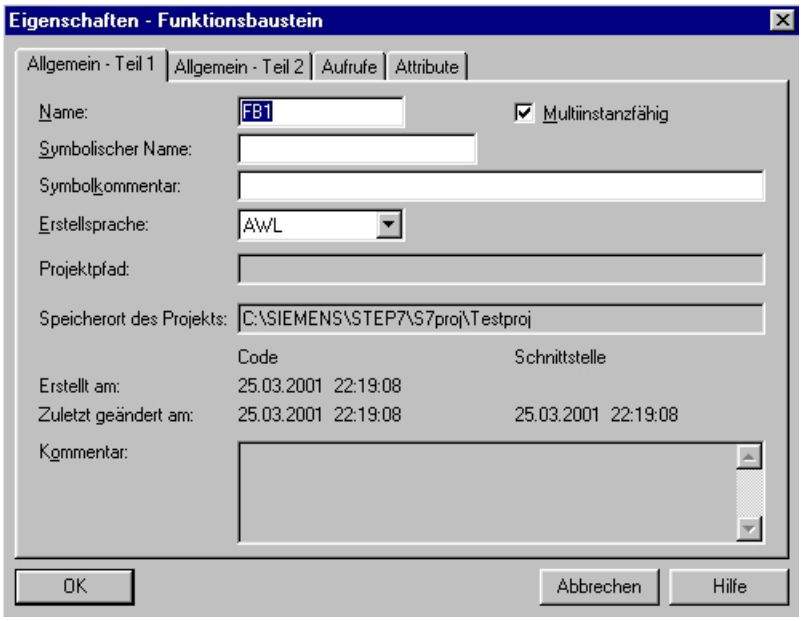
5. Ordner **‚Bausteine'** markieren (→ Bausteine).



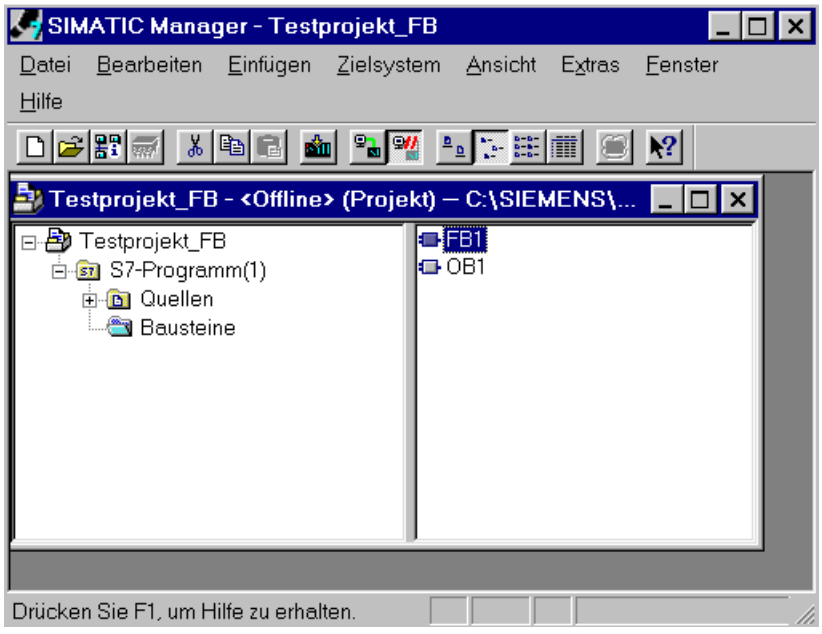
6. **‚Funktionsbaustein'** einfügen (→ Einfügen → S7-Baustein → Funktionsbaustein).



7. Name des FBs mit **‚FB1'** eingeben und Eigenschaften mit **‚OK'** übernehmen. (→ FB1 → OK).



8. Funktionsbaustein **‚FB1'** mit Doppellklick öffnen. (→ FB1)



9. Mit **'KOP, AWL, FUP- S7 Bausteine programmieren'** haben Sie jetzt einen Editor, der Ihnen die Möglichkeit gibt Ihren Funktionsbaustein zu editieren.
   Dazu müssen in der Variablendeklarationstabelle, die oben im FB1 angezeigt wird, Variablen definiert und deren Namen festgelegt werden.
   Diese Variablen gibt es vom Typ 'in', 'out', 'in_out', 'stat' und 'temp'.

Eingangsparameter (IN) nur in FBs, FCs, SFBs und SFCs
Mit Hilfe der Eingangsparameter werden Daten zur Verarbeitung an den Baustein übergeben.

Ausgangsparameter (OUT) nur in FBs, FCs, SFBs und SFCs
Mit dem Ausgangsparameter werden Ergebnisse an den aufrufenden Baustein übergeben.

Durchgangsparameter (IN_OUT) nur in FBs, FCs, SFBs und SFCs
Mit Durchgangsparametern werden Daten an den aufgerufenen Baustein übergeben, dort verarbeitet, und die Ergebnisse vom aufgerufenen Baustein wieder in der gleichen Variablen abgelegt.

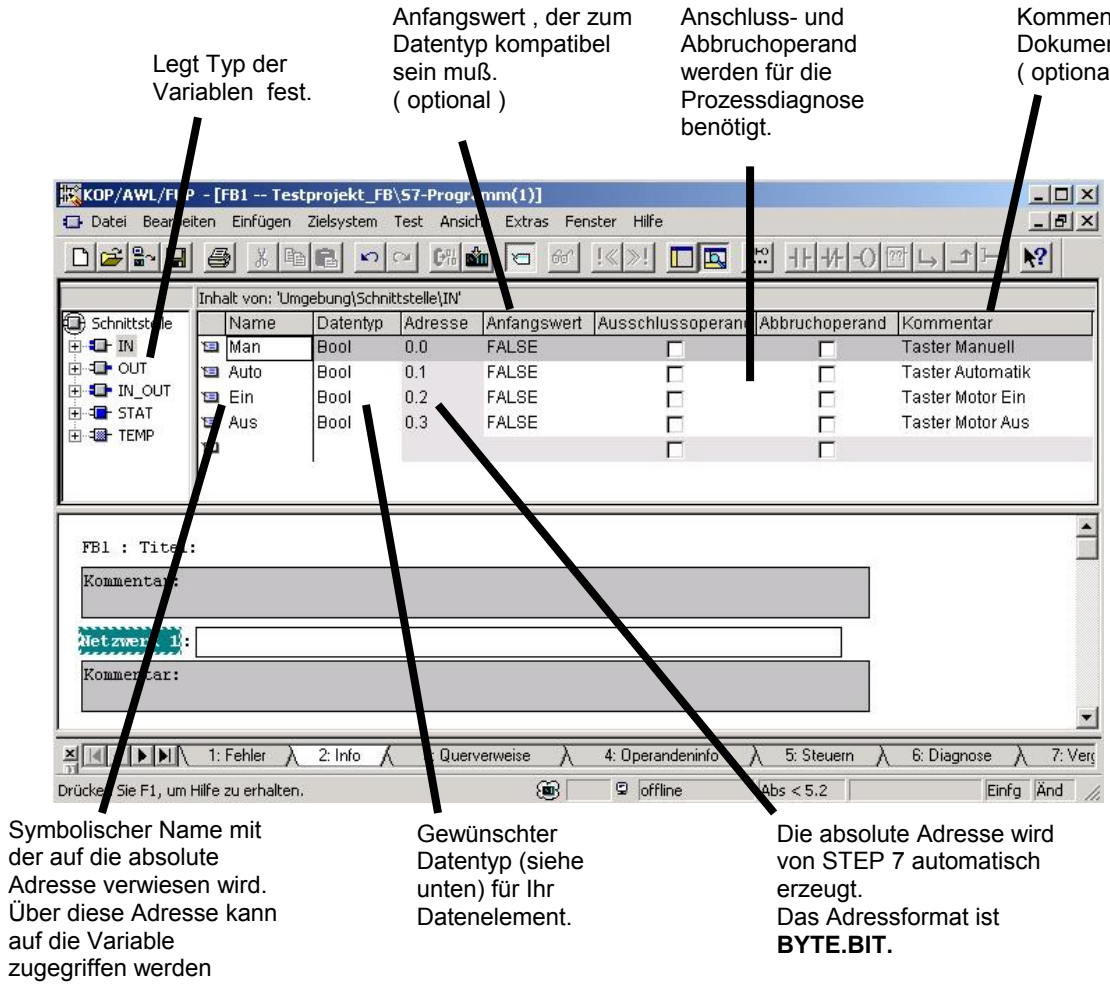
Statische Daten (STAT) nur in FBs und SFBs
Statische Daten sind Lokaldaten eines Funktionsbausteins, die im Instanz-Datenbaustein gespeichert werden und deshalb bis zur nächsten Bearbeitung des Funktionsbausteins erhalten bleiben.

Temporäre Daten (TEMP) in allen Bausteinen
Temporäre Daten sind Lokaldaten eines Bausteins, die während der Bearbeitung eines Bausteins im Lokaldaten-Stack (L-Stack) abgelegt werden und nach der Bearbeitung nicht mehr verfügbar sind.

Rückgabeparameter (RETURN) nur in FCs und SFBs
Beinhaltet den Rückgabewert (RET_VAL) einer Funktion.

Hinweis: Hier liegt auch der Unterschied zwischen FB/SFB und FC/SFC. Im FC stehen keine statischen Variablen **‚stat'** zur Verfügung, da es hier keinen Speicher für den Erhalt von Variableninhalten nach der Bearbeitung des FC gibt. Im FB werden diese statischen Variablen in dem dazugehörigen lokalen Instanz- DB bis zur nächsten Bearbeitung des FB zwischengespeichert.
Aus diesem Grund eignet sich auch nur der FB für die Erstellung von Programmen, in denen Daten wie z.B. Schrittmerker über mehrere Programmzyklen hinweg erhalten bleiben sollen.

Diese Festlegung der Variablen erfolgt indem zuerst ein Name vergeben, der Datentyp festgelegt und wahlweise ein Anfangswert und ein Kommentar eingegeben wird. Dies kann für die **‚IN'**-Variablen in dem Beispiel folgendermaßen aussehen:

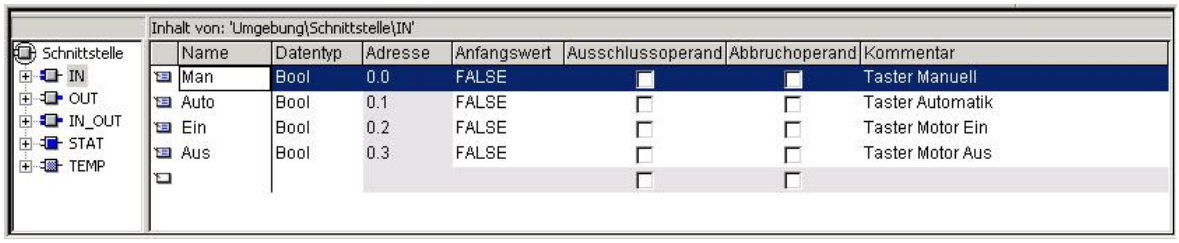


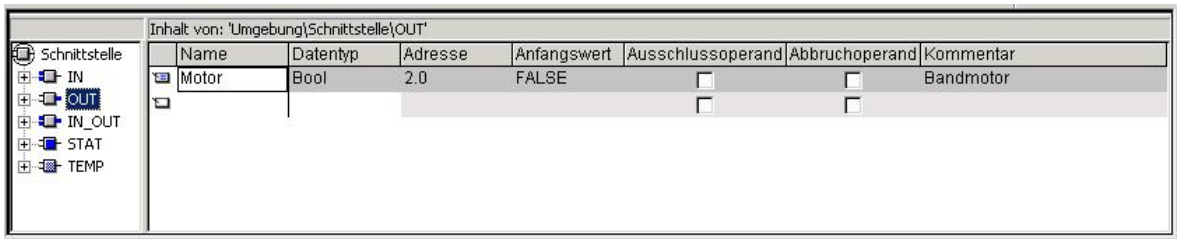
Hinweis: In der Deklaration wird von jedem zur Wahl stehenden Variablentypen eine angezeigt. Bei FCs also Variablen vom Typ **‚in', 'out'**, **'in_out'** und **'temp'** und bei FBs Variablen vom Typ **‚in', 'out', 'in_out', 'stat'** und **'temp'**. Wurde eine Variable vollständig angelegt, so erscheint automatisch eine weitere leere Zeile mit diesem Variablentyp.

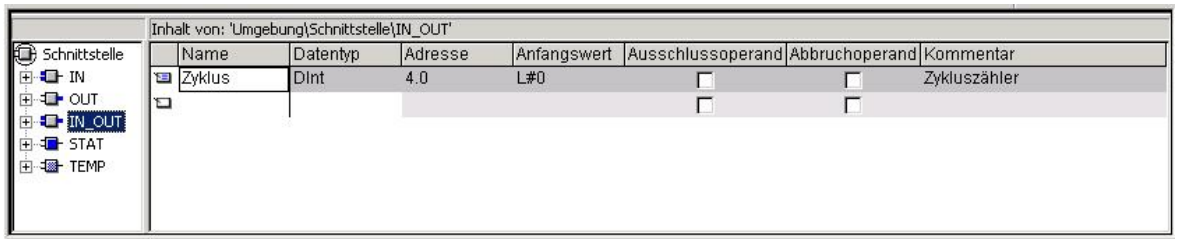
Daten in einer Variablendeklarationstabelle müssen wie in einem DB durch Datentypen bestimmt werden.
Folgende Standard- Datentypen sind unter anderen in der S7 definiert :

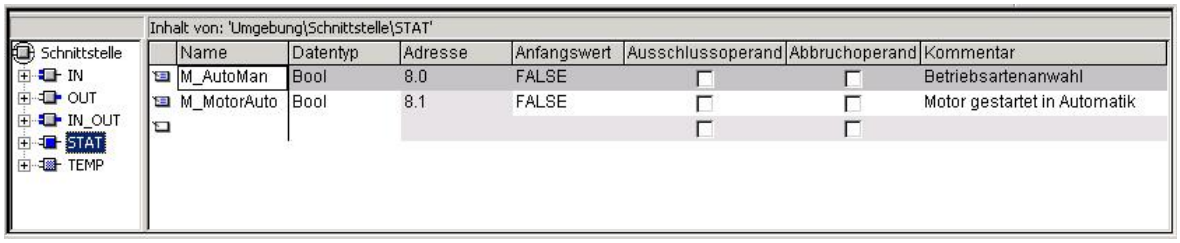
Typ und Beschreibung	Größe in Bits	Format-option	Bereich und Zahlendarstellung niedrigster bis höchster Wert	Beispiel
BOOL (Bit)	1	Bool-Text	TRUE/FALSE	TRUE
BYTE (Byte)	8	Hexadezimal	B#16#0 bis B#16#FF	B#16#10
WORD (Wort)	16	Dualzahl	2#0 bis 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Hexadezimalzahl	W#16#0 bis W#16#FFFF	W#16#1000
		BCD	C#0 bis C#999	C#998
		Dezimalzahl (o.V.)	B#(0,0) bis B#(255,255)	B#(10,20)
DWORD (Doppelwort)	32	Dualzahl	2#0 bis 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Hexadezimalzahl	DW#16#0000_0000 bis DW#16#FFFF_FFFF	DW#16#00A2_1234
		Dezimalzahl (o.V.)	B#(0,0,0,0) bis B#(255,255,255,255)	B#(1,14,100,120)
INT (Ganzzahl)	16	Dezimalzahl	-32768 bis 32767	1
DINT (Ganzzahl,32 bit	32	Dezimalzahl	L#-2147483648 bis L#2147483647	L#1
REAL (Gleitpunktzahl)	32	IEEE Gleitpunktzahl	Oberer Grenze: +/-3.402823e+38 Untere Grenze: +/-1.175495e-38	1.234567e+13
S5TIME (Simatic-Zeit)	16	S7-Zeit in Schritten von 10 ms	S5T#0H_0M_0S_10MS bis S5T#2H_46M_30S_0MS und S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (IEC-Zeit)	32	IEC-Zeit in Schritten von 1ms, Ganzzahl mit Vorzeichen	-T#24D_20H_31M_23S_648MS bis T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (IEC-Datum)	16	IEC-Datum in Schritten von 1 Tag	D#1990-1-1 bis D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Uhrzeit)	32	Uhrzeit in Schritten von 1ms	TOD#0:0:0.0 bis TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Zeichen)	8	ASCII-Zeichen	´A´, ´B´ usw.	´B´

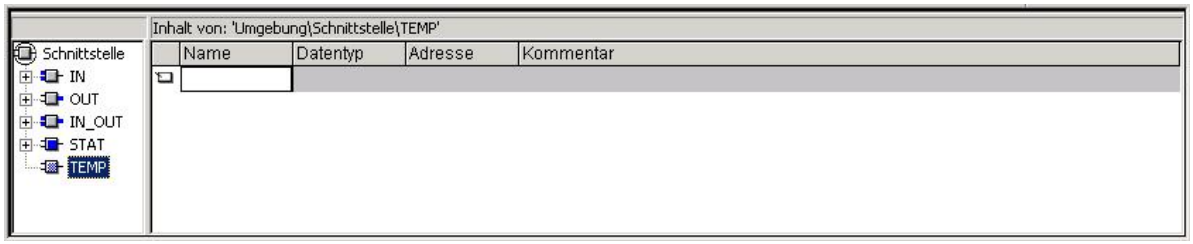
Die Deklaration für die **‚IN'** ,**OUT', ‚IN_OUT', ‚STAT'** und **‚TEMP'-** Variablen in dem Beispiel könnten folgendermaßen aussehen:





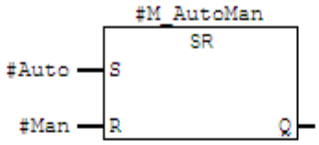




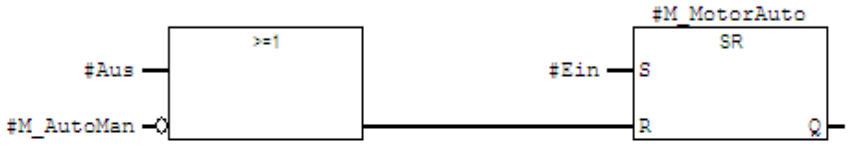


10. Nun kann unter Verwendung der Variablennamen das Programm eingegeben werden. (Variablen werden dabei durch das Symbol ‚#' gekennzeichnet.) Dies könnte für das Beispiel in FUP folgendermaßen aussehen. Dann soll der Funktionsbaustein FB1 gespeichert und in die CPU geladen werden. Der Schlüsselschalter der CPU steht dabei auf Stop! (→ →)

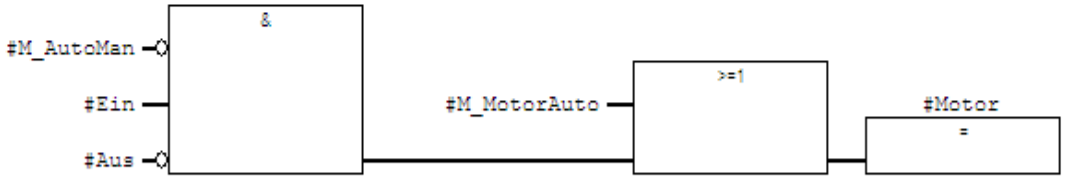
```
FB1 : Bandansteuerung mit Zykluszähler
Netzwerk 1 : Merker Betriebsart Automatik/Manuell angewählt
```



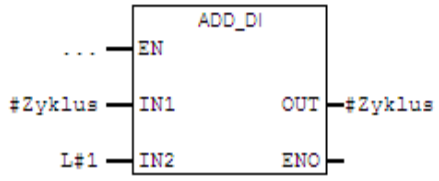

```
Netzwerk 2 : Ansteuerung Bandmotor im Automatik
```



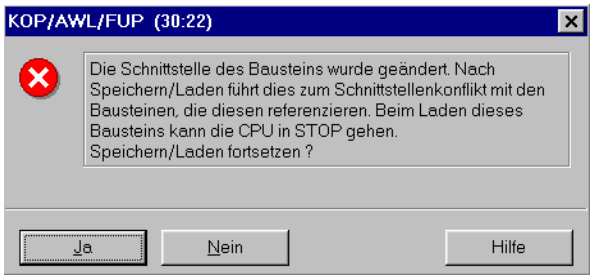

```
Netzwerk 3 : Ansteuerung Bandmotor im Manuell
```



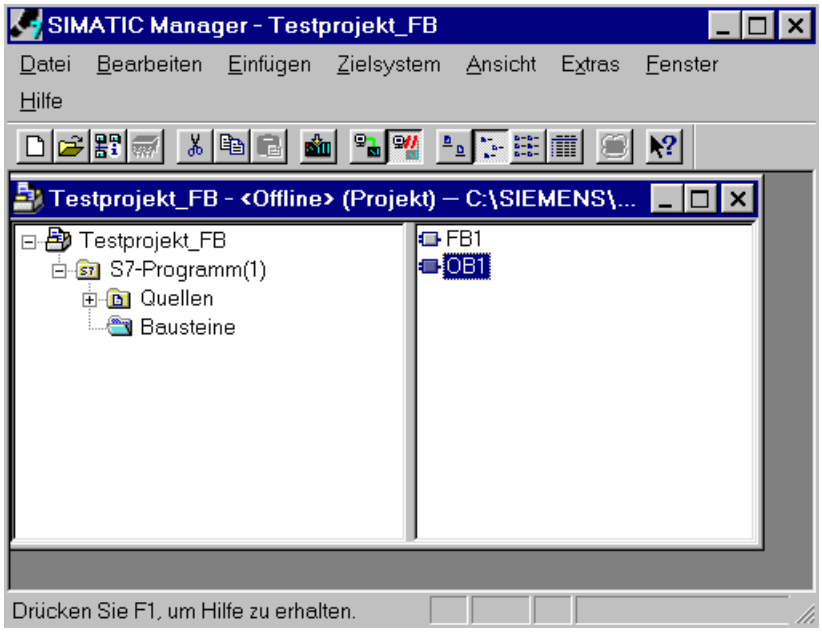

```
Netzwerk 4 : Zykluszähler  erhöht die Variable #Zyklus bei jedem Zyklus um 1
```



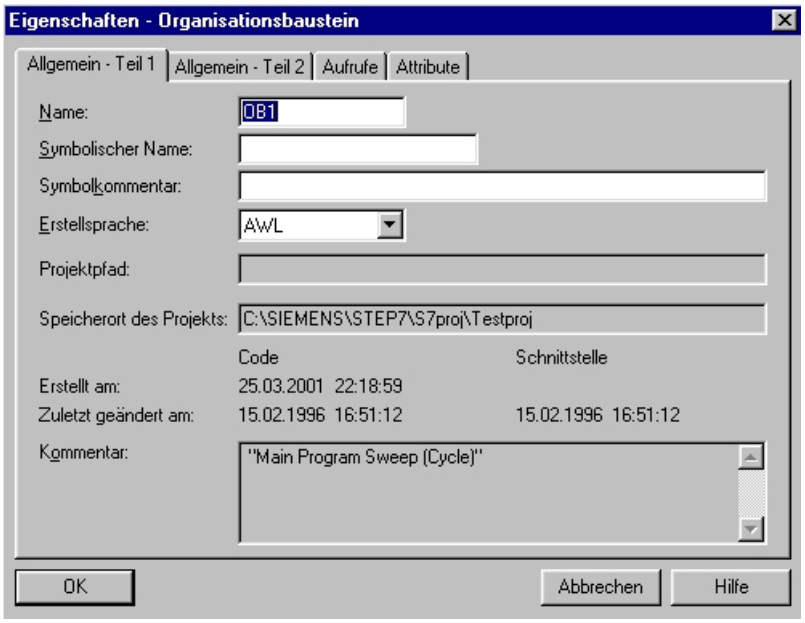
11. Die Warnung weist darauf hin, dass jeder bereits bestehende Aufruf dieses Bausteins FB1 neu Programmiert werden muss. Mit **‚Ja'** bestätigen. (→ Ja).



12. Im **‚SIMATIC Manager'** kann nun der **‚OB1'** geöffnet werden um den Aufruf des FB1 zu programmieren. (→ OB1)

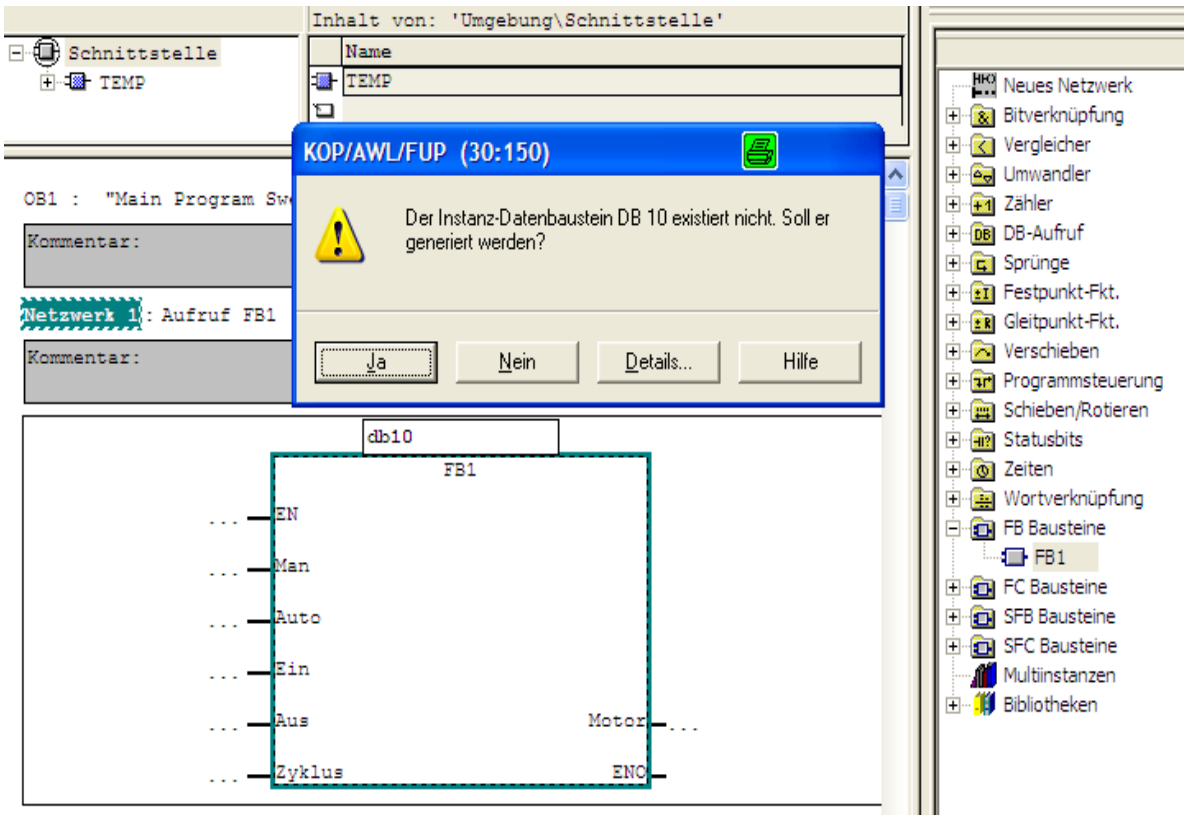


13. Einstellungen übernehmen und mit **‚OK'** bestätigen(→ OK).



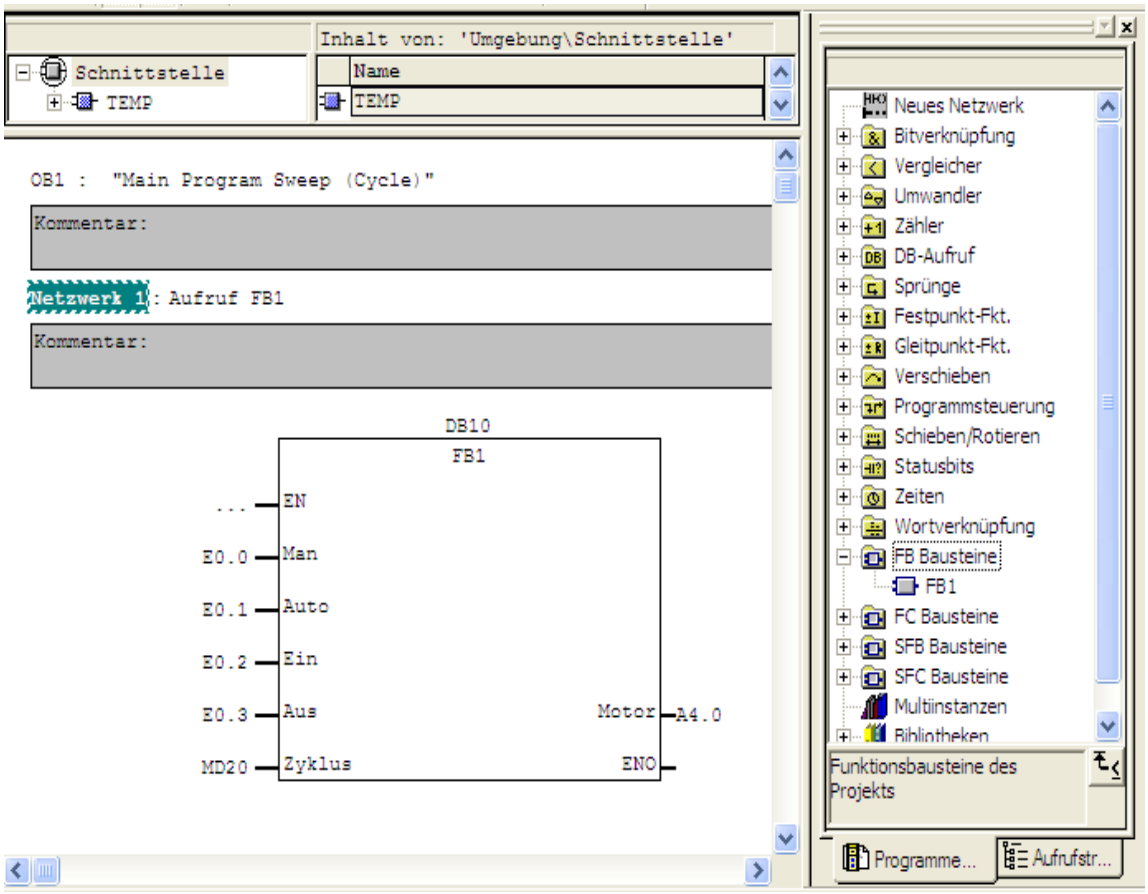
14. Mit ‘KOP, AWL, FUP- S7 Bausteine programmieren’ haben Sie jetzt einen Editor, der Ihnen die Möglichkeit gibt Ihren OB1 zu erstellen. Und zwar soll dort der FB1 zusammen mit seinem zugehörigen Instanz-DB (auch lokaler DB genannt) aufgerufen werden:

Dabei kann der Instanz-DB (hier: DB10) automatisch generiert werden, wenn die Abfrage mit **‚Ja’** bestätigt wird. (→ DB10 → Ja)



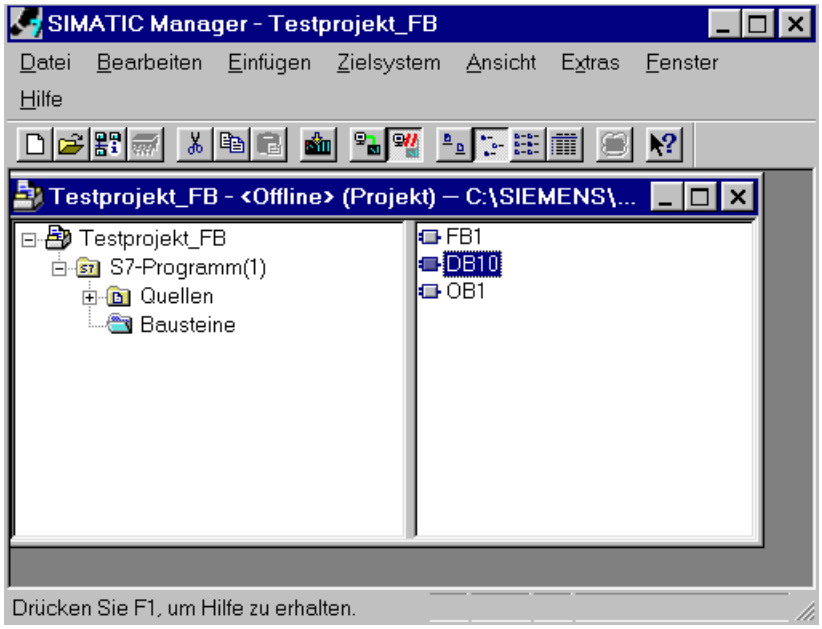
15. Dann werden alle Variablen vom Typ **‘in’, ‘out’** und **‘in_out’** angezeigt damit diesen Variablen Aktualparameter (z.B.: E 0.0, MW2 etc ...) zugeordnet werden können.

16. In unserem Beispiel soll diese Zuordnung wie unten gezeigt erfolgen. Ist diese Zuordnung erfolgt, kann der Organisationsbaustein OB1 gespeichert ,  ' und geladen ,  ' werden. Der Schlüsselschalter der CPU steht dabei auf Stop! (→  → )



Hinweis: Auf diese Art kann der FB1 unter Angabe unterschiedlicher Datenbausteine und Ein- / Ausgangsadressen mehrmals aufgerufen werden. Somit stellt er einen Standardbaustein für diese spezielle Aufgabenstellung dar.

17. Jetzt muss im ‚SIMATIC Manager' noch der Instanz- DB (lokaler DB) ‚DB10' angewählt, und in die in die CPU geladen ‚  'werden. Der Schlüsselschalter der CPU steht dabei auf Stop! (→ DB10 → )



18. Durch Schalten des Schlüsselschalters auf RUN wird das Programm gestartet.
Im Manuell-Betrieb (Vorwahl mit E0.0) läuft der Motor, solange wie Taster E0.2 betätigt wird.
Im Automatik-Betrieb (Vorwahl mit E0.1) schaltet der Motor ein, wenn Taster E0.2 kurz betätigt wird. Ausgeschaltet wird er, indem der Taster E 0.3 betätigt wird oder wieder auf die Betriebsart Manuell umgestellt wird.
In dem Merker MD20 wird mitgezählt, wie oft der FB1 aus dem OB1 heraus aufgerufen wird. Sie bekommen so ein Gefühl für die Zykluszeit des OB1. Dies geschieht mit einer hohen Frequenz, da der Programmzyklus im OB1 sehr kurz ist.