4.2 Einstieg in das Programmiersystem Step7 für Automatisierungstechnik Simatic S7

4.2.1 Installation Step7

Mit der lizensierten Software Step7 Professional werden installiert (**Bild 4-1**):

- der Simatic Manager
- der Simatic License Manager

Alle erforderlichen Operationen in S7 Projekten wie Erstellen, Löschen, Umbenennen, Archivieren etc. werden allein mit dem Simatic Manager ausgeführt.

Die Lizenz wird mit dem License Manager durch Übertragen eines Key – bisher auf einer Authorisierungsdiskette geliefert - auf den Programmierrechner freigeschaltet. Ein Einzellizenz kann mit diesem Key wahlweise und zeitweise auf verschiedene Programmierrechner übertragen werden (**Bild 4-2**).

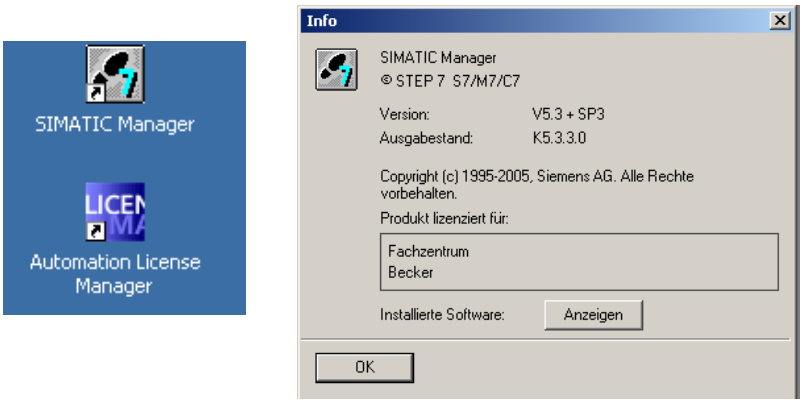


Bild 4-1: Simatic Manager und License Manager

Bild 4-2 : Verschieben des License Key für die zeitweise Authorisierung unterschiedlicher PC

4.2.2 Anlegen (oder Öffnen) eines Projektes

Die Automatisierungsaufgaben werden in Projekten gelöst. Projekte können enthalten die **Objekte**
- Hardware-Stationen
- Programme und
- Netze.

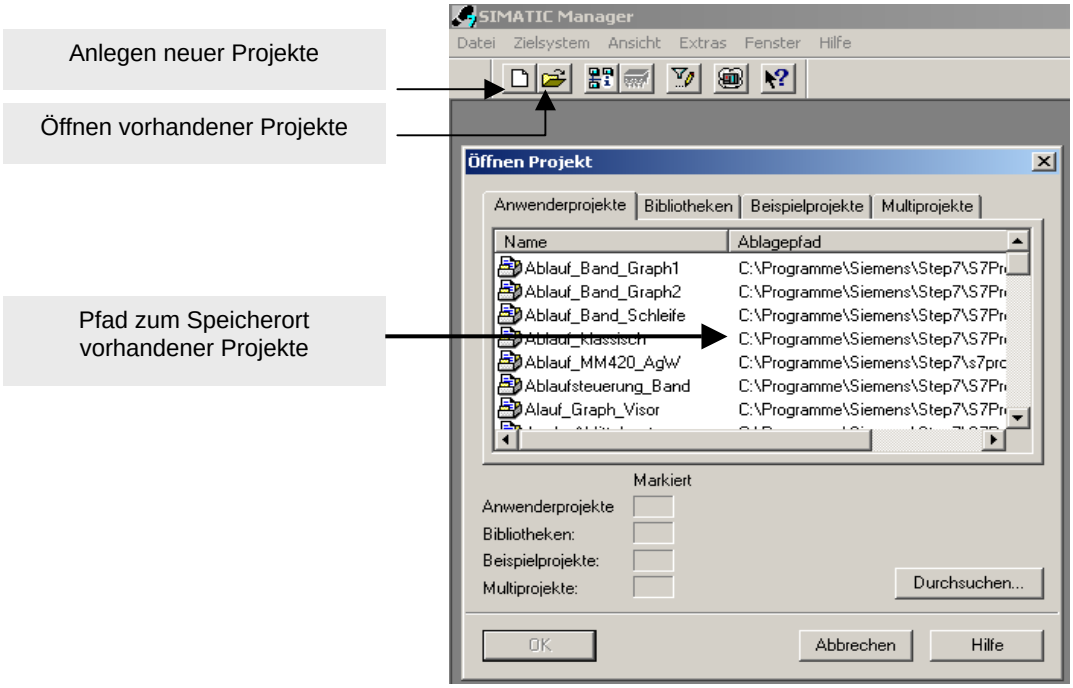


Bild 4-3: Anlegen neuer und Öffnen vorhandener Simatic Projekte

Für die nachfolgenden Darlegungen wird ein neues Projekt mit Namen *Einstieg_S7* angelegt. Der Simatic Manager legt dann einen Projektbaum an – vergleichbar mit dem Windows Explorer. Die Projektstruktur wird deshalb auch **Projektexplorer** bezeichnet..

Im leeren Projekt ist zunächst allein ein **MPI-Objekt** verfügbar **Bild 4-4**). Durch -> *Einfügen* werden weitere Objekte eingefügt .(**Bild 4-5**).

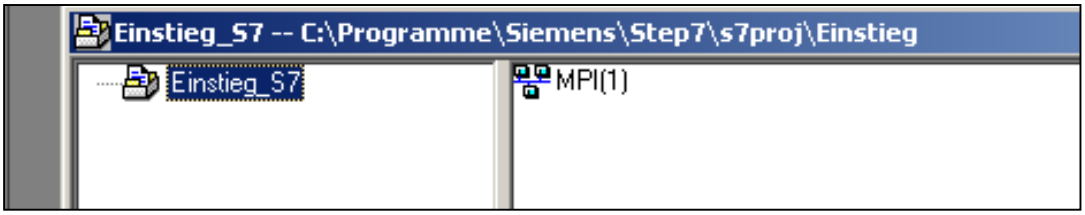


Bild 4-4: Leeres Projekt nach dem Anlegen

Das Mehrpunkt-Interface MPI von Simatic S7 erlaubt die einfache Vernetzung von S7-Komponenten. Damit kann über Globaldatenkreise ein begrenzter Datenaustausch zwischen S7-CPU's erfolgen. MPI hat gegenüber Profibus-DP an Bedeutung verloren. Allerdings wird MPI noch häufig für die Verbindung von Programmierrechner und S7-CPU gewählt, um Programme und Hardware-Konfigurationen in die CPU einzutragen.

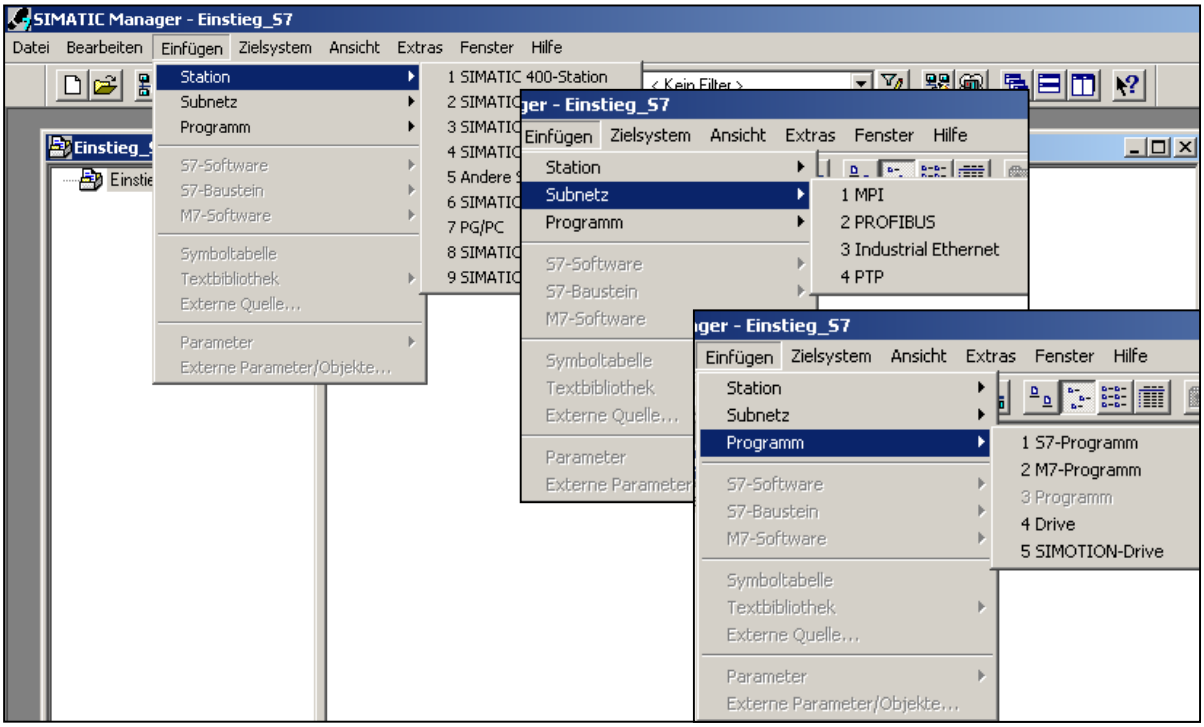


Bild 4-5: Einfügen von Stationen, Netzen und Programmen in ein Projekt

4.2.3 Die Hardware-Konfiguration

Programme sind in allen Projekten erforderlich, nicht aber zwingend auch Netze und Hardware-Stationen.
Die Hardware **muss** immer dann konfiguriert werden, wenn Stationen **vernetzt** oder wenn **Parameter** gegenüber den Default-Werten der Komponenten verändert werden müssen!
Dies ist in aktuellen Aufgabenstellungen fast immer der Fall!

HW-Konfiguration und Parametrierung hängen eng zusammen:
Konfiguration ist die Anordnung der Baugruppenträger und zentralen sowie dezentralen Baugruppen mit Hilfe des Softwarewerkzeuges HW-Konfig. Mit der HW-Konfiguration wird ein Abbild aller Baugruppen der Automatisierungseinrichtung erstellt.
Parametrierung ist das Festlegen des Verhaltens der Baugruppen durch Einstellung bestimmter Parameter.

Bild 4-6 zeigt beispielhaft die Vorgehensweise bei der Hardware-Konfiguration. Dabei sind folgende Details sind sichtbar:

- im Arbeitsfeld unten links die Konfiguration der (zentralen) Automatisierungsstation:
  Es wurden bereits eingefügt das Rack-300 (Profilschiene), ein Netzteil auf Steckplatz 1 und eine CPU 315-2PN/DP auf Steckplatz 2.
- im Arbeitsfeld oben links die Übersicht über alle (vernetzten) Komponenten, im Beispiel kein Netz.
- im Katalog rechts die Übersicht über die verfügbaren Komponenten, darunter im Schriftfeld Detailbeschreibungen der ausgewählten Komponente

Wichtige Begriffe und Abkürzungen für HW-Konfig:

- **Baugruppen:**

Rack Profilschiene als Träger der Baugruppen
PS Power Supply (Netzteile)
CPU Central Processor Unit (Zentraleinheit)
IM Interface-Module mit Sende- oder Empfangsfunktion für mehrzeiligen Aufbau
CP Communication Processor (Kommunikationseinheit)

FM Funktionsmodul (für speziellen hardwareunterstützte Funktionen)
SM Signalmodul

DI	Digitale Eingänge (Digital Input)	AI	Analoge Eingänge (Analog Input)
DO	Digitale Ausgänge (Digital Output)	AO	Analoge Ausgänge (Analog Output)

- **Steckplatznummer:**

Die Steckplätze im Rack werden nummeriert. Für einzeiligen Aufbau von S7-300 gilt:
Steckplatz 1: PS
Steckplatz 2: CPU
Steckplatz 3: IM oder Dummy
Steckplatz 4... 11: SM

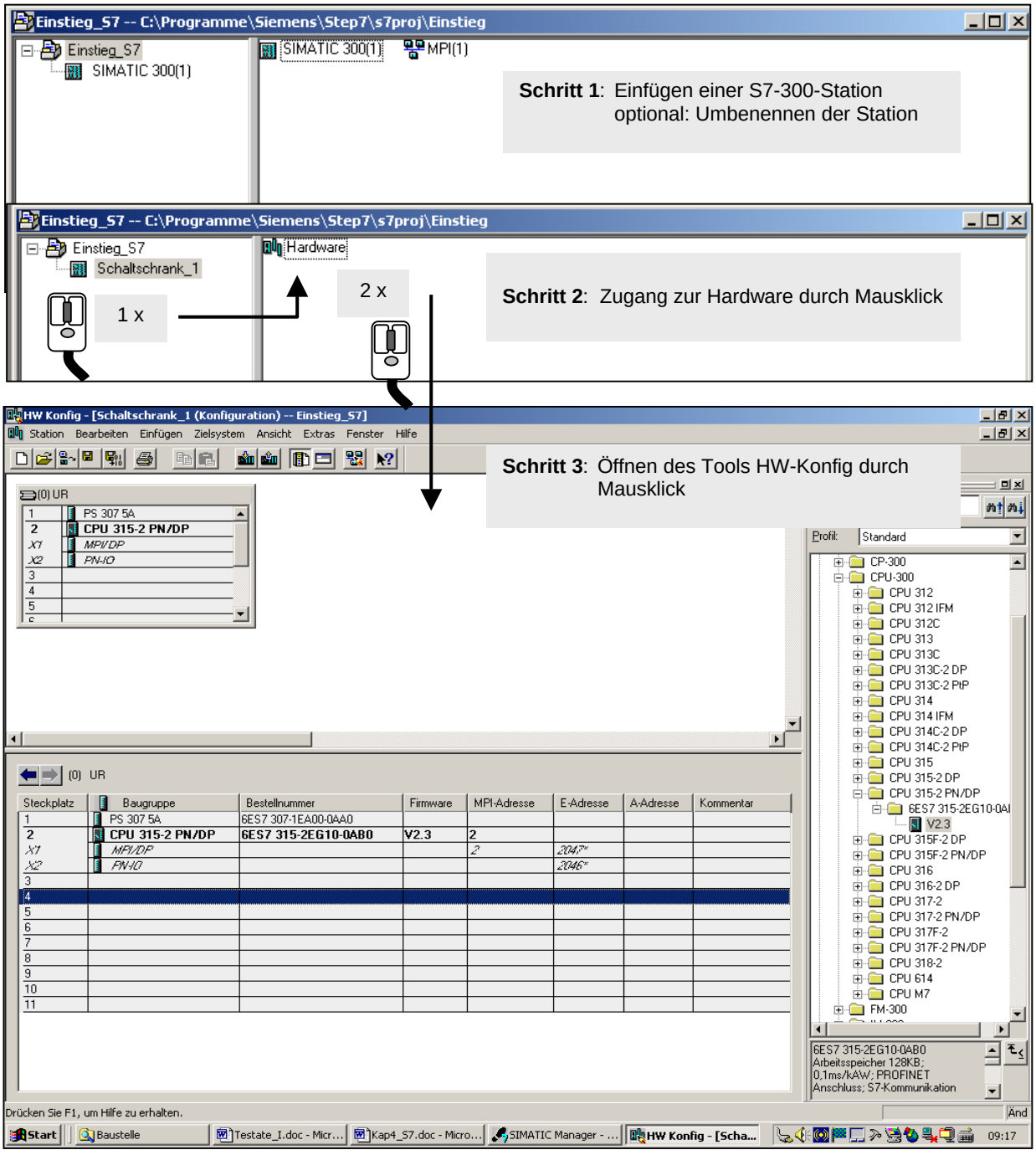


Bild 4-6: Der Weg zur HW-Konfiguration

- **Steckplatzabhängige Adressierung (Bild 4-7):**

Digitale Baugruppen: Pro Steckplatz werden 4 Byte vergeben (für bis zu 32 Ein- oder Ausgänge).
Standard S7-300: Adressbereich Byte 0 bis Byte 127
Grenzen sind abhängig von der Größe des Prozessabbildes der speziellen CPU.

Analoge Baugruppen: Pro Steckplatz werden 8 Byte vergeben (für bis zu 8 analoge Kanäle)
Standard S7-300: Adressbereich Byte 256 bis 766
Diese Grenzen sind abhängig von der Größe des Prozessabbildes der speziellen CPU

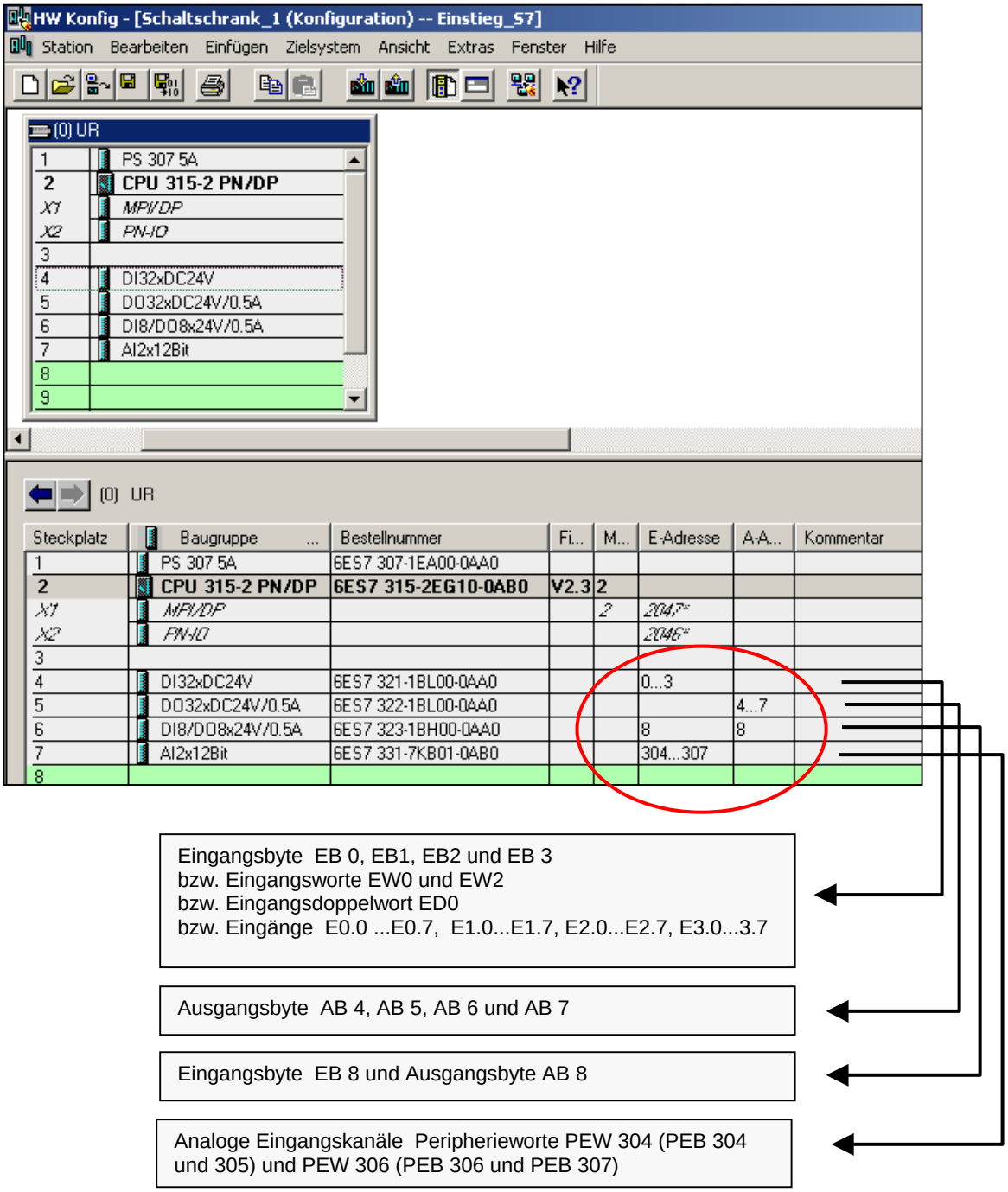


Bild 4-7: HW-Konfiguration mit steckplatzabhängiger Adressierung der Signalmodule

- **Freie Adressierung (Bild 4-8):**

Leistungsfähige CPU wie z.B. die CPU 315-2DP/PN erlauben im Rahmen gültiger Grenzen auch freie Adressierung.

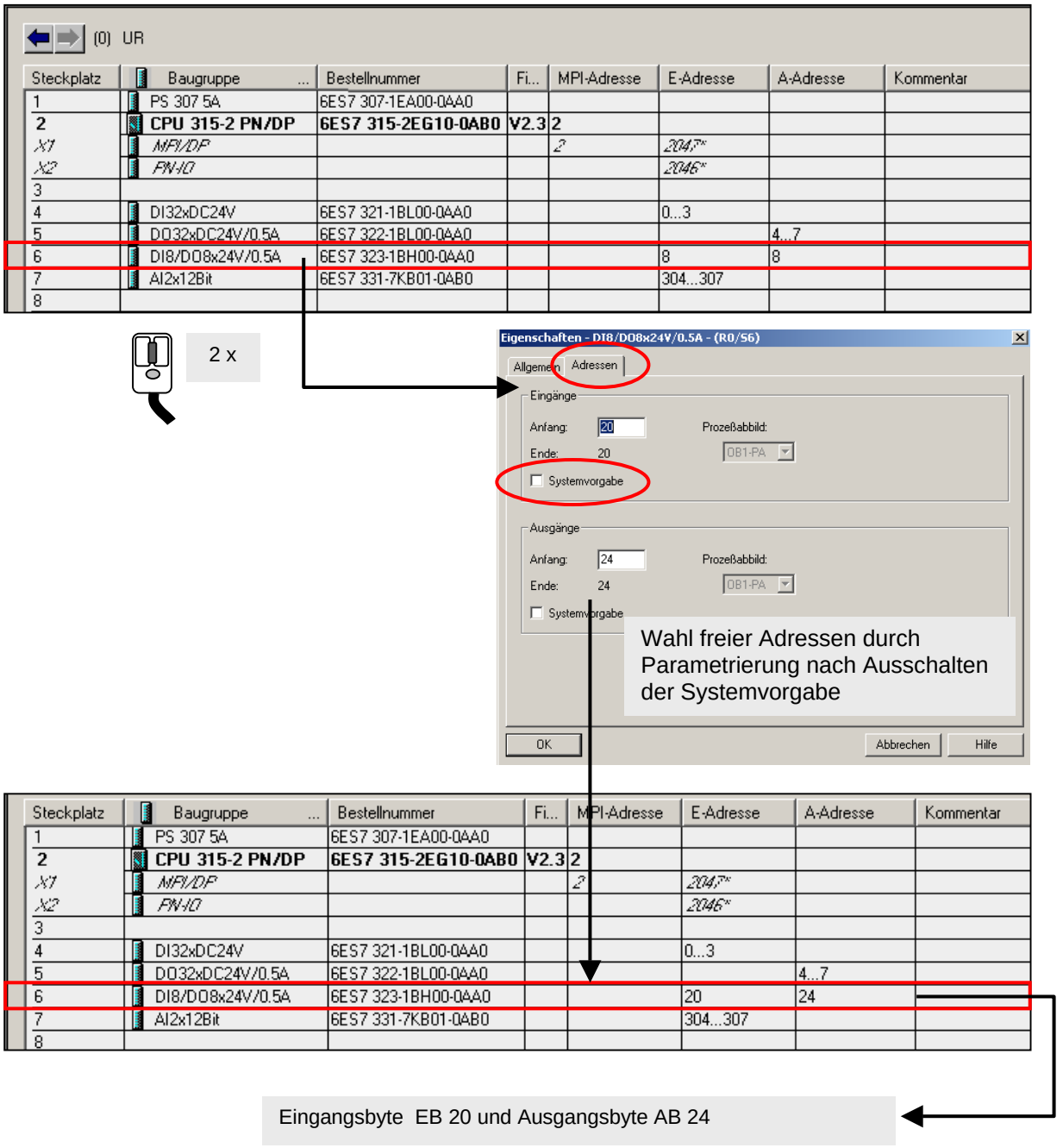


Bild 4-8: Vorgehensweise bei freier Adressierung

- **Parametrierung von Baugruppen**

Aktuelle Baugruppen erlauben die Einstellung von Parametern per Software anstelle Wickelbrücken, DIL-Schaltern etc. Die Parametrierung erfolgt durch „Öffnen“ der Baugruppen in HW-Konfig.
Bild 4-9 zeigt beispielhaft den Zugang zur Parametereinstellung in einer CPU und in einer analogen Eingangsbaugruppe.

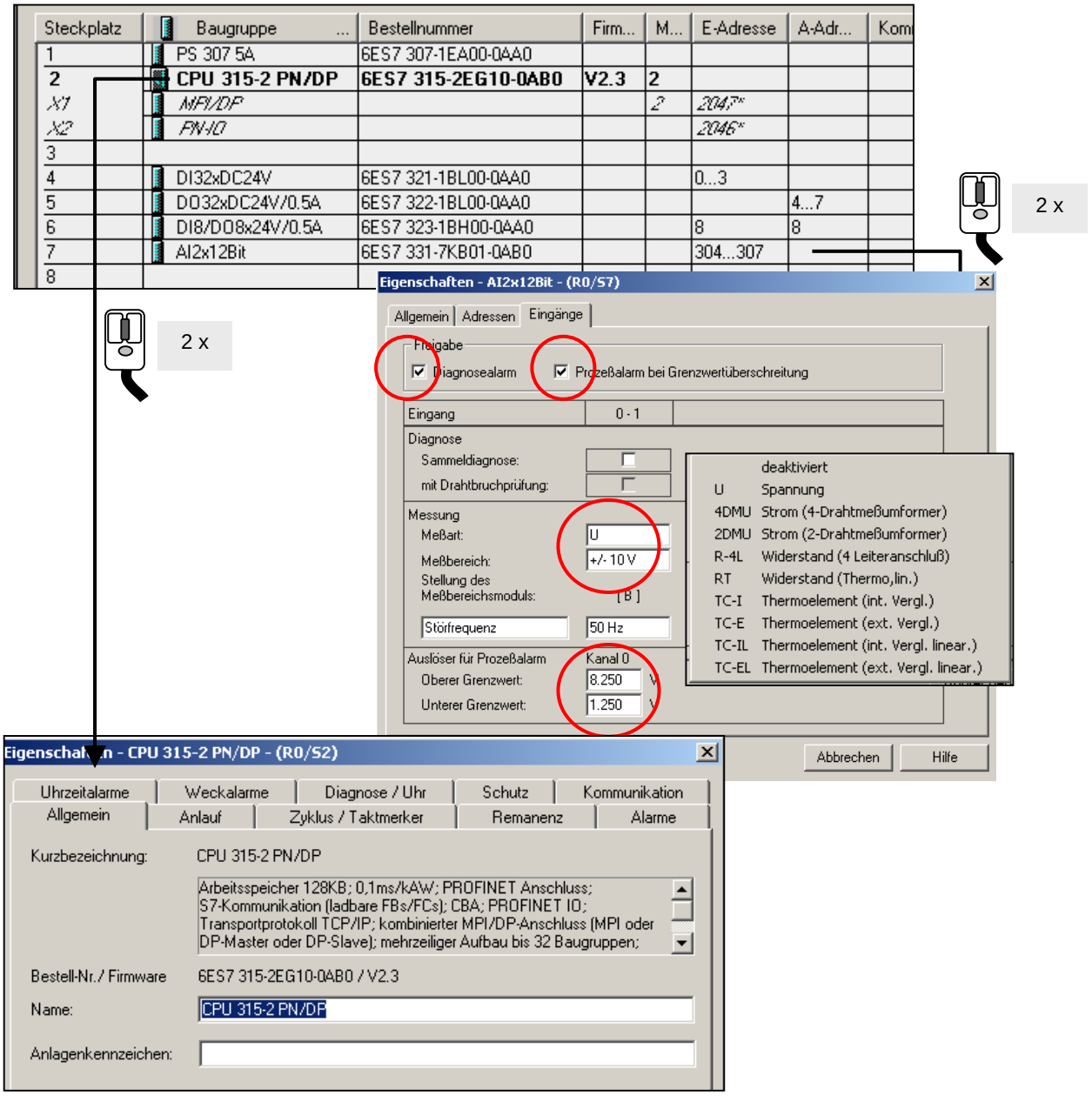


Bild 4-9: Beispiel für die Parametrierung einer CPU und einer analogen Eingangsbaugruppe

- **Systemdaten**

Sobald eine programmierbare Baugruppe – zumeist eine CPU – in eine HW-Konfiguration eingefügt wird, ist untrennbar damit ein (hardwareabhängiges) Programm verbunden.

Durch *Speichern und Übersetzen* der HW-Konfiguration werden Systemdaten erzeugt. Diese sind ein Abbild der Konfiguration. Sie werden im Bausteinbehälter des hardwareabhängigen Programms abgelegt (**Bild 4-10**).

Sobald Konfigurationen und Parameter festzulegen waren, sind diese anschließend in Form der Systemdaten in die CPU zu laden. Dies kann direkt durch Laden der Systemdaten erfolgen oder aber auch durch Laden der HW-Konfiguration.

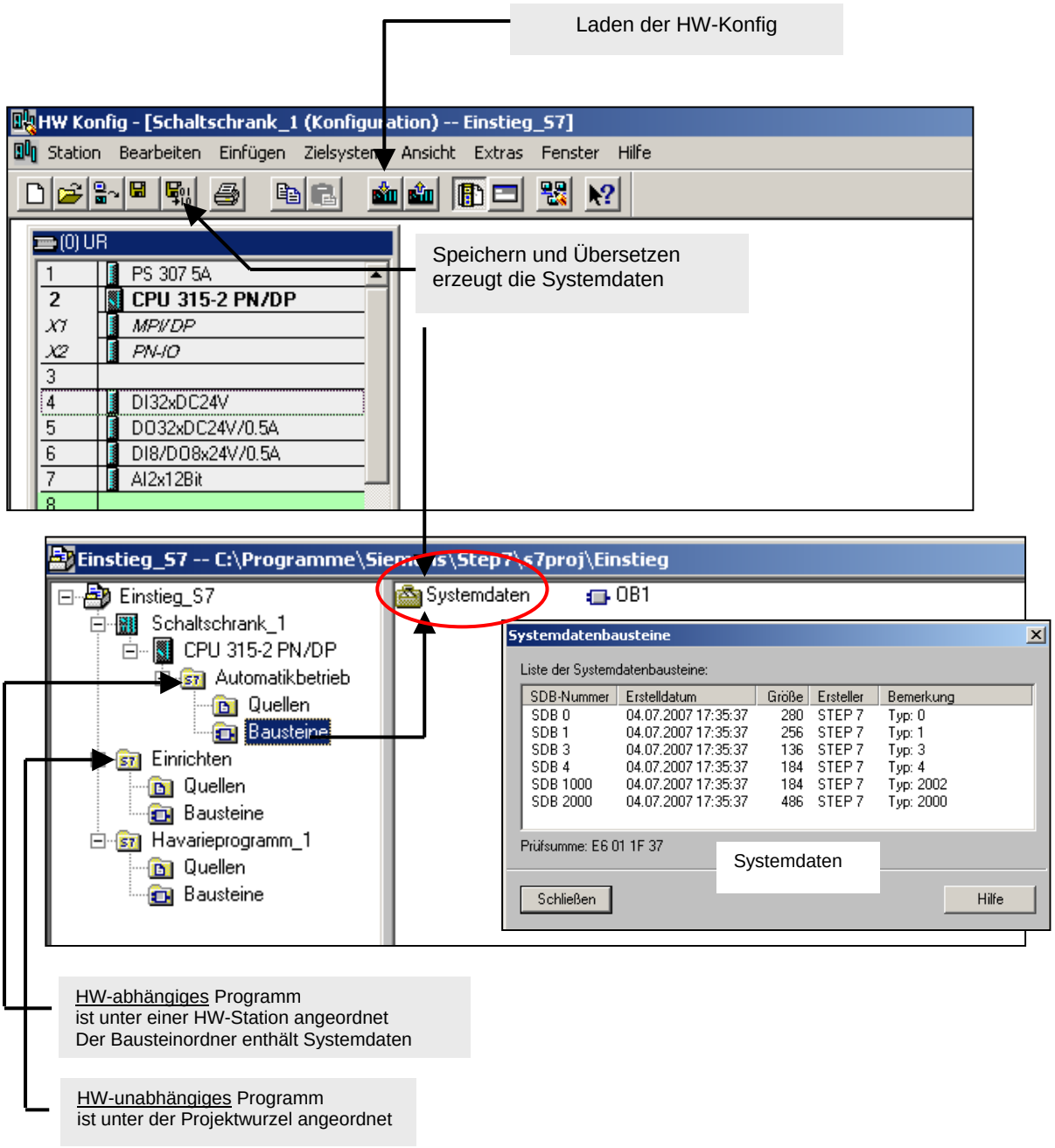


Bild 4-10: Erzeugen der Systemdaten durch -> *Speichern und Übersetzen*.
Die Systemdaten werden im Bausteinordner des hardwareabhängigen Programms (hier *Automatikbetrieb*) abgelegt.

4.2.4 Der Projektbaum (Projektexplorer)

Bild 4-10 zeigt eine Projektstruktur, wie sie durch Einfügen von Hardware und Programmen entsteht. Das Beispielprojekt enthält eine HW-Konfiguration mit hardwareabhängigem Programm „Automatikbetrieb“ sowie zwei hardwareunabhängige Programme „Havarieprogramm_1“ sowie „Einrichtbetrieb“. Alle Programme enthalten neben Quellen einen „Bausteinordner“. Quellen dienen zum Überleiten strukturierter Texte von Texteditoren in Step7-Programme.

In die CPU werden neben erforderlichen Systemdaten die jeweils erforderlichen Programmbausteine geladen.

4.2.5 Programmeditor und Programm

Ein S7-Programm besteht aus Bausteinen. Mit ihnen kann ein Programm gegliedert und strukturiert werden.

- Organisationsbausteine (OB’s) unterschiedlicher Priorität werden vom Betriebssystem aufgerufen. Der Baustein OB1 mit der niedrigsten Priorität 1 organisiert die zyklische Programmbearbeitung.
- Funktionen (FC’s) nehmen Programme auf. Sie fungieren wie Unterprogramme. FC’s verfügen über keinen speziell zugeordneten Datenspeicher. Da FC’s nicht vom Betriebssystem erkannt werden, müssen sie in das zyklisch bearbeitete Programm eingebunden werden (Unterprogramm aufrufen!). Funktionen können als parametrierbare Bausteine geschrieben werden.
- Funktionsbausteine (FB’s) nehmen Programme auf. Sie fungieren wie Unterprogramme. FB’s verfügen mit den Instanzdatenbausteinen über einen speziell zugeordneten Datenspeicher („Gedächtnis“). Da FB’s nicht vom Betriebssystem erkannt werden, müssen sie in das zyklisch bearbeitete Programm eingebunden werden (Unterprogramm aufrufen!). Funktionsbausteine können als parametrierbare Bausteine geschrieben werden.
- Datenbausteine (DB’s) enthalten globale Anwenderdaten und keine Programme. Sie werden nicht aufgerufen. Vielmehr greifen OB’s, FC’s und FB’s lesend oder schreibend auf Datenbausteine zu.

- **Bausteine in den Bausteinordner eines Programms einfügen**

Nach Markieren des Bausteinordners kann man mit -> *Einfügen* S7-Bausteine einfügen. Dazu wählt man den Typ Funktion, Funktionsbaustein, Organisationsbaustein oder Datenbaustein (**Bild 4-11**).

Variablentabellen (VAT) dienen zum Beobachten und Steuern von Operanden und sind an sich keine S7-Bausteine, sie werden aber wie solche behandelt (siehe Abschnitt 4.18).

In Bild 4-11 wurde eine Funktion FC1 eingefügt und dabei ein symbolischer Name sowie ein erklärender Kommentar festgelegt. Weiter wurde Funktionsbausteinsprache (Funktionsplan, FUP) als Programmiersprache gewählt (siehe Abschnitt 1.5.5 Die klassischen Programmiersprachen).

- **Den Programm-Editor öffnen**

Durch Mausklick auf den wünschten Baustein wird der Editor geöffnet (**Bild 4-12**). Sein Erscheinungsbild ist von der gewählten Sprache abhängig. Nur in den grafischen Sprachen KOP und FUP, nicht aber in AWL, steht ein Katalog der wesentlichsten Operationen zur Verfügung.

Das Fenster des Editors ist bezeichnet mit „KOP/AWL/FUP“.

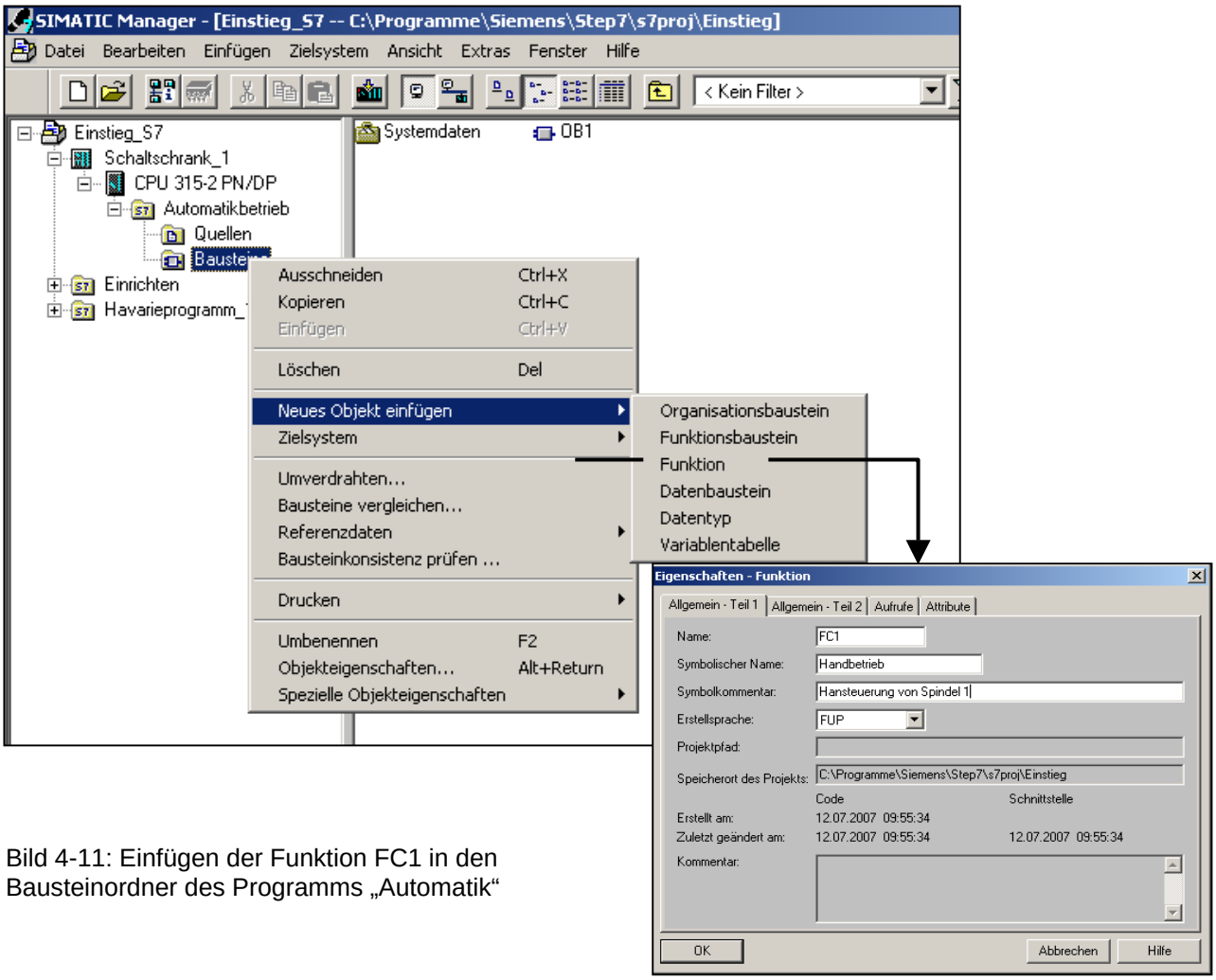


Bild 4-11: Einfügen der Funktion FC1 in den Bausteinordner des Programms „Automatik“

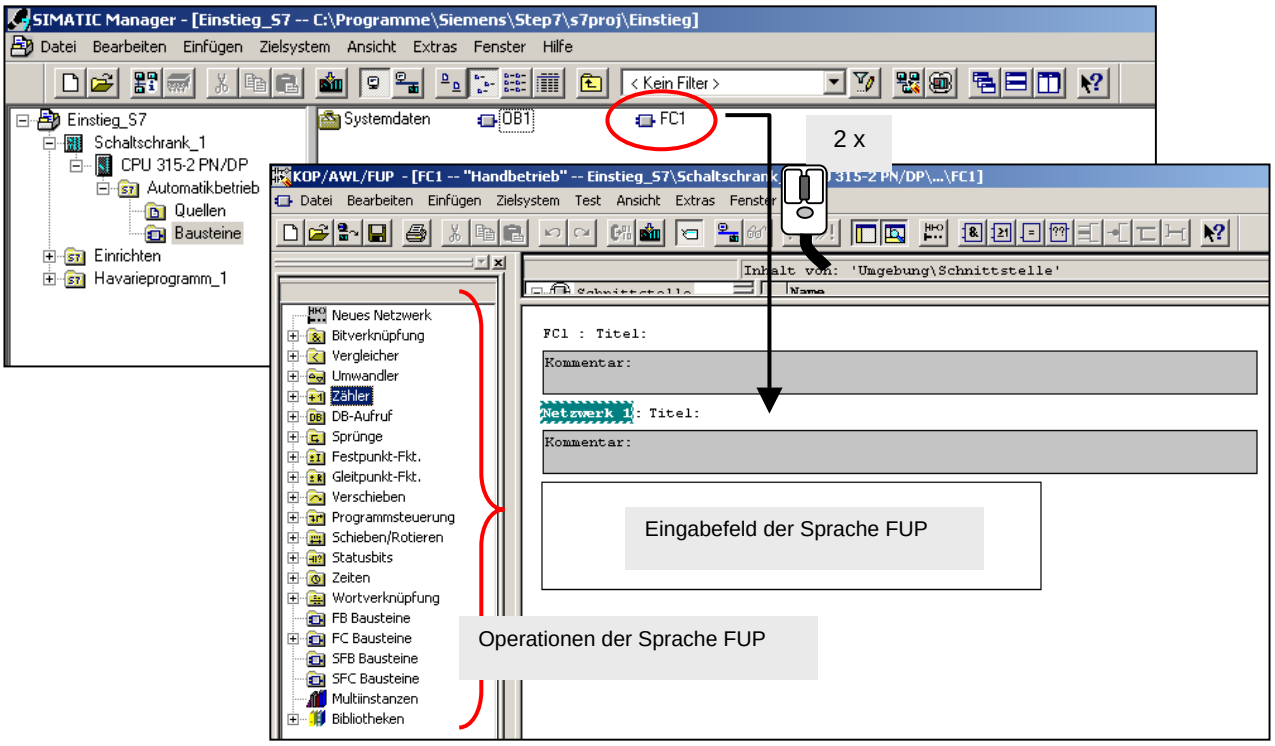


Bild 4-12: Öffnen des Programmeditors durch Mausklick auf Baustein im Simatic Manager

- **Programmanweisungen schreiben**

Der Einstieg erfolgt in der Funktionsbaustein-Sprache FUP

Nach dem Öffnen des Editors werden die Programmanweisungen geschrieben (**Bild 4-13**).
Für Sprache FUP gilt: Rote Fragezeichen fordern zwingend den Eintrag von Operanden. Drei Punkte stehen für Wahlfreiheit, ob an dieser Stelle ein Operand geschrieben wird oder nicht.

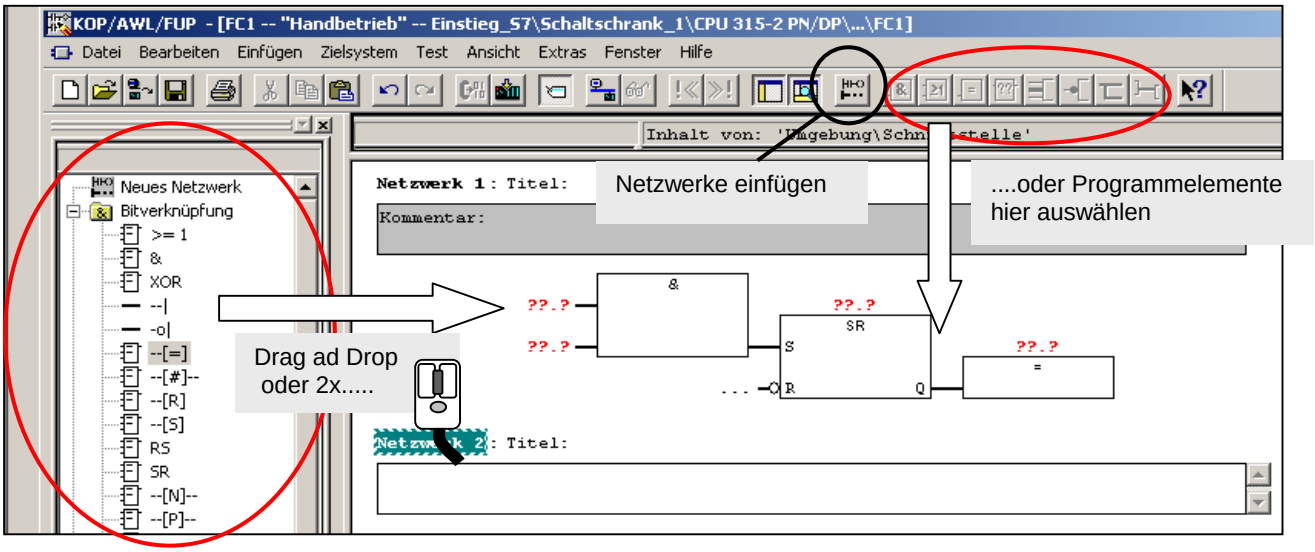


Bild 4-13: Arbeit im FUP-Editor mit Operationen der Bitverknüpfung. Auch in der Symbol-Leiste finden sich häufig benutzte Bit-Operationen

Mit **Netzwerken** werden Programme innerhalb eines Bausteins gegliedert. Sie sind zwingend erforderlich in den grafischen Sprachen KOP und FUP, nicht aber in AWL. Netzwerke sollten **Überschriften (Titel) und Kommentare** erhalten!

- **Funktionen oder Funktionsbausteine „aufrufen“**

Unter „Aufrufen“ versteht man umgangssprachlich, die „Unterprogramme“ in Form von FC’s oder FB’s so in das Gesamtprogramm einzubinden, dass diese bearbeitet werden.
In den grafischen Sprachen wie FUP und KOP werden alle bereits geschriebenen FC’s und FB’s genau wie Programmanweisungen im Katalog derselben abgelegt. Ihr Aufruf in anderen Bausteinen kann deshalb ähnlich der Vorgehensweise bei Programmanweisungen per Drag an Drop erfolgen (**Bild 4-14**).

Nicht zugeordnete Funktionen und Funktionsbausteine werden nicht bearbeitet!

- **Verwendung von Eingang EN (Enable = Freigabe) und Ausgang ENO (Enable Out):**

In den grafischer Sprachen wie FUP und KOP erleichtert der Eingang EN die Programmierung bedingter Wirkung von Programmanweisungen. In AWL müssen diese dagegen durch Sprungbefehle programmiert werden!

EN erlaubt Freigabe oder Sperrung der Operation (hier Aufruf von FC1): Wird an EN kein Boolescher Operator geschrieben, wird die Operation immer ausgeführt.
Bei Anschreiben eines Booleschen Operators wird die Operation nur dann ausgeführt, wenn dieser den Wert TRUE hat.

Der Boolesche Ausgang ENO ist TRUE, wenn die Operation fehlerfrei ausgeführt wird. An diesem Ausgang können Programmanweisungen oder aber andere EN-Eingänge angeschaltet werden. Durch Verschalten von ENO und EN entstehen Ketten grafischer Programmanweisungen.

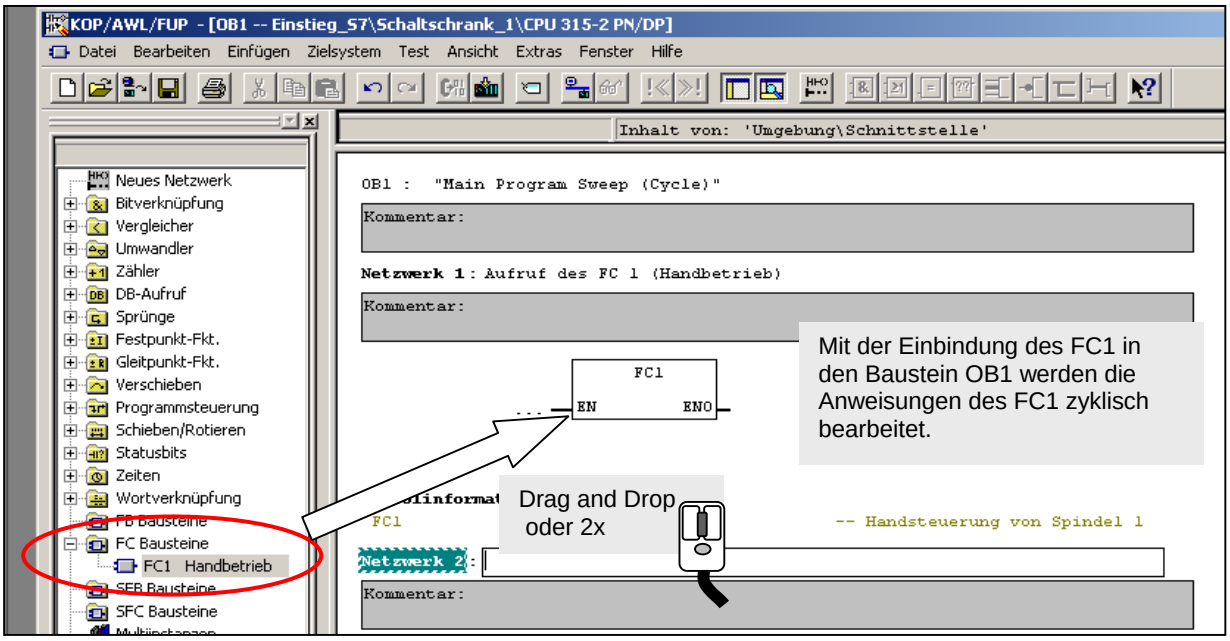


Bild 4-14: Einbinden eines FC in den Organisationsbaustein OB1

4.2.6 Zyklische Programmbearbeitung

(Siehe hierzu Abschnitt 1.5.3: Programmbearbeitung, Prozess-Abbilder und Mehrfachzuweisungen)

Im System Simatic S7 kann die Programmbearbeitung **zyklisch** (OB1), **zeitzyklisch** (z.B. OB 35) und **ereignisgesteuert** (z.B. OB 86) erfolgen.
OB1 organisiert die zyklische Programmbearbeitung. Nur Organisationsbausteine werden vom Betriebssystem aufgerufen, alle anderen Bausteine müssen durch „Aufruf“ eingebunden werden (im **Bild 4-15** mit AWL Operation CALL).

Das Schema zeigt einen vom OB1 organisierten Zyklus. Der Zyklus muss in jedem Falle – auch bei zeitweiliger Nichtbearbeitung von Programmteilen – erhalten bleiben! Seine Zeitdauer wird von der CPU überwacht. (Defaultwert der Zyklusüberwachungszeit 150 ms). Bei Überschreitung geht die CPU in STOPP:

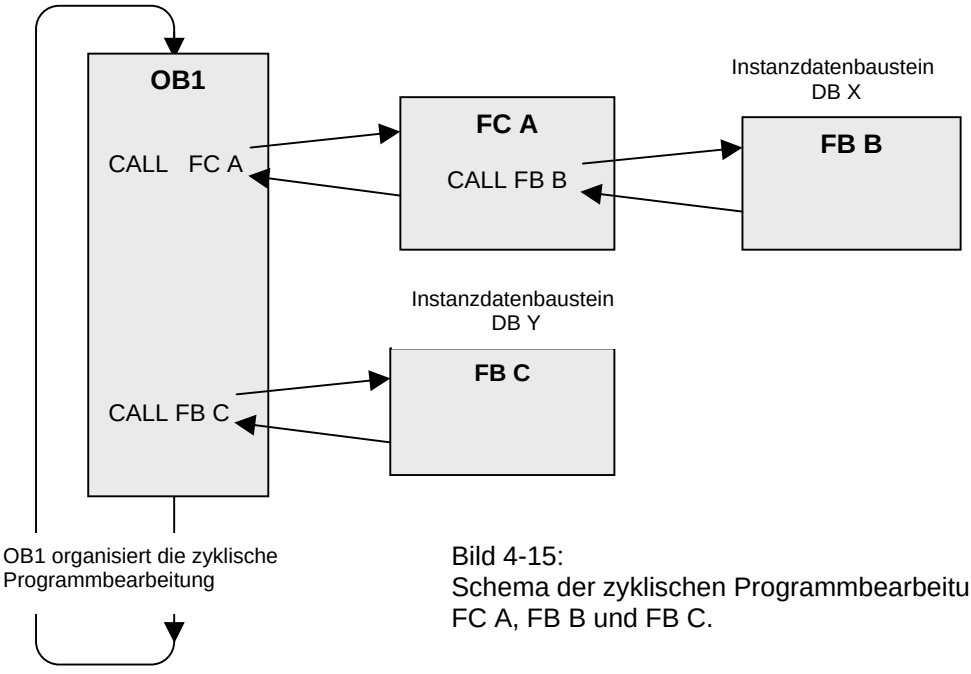


Bild 4-15:
Schema der zyklischen Programmbearbeitung von OB1, FC A, FB B und FB C.

4.2.7 Gliederung und Strukturierung von Programmen

Wird die Abarbeitung eines Programms nicht durch OB's organisiert, muss das Programm den Baustein OB1 enthalten. Ein vom Umfang her begrenztes Programm kann durchaus allein im OB1 ohne Verzweigungen zu anderen Bausteinen niedergeschrieben werden. Ein solches Programm bezeichnet man als **lineares Programm (Bild 4-16).**

In einzelne Bausteine FC's und FB's **gegliederte Programme** sind gegenüber linearen Programmen übersichtlicher. Bei Änderungen müssen nur bestimmte Bausteine berücksichtigt werden.

In der Automatisierungstechnik müssen oftmals gleichartige Programmteile mehrfach angewendet werden. Dann ist zu empfehlen, Programme ohne Bezug auf Hardware (keine Operanden E, A, M, T, T, DB!) als parametrierbare Bausteine zu schreiben. Erst beim Aufruf werden diesen Programmen die aktuellen Parameter übergeben (siehe Abschnitt 4.13).

Strukturiert man mit parametrierbaren Bausteinen, dann können Programmteile in anderen Programmen wiederwendet werden oder aber mehrmals im gleichen Programm.

In Fachkreisen spricht man erst bei Anwendung parametrierbarer Bausteine von **strukturierten Programmen**.

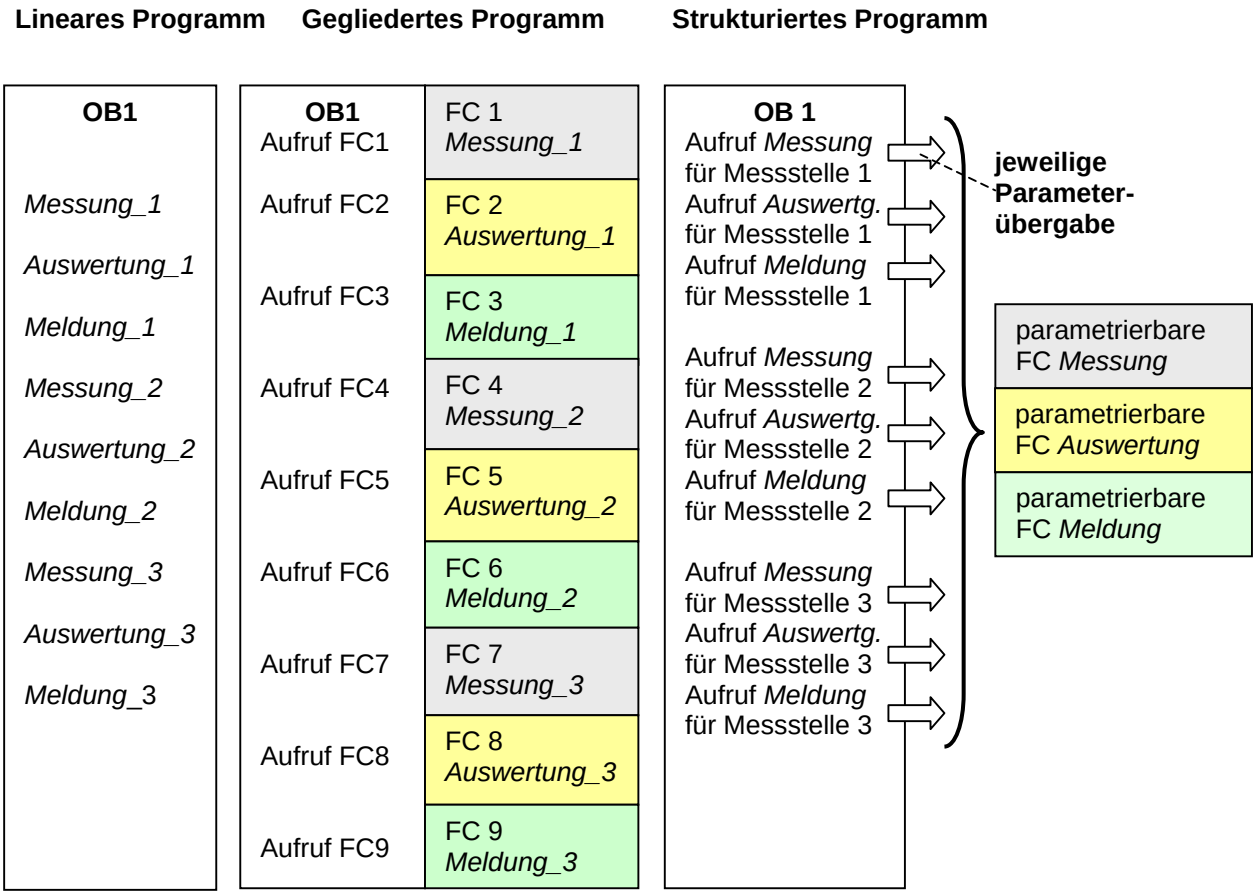


Bild 4-16: Schema für lineare, gegliederte und strukturierte Programme
(anstelle mit Funktionen (FC) könnte prinzipiell auch mit Funktionsbausteinen (FB) gearbeitet werden)

4.2.8 Datenhaltung

Daten der Anwenderprogramme können global oder lokal deklariert werden:
Globale Daten können von allen Bausteinen des Programms geschrieben und gelesen werden. Lokale Daten gelten nur in dem Baustein, in welchem sie deklariert wurden. Gleiche Namen lokaler Daten in unterschiedlichen Bausteinen beeinflussen sich nicht.

Eine Besonderheit des Systems Simatic S7 gegenüber der Norm IEC 61131 ist die Pflicht des Programmierers, globale Daten selbst zu adressieren. Dazu stehen unterschiedliche Speicherbereiche zur Verfügung.
Weiter erfolgt bei Simatic S7 die Einbindung von Ein- und Ausgangssignalen in ein Programm nicht durch Legen von Variablen auf Adressen, sondern durch Ansprechen der Werte in den Prozessabbildern PAE und PAA mit den Operanden E und A und deren Adressen.

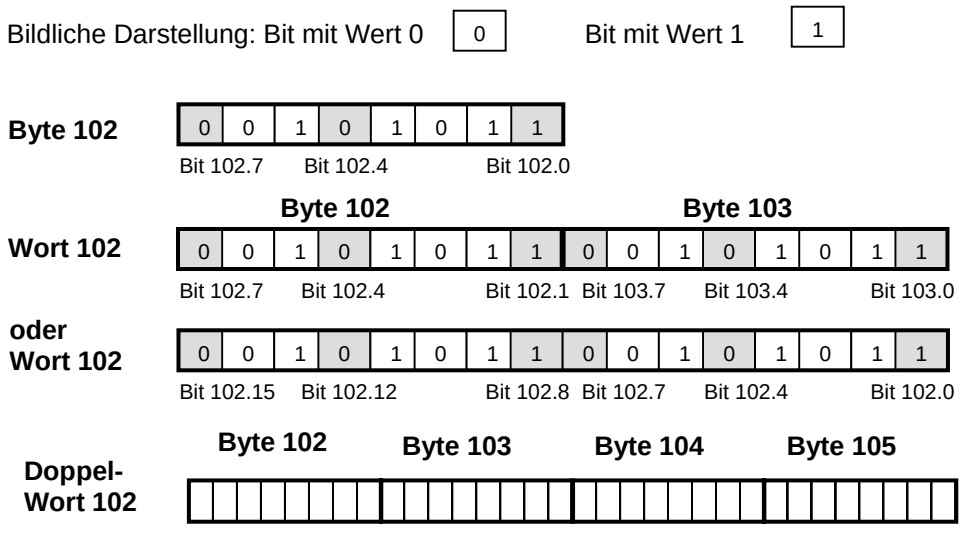
Der Begriff der Variablen und die Variablendeklaration spielen deshalb in Step7 nicht die zentrale Rolle wie in IEC-Programmiersystemen. Lokale Variablen in FC's und FB's werden allerdings strikt gemäß IEC 61131-3 angewendet.

Bild 4-18 zeigt die globalen Daten in der Übersicht mit ihren Kennungen sowie Beispiele der Adressierung

→ Hierzu auch Abschnitt 1.5.4 Signale, Daten und Speicher

Der **Merkerbereich** dient zur Ablage von Zwischenergebnissen jeder Art, wenn diese global verfügbar sein müssen. Seine Größe ist abhängig von der CPU und kann z.B. 2048 Byte betragen. Auf Merker kann im Format Bit, Byte, Wort oder Doppelwort zugegriffen werden. Binäre Merker werden mit Kennzeichen M und Bytenummer **.** Bitnummer adressiert. Die Adresse von Wort und Doppelwort richtet sich stets nach dem niederwertigsten enthaltenen Byte (**Bild 4.17**).

Adressieren eines BOOL / Bit:	Byteadresse **.** Bitadresse
Adressieren eines BYTE:	Byteadresse
Adressieren eines WORD:	Byteadresse des niederwertigen BYTE
Adressieren eines DWORD:	Byteadresse des niederwertigsten der vier BYTE.



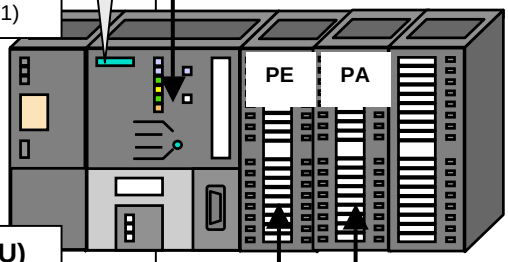
In gleicher Weise werden Eingänge und Ausgänge als Bit, Bit, Wort und Doppelwort angesprochen.
Beispiele: E 22.0; EB 22; EW 22; ED 22
A 40.2; AB 40; AW 40; AD 40

Bild 4-17: Regeln der Adressierung im System Simatic

Programmierrechner			
HW-Konfiguration	Anwender-programm	Kommentare Datenvorköpfe	Symbol-tabelle

Systemspeicher (CPU)			
Operanden			Beispiel
Prozessabbild der Eingänge PAE	Binärer Eingang (Eingangsbit) Eingangsbyte Eingangswort Eingangsdoppelwort	**E**	E 4.0 EB 4 EW 4 ED 4
Prozessabbild der Ausgänge PAA	Binärer Ausgang (Ausgangsbit) Ausgangsbyte Ausgangswort Ausgangsdoppelwort	**A**	A 12.0 AB 12 AW 12 AD12
Merker	Merkerbit Merkerbyte Merkerwort Merkerdoppelwort	**M**	M 1.0 MB 1 MW 2 MD 2
Zeitglieder	Zeitworte der Timer	**T**	T0 ... T63 ...
Zähler	Zählerworte der Counter	**Z**	Z0 ... Z63 ...
Datenbausteine	Datenbit Datenbyte Datenwort Datendoppelwort	**DB**	DB2.DBX 4.0 DB2.DBB 4 DB2.DBW 4 DB2.DBD 4

Ladespeicher (CPU)	
System-daten	Maschinencode des Anwender-programms 1)



Arbeitsspeicher (CPU)	
Aktuell erforderliche Teile der Systemdaten	Aktuell erforderliche Teile des Anwender-programms

Systemspeicher (Direkter Zugriff auf E/A Baugruppen)			
Peripherie von Eingangskarten	Peripherieeingangsbyte Peripherieeingangswort Peripherieeingangsdoppelwort	**PE**	PEB 8 PEW 8 PED 8
Peripherie von Ausgangskarten	Peripherieausgangsbyte Peripherieausgangswort Peripherieausgangsdoppelwort	**PA**	PAB 20 PAW 20 PAD 20

1) Maschinencode des Anwenderprogramms ohne Symbole, Kommentare und Datenvorköpfe !

Bild 4-18: Übersicht über die Datenhaltung im System Simatic S7

Das Gros der Anwenderdaten wird in **globalen Datenbausteinen** verwaltet. Abhängig von der Leistungsfähigkeit der CPU kann man z.B. 511 Datenbausteine DB 1 bis DB 512 nutzen. Jeder Datenbaustein kann beispielsweise bis zu 16 420 (16k) Byte enthalten.
Vom Grundsatz her werden Daten in Datenbausteinen byteweise organisiert, aber als Bit, Byte, Wort oder Doppelwort angesprochen (**Bild 4-19)**. Im Datenbaustein selbst erscheinen Wortadressen (**Bild 4-21**).

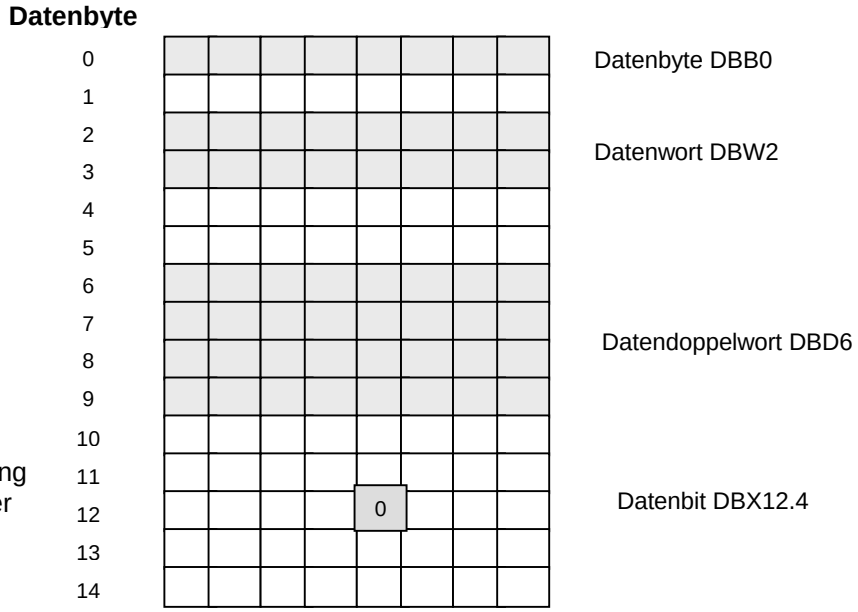


Bild 4-19:
Regeln der Adressierung von Elementen globaler Datenbausteine

Globale Datenbausteine werden wie Funktionen und Funktionsbausteine in den Bausteinordner eines Programms eingefügt (**Bild 4-20**).

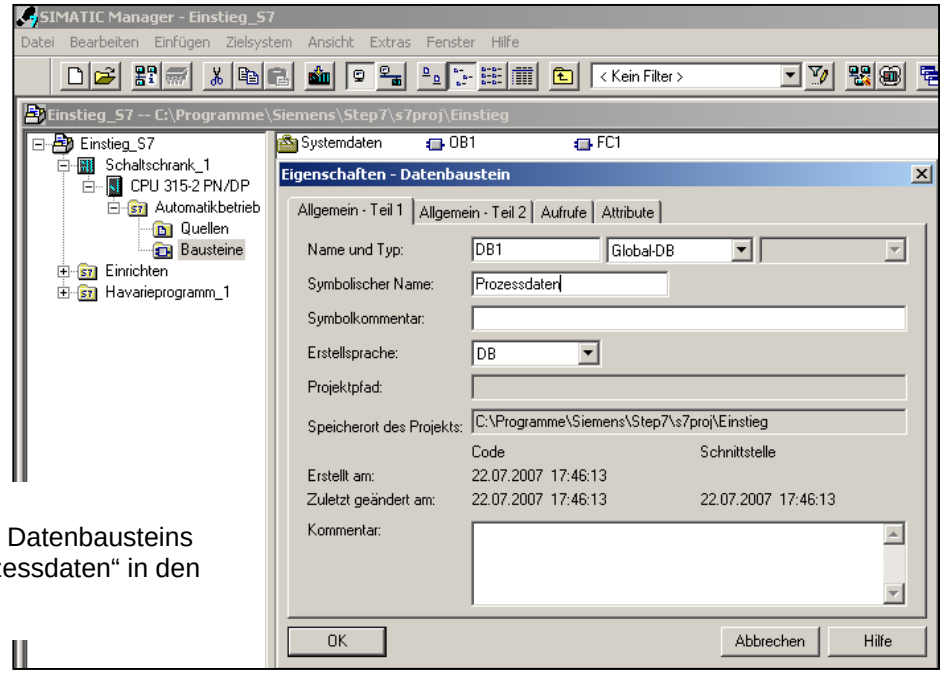


Bild 4-20:
Einfügen des globalen Datenbausteins DB 1 mit Namen „Prozessdaten“ in den Bausteinordner

Beim Anlegen der Datenworte im Datenbaustein werden Name und Datentyp und bei Bedarf ein Anfangswert festgelegt Standarddatentypen können dabei aus einer Liste ausgewählt werden.
Im **Bild 21** wurden 3 Daten deklariert:
DB1.DBW0 mit Namen „Zeitwert1“ vom Typ S5Time und Anfangswert 2s
DB1.DBW2 mit Namen „Teilezahl“ vom Typ INTEGER

DB1.DBX4.0 mit Namen „Meldebit“ vom Typ BOOL.
Der „Sollwert“ vom Datentyp REAL würde Adresse DB1.DBD6 erhalten.

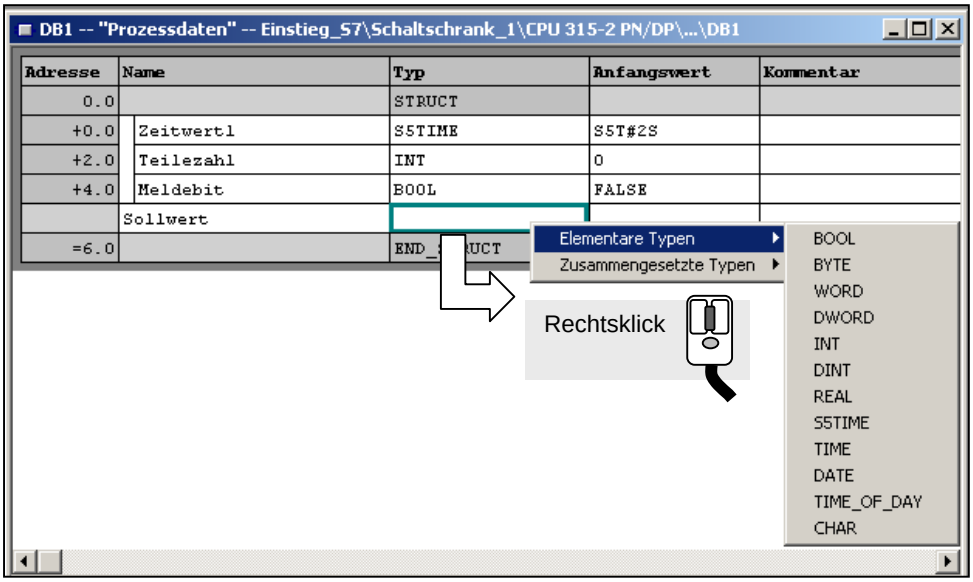


Bild 4-21 : Deklaration von globalen Daten in einem Datenbaustein.

4.2.9 Datenformate

Step7 verwendet weitgehend die nach IEC 61131-3 definierten **elementaren und zusammengesetzten Datentypen**. Zusammengesetzte Datentypen wie ARRAY und STRUCT sowie spezielle **User defined Typs (UDT)** werden im Abschnitt 14 behandelt.
Nachfolgende Tabelle zeigt die allgemeinste Übersicht der elementaren IEC-Datentypen. Mit dem Schlüsselwort ANY_ fasst man elementare Datentypen in fünf Gruppen zusammen:

- Bitfolgen: ANY_BIT
- Alle Zahlen: ANY_NUM
- Ganzzahlen mit und ohne Vorzeichen: ANY_INT
- Gleitpunktzahlen: ANY_REAL
- Datum, Zeitpunkt: ANY_DATE
- Zeichenfolge, Zeitdauer ANY (allgemeinste Vorgabe)

<table>
<tr><th colspan="5">ANY</th></tr>
<tr><th>ANY_BIT</th><th colspan="3">ANY_NUM</th><th>ANY_DATE</th><th rowspan="3"></th></tr>
<tr><td rowspan="2">BOOL
BYTE
WORD
DWORD
LWORD</td><th colspan="2">ANY_INT</th><th>ANY_REAL</th><td rowspan="2">DATE
TIME_OF_DAY
DATE_AND_TIME</td></tr>
<tr><td>INT
SINT
DINT
LINT</td><td>UINT
USINT
UDINT
ULINT</td><td>REAL
LREAL</td><td>TIME
STRING</td></tr>
</table>

Bedeutung der Schlüsselworte:

DWORD:	Double Word:	Doppelwort	
LWORD:	Long Word:	Langwort	
INT:	Integer:	Ganzzahl	
SINT:	Short Integer:	kurze Ganzzahl	
DINT:	Double Integer:	Doppelte Ganzzahl	
LINT:	Long Integer:	Lange Ganzzahl	
UINT:	Unsigned Integer:	Ganzzahl ohne Vorzeichen	(0..+65 535)
USINT:	Unsigned Short Integer:	kurze Ganzzahl ohne Vorzeichen	(0..+255)
UDINT:	Unsigned Double Int:	Doppelte Ganzzahl ohne Vorzeichen	(0..+2^{32}-1)
ULINT:	Unsigned Long Integer:	Langzahl ohne Vorzeichen	(0..+2^{64}-1
REAL:	Real:	Gleitpunktzahl	
LREAL:	Long Real:	Lange Gleitpunktzahl	

Step7 unterstützt nicht alle Datentypen. Nachfolgende elementare Datenformate bilden das unverzichtbare **Grundgerüst der Datentypen (Bild 4-22).**

Schlüsselwort	Datentyp	Größe	Schreibweise / Wertebereiche
Bit-Datentypen			
BOOL	Boolesche Variable	1 Bit	FALSE (0) , TRUE (1)
BYTE	8 Bit-Folge oder 2 Hex-Zahlen	8 Bit	B#16#00...FF
WORD	16 Bit-Folge oder 4 Hex-Zahlen	16 Bit	W#16#0000...FFFF
DWORD	32 Bit-Folge oder 8 Hex-Zahlen	32 Bit	DW#16#0000_0000...FFFF_FFFF
LWORD	64 Bit-Folge oder 16 Hex-Zahlen	64 Bit	LW#16#0...FFFF_FFFF_FFFF_FFFF
CHAR	ASCII-Zeichen	8 Bit	‘ X‘ , ‘ +‘, ‘ &‘
Arithmetische Typen			
INT	Ganze Zahlen (Festpunktzahlen)	16 Bit	-32768 .... +32767 (0...65535)
DINT	Ganze Zahlen (Festpunktzahlen)	32 Bit	L# -2 147 483 648 ... 2 147 483 647
REAL	Reelle Zahlen (Gleitpunktzahlen)	32 Bit	Beispiel: 634.57 oder 6.3457e+02
Zeittypen			
TIME	Zeitdauer (IEC)	32 Bit	TIME# -24d20h31min ... + 24d20h31min
TIME OF DAY	Uhrzeit (Tageszeit)	32 Bit	TOD# 23:59:59.9
DATE	Datum	32 Bit	DATE#1990-01-01
DATE_AND_TIME	Zeitstempel: Datum und Uhrzeit	32 Bit	DT#2007-02-04-10:15:30
S5TIME	Zeitdauer (S5-Format, nur bei Step7)	16 Bit	S5T# 0ms... 9990s

Bild 4-22: Die wichtigsten Datentypen bei der Arbeit mit Step7

Erläuterungen zu den Datentypen für Zeit und Datum

Datentyp	Schlüsselwort	Interne Wertung
Zeitdauer	TIME	Zeitdauer in Millisekunden
Tageszeit	TIME_OF_DAY oder TOD	Zeit im Millisekunden ab 00:00Uhr
Datum	DATE	Zeit in Sekunden ab 01.01.1970 00:00 Uhr
Datum und Uhrzeit (Zeitstempel)	DATE_AND_TIME oder DT	Zeit in Sekunden ab 01.01.1970 00:00 Uhr

TIME: Zeitbasis ist die Millisekunde. Jede andere Zeitangabe wird intern in eine Anzahl von Millisekunden umgerechnet. Die Gesamtzahl von Millisekunden steht dualcodiert in einem 32 Bit breiten Doppelwort (DWORD). Der größtmögliche Dezimalwert eines Doppelwortes ist 4.294.967.295.
Werden größere Werte vorgegeben, meldet das System CoDeSys den Fehler „Überlauf in Zeitkonstante“. Es ist aber zu beachten, dass die Datenbreite in unterschiedlichen Programmiersystemen auch systemabhängig sein kann.

TOD: Auch wenn die Tageszeit im „Klartext“ eingetragen werden kann, so wird sie intern doch als Anzahl der Millisekunden ab 00:00 Uhr gewertet.

DATE: Auch hier wird die Angabe in „Klartext“ intern in eine Anzahl von Sekunden ab dem festgelegten Zeitpunkt 01.01.1970 00:00 Uhr umgerechnet.

DT: ebenfalls Umrechnung in die Anzahl von Sekunden ab dem festgelegten Zeitpunkt 01.01.1970 00:00 Uhr

- **Intern werden alle Zeitdaten-Typen wie DWORD behandelt!**

Erläuterungen zu den Datentypen INT, DINT und REAL

- Die Inhalte von Ganzahlen im Umfang von Byte, Wort und Doppelwort können im **Dezimal-Code**, **Dual-Code oder Hexadezimal-Code** angegeben werden.

- Das Automatisierungsgerät arbeitet grundsätzlich nur auf Basis des Dualcodes. Aus ihm werden alle anderen Codes abgeleitet.

- Der Hexadezimalcode dient zur Eingabe und zur Anzeige des Inhaltes von Datenelementen anstelle des unübersichtlichen Dualcodes. Dazu wird jeweils eine Tetrade von 4 Bit zu einer Hexadezimalziffer zusammengefasst.

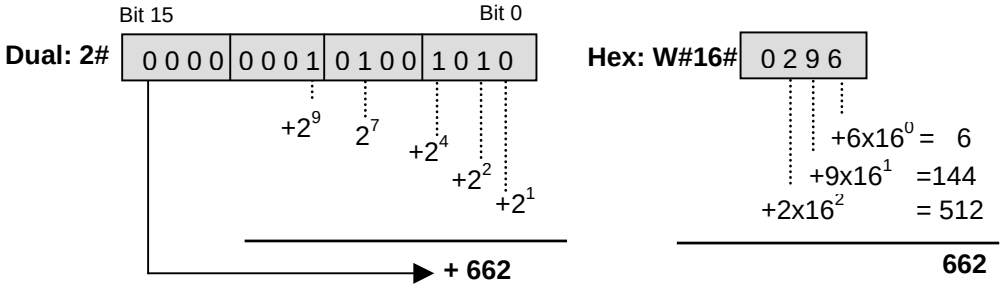
- Bei INT-Datentypen gilt: Das höchstwertige Bit enthält das Vorzeichen: Wert 0 = positiv, Wert 1 = negativ

- Negative Zahlen werden im Zweierkomplement angegeben. Zweierkomplemente ermittelt man durch Umkehr aller Bitwerte und Addition eines Bitwertes, wobei Überträge zu berücksichtigen sind.

- REAL werden nicht mit Komma, sondern mit Punkt eingegeben
  Um sowohl sehr kleine Kommazahlen wie z.B. 0.000000000000342 als auch große Kommazahlen wie z.B.1233231456. 012 mit 32 Bit verschlüsseln zu können, wird bei REAL das Prinzip des gleitenden Kommas benutzt. Intern wird dazu die exponentielle Darstellung der Zahlen und eine festgelegte Anzahl Bits jeweils für Mantisse und für Exponent verwendet (**Bild 4-24**).

Beispiele:

Dezimalzahl 662, darstellbar als INT := 662, im Dualcode 2# 0000_0001_0100_1010 oder im Hexacode W#16# 0296



Die Hexaziffern für die Darstellung von vier Bit:

Tetrade	Hex-Ziffer	Dezimal	Hex-Ziffer	Tetrade	Dezimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

Beispiel: 2# 1110_1010_0001_1000_1111_0111_1011_1001

DW#16# E A 1 8 F 7 B 9

Dezimal: 3 927 504 825

Bild 4-23: Erläuterungen zu Dual- und Hexadezimalcode

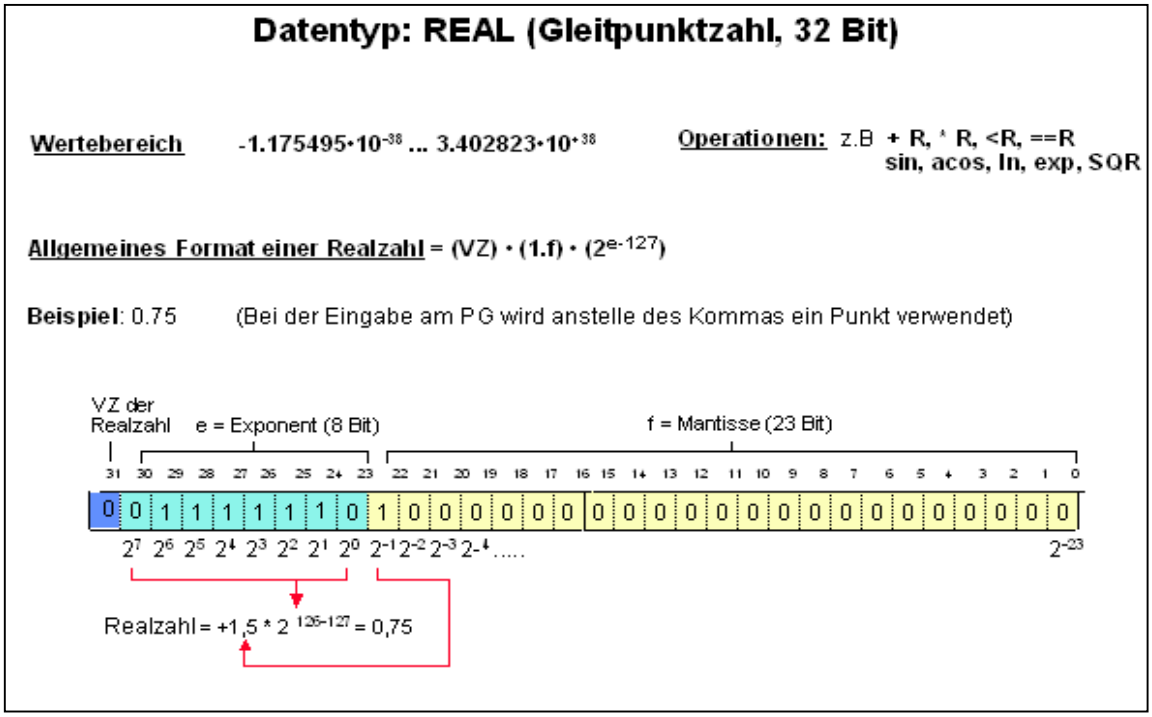


Bild 4-24: Prinzip der Verschlüsselung von 32 Bit-Gleitpunktzahlen
(Bildquelle: Siemens AG Lehrgang SPS-Techniker)

- **Binär codierte Dezimalzahlen (BCD-Code)**

Zifferncodierschalter und 7-Segment-Anzeigen enthalten Codewandler, die eine Tetrade in eine Dezimalziffer wandeln. Grundlage für die Darstellung einer Dezimalziffer durch vier Bit ist der Dualcode.

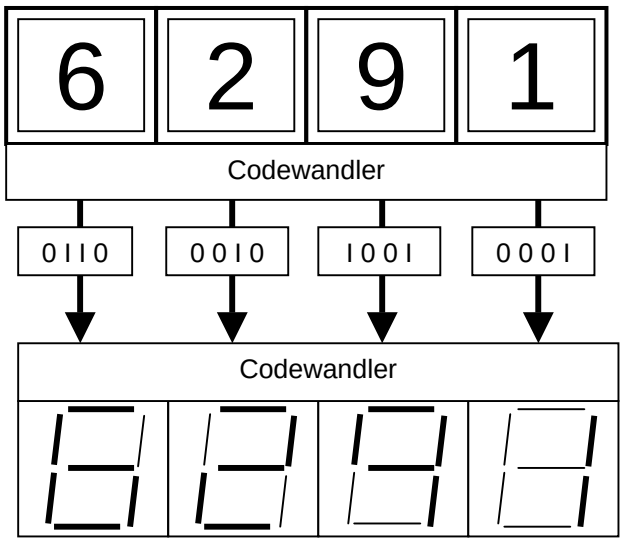
Der BCD-Code ist ein Code, bei dem eine Dezimalzahl nicht als Ganzes in den Dualcode überführt wird, sondern jede einzelne Dezimalziffer für sich.

Dezimalziffer	Dualcode der Dezimalziffer = BCD		Hexacode
0	2# 0000		16# 0
1	2# 0001		16# 1
2	2# 0010		16# 2
3	2# 0011		16# 3
4	2# 0100		16# 4
5	2# 0101		16# 5
6	2# 0110		16# 6
7	2# 0111		16# 7
8	2# 1000		16# 8
9	2# 1001		16# 9
Weiter verfügbare Tetraden	2# 1010	**In BCD nicht erlaubt!**	16# A
	2# 1011		16# B
	2# 1100		16# C
	2# 1101		16# D
	2# 1110		16# E
	2# 1111		16# F

Erkenntnis:
Im Bereich der Dezimalziffern 0..9 sind BCD- und Hexacode formal gleich.

Die Hexa-Ziffern A..F sind im BCD-Code nicht erlaubt!
Bei Simatic S7 wirken fehlerhafte BCD-Tetraden, die zu Ziffern A..F führen würden, als Stopp-Fehler!

Beispiel:



vierstellige Zahl im BCD-Code, dargestellt durch vier Tetraden:

0 I I 0_0 0 I 0_I 0 0 I_0 0 0 I

Im Gegensatz dazu ist der Dualcode von 6291:

2# 0 0 0 I_I 00 0_I 0 0 I_ 0 0 I I

Bild 4-25: Erläuterungen zum BCD-Code

Bei den speziellen Simatic **S5–Timern und S5-Zählern** – die auch in Step7 weiter verwendet werden -, liefern die Ausgänge „DEZ“ Inhalte im BCD-Code. Die Ausgänge „DUAL“ liefern die gleichen Inhalte im Dualcode.

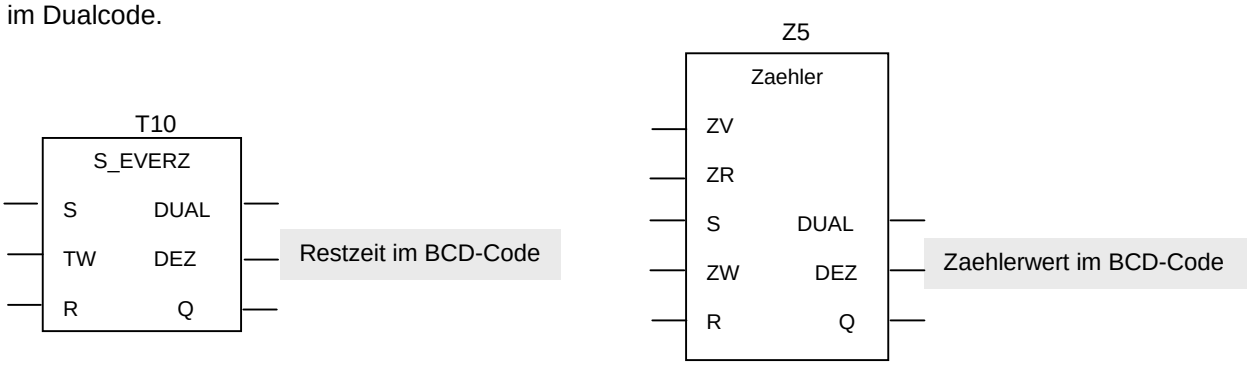


Bild 4-26: BCD-Code bei S5-Timern und S5-Zählern

So wie zunehmend Codierschalter und 7-Segment-Anzeigen durch Touch Panels ersetzt werden, verliert der BCD-Code an Bedeutung. BCD ist in IEC 61131-3 nicht explizit definiert!

- **Ansprechen einzelner Bit’s in Datenelementen:**

Einzelne Bits von elementaren Daten im Merker- oder Datenbaustein-Bereich spricht man mit der Bitadresse an, z.B.

Bit 3 des WORD „Teilezahl“ in MW 100: M 101.3
Bit 3 des WORD „Teilezahl“ in DB1.DBW2: DB1.DBX2.3

- **Remanenz von Daten**

Remanente Daten erhalten ihren Inhalt auch nach CPU in Stopp und nach Spannungsausfall. Remanent gestaltet werden können Merkerbereiche, Timer, Zähler und Bereiche von Datenbausteinen. Die Einstellung erfolgt in der HW-Konfig durch Parametrierung der CPU (**Bild 4-27**). Remanenz ist CPU-abhängig. Die CPU im Bild erlaubt z.B. keine remanente Daten in globalen Datenbausteinen.

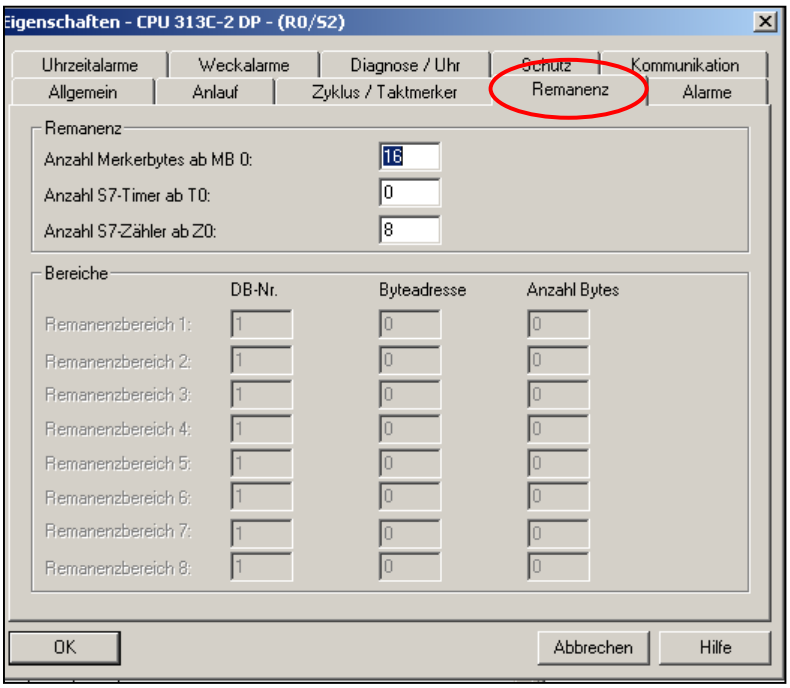
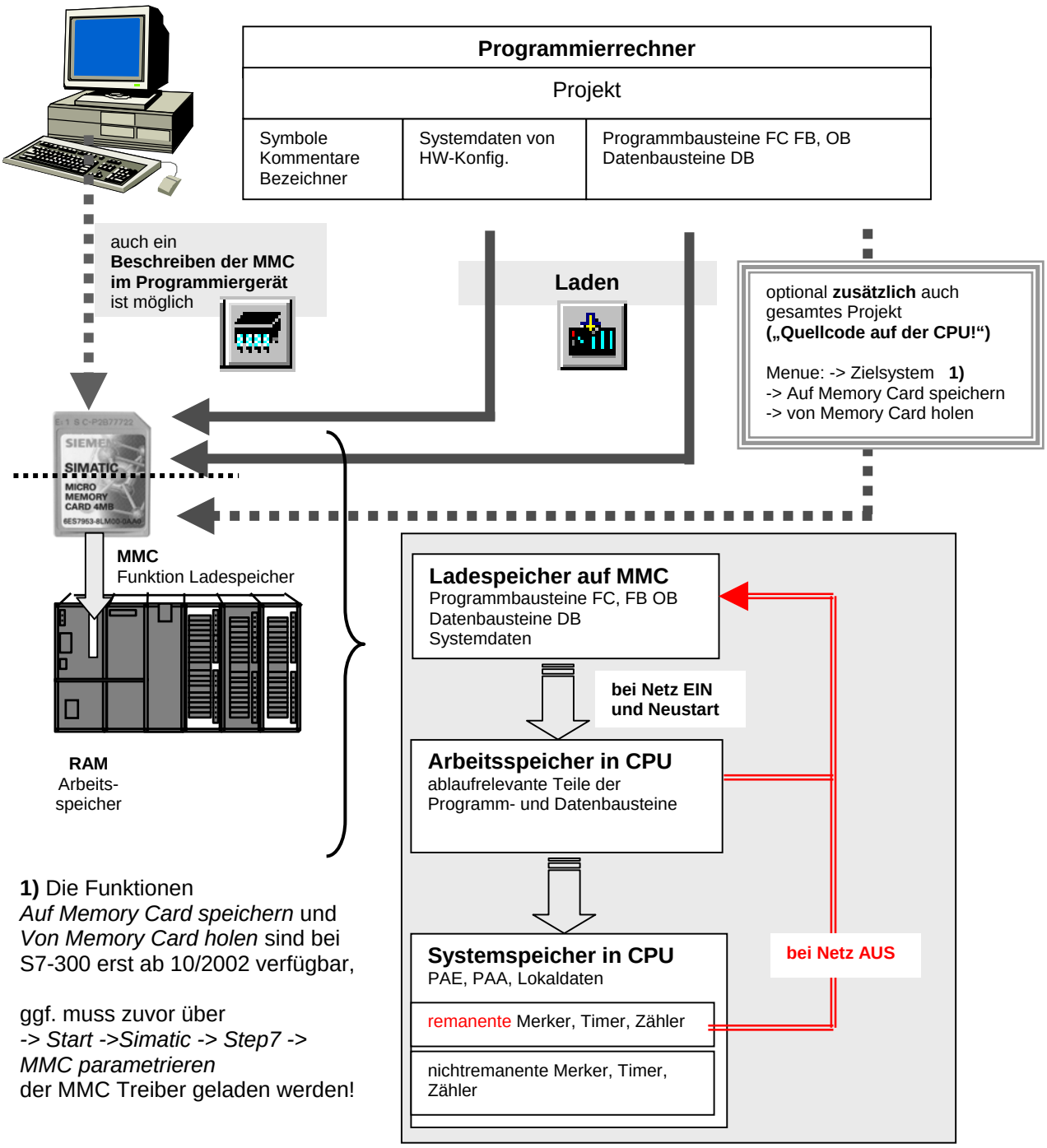


Bild 4-27 :
Parametrierung remanenter Datenbereiche in der CPU

4.2.10 Speicherkonzept S7-300

Seit IV / 2002 werden S7-300-CPU mit **Micro Memory Card** (MMC) ausgeliefert. MMC's übernehmen u.a. die Funktion des Ladespeichers, so dass solche CPU ohne MMC nicht funktionsfähig sind.

- Die MMC sichert die Remanenz von Daten. Der Inhalt der MMC ist grundsätzlich remanent. Damit entfallen die zuvor üblichen, service-aufwändigen Batterien für die Pufferung von Programmcode und remanenten Daten! Mit Einführung der MMC verlieren auch Flash-EPROM's und EEPROM's sowie die vormals üblichen EPROM's ihre Bedeutung!



1) Die Funktionen *Auf Memory Card speichern* und *Von Memory Card holen* sind bei S7-300 erst ab 10/2002 verfügbar,

ggf. muss zuvor über *-> Start ->Simatic -> Step7 -> MMC parametrieren* der MMC Treiber geladen werden!

Bild 4-28: Aktuelles Speicherkonzept im System S7-300

- Die CPU codiert nach Netz EIN und Neustart ablaufrelevante Bestandteile des Ladespeichers automatisch in ihren Arbeitsspeicher (RAM).
  Bestimmte Daten werden – getrennt in remanente und nichtremanente Daten – im Systemspeicher der CPU abgelegt.

- Bei Netzspannungsverlust werden aktuelle remanent definierte Daten auf die MMC geschrieben.

- Hat die MMC ausreichende Speichergröße, so kann unabhängig von ihrer unverzichtbaren Funktion als Ladespeicher das gesamte Projekt einschließlich der Symbolik und Kommentare dort zusätzlich abgespeichert werden.
  Vorteil: Der Servicetechnik findet vor Ort den aktuellen Quellcode des Projektes vor. Er lädt das Projekt von der MMC in das Programmiergerät, ändert und ergänzt soweit erforderlich und lädt veränderte Teile in die CPU (Ladespeicher auf MMC). Zusätzlich schreibt er am Ende das geänderte Projekt wieder zurück auf die MMC für einen nachfolgenden Service.

- Der Einsatz von MMC hat Auswirkungen auf die Durchführung des Urlöschens (Abschnitt)

- Im **System S7-400** sind einige Details dieses Speicherkonzepts geändert ausgeführt!

4.2.11 Symbolik

Step7 ermöglicht für alle globalen Operanden die Verwendung symbolischer Namen. Diese werden kurz **Symbole** genannt und in der **Symboltabelle** verwaltet. Bei symbolischer Programmierung vermeidet man das fehleranfällige und anstrengende Programmieren mit absoluten Operanden und deren Adressen. Die Programme werden leichter lesbar.

Die Zuordnung von Symbolen zu absoluten Operanden darf nicht mit der Variablendeklaration nach IEC 61131 verwechselt werden, auch wenn im Ergebnis ein symbolisch geschriebenes Programm gleiche Qualitäten aufweist. Hinter Symbolen stehen stets vom Programmierer festzulegende absolute Adressen, wogegen in IEC-Programmiersystemen die Adressen von Variablen ohne Zutun des Programmierers vom System selbst verwaltet werden. Nur im Sonderfall werden Variablen mit dem Schlüsselwort AT auf Adressen von Eingangs- und Ausgangskanälen gelegt.

Symbole werden vergeben für die Operanden im Merkerbereich, im Prozessabbild der Eingänge (PAE) und Ausgänge (PAA), für Peripheriebereiche und für Timer und Zähler. Weiter können alle Bausteine, u.a. auch die Datenbausteine Symbole erhalten. Die Daten in den Datenbausteinen selbst erhalten ihre Namen bei der Deklaration der Datenbausteine und werden nicht als Symbole füh t

Die Symboltabelle (**Bild 4-29**) ist wie die Bausteine Bestandteil eines Programms (**Bild 4-30**). Im Programmeditor kann die Symboltabelle über das Menu -> Extras -> Symboltabelle geöffnet werden.

Spindelantrieb (Symbole) -- BA_Spindel_2007\BA_Rack_313\CPU 313C-2 DP

	Status	Symbol	Adresse	Datentyp	Kommentar
1		Analoge_Wegsteueru...	FC 12	FC 12	
2		Antr_links	A 4.3	BOOL	
3		Antr_rechts	A 4.4	BOOL	
4		Betriebsarten	FC 1	FC 1	
5		Funktion_1	FC 11	FC 11	
6		Hand_Auto	E 1.0	BOOL	
7		Impulsgeber	E 1.1	BOOL	
8		Impulszähler	FC 10	FC 10	
9		Inbetriebnahme	FC 2	FC 2	
10		INI_links	E 0.5	BOOL	
11		INI_Lochl	E 0.7	BOOL	
12		INI_rechts	E 0.6	BOOL	
13		LED_Anl_ein	A 4.0	BOOL	

Bild 4-29: Auszug aus einer Symboltabelle, hier in alphabetischer Ordnung der Symbole

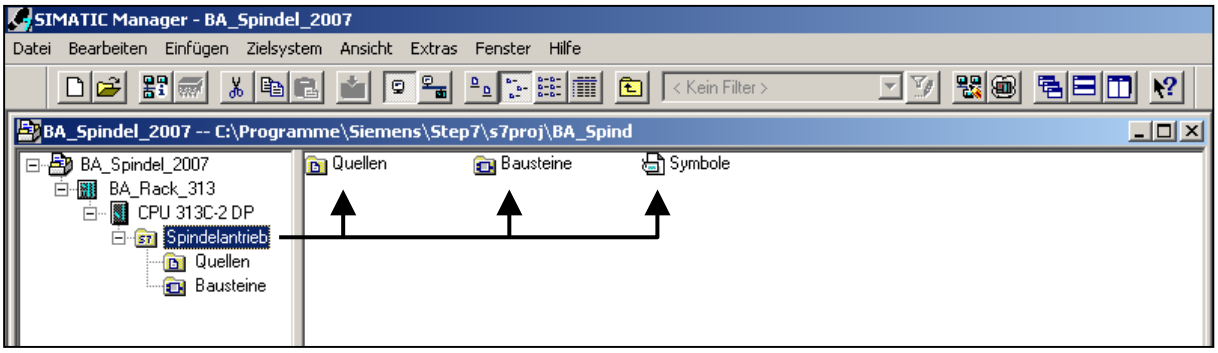


Bild 4-30: Die Symboltabelle ist Bestandteil eines Programms

Symbole erscheinen in Step7-Programmen eingebettet in doppeltes Hochkomma, z. B. “ T_Ein“ (**Bild 4-31**).

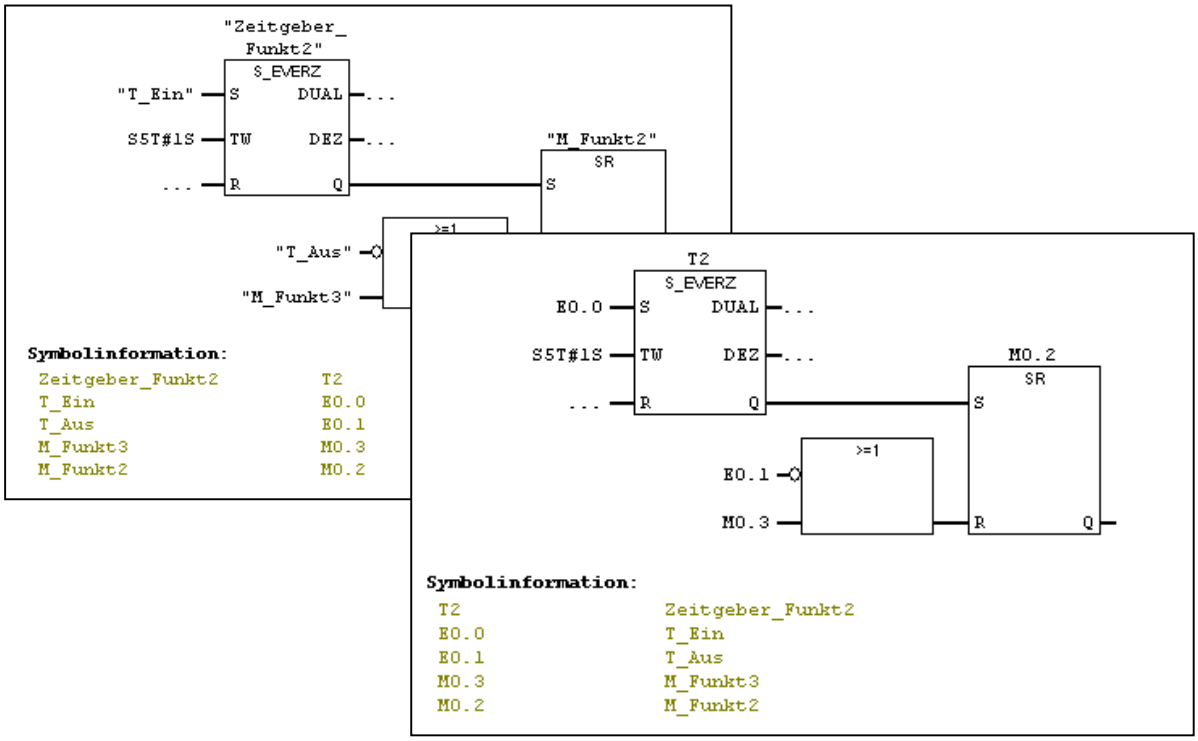


Bild 4-31: Detail eines Programms mit symbolischen Operanden (oben) und absoluten Operanden (unten). Zusätzlich werden alle erforderlichen Symbolinformationen aufgeführt.

Stellt man die Eigenschaften der Symboltabelle so ein, dass das Symbol Vorrang vor dem absolut adressierten Operanden hat (**Bild 4-32**), ergibt sich ein weiterer Vorteil der symbolischen Programmierung:
Eine erforderliche „**Umverdrahtung**“ auf andere Operanden erfolgt automatisch im gesamten Programm allein durch einmalige Änderung des Operanden in der Symboltabelle. Das unveränderte Symbol übernimmt in diesem Fall im gesamten Programm die dominierenden Funktion.
(Hinweis: Dieser Vorgang ist in der Wirkung gleichbedeutend mit einer einmaligen Adress-änderung einer IEC- Variablen bei ihrer Deklaration
z.B. T_Ein AT %IX2.0:BOOL; geändert in -> T_Ein AT % IX 4.0:BOOL;).

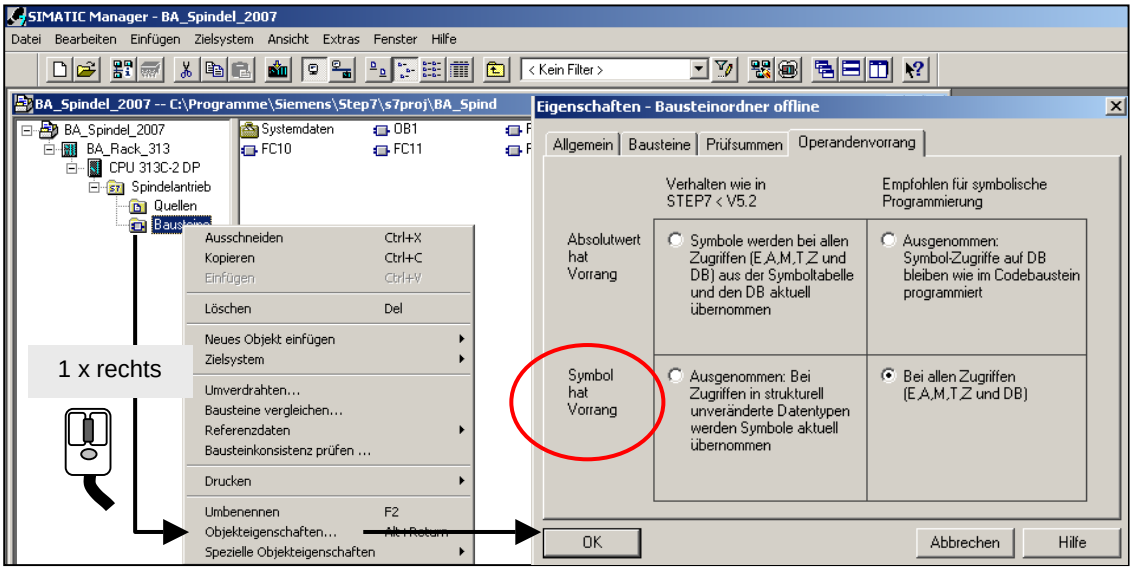


Bild 4-32 : Einstellung des Vorrangs der Symbolik

4.2.12 Verwendung lokaler Variablen TEMP und STAT

Neben den bisher verwendeten globalen Daten im Merkerbereich und in Datenbausteinen können in Organisationsbausteinen (OB), Funktionen (FC) und in Funktionsbausteinen (FB) mit Hilfe der **Deklarationstabelle** auch **lokale** Daten verwendet werden. Diese Deklaration ist im wesentlichen konform zu IEC 61131-3 (zu Unterschieden siehe auch Abschnitt 3.5).

Die Norm definiert die Variablen als *„Mittel zur Identifizierung von Datenobjekten, deren Inhalt sich ändern darf, d.h. Daten, die mit Eingängen, Ausgängen oder Speicherplätzen der SPS verbunden sind.“*

Lokale Variablen werden nach den Regeln der Variablendeklaration mit **Namen** (Bezeichner) **und Datentyp** deklariert. Im Step7-Programm erscheinen lokale Variablen mit dem vorgesetzten Kennzeichen # .

Alle lokalen Variablen gelten grundsätzlich **nur in dem Baustein**, in welchem sie deklariert wurden! Das bedeutet, dass in unterschiedlichen Bausteinen gleiche Namen für unterschiedliche lokale Variablen verwendet werden können. Auch die Symbolbezeichnungen können schadlos für lokale Variablen genutzt werden.

Funktionen und Organisationsbausteine erlauben nur **temporäre lokale Variablen** (TEMP). Eine temporäre Variable dient als Zwischenspeicher („Schmierzettel“), um Werte innerhalb eines Bausteins im Programmablauf „von oben nach unten“ weiter zu geben (Prinzip: erst schreiben, dann lesen!) Die Information der TEMP-Variablen steht nach einem Programmzyklus im allgemeinen nicht mehr zur Verfügung!

Funktionsbausteine erlauben neben temporären Variablen (TEMP) auch **statische lokale Variablen** (STAT). Diese werden in einem **Instanzdatenbaustein** verwaltet, so dass der Baustein über ein „Gedächtnis“ verfügt. Die Daten bleiben solange erhalten, bis sie mit neuen Werten überschrieben werden, auch über Zyklen und Bausteinaufrufe hinaus. FB’s sind deshalb stets mit zugeordnetem Instanzdatenbaustein zu verwenden.

Bild 4-33 zeigt die Deklaration von temporären Variablen in einer Funktion und weiter ein Programmdetail, in welchem diese zusammen mit globalen Variablen in symbolischer Darstellung verwendet werden.

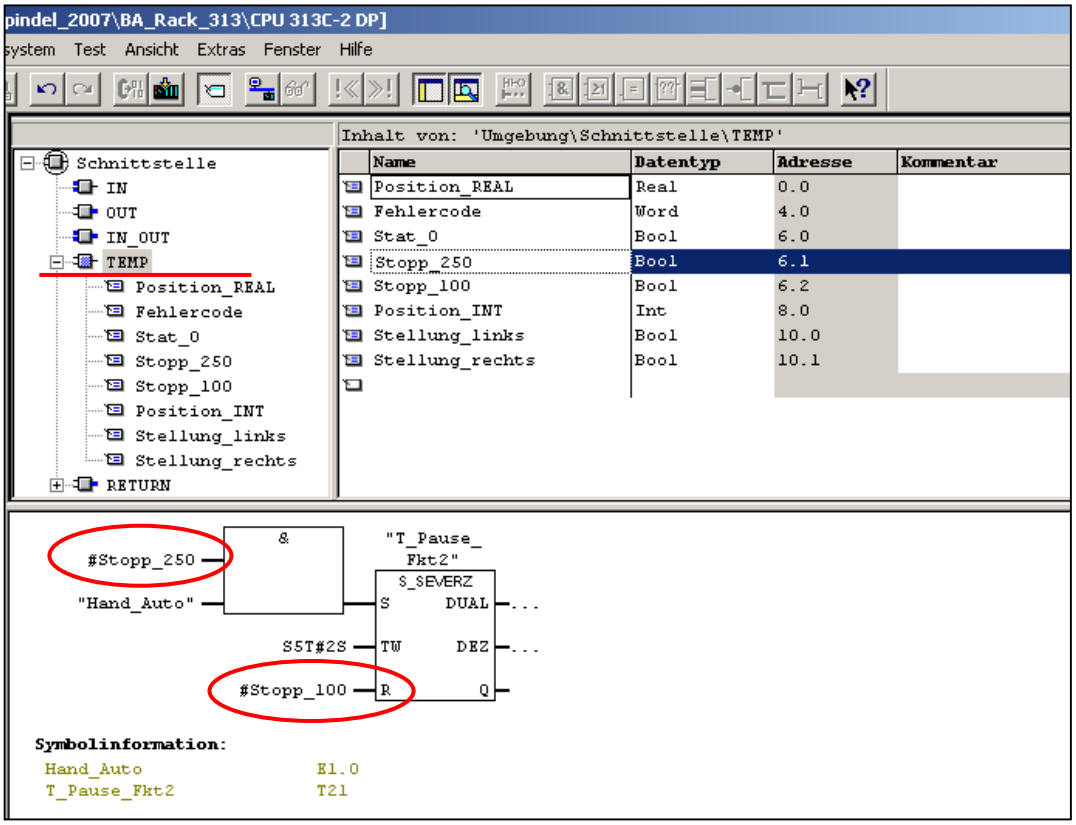


Bild 4-33: Deklaration und Verwendung lokaler temporärer Variablen in einer Funktion (FC)

Bild 4-34 zeigt, dass in Funktionsbausteinen (FB) neben temporären Variablen auch statische deklariert werden können.
Im Bild sind es die statischen Variablen *Gutteile:INT* und *Zaehler_Null:BOOL* und die temporäre Variable *Stopp:BOOL*.

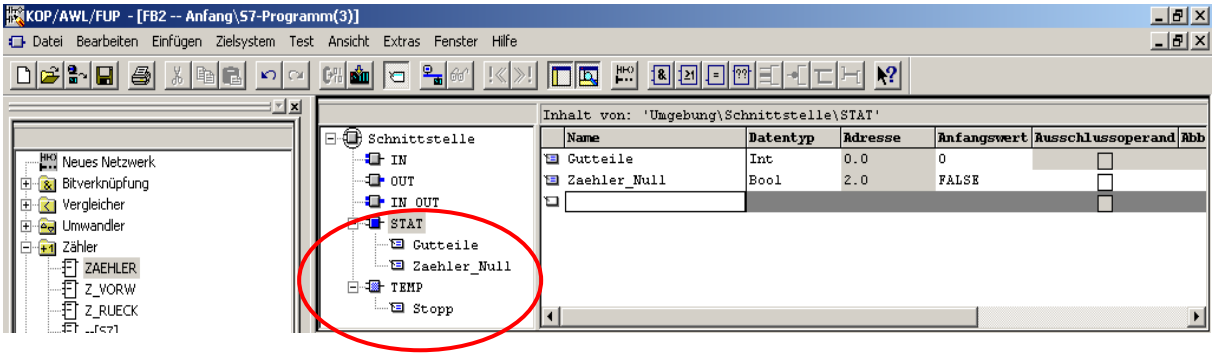


Bild 4-34: Deklaration lokaler Variablen in Step7: Der Funktionsbaustein erlaubt neben den Parametern

Variablen vom Typ STAT sind am ehesten anstelle von Merkern zu verwenden, allerdings nur dann, wenn diese Daten nur innerhalb eines Bausteins von Bedeutung sind. Bausteinübergreifende Daten müssen grundsätzlich globale Daten – also Merker oder Elemente von Datenbausteinen - sein!

4.2.13 Parametrierbare Bausteine

Sowohl Funktionen als auch Funktionsbausteine können **ohne jeglichen Hardwarebezug und ohne Bezug auf konkrete Operanden und Parameter** allein mit Variablen der Typen IN, OUT und IN_OUT geschrieben werden. Es entstehen parametrierbare Bausteine mit **formalen** Operanden. Beim Aufruf eines solchen Bausteins werden ihm konkrete **aktuelle** Operanden und Parameter übergeben.

Diese Vorgehensweise ist weitgehend konform zur Norm IEC 61131-3 und den dort verwendeten Variablen vom Typ VAR_INPUT, VAR_OUTPUT sowie VAR_IN_OUT. Allerdings erlaubt die Norm bei Funktionen nur Parameter vom Typ VAR_INPUT.

Variablen Typ	**Funktion der Variablen**
IN	**von außen kommend (gelesen), im Baustein nicht änderbar** Diese Variablen dienen zur Parameterübergabe beim Aufruf einer Funktion oder eines Funktionsbausteins. In der graphischen Darstellung stehen diese Variablen auf der linken Seite des Baustein-Symbols. Es können nur Werte des deklarierten Datentyps übergeben werden.
OUT	**vom Baustein nach außen geliefert (geschrieben)** Mit diesen Variablen gibt der parametrierbare Baustein Informationen zurück. In der graphischen Darstellung stehen diese Variablen auf der rechten Seite des Baustein-Symbols
IN_OUT	**von außen kommend (gelesen), kann innerhalb des Bausteins geändert werden** Mit diesen Variablen erhält der Baustein Eingangsinformationen und gibt diese ggf. geändert zurück. In der graphischen Darstellung stehen diese Variablen mit besonderer Kennzeichnung auf der linken Seite des Baustein-Symbols wie die Variablen des Typs IN.

Die Variablen der Typen IN, OUT und IN_OUT stellen die **Schnittstelle** eines parametrierbaren Bausteins zu anderen Programmteilen dar **(Bild 4-35**).

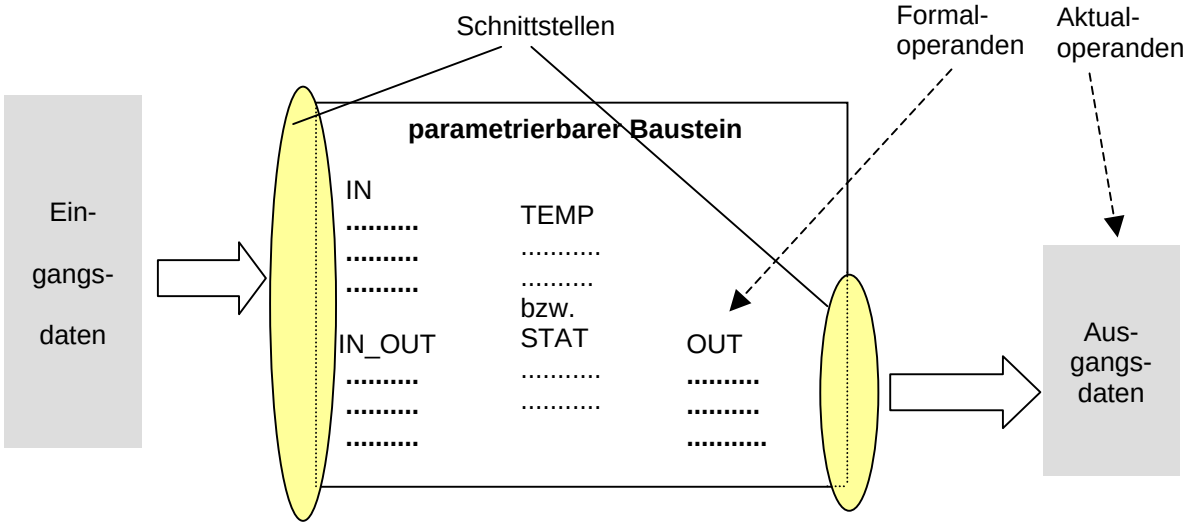


Bild 4-35: Demonstration des Begriffes Schnittstelle eines parametrierbaren Bausteins zu anderen Programmteilen.

Mit parametrierbaren Bausteine schafft sich der Programmierer **wiederverwendbare Programme**, wie es die Bibliotheken sind. Vom Grundsatz her sind alle Bibliotheksbausteine parametrierbare Bausteine.

Hinweis:

In **Funktionen** werden interne Werte, die nicht nach außen in Erscheinung treten müssen, als temporäre Variablen TEMP deklariert. Allerdings besteht die bekannte Einschränkung, dass TEMP's ihre Werte innerhalb eines Programms nur „von oben nach unten" weiter geben können (Prinzip: erst schreiben, dann lesen!)

Kann diese Einschränkung nicht akzeptiert werden, muss man bei Funktionen Variablen IN_OUT deklarieren, auch wenn die internen Werte nicht nach außen in Erscheinung treten müssen. Man übergibt dann Merker als Aktualparameter.

In **Funktionsbausteinen** kann man dagegen in solchen Fällen anstelle der temporären Variablen statische Variablen STAT deklarieren. Die Zahl der erforderlichen Schnittstellenvariablen verringert sich dadurch! Deshalb sind auch bei Step7 parametrierbare Funktionsbausteine vorteilhafter einzusetzen.

4.2.14 Die Verwendung von Bibliotheken

Eine Vielzahl oft benötigter Programmdetails liegen als fertige Programmelemente in Bibliotheken bereit. Der Form nach sind es parametrierbare Bausteine, die ihre aktuellen Variablen durch Parameterübergabe mit Variablen der Typen VAR_INPUT, VAR_OUTPUT und VAR_IN_OUT erhalten.

Die wichtigste Bibliothek ist die Standardbibliothek (Standard Library), die nach IEC 61131-3 allen Programmiersystemen beigefügt werden muss (**Bild 4-36**).

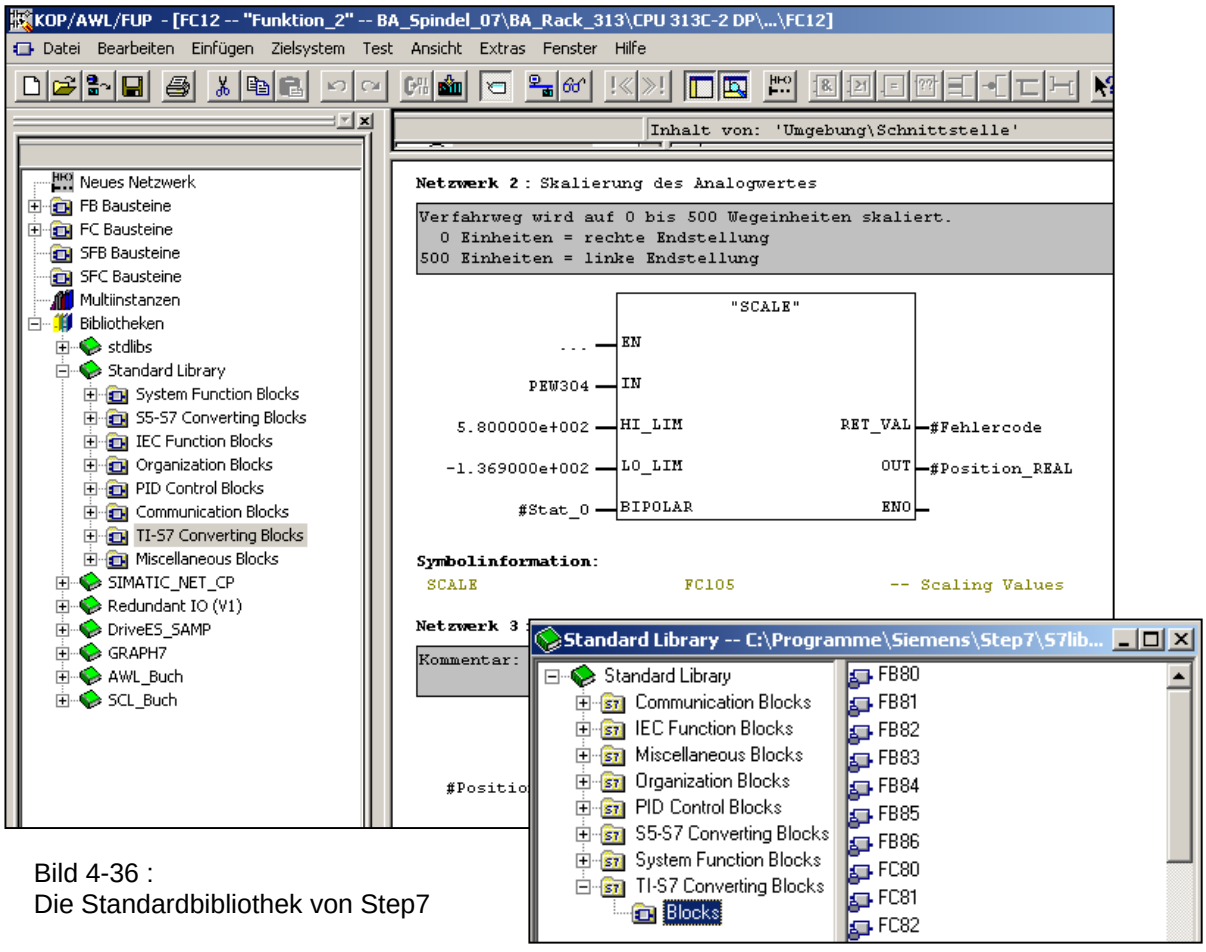
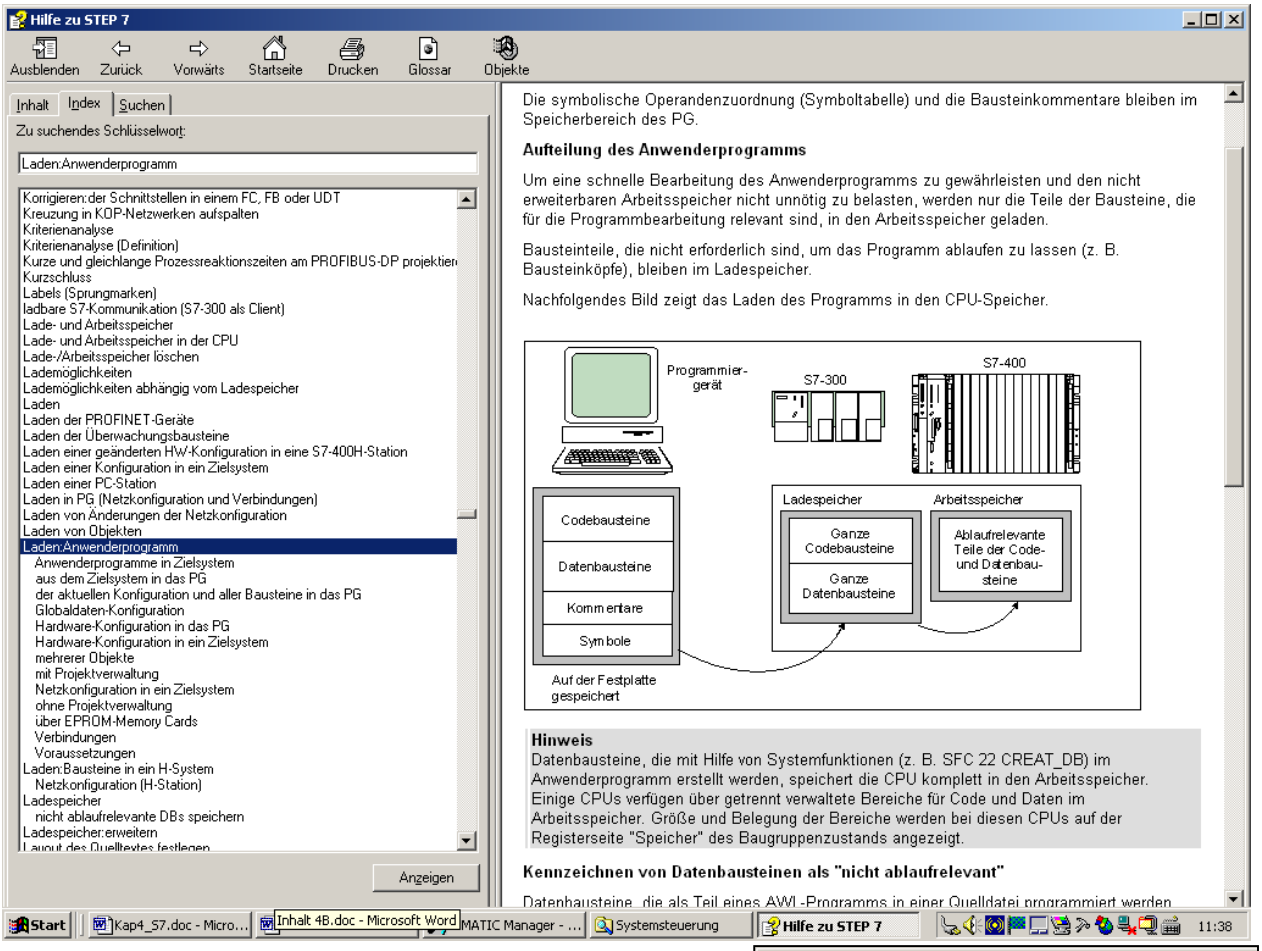


Bild 4-36 :
Die Standardbibliothek von Step7

4.2.15 Online-Hilfe als Lernmedium

Step7 verfügt über eine ausgezeichnete, vielfältige und vielseitig anwendbare Online-Hilfe. Ein Beispiel zeigt **Bild 4-37.** Die Online-Hilfe übertrifft bei weitem jedes andere Lehrmittel!



Darüber hinaus kann bei der Programmierung nach dem Markieren mit Taste F1 jederzeit Auskunft zu allen fraglichen Unklarheiten eingeholt werden.

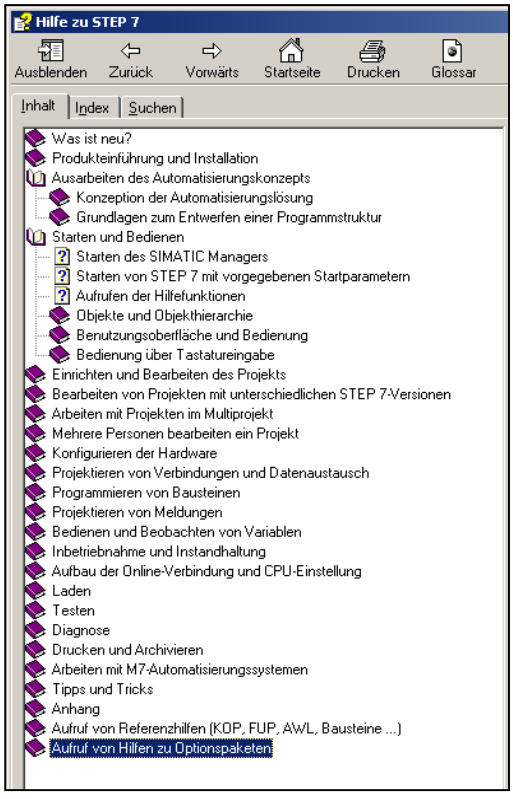


Bild 4-37: Lehrbuchqualität der Online-Hilfe, oben ein Beispiel der Auswahl nach einem Index, rechts Angebot aller Themen.

4.2.16 Grundlegende Online-Funktionen

- **Zugang zur CPU festlegen**

Die Kommunikation zwischen Programmierrechner / Step7 und CPU erfolgt mit dem Kommunikationsprozessor CP 5611. Dieser kann als Rechnerkarte oder als spezieller Leitungs-Adapter mit USB-Schnittstelle vorliegen.

Aus verschiedenen Menue's heraus kann die Parametrierung des CP 5611 erfolgen, z.B. im Simatic-Manager mit -> *Extras -> PG/PC-Schnittstelle einstellen*.

Bild 4-38 zeigt beispielhaft die Einstellmöglichkeit des Adapters auf MPI-, PPI oder Profibus-Protokoll. Die Leitung selbst ist davon unabhängig gleichbleibend eine Zweidrahtleitung entsprechend Schnittstelle RS 485. Unter -> *Eigenschaften* werden hier weitere Parameter wie Übertragungsrate u.a. festgeschrieben.

Für den Einstieg sinnvoll:
Protokoll MPI (Mehrpunkt-Interface) mit 187,5 kBit/s.

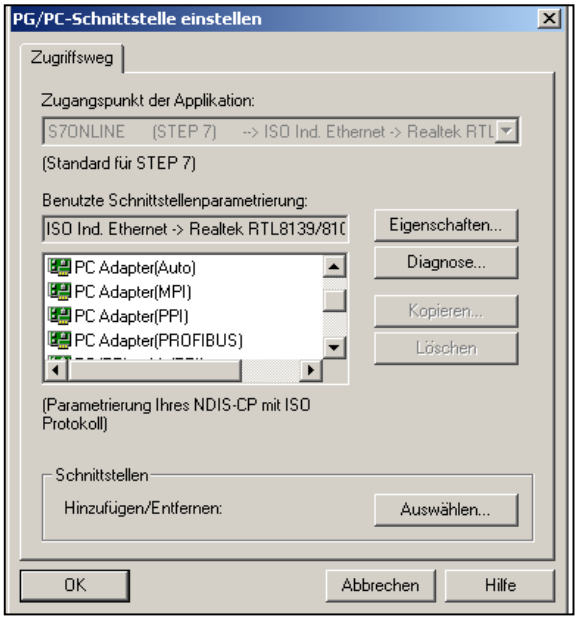


Bild 4-38:
Einstellung des Protokolls für den Online-Zugang zur CPU im Menue *PG/PC-Schnittstelle einstellen*

- **Erreichbare Teilnehmer anzeigen**

Den erfolgreichen Online-Zugang zur CPU testet man in verschiedenen Menue-Ebenen z.B. mit -> *Zielsystem -> Erreichbare Teilnehmer anzeigen.*

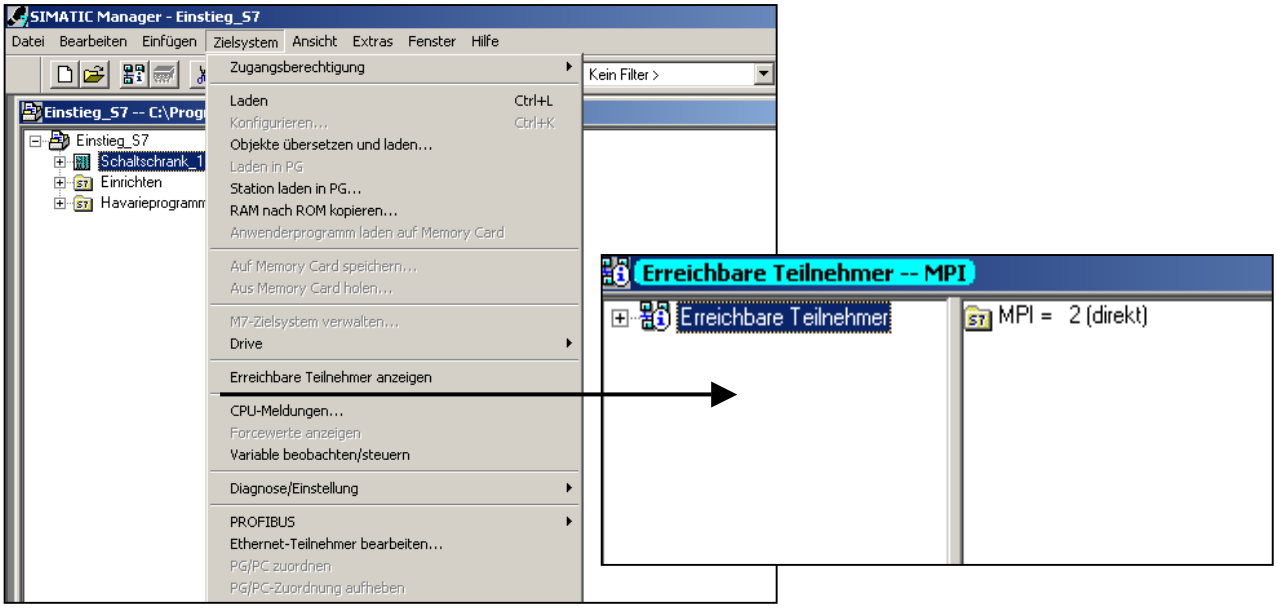


Bild 4-39: Test des Online-Zugangs zur CPU

- **Urlöschen**

Urlöschen löscht die Inhalte aller Speicher in der CPU. Das Löschen der Systemdaten als Abbild der HW-Konfiguration ist optional im Rahmen des Urlöschens ebenfalls möglich.

Mit **Einsatz der Micro Memory Card** als Ladespeicher der CPU erfordert das Urlöschen zwei Schritte:

1. Das Löschen aller Programmbausteine (OB, FC, FB und DB) auf der MMC in der Online-Sicht des Simatic-Managers (**Bild 4-40**)

2. Das nachträgliche Urlöschen von Arbeits- und Systemspeicher der CPU im Menue *-> Zielsystem -> Diagnose/Einstellung -> Urlöschen* (**Bild 4-41**)

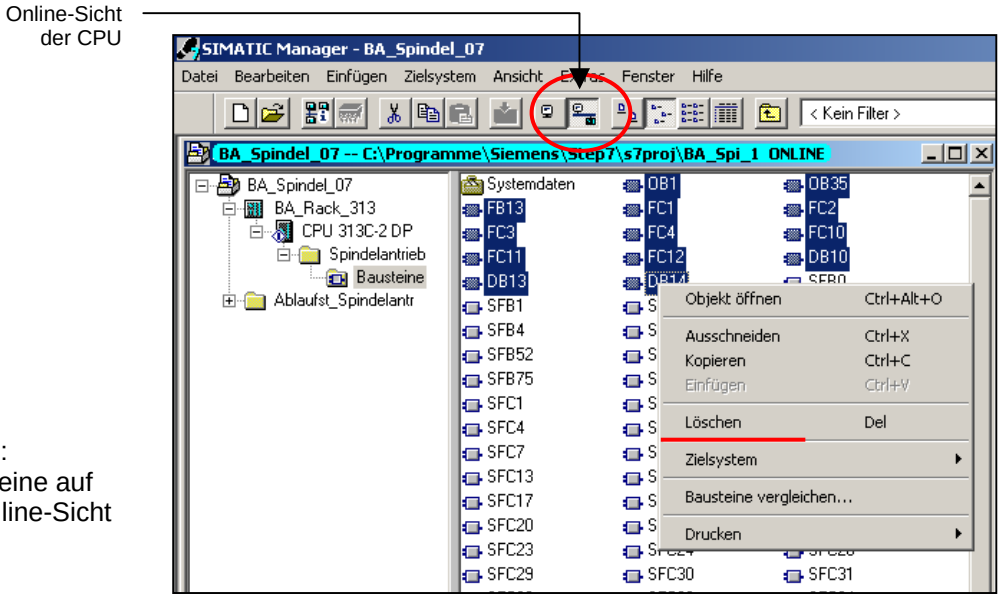


Bild 4-40:
Urlöschen Schritt 1:
Löschen der Bausteine auf der MMC in der Online-Sicht der CPU

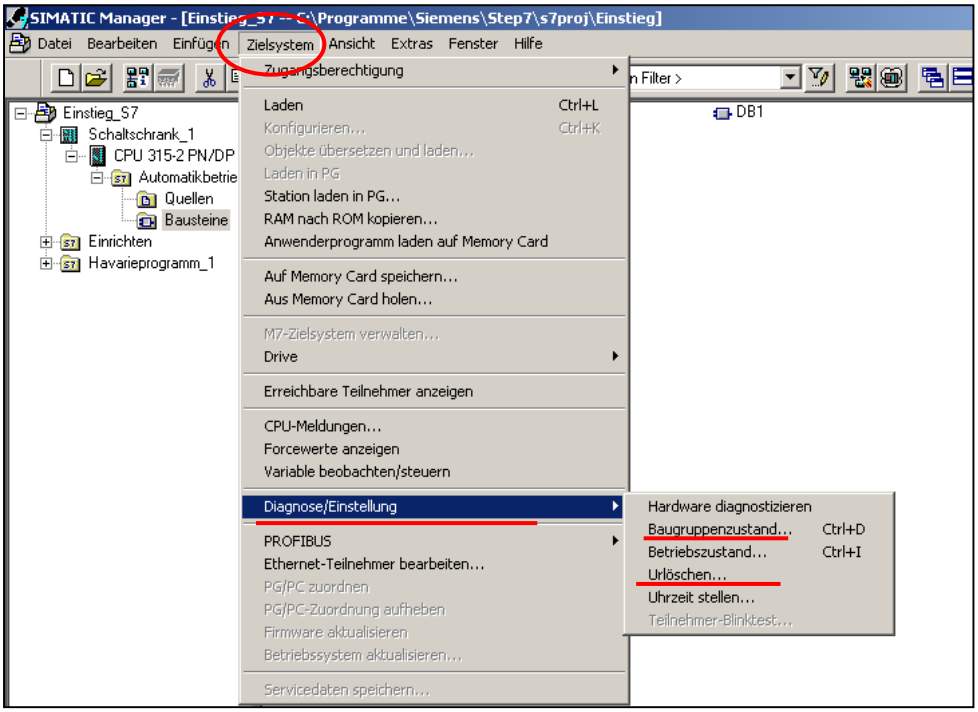


Bild 4-41 : Urlöschen Schritt 2: Löschen der RAM-Speicher der CPU über Menue -> *Zielsystem*

- **Baugruppenzustand der CPU**

Im Menue -> *Zielsystem -> Diagnose/Einstellung* findet sich auch der Aufruf des -> *Baugruppen-zustand* (**Bild 4-42**). Dieser ist nicht zu verwechseln mit -> *Betriebszustand*, wo allein Start und Stop der CPU veranlasst werden kann.

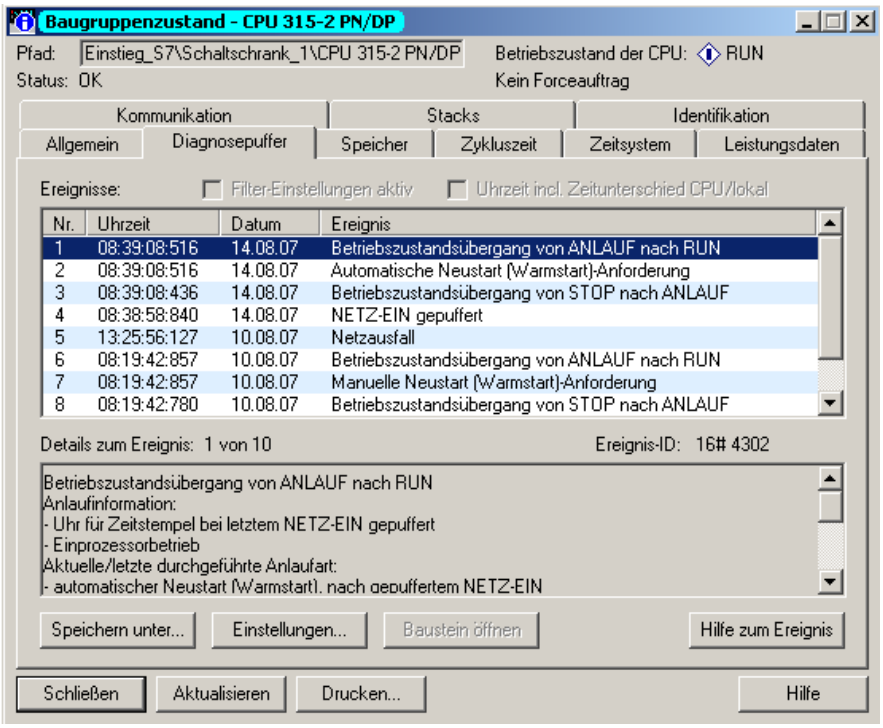


Bild 4-42: Online-Sicht *Baugruppenzustand*, bei dem das Untermenue *Diagnosepuffer* geöffnet wurde

Der Baugruppenzustand offeriert eine Vielzahl Untermenue's, die für S7-Servicetechniker von Bedeutung sind und Teil der excellenten Diagnosefunktionen von Simatic S7 sind. Einige Beispiele sind:
Leistungsdaten geben Auskunft über verfügbare Parameter der CPU
Diagnosepuffer und *Stack's* unterstützen die Fehlersuche
Zykluszeit gibt Auskunft über die aktuelle Zykluszeit und die Zyklusüberwachungszeit
Speicher zeigt den Auslastungsgrad der internen Datenspeicher

- **Station laden in PG**

Mit dem Menue ->*Zielsystem -> Station laden in PG* kann eine komplette Station, bestehend aus HW-Konfiguration und den in der CPU abgelegten Programmbausteinen, in ein Projekt auf dem Programmierrechner übertragen werden. Danach ist ein Umparametrieren der Hardware möglich. Die Funktion fügt stets eine neue HW-Station in das Projekt ein, d.h. vorhandene Stationen werden nicht überschrieben.

Man kann auch lediglich bestimmte oder alle Programmbausteine aus der CPU auf den Rechner übertragen. Dazu sind diese Bausteine in der Online-Sicht zu markieren.

Die Funktion verlangt die Angabe der MPI-Adresse der CPU, welche erreicht werden soll (**Bild 4-43)**

Achtung! Diese Funktion ist **kein Alibi** für „verloren gehende" Quellcodes von Programmen. Symbole, Kommentare und die Bezeichner der Daten in Datenbausteinen sowie der lokalen Variablen STAT und TEMP werden nicht in die CPU übertragen und sind dann verloren (siehe Speicherkonzept Abschnitt 4B.10)!

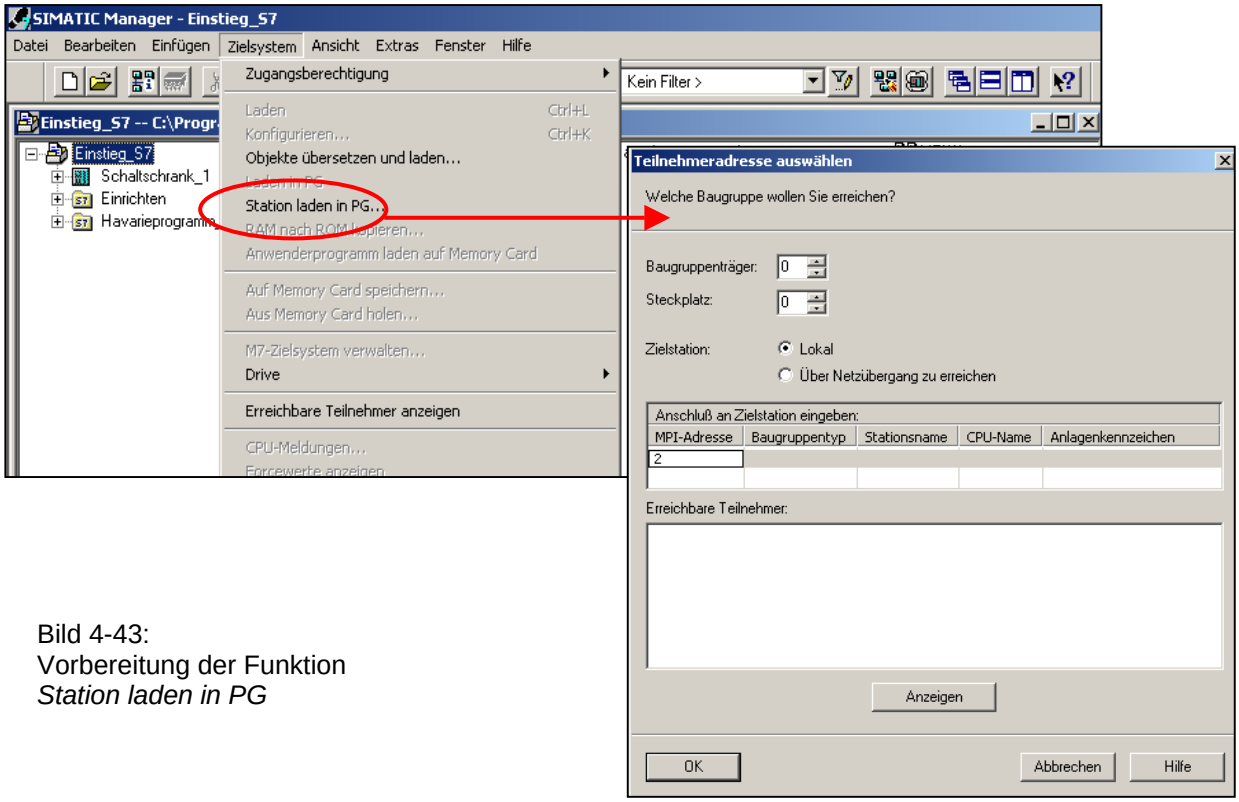


Bild 4-43:
Vorbereitung der Funktion
Station laden in PG

4.2.17 Laden und Testen von Programmen

Durch Markieren des Bausteinordners eines Programms oder aber ausgewählter Programmbausteine wird entschieden, ob alle oder nur einzelne Bausteine in die CPU geladen werden. Weiter ist jeweils zu entscheiden, ob auch die Systemdaten geladen werden sollen.

Im Beispiel des **Bildes 4-44** werden die Systemdaten und allein der Baustein FC1 geladen.

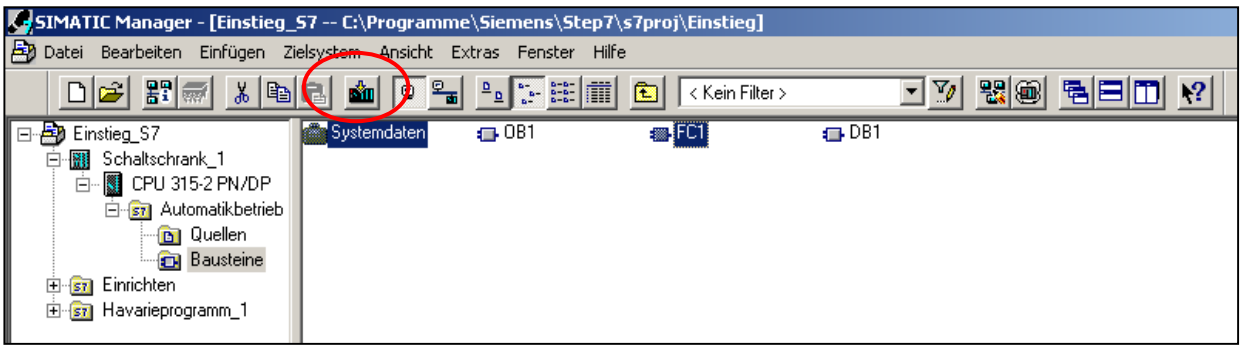


Bild 4-44: Laden von Programmelementen in die CPU

Das Programm eines zyklisch bearbeiteten Bausteins kann im „Programmstatus“ beobachtet werden (**Bild 4-45**). Je nach Farbeinstellung im Editors erscheinen durchgeschaltete Signalflüsse farbig, im Beispiel grün. Weiter wird der Wert von Operanden angezeigt, im Beispiel Wert 0 oder 1 bei Boole-schen Operatoren.
Zu beachten ist, dass im Programmstatus stets nur ein aktuelles Detail des Programms beobachtet wird, nicht aber das Ergebnis am Ende der zyklischen Bearbeitung des Gesamtprogramms! Dieses kann nur in der Variablentabelle beobachtet werden.

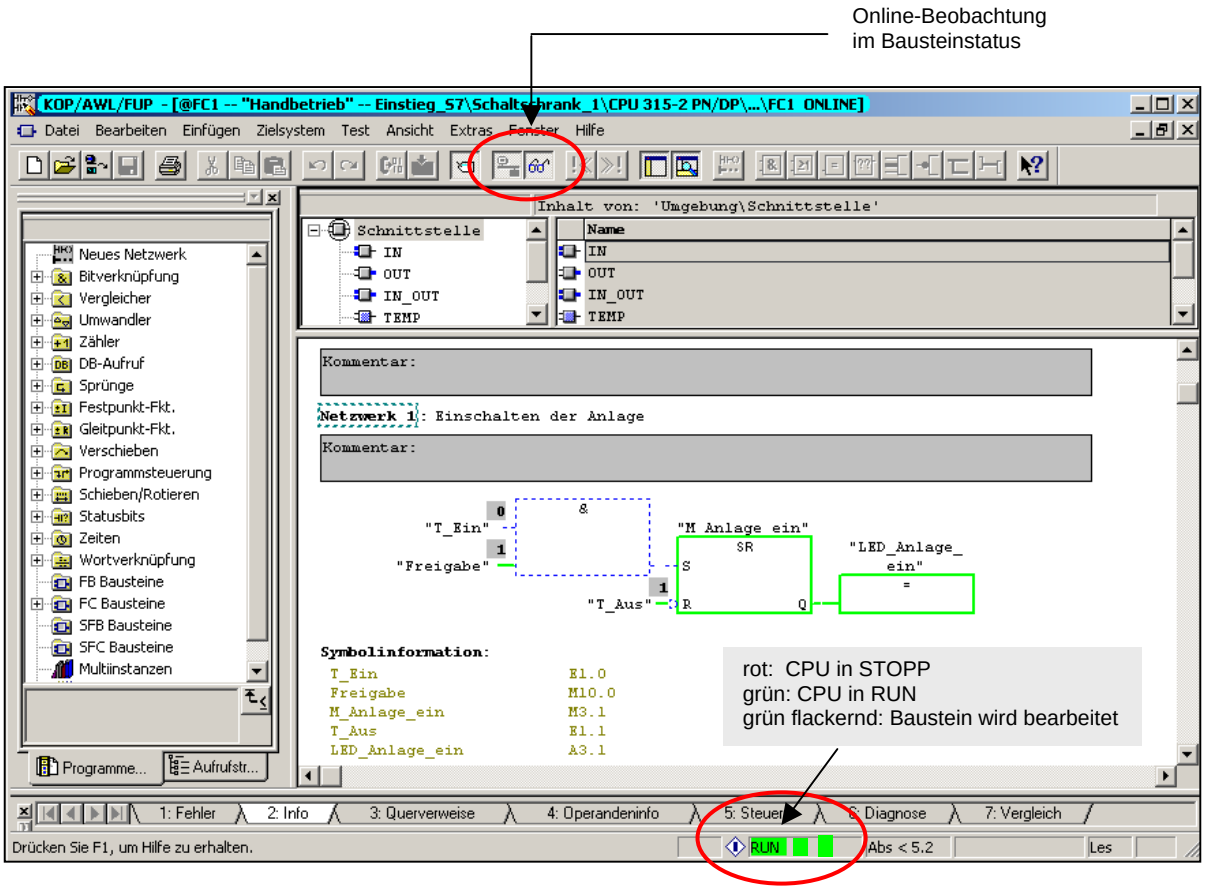


Bild 4-44: Online Bausteinbeobachtung in FUP Bausteinstatus

4.2.18 Beobachten und Steuern von Operanden mit der Variablentabelle VAT

Variablentabellen (VAT) erlauben das Beobachten der Werte von Operanden (E, A, M, T, Z sowie Peripherie PE und PA) und das Steuern von Ausgangssignalen. Die VAT werden wie Bausteine FC, FB, OB oder DB in den Bausteinordner eines Programms eingefügt und bei Bedarf geöffnet. VAT's können symbolisch benannt werden (**Bild 4-45**).

In einer VAT stellt man jeweils die Operanden zusammen, welche man beobachten oder steuern will. (**Bild 4-46**): Im Gegensatz zum Bausteinstatus – welcher den Wert von Operanden zu einem bestimmten Zeitpunkt der Programmbearbeitung liefert - zeigt die Beobachtung in der VAT tatsächliche und endgültige Werte. Allerdings müssen diese wiederum von Triggereinstellungen aus bewertet werden.

Beim Steuern werden in der VAT Steuerwerte vorgegeben und mit dem Button *Steuern* aktiviert.

Für den Einstieg empfohlen:
Beobachten: Permanent und am Beginn der zyklischen Programmbearbeitung
Steuern: Permanent und am Ende der zyklischen Programmbearbeitung

Daten in Datenbausteinen können mit VAT nicht beobachtet werden. Diese kontrolliert man in der Datensicht der Datenbausteine.

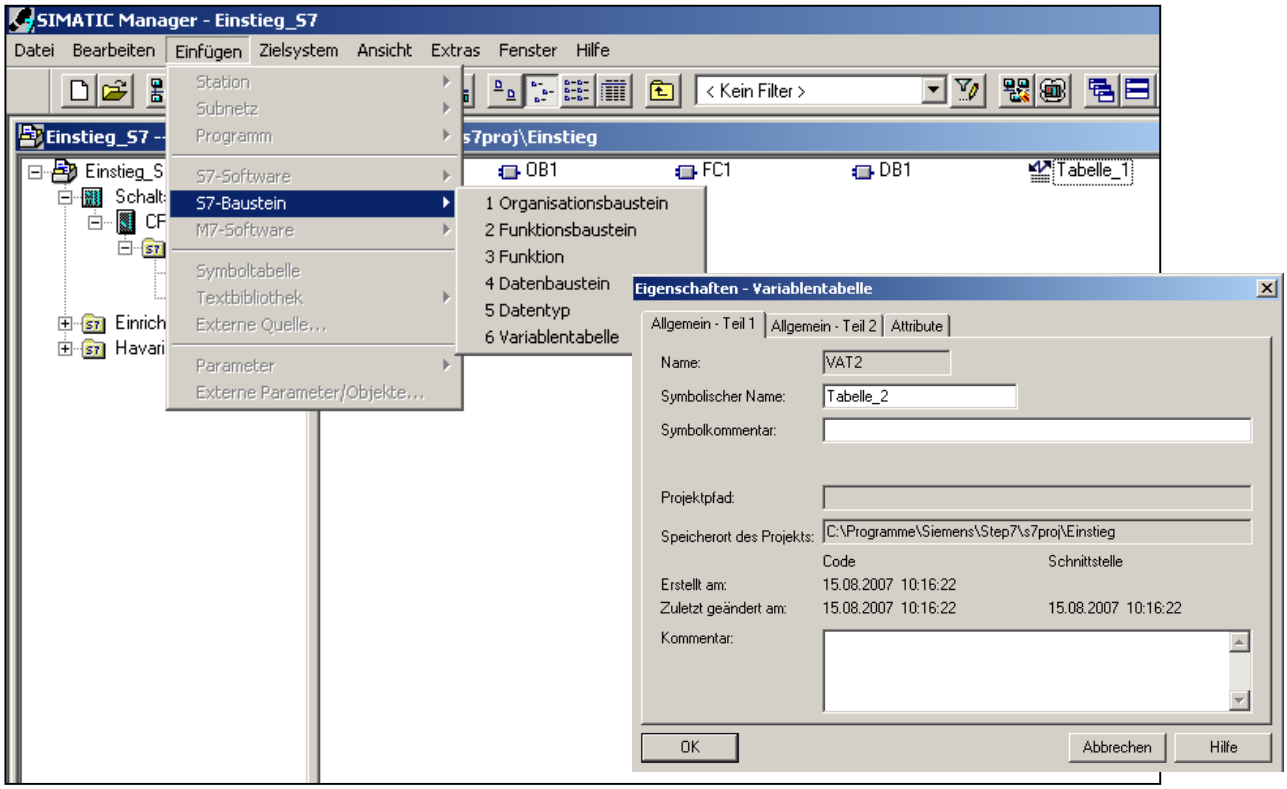


Bild 4-45 : Einfügen einer Variablentabelle als Baustein in ein Programm

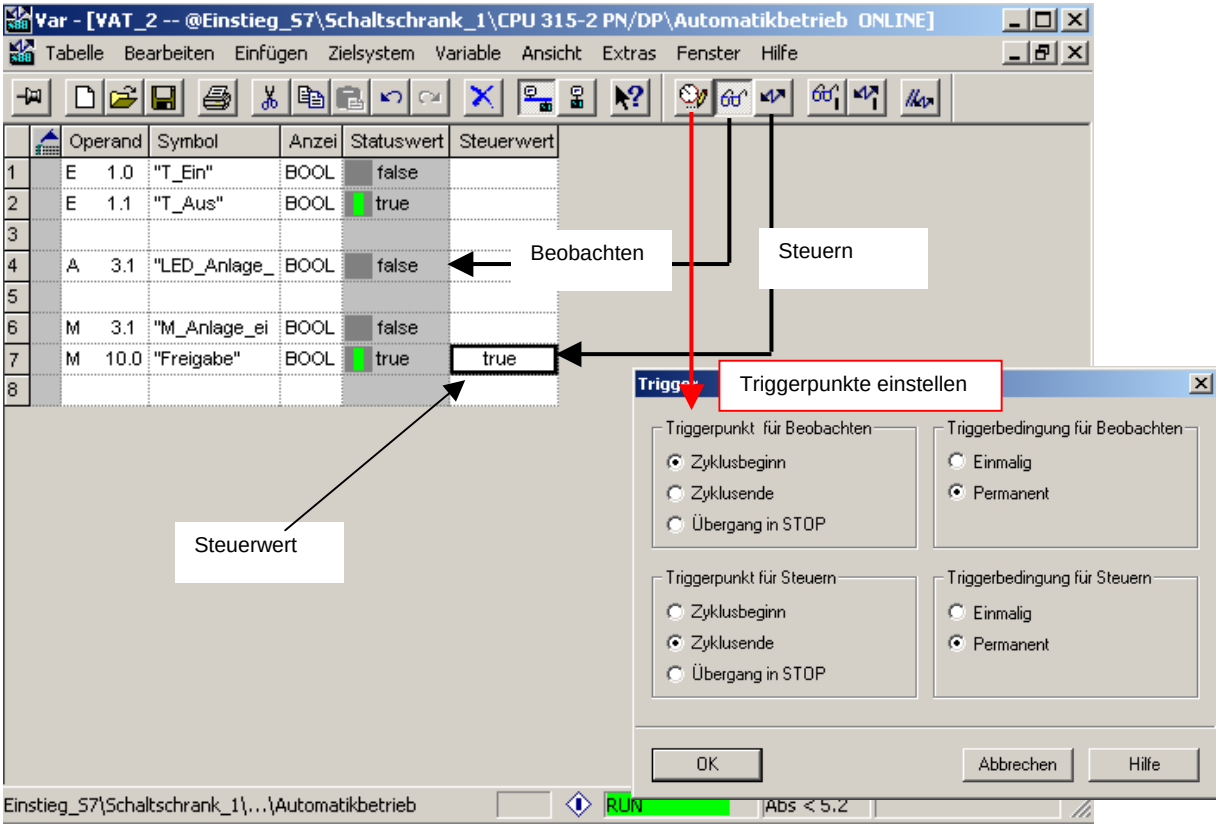


Bild 4-46: Beobachten und Steuern Boolescher Operanden in der Variablentabelle, rechts die für den Einstieg empfohlenen Triggereinstellungen.