- Ein- und Ausgänge (jeweils das Prozessabbild zu Beginn und am Ende eines Zyklus)
- Merker bzw. Merkerbereiche
- Bereiche in Datenbausteinen
- Zeiten und Zähler

Um Globaldaten festlegen zu können muss zunächst in NetPro der MPI-Bus markiert werden. Dann kann über „Extras > Globaldaten definieren“ das Fenster für die mit der GD-Tabelle des MPI-Busses geöffnet werden.

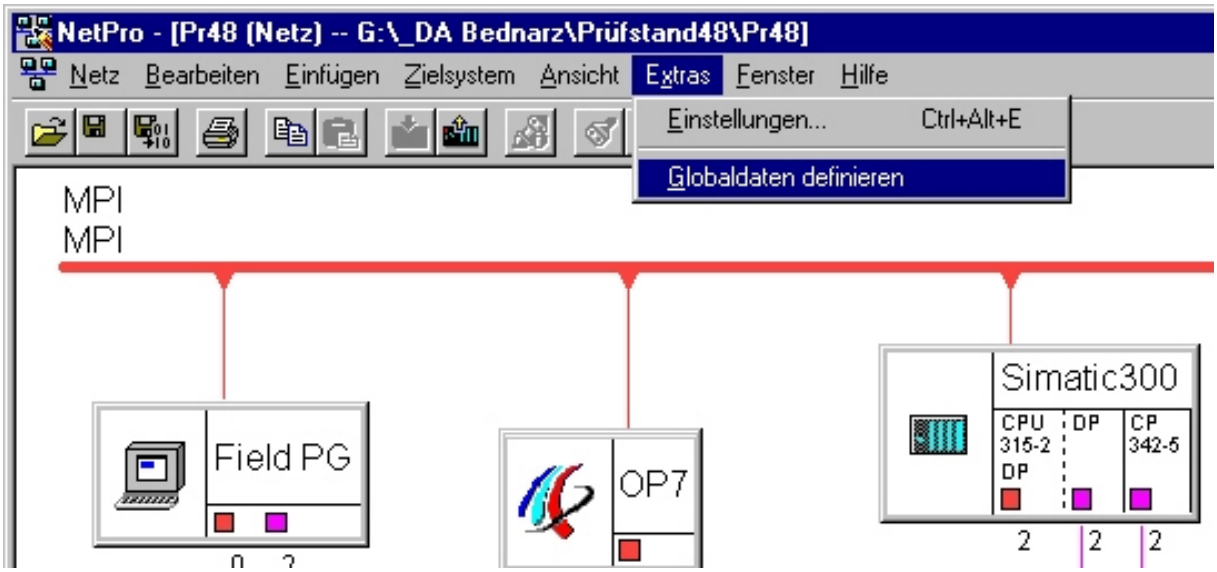


Abbildung 7.3-11: Globaldaten definieren

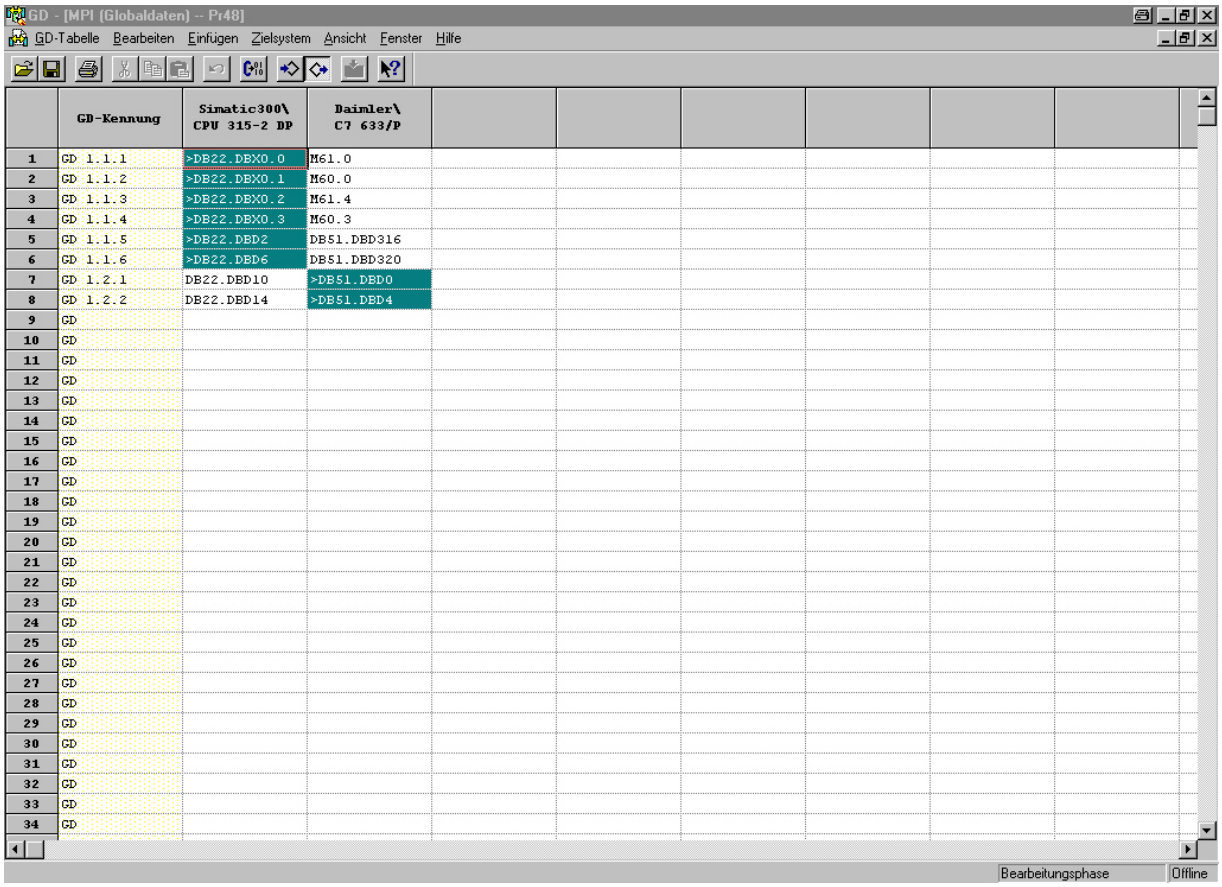


Abbildung 7.3-12: GD-Tabelle

In den Spalten der GD-Tabelle werden die an der GD-Kommunikation beteiligten CPUs eingetragen. Die Spalte GD Kennung wird beim Übersetzen und Speichern (beides unter dem Menüpunkt GD-Tabelle) automatisch erzeugt.

In die Zeilen trägt man dann die Globaldatenbereiche ein. Pro Zeile kann jeweils ein Eintag grün markiert werden. Die Markierung kennzeichnet jeweils den Ursprung der Daten (Sender). Pro Zeile kann jeweils nur eine CPU der Sender sein, die anderen CPUs empfangen nur. Eine Empfangsbestätigung wird nicht gesendet. Die GD habe ich prüfstandsseitig alle in Datenbaustein 22 (DB22) hinterlegt, während in der Klimaschranksteuerung auf sowohl auf einen Datenbaustein, als auch auf Merker zugegriffen wird. Die Adressbereiche der Klimaschranksteuerung habe ich der Programmdokumentation des Herstellers entnommen. Im Steuerprogramm von L&R war bereits die Vorgabe von externen Sollwerten vorgesehen und man hat dort entsprechenden Speicherplatz reserviert.

Die Globaldaten sind wie folgt aufgebaut: in Zeile 1-6 ist jeweils die Prüfstand-CPU der Sender, in Zeile 7 und 8 ist es die Klimaschranksteuerung die sendet.

GD - [MPI (Globaldaten) -- Pr48]

GD-Tabelle Bearbeiten Einfügen Zielsystem Ansicht Fenster Hilfe

	GD-Kennung	Simatic300\ CPU 315-2 DP	Daimler\ C7 633/P
1	GD 1.1.1	>DB22.DBX0.0	M61.0
2	GD 1.1.2	>DB22.DBX0.1	M60.0
3	GD 1.1.3	>DB22.DBX0.2	M61.4
4	GD 1.1.4	>DB22.DBX0.3	M60.3
5	GD 1.1.5	>DB22.DBD2	DB51.DBD316
6	GD 1.1.6	>DB22.DBD6	DB51.DBD320
7	GD 1.2.1	DB22.DBD10	>DB51.DBD0
8	GD 1.2.2	DB22.DBD14	>DB51.DBD4
9	GD		
10	GD		
11	GD		
12	GD		

Abbildung 7.3-13: GD

Im Steuerungsprogramm wird jetzt also nur noch mit dem von mir programmierten Datenbaustein 22 gearbeitet, die GD-Tabelle sorgt dafür, dass die Daten in der Klimaschranksteuerung an die richtige Stelle geschrieben bzw. an der richtigen Stelle gelesen werden.

Beispiel Zeile 5:

Hinter Zeile 5 verbirgt sich der Temperatursollwert, der an das Aggregat geschickt werden soll. Wird nun durch den Bediener eine Temperatur in die Adresse DB22.DBD2 (Datenbaustein 22 Datenbausteindoppelwort 2) eingetragen, wird diese an die für die externe Temperatursollwert-Vorgabe vorgesehene Adresse im Klima-Aggregat geschickt und zwar an die Adresse DB51.DBD316.

7.4. Die Programmierung einer S7-300

Der S7-Programmbehälter

Nachdem die Station konfiguriert wurde, kann mit der Erstellung des Steuerprogramms begonnen werden. Da das fertige S7-Programm in der CPU abgelegt wird, ist es als Unterobjekt der CPU zugeordnet. Ein S7-Programmordner wird automatisch nach der Stationskonfiguration angelegt.

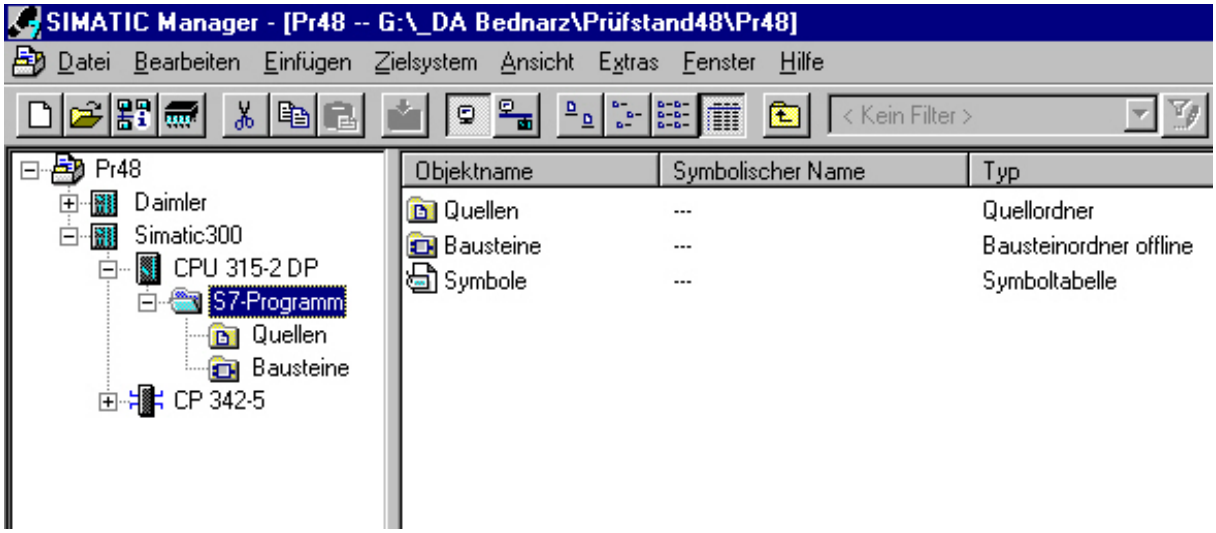


Abbildung 7.4-1: Der S7-Programmbehälter

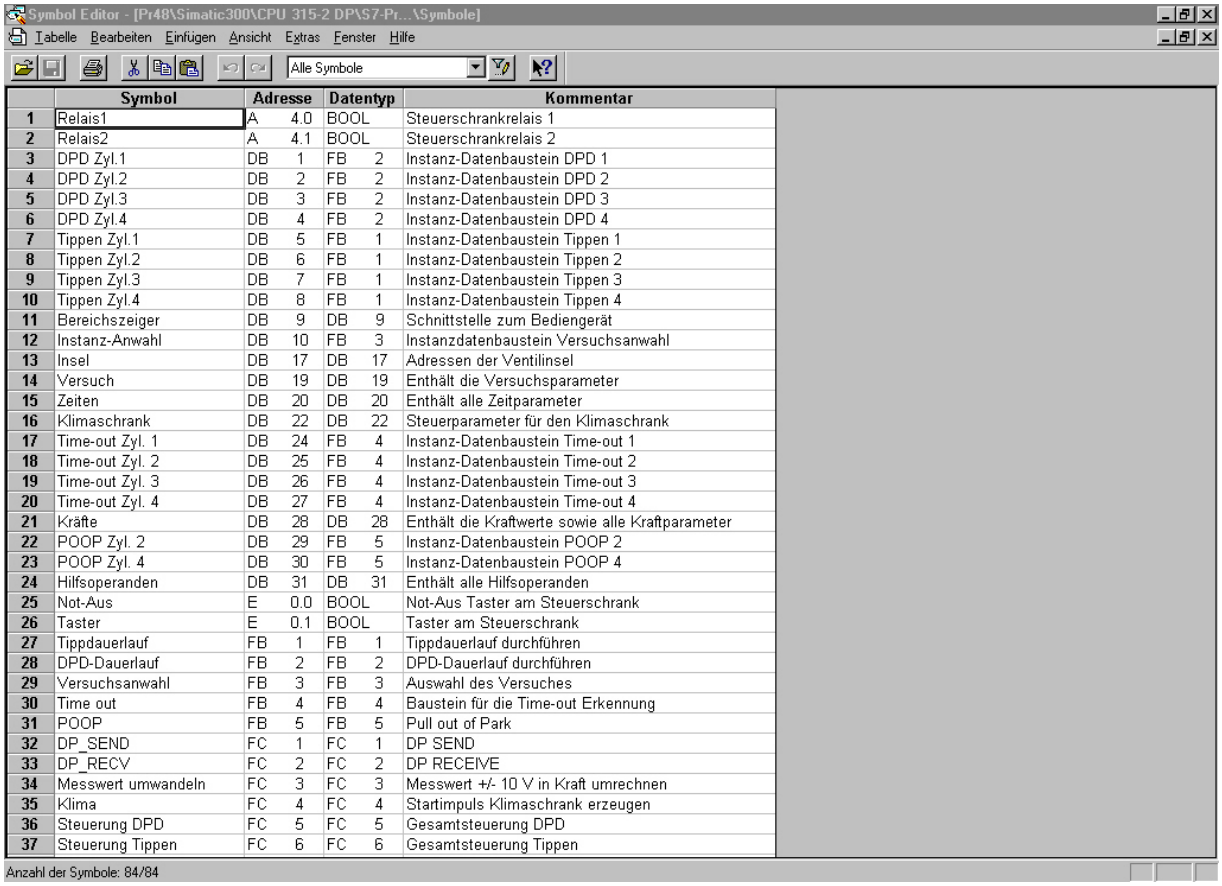
Quell- und Bausteinordner

S7-Programme können sowohl in Quell- als auch in Bausteinform erzeugt werden, auf die CPU können allerdings nur Bausteine geladen werden. Werden Programmteile in Quellform erstellt, müssen sie mit Hilfe eines Compilers in Bausteine umgewandet werden. Quellen dienen also nur als Basis für die Bausteinerzeugung. Darauf wird hier nicht näher eingegangen, da ich das gesamte Programm in ausschließlich Bausteinform geschrieben habe.

Symbole

Hinter dem Objekt Symbole verbirgt eine Tabelle, in der allen Operanden und Bausteinen, die im Programm vorkommen, symbolische Namen gegeben werden können. Dies erleichtert die Programmerstellung und verbessert die Lesbarkeit von S7 Programmcode enorm. Ich habe allen Operanden und Bausteinen

symbolische Namen zugeteilt. Dies geschieht mit Hilfe des Symboleditiors, der sich durch Klick auf Symbole öffnen lässt.



	Symbol	Adresse	Datentyp	Kommentar
1	Relais1	A 4.0	BOOL	Steuerschrankrelais 1
2	Relais2	A 4.1	BOOL	Steuerschrankrelais 2
3	DPD Zyl.1	DB 1	FB 2	Instanz-Datenbaustein DPD 1
4	DPD Zyl.2	DB 2	FB 2	Instanz-Datenbaustein DPD 2
5	DPD Zyl.3	DB 3	FB 2	Instanz-Datenbaustein DPD 3
6	DPD Zyl.4	DB 4	FB 2	Instanz-Datenbaustein DPD 4
7	Tippen Zyl.1	DB 5	FB 1	Instanz-Datenbaustein Tippen 1
8	Tippen Zyl.2	DB 6	FB 1	Instanz-Datenbaustein Tippen 2
9	Tippen Zyl.3	DB 7	FB 1	Instanz-Datenbaustein Tippen 3
10	Tippen Zyl.4	DB 8	FB 1	Instanz-Datenbaustein Tippen 4
11	Bereichszeiger	DB 9	DB 9	Schnittstelle zum Bediengerät
12	Instanz-Anwahl	DB 10	FB 3	Instanzdatenbaustein Versuchsanwahl
13	Insel	DB 17	DB 17	Adressen der Ventilinsel
14	Versuch	DB 19	DB 19	Enthält die Versuchsparameter
15	Zeiten	DB 20	DB 20	Enthält alle Zeitparameter
16	Klimaschrank	DB 22	DB 22	Steuerparameter für den Klimaschrank
17	Time-out Zyl. 1	DB 24	FB 4	Instanz-Datenbaustein Time-out 1
18	Time-out Zyl. 2	DB 25	FB 4	Instanz-Datenbaustein Time-out 2
19	Time-out Zyl. 3	DB 26	FB 4	Instanz-Datenbaustein Time-out 3
20	Time-out Zyl. 4	DB 27	FB 4	Instanz-Datenbaustein Time-out 4
21	Kräfte	DB 28	DB 28	Enthält die Kraftwerte sowie alle Kraftparameter
22	POOP Zyl. 2	DB 29	FB 5	Instanz-Datenbaustein POOP 2
23	POOP Zyl. 4	DB 30	FB 5	Instanz-Datenbaustein POOP 4
24	Hilfsoperanden	DB 31	DB 31	Enthält alle Hilfsoperanden
25	Not-Aus	E 0.0	BOOL	Not-Aus Taster am Steuerschrank
26	Taster	E 0.1	BOOL	Taster am Steuerschrank
27	Tippdauerlauf	FB 1	FB 1	Tippdauerlauf durchführen
28	DPD-Dauerlauf	FB 2	FB 2	DPD-Dauerlauf durchführen
29	Versuchsanwahl	FB 3	FB 3	Auswahl des Versuches
30	Time out	FB 4	FB 4	Baustein für die Time-out Erkennung
31	POOP	FB 5	FB 5	Pull out of Park
32	DP_SEND	FC 1	FC 1	DP SEND
33	DP_RECV	FC 2	FC 2	DP RECEIVE
34	Messwert umwandeln	FC 3	FC 3	Messwert +/- 10 V in Kraft umrechnen
35	Klima	FC 4	FC 4	Startimpuls Klimaschrank erzeugen
36	Steuerung DPD	FC 5	FC 5	Gesamtsteuerung DPD
37	Steuerung Tippen	FC 6	FC 6	Gesamtsteuerung Tippen

Abbildung 7.4-2: Symboltabelle

Nach Eingabe eines Symbols kann im Steuerprogramm - wenn symbolische Programmierung aktiv ist – der symbolische Operand statt des absoluten Operanden benutzt werden. Würde im Steuerprogramm beispielsweise lediglich der absolute Operand A 4.0 auftauchen, wüsste der Leser des Programms nicht, was sich hinter der Adresse verbirgt. Ist, wie in diesem Fall, ein symbolischer Name für A 4.0 vergeben, wird dieser im Steuerungsprogramm angezeigt und man erkennt, dass es sich um Relais1 handelt. Soll ein Symbol im Steuerungsprogramm angesprochen werden, muss der Name mit Anführungsstrichen versehen werden. Alternativ kann auch der absolute Operand hingeschrieben werden. Ist ein Symbol definiert, wird der absolute Operand nach der Eingabe automatisch ersetzt.

Die Symboltabellen werden übrigens beim Laden des Programms in die Steuerung **nicht** mit gesendet, weil die symbolischen Namen nur als

Erleichterung beim Lesen und Programmieren dienen sollen. Für die eigentliche Programmausführung spielen sie keine Rolle, daher würden sie nur unnütz Speicherplatz im Zielsystem vergeuden.
Überträgt man also ein Programm, über dessen Projektierungdaten (S7-Projekt) man nicht verfügt, von einer S7-Steuerung auf ein PG, werden in den Bausteinen nur die absoluten Operanden angezeigt, die Symbole stehen nicht zur Verfügung.

Die Programmbausteine

Im Bausteinbehälter ist der gesamte Quellcode des Steuerungsprogramms in Bausteinform untergebracht. Zur Programmerstellung stehen verschiedene Arten von Bausteinen zur Verfügung. Die in diesem Steuerungsprogramm verwendenden Bausteinarten sind:

Codebausteintypen

- Organisationsbausteine (OB)
- Funktionsbausteine (FB)
- Funktionen (FC)

Datenbausteintypen

- Globaldatenbausteine (DB)
- Instanz-Datenbausteine (DB)

Diese fünf Bausteintypen sind die Standardbausteintypen in S7. Es gibt in Step 7 noch einige Sondertypen, die aber nicht benutzt wurden. Codebausteine enthalten ablauffähigen Programmcode, Datenbausteine dienen der Datenhaltung. Ein neuer Baustein kann durch Rechtsklick im Bausteinbehälter hinzugefügt werden. Außer den Bausteinen enthält der Behälter noch die Systemdaten in denen u.a. die Daten zur Hardwareprojektierung, Netzkonfiguration und GD-Kommunikation abgelegt sind.

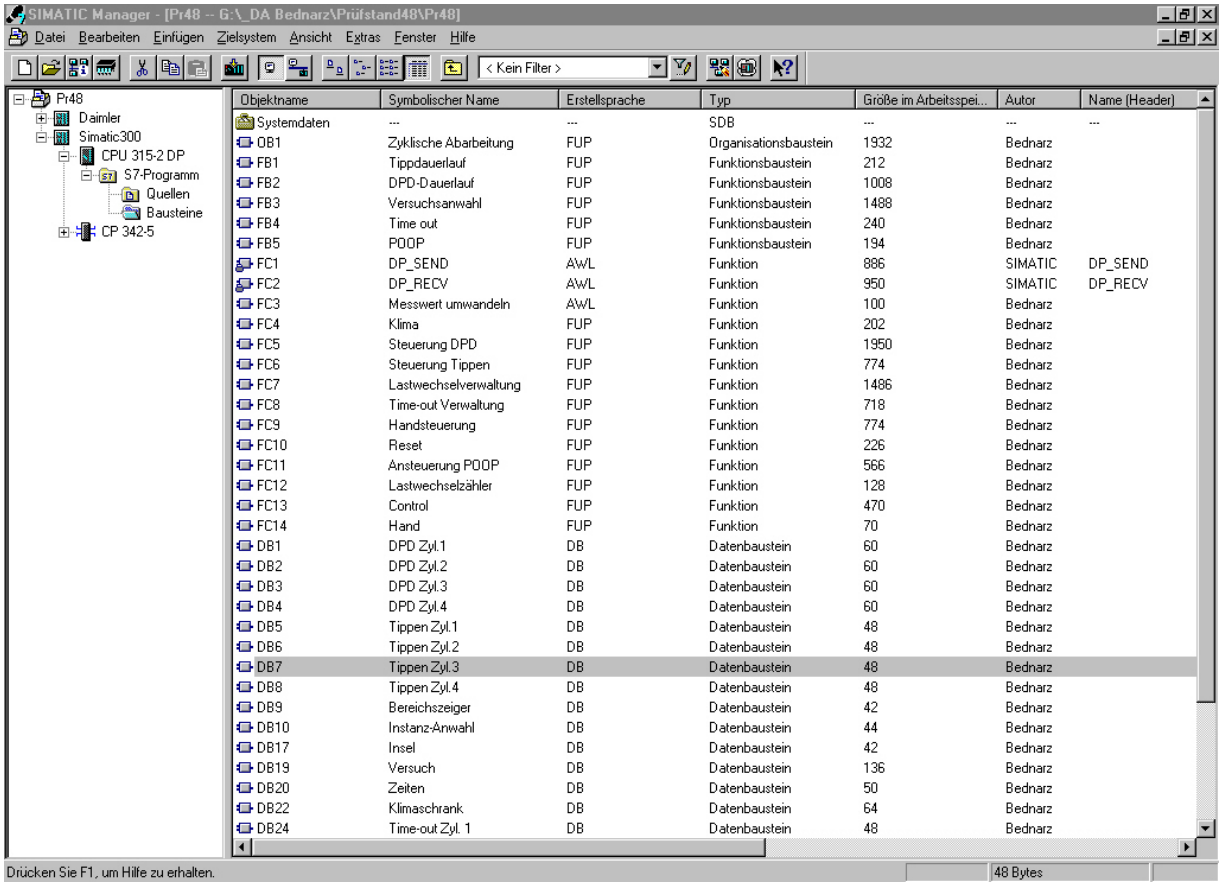


Abbildung 7.4-3: Der Bausteinbehälter

Organisationsbausteine (OB)

Ähnlich wie bei Computer-Betriebsystemen werden Programmorganisationseinheiten durch so genannte Tasks ausgeführt. Das Taskkonzept eines Automatisierungssystems muss über eine priorisierbare, zeitzyklische sowie ereignisorientierte Programmsteuerung verfügen[4]. Dies wird mit Hilfe von Organisationsbausteinen realisiert.

Sie bilden die Schnittstelle zwischen dem Betriebssystem der CPU und des Anwenderprogramms. Organisationsbausteine können nur vom Betriebssystem selbst beim Eintreten von bestimmten Ereignissen aufgerufen werden. Der

wichtigste Organisationsbaustein heißt OB1, der zyklisch abgearbeitet wird und somit das Programm mit sämtlichen Unterprogrammen organisiert. Der OB1 kann jederzeit bei Eintreffen bestimmter Ereignisse, z.B. Zykluszeitfehler, durch einen anderen OB (sofern projektiert) unterbrochen werden (siehe Grafik). Nach Abarbeitung des anderen OB erfolgt ein Rücksprung in den OB1, und zwar an die Stelle, an der er unterbrochen wurde. Auch der Aufruf aller Unterprogramme erfolgt vom OB1 aus.

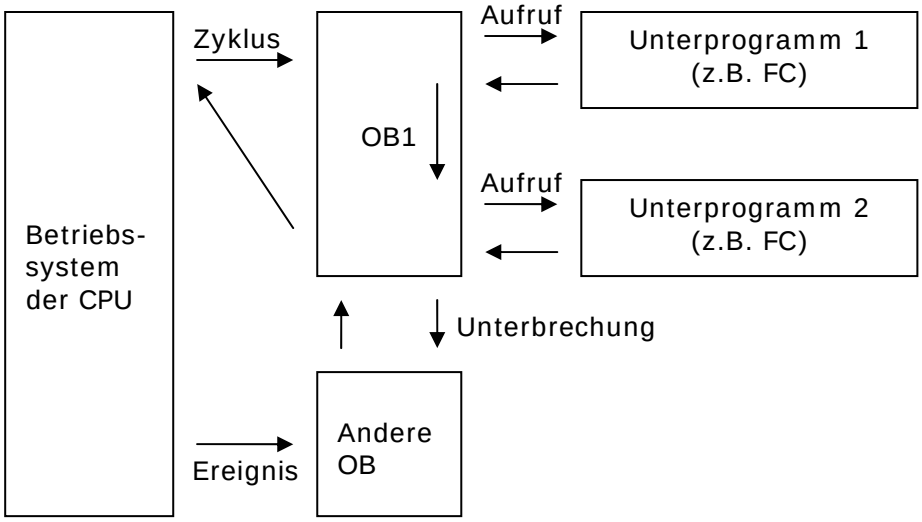


Abbildung 7.4-4: Programmablauf

Funktionen (FC, SFC):

Je umfangreicher ein Programm ist, desto mehr Anweisungen müssen im OB1 (zyklische Abarbeitung) stehen. Dies führt dazu, dass der Baustein sehr groß und sehr unübersichtlich wird. Des Weiteren kommt es im Verlauf eines Programms immer wieder vor, dass ein- und dieselbe Befehlskette mehrfach benötigt wird, die Berechnung der Kraft beispielsweise muss für alle vier Zylinder durchgeführt werden. Damit man nicht dieselben Befehlsketten mehrfach programmieren muss, definiert man als Unterprogramme Funktionen, in denen die Befehlsketten einmal abgelegt werden. Funktionen kann man mit Bausteinparametern zur Übergabe von Daten ausstatten. Diese Parameter werden auch **Formalparameter** genannt und werden im Deklarationsteil eines Bausteins festgelegt. Innerhalb der Funktion können diese dann wie CPU-Operanden behandelt werden.

Ebenso können Funktionen zur Programmstrukturierung verwendet werden, um durch klein gehaltene Bausteine eine bessere Übersicht über das Gesamtprogramm zu haben. Einzelne Funktionen lassen sich auch in andere Projekte kopieren. Als Systemfunktionen (SFCs) bezeichnet man Funktionen, die vom Steuerungshersteller stammen und Aufgaben übernehmen, die typischerweise beim Erstellen von Steuerungsprogrammen häufiger benötigt werden. Ein typisches Beispiel ist der FC2, dessen Aufgabe das Senden von Daten über einen Profibus-Kommunikationsprozessor (CP). Die Systemfunktionen befinden sich im Ladespeicher der CPU, die Anweisungen können nicht eingesehen werden. Man kann SFC's auch nicht ändern oder löschen, nur benutzen.

Funktionsbausteine (FB, SFB)

Funktionsbausteine sind Funktionen sehr ähnlich. Sie können alles was die Funktionen auch können, es gibt allerdings zwei Unterschiede:
Innerhalb einer Funktion können nur temporäre Variablen deklariert werden. Temporäre Variablen dienen lediglich zur Speicherung von Zwischenergebnissen und verlieren bei der Abarbeitung des Programms sofort nach Verlassen der Funktion ihre Gültigkeit. Innerhalb eines Funktionsbausteins hingegen hat man zusätzlich die Möglichkeit, so genannte statische Variablen zu deklarieren. Daten, die in solchen Variablen abgelegt wurden, bleiben, auch nach Verlassen des FB, erhalten und verleihen dem Baustein damit eine Art „Gedächtnis".
Zu jedem Funktionsbaustein gehört mindestens ein Instanz-Datenbaustein, der die Formalparameter und die Variablen des FB speichert. Diese Funktionsweise wird im auch Prüfstandsprogramm ausgenutzt: der FB DPD-Dauerlauf wird nur einmal programmiert, aber insgesamt vier Mal aufgerufen. Der Aufruf erfolgt jedes Mal mit den Parametern eines anderen Prüfplatzes.
Für Systemfunktionsbausteine (SFB) gilt dasselbe wie für die Systemfunktionen, SFB wurden bei der Programmierung nicht einsetzt.

Datenbausteine

Als Datenbausteine bezeichnet man fest definierte Datenbereiche, in denen Daten des Anwenderprogramms abgelegt werden können.

Global-DBs

Global-Datenbausteine enthalten Daten, die von allen Codebausteinen gelesen und geschrieben werden können. Sie müssen durch den Anwender programmiert werden, um festzulegen wie viele Daten aufgenommen werden sollen und von welchem Datentyp sie sind.

Instanz-DBs

Instanz-Datenbausteine sind immer einem Funktionsbaustein zugeordnet. Sie dienen dazu, die Formalparameter sowie die statischen Variablen des FB aufzunehmen. Da sie identisch mit dem Deklarationsteil des FB sind, müssen sie nicht extra programmiert werden.

Programmierung von Codebausteinen

Die Programmierung von Codebausteinen erfolgt mit dem Editor für KOP/AWL/FUP. Der Editor kann für alle drei Sprachen verwendet werden. Die Anweisungsliste (AWL) ist eine reine Textsprache, der Kontaktplan und der Funktionsplan sind hingegen grafische Sprachen.

Der Kontaktplan orientiert sich an der Elektrotechnik, er ist so ähnlich aufgebaut wie die Stromlaufpläne in der Relaistechnik. Der Funktionsplan orientiert sich an der Digitaltechnik. Die meisten meiner Bausteine habe ich aufgrund der grafischen Darstellung in FUP programmiert, die ich für die bessere Variante gegenüber KOP halte, aber das ist teilweise auch Geschmackssache des Programmierers. Grundsätzlich kann man sagen, dass jede Sprache ihre Vor- und Nachteile hat. Es kommt halt auf die Anwendung an, manche Sachen lassen sich in AWL besser programmieren, andere wiederum in FUP oder KOP.

Das nachfolgende Bild zeigt den FB 1 „Tippdauerlauf“, der für die Bewegung der Tippzylinder sorgt.

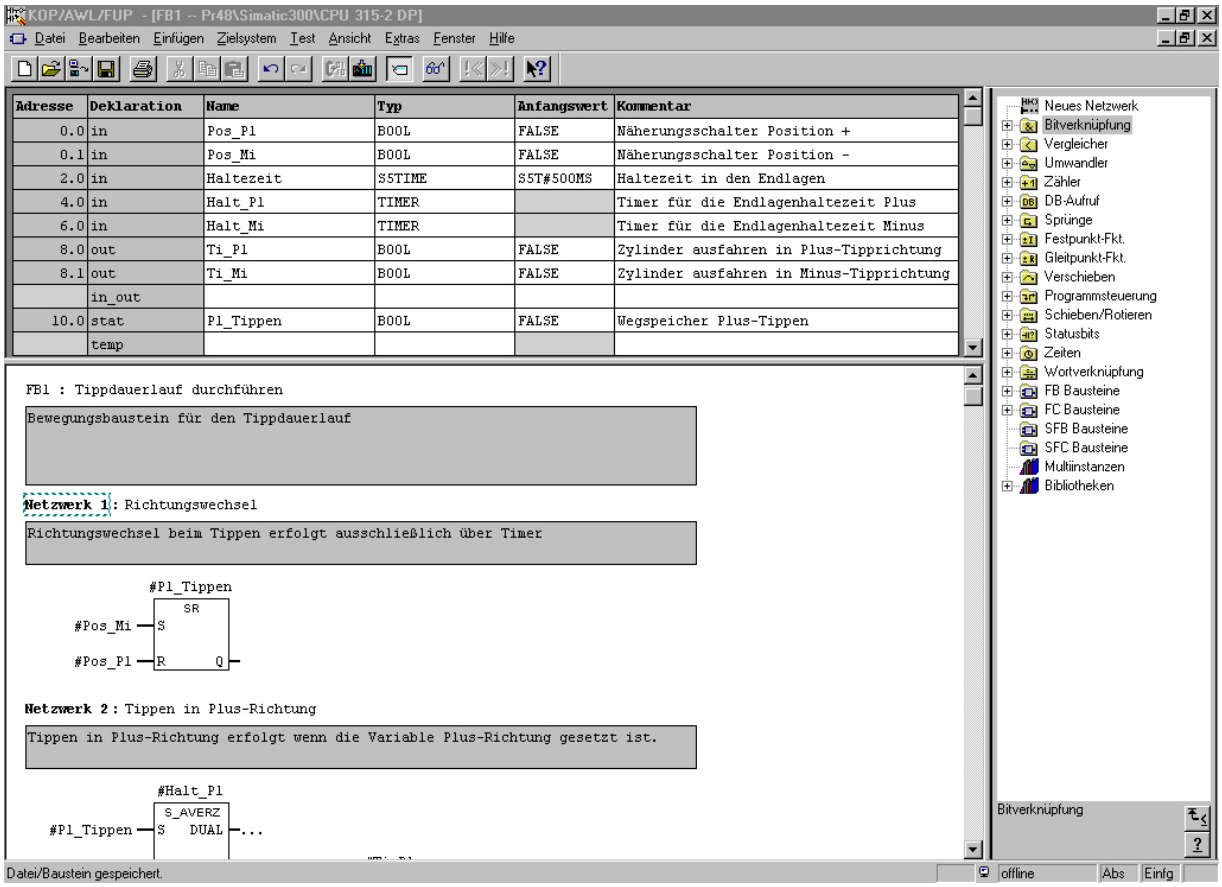


Abbildung 7.4-5: Editor für KOP/AWL/FUP

Die Deklarationstabelle

Im oberen Teil des Fensters erkennt man die Deklarationstabelle des Bausteins. Hier werden die Schnittstellen des Bausteins, die Formalparameter eingefügt. Über diese Ein- und Ausgänge werden Daten an den Baustein übergeben und Daten nach außen geschrieben. Damit dieser Baustein mehrfach verwendbar bleibt, darf im Programmcode des Bausteins dann nur noch mit diesen Formalparametern gearbeitet werden. Auf keinen Fall darf man anlagenspezifische Operanden verwenden, da diese garantiert in der nächsten Anlage so nicht vorkommen werden.

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Pos_Pl	BOOL	FALSE	Näherungsschalter Position +
0.1	in	Pos_Mi	BOOL	FALSE	Näherungsschalter Position -
2.0	in	Haltezeit	S5TIME	S5T#500MS	Haltezeit in den Endlagen
4.0	in	Halt_Pl	TIMER		Timer für die Endlagenhaltezeit Plus
6.0	in	Halt_Mi	TIMER		Timer für die Endlagenhaltezeit Minus
8.0	out	Ti_Pl	BOOL	FALSE	Zylinder ausfahren in Plus-Tipprichtung
8.1	out	Ti_Mi	BOOL	FALSE	Zylinder ausfahren in Minus-Tipprichtung
	in_out				
10.0	stat	Pl_Tippen	BOOL	FALSE	Wegspeicher Plus-Tippen
	temp				

Screenshot 8.4.3-2: Deklarationsteil eines Codebausteins

Es können verschiedene Arten von Formalparametern deklariert werden:

in: Eine Variable, die an einen In-Eingang gelegt wird, kann durch den Baustein nur abgefragt, aber nicht beschrieben werden.

out: Eine Variable, die an einen Ausgang gelegt wird, kann nicht abgefragt, sondern nur beschrieben werden.

in_out: Die Variable kann gelesen und beschrieben werden.

stat: Eine statische Variable innerhalb des Baussteins. Die gespeicherten Werte bleiben nach Ende der Bausteinbearbeitung erhalten. Über statische Variablen verfügen nur die Funktionsbausteine, nicht aber Funktionen. Die Werte werden in den Instanz-Datenbausteinen gespeichert.

temp: Über die Möglichkeit temporäre Variablen zu deklarieren, verfügen wiederum beide Bausteintypen. Sie dienen nur als Speicher für Zwischenergebnisse und verfallen nach Beenden der Bausteinbearbeitung.

Im unteren Teil des Fensters findet die eigentliche Programmierung statt. Die Anweisungen bzw. Funktionsblöcke werden dabei in so genannten Netzwerken abgelegt. Die Aufteilung in Netzwerke erleichtert die Lesbarkeit des Programms. Die grau unterlegten Felder sind für Kommentare vorgesehen. Zu jedem Netzwerk gehören ein Titel und ein Kommentar. Programmiert man in AWL ist es theoretisch möglich, alle Bausteinanweisungen in ein Netzwerk zu schreiben, was aber aus Übersichtsgründen nicht empfehlenswert ist. Programmiert man in FUP, ist nur eine Zuweisung pro Netzwerk möglich. Man ist also gezwungen die Blöcke aufzuteilen.

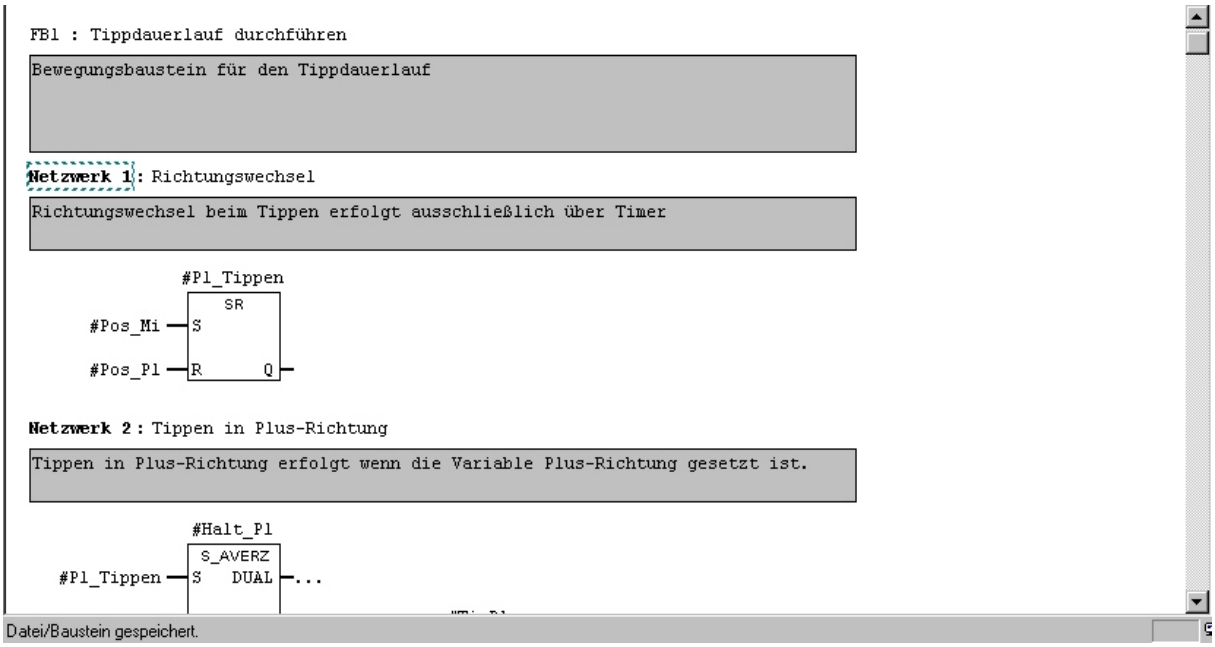


Abbildung 7.4-6: Deklarationsteil eines Codebausteins

Die für die Programmierung benötigten Funktionen und Funktionsblöcke können aus dem Katalog auf der rechten Seite ausgewählt und in das Programm eingefügt werden. Nachdem alle Anweisungen eingefügt wurden, muss der Baustein gespeichert werden. Er kann ab jetzt an beliebiger Stelle aufgerufen und parametriert werden. Er steht jetzt im Katalog (im Ordner FB Bausteine) zur Verfügung.

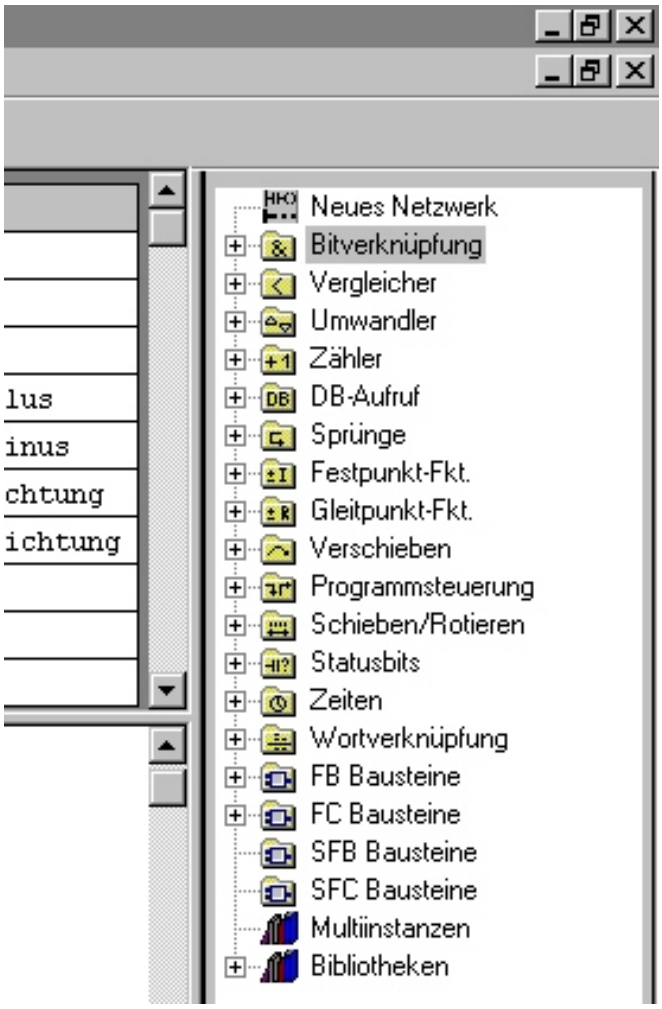


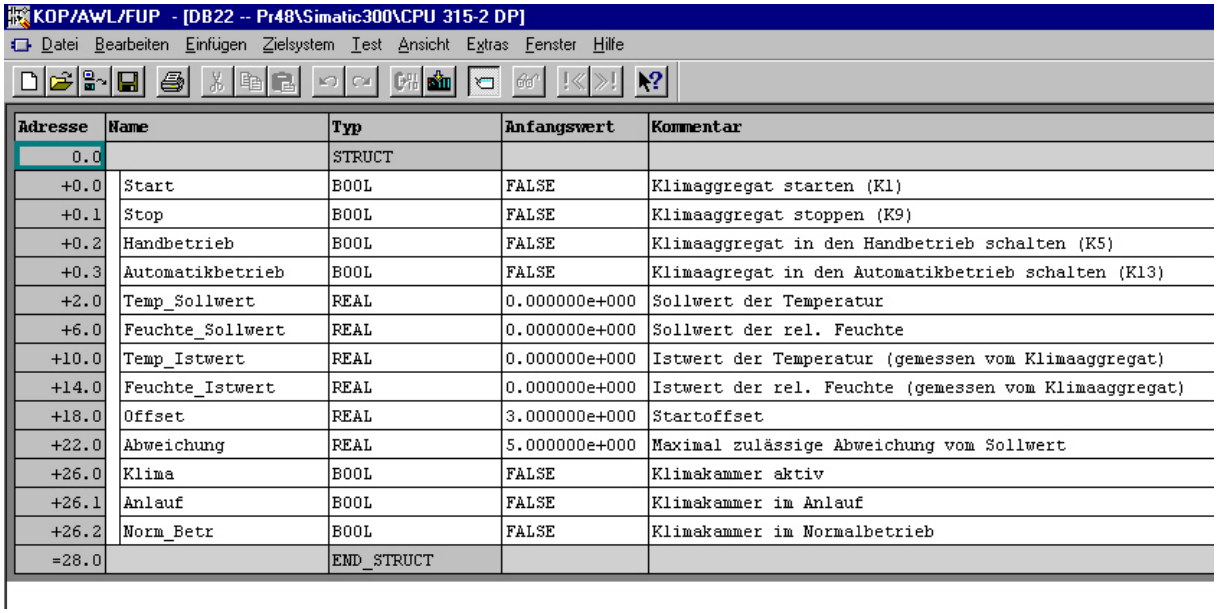
Abbildung 7.4-7: Funktionsblöcke für die Programmierung in FUP

Parametrierung von Datenbausteinen

Datenbausteine dienen in Step 7 der Datenhaltung. Die Daten werden in Form von Variablen unterschiedlicher Datentypen tabellarisch hinterlegt. Die Instanz-Datenbausteine sind immer Funktionsbausteinen zugeordnet. In ihnen sind die Daten der Formalparameter gespeichert, sie haben genau denselben Aufbau wie der Deklarationsteil des FB. Daher müssen sie **nicht** extra projektiert werden, sondern können bei Aufruf des FB automatisch erzeugt werden. Sie können von Hand auch nicht geändert werden, man kann sie nur ändern indem man den Deklarationsteil des FB ändert. In diesem Fall muss man den Instanz-DB neu erzeugen lassen.

Anders sieht es bei den Global-DBs aus. Sie müssen per Hand programmiert werden. Dazu müssen die Variablen mit Namen, Datentyp, Anfangswert und

Kommentar in die Tabelle eingetragen werden. Der Kommentar ist dabei optional, die anderen Felder müssen ausgefüllt werden. Auf dem Screenshot ist als Beispiel der DB22 zu sehen, in ihm sind alle Klimadaten gespeichert.



Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Start	BOOL	FALSE	Klimaggregat starten (K1)
+0.1	Stop	BOOL	FALSE	Klimaaggregat stoppen (K9)
+0.2	Handbetrieb	BOOL	FALSE	Klimaaggregat in den Handbetrieb schalten (K5)
+0.3	Automatikbetrieb	BOOL	FALSE	Klimaagregat in den Automatikbetrieb schalten (K13)
+2.0	Temp_Sollwert	REAL	0.000000e+000	Sollwert der Temperatur
+6.0	Feuchte_Sollwert	REAL	0.000000e+000	Sollwert der rel. Feuchte
+10.0	Temp_Istwert	REAL	0.000000e+000	Istwert der Temperatur (gemessen vom Klimaaggregat)
+14.0	Feuchte_Istwert	REAL	0.000000e+000	Istwert der rel. Feuchte (gemessen vom Klimaaggregat)
+18.0	Offset	REAL	3.000000e+000	Startoffset
+22.0	Abweichung	REAL	5.000000e+000	Maximal zulässige Abweichung vom Sollwert
+26.0	Klima	BOOL	FALSE	Klimakammer aktiv
+26.1	Anlauf	BOOL	FALSE	Klimakammer im Anlauf
+26.2	Norm_Betr	BOOL	FALSE	Klimakammer im Normalbetrieb
=28.0		END_STRUCT		

Abbildung 7.4-8: Datenbaustein 22

7.5. Projektierung von Bediengeräten

Simatic Bediengeräte und Panels werden mit Hilfe der Software ProTool projektiert. ProTool ist nicht Bestandteil der Step 7 Basissoftware, lässt sich aber nach separater Installation entweder in Step 7 integrieren oder Stand-Alone betreiben. Der Stand-Alone Modus ist aber nur sinnvoll, wenn keine Step 7-Basisversion auf den Rechner vorhanden ist. Die Integration in Step 7 hat den entscheidenden Vorteil, dass bei der Projektierung des Datenaustausches direkt auf die Symboltabelle der Steuerung zugegriffen werden kann. Bediengeräte können dem Projekt genau wie Netze und Stationen hinzugefügt werden, und zwar unter dem Menüpunkt 8 SIMATIC OP. OP ist die Abkürzung für Operator-Panel.

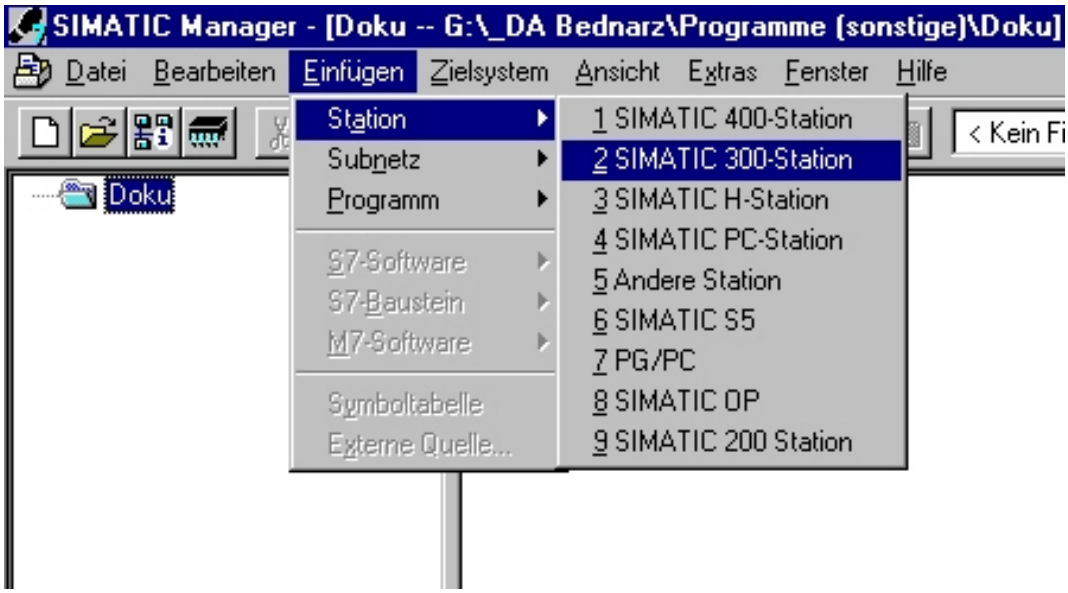


Abbildung 7.5-1: OP einfügen

Durch Klicken auf eins der OP-Icons öffnet sich ProTool und die Projektierung des betreffenden OP's wird geladen. Ist für das betreffende OP noch keine Projektierung vorhanden, öffnet sich zunächst ein Konfigurationsassistent in dem unter Anderem ausgewählt werden muss, um welchen Bediengerätetyp es sich handelt. Es können verschiedene Panelarten mit ProTool projektiert werden. Da sie sich alle in Funktionsumfang und in Ausstattung unterscheiden, muss man ProTool auf den entsprechenden Typ (hier: OP7) einstellen. Beim Erstellen der Projektierung werden dann auch nur die Funktionalitäten angeboten, die das vorhandene Bediengerät unterstützt. Beispielsweise ist die Anzeige einer Grafik auf einem OP7 Display mit 4x20 Zeichen nicht möglich, sehr wohl aber auf einem Bediengerät mit Grafikdisplay. Nach Fertigstellung des Assistenten erscheint die Oberfläche von ProTool.

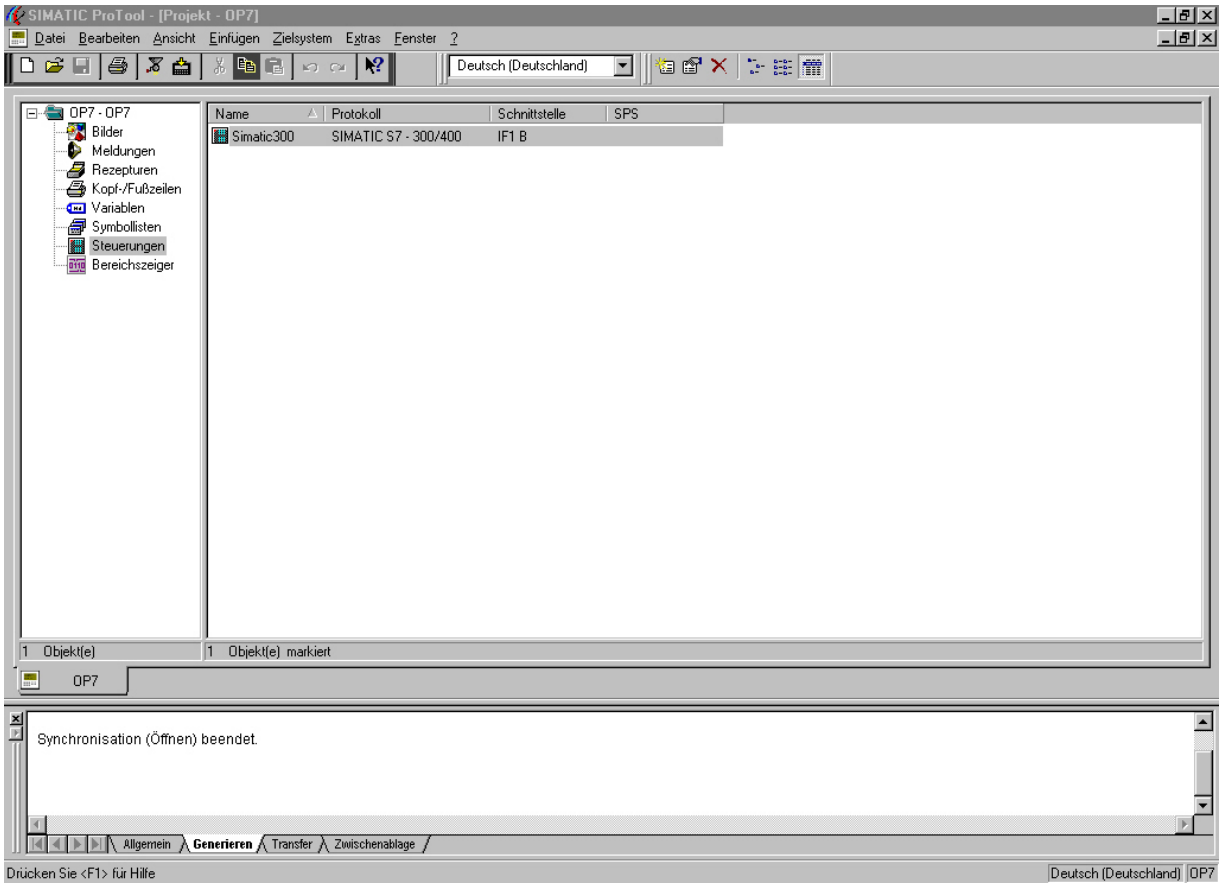


Abbildung 7.5-2: Pro Tool Benutzeroberfläche

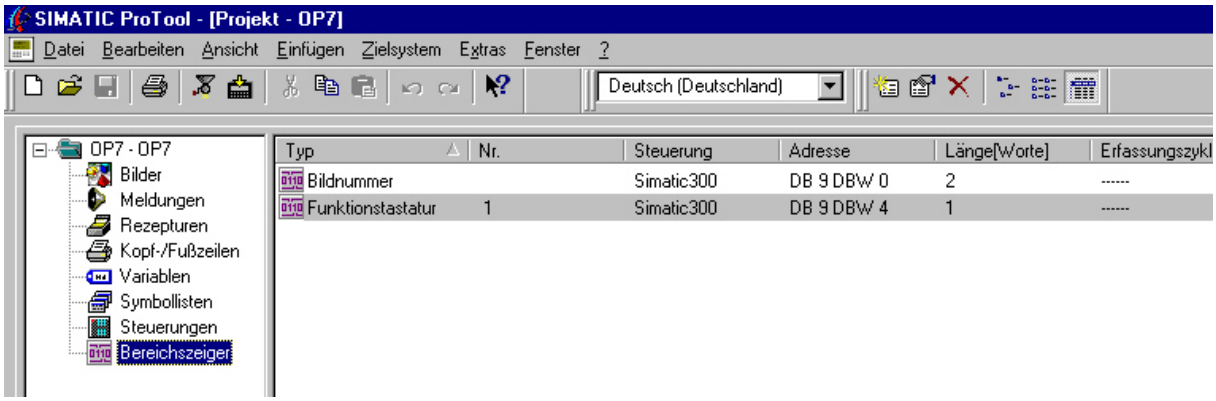
Für die Projektierung in ProTool stehen im linken Fenster verschiedene Objekte zur Verfügung. In der Liste stehen alle Objekte, die dem OP7 zur Verfügung stehen.

Bereichszeiger

Bereichszeiger sind Zeiger auf Adressenbereiche in der SPS. Es gibt verschiedene Bereichszeiger für verschiedene Aufgaben, sie dienen aber alle dem Datenaustausch mit der SPS.

Ich habe in der Anlage insgesamt zwei Bereichszeiger programmiert:

- Bildnummer: Überträgt Informationen zum aktuell aufgerufen Bild am Display an die SPS. Diese Informationen können dann im Steuerprogramm ausgewertet werden, um bestimmte Aktionen bildabhängig auszulösen.
- Funktionstastatur: Überträgt die Betätigung der Funktionstasten an die Steuerung, so dass ein Druck auf eine bestimmte Taste im Steuerprogramm ausgewertet werden kann.

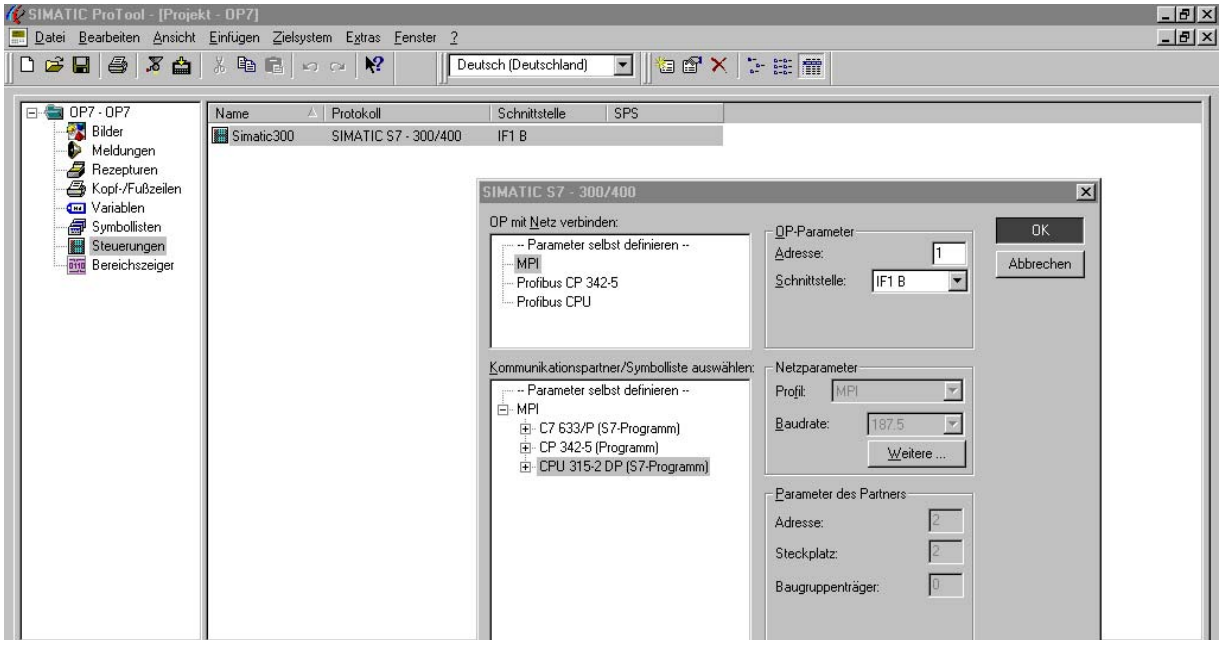


Screenshot 8.5-3: Bereichszeiger

Für die Bereichszeiger habe ich einen eigenen Datenbaustein (DB9) vorgesehen.

Steuerungen

Hier legt das Bediengerät die Informationen zu den angekoppelten Steuerungen ab. Hier werden nur die Steuerungseigenschaften eingestellt.

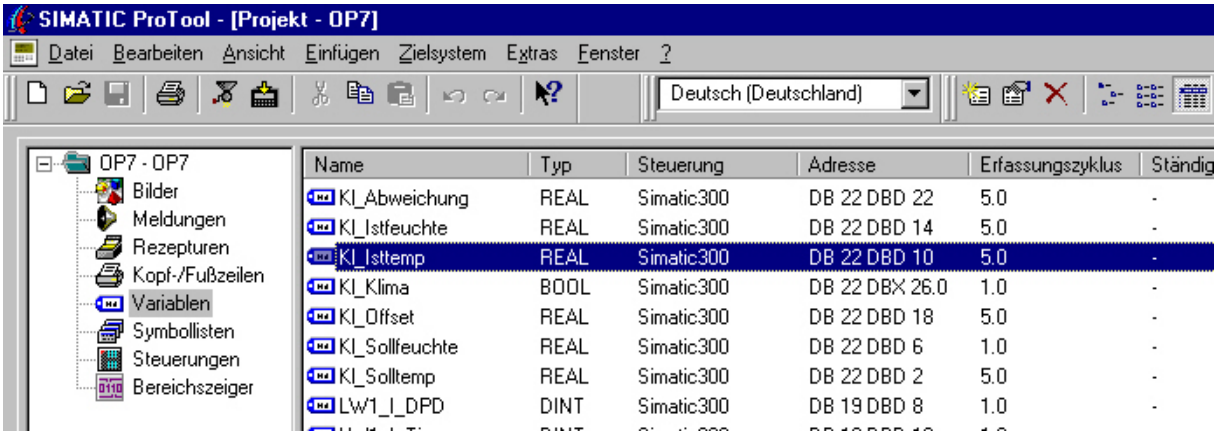


Screenshot 8.5-4: Steuerungseigenschaften

Variablen

Variablen sind das wichtigste Mittel um Daten mit der Steuerung austauschen zu können. Es gibt Variablen mit oder ohne Steuerungsanbindung. Eine Variable ohne Steuerungsanbindung existiert nur im Bediengerät, eine Variable mit Anbindung hingegen kann sowohl vom OP als auch von der SPS gelesen und

beschrieben werden. Betreibt man ProTool integriert in Step 7, kann beim Anlegen einer Variable direkt auf die Symboltabelle der Steuerung zugegriffen werden.



Screenshot 8.5-5: Variablen

Beispiel: Bei der blau markierten Variable handelt es sich um die aktuelle Temperatur des Klima-Aggregats, der Wert wird alle 5 Sekunden aus der Steuerung geholt und in die Variable KI_Isttemp geschrieben. Er steht dann im OP zur Verfügung. Die Variable kann jetzt z.B. dazu benutzt werden, den Wert auf dem Display auszugeben.

Kopf- und Fußzeilen

Kopf und Fußzeilen sind nur von Belang, wenn an das Bediengerät ein Drucker zur Protokollierung angeschlossen wird. Diese Möglichkeit gibt es in dieser Anlage nicht, da kein Drucker vorhanden ist.

Rezepturen

Mit Rezepturen können mehrere zusammenhängende Daten gemeinsam und synchron zur Steuerung übertragen werden. Rezepturen habe ich bei der Projektierung nicht verwendet.

Meldungen

Unter dem Menüpunkt Meldungen können Meldungen an Display angezeigt werden. Meldungen habe ich ebenfalls nicht projektiert.

Bilder

Bilder sind dazu da, um Werte aus dem Prozess darzustellen. Alles was auf dem Display angezeigt werden soll, muss in Bildern abgelegt werden. Bilder können beim OP7 statischen Text und Ein- bzw. Ausgabefelder enthalten. Pro Bild können mehrere Bildeinträge projektiert werden. Jeder Bildeintrag stellt einen Bildschirm am OP dar. Zusätzlich können in jedem Bild die Funktionstasten F1 – F4 und K1 – K8 mit Funktionen belegt werden. Funktionen lösen ein bestimmtes Ereignis, z.B. das Aufrufen eines anderen Bilds, aus.

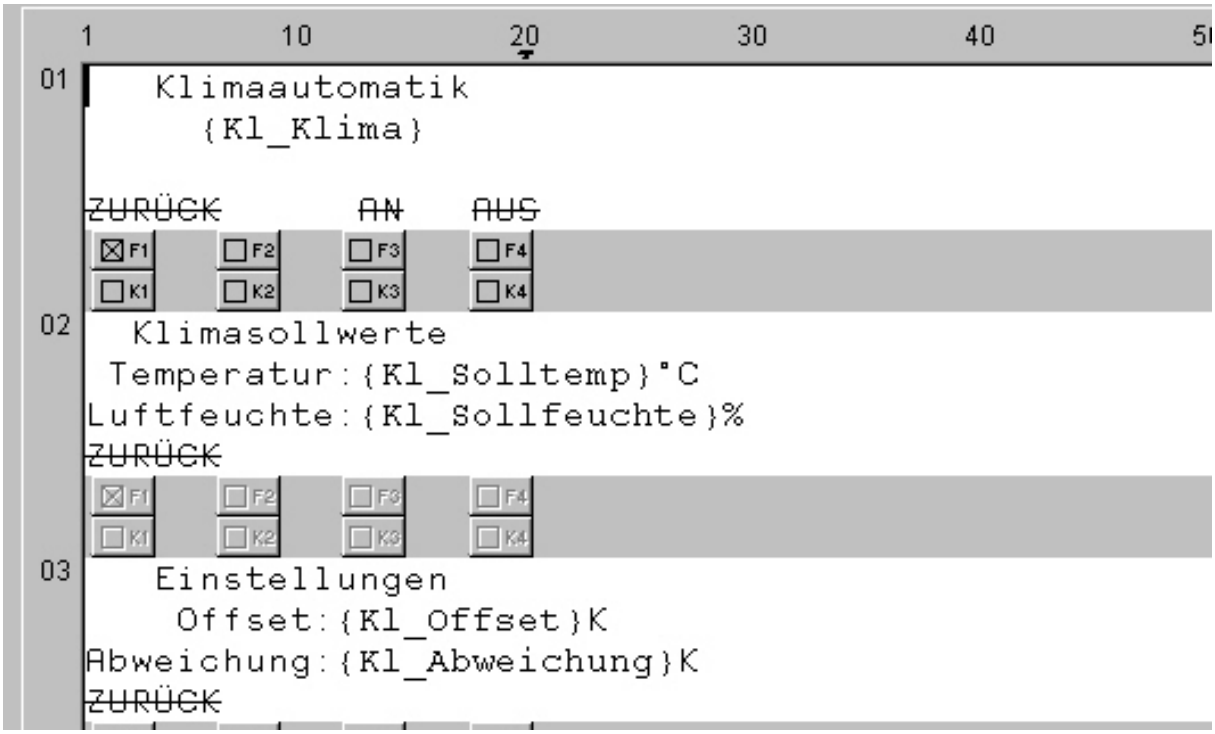


Abbildung 7.5-6: Das Bild Klima

Das Bild „Klima“ verfügt, wie man auf den Screenshot sehen kann, über insgesamt 3 Einträge.

Für die Umschaltung zwischen den Bildeinträgen werden die Cursortasten des OP verwendet.

Die Klammern kennzeichnen Ein- und Ausgabefelder. Hier wird der Name der Variable angezeigt, auf die sich das Ein/Ausgabefeld bezieht. In Eintrag 2 werden neben den Beschriftungen die aktuellen Klimasollwerte angezeigt. Da es sich hier um eine Kombination von Ein- und Ausgabefeld handelt, können die Werte auch geändert werden. Dazu ist der Cursor auf den gewünschten Eintag zu setzten. Nun kann ein neuer Wert in das Feld eingetragen werden. Der Wert wird nur in die Steuerung übernommen, wenn am Ende der Eingabe die Enter-Taste