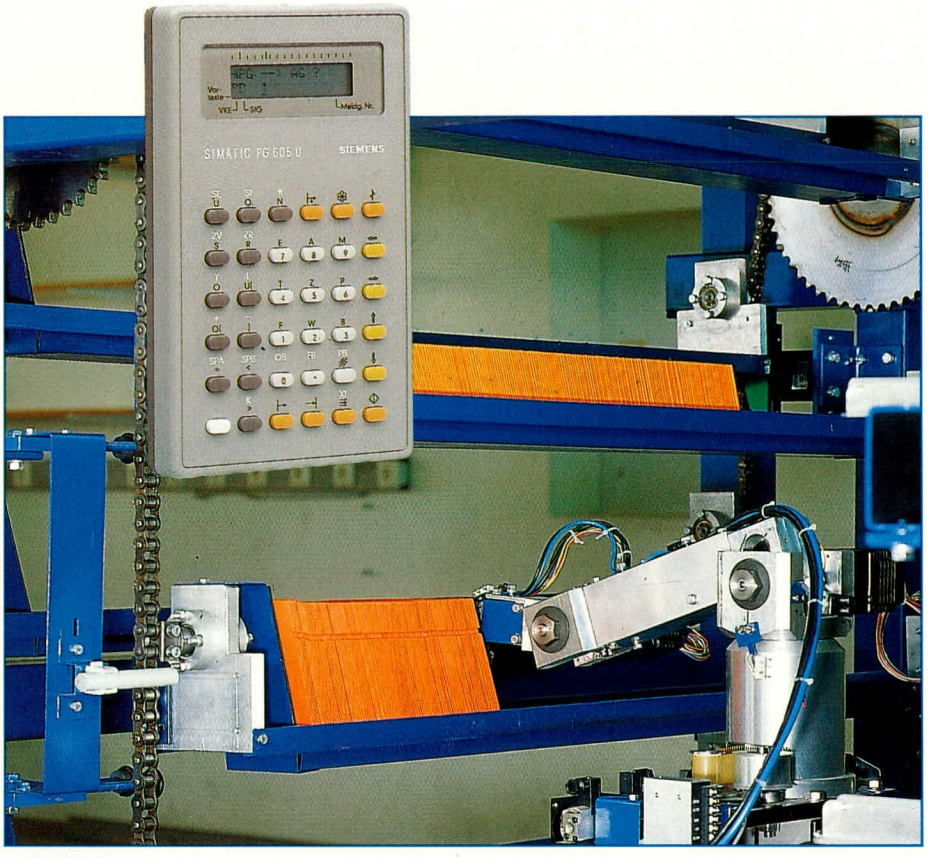


Speicherprogrammierbare Steuerungen (SPS)

Automatisierungsmittel für die verschiedensten Aufgaben — Teil 3

Der vorliegende dritte Teil der Artikelserie behandelt weitere, im SPS-Bereich verwendete Programmiersprachen sowie Möglichkeiten zur Strukturierung von SPS-Programmen.

Prof. Dr.-Ing. Ewald Matull

8. Weitere SPS-Programmiersprachen

In den ersten beiden Folgen dieser Artikelserie haben wir im wesentlichen die Kontaktplansprache (KOP) als Möglichkeit zur Erstellung von SPS-Programmen kennengelernt. Sie wurde gerade deshalb entwickelt, um im SPS-Bereich einer bestimmten Anwendergruppe eine weitgehend bekannte Darstellungsart (nämlich ein Äquivalent zu den in der Schütztechnik verwendeten Stromlaufplänen) zu bieten.

Auch die anderen SPS-Programmiersprachen entstanden mit ähnlicher Zielsetzung. Hier sind zu nennen:

- die Funktionsbausteinsprache bzw. Funktionsplansprache (**FBS** bzw. **FUP**), die mit einer Symbolik arbeitet, die in der Entwicklung elektronischer Schaltungen verwendet wurde. Da gerade in der Verfahrenstechnik zeitweilig aus Elektronik-Baugruppen aufgebaute verbindungsprogrammierte Steuerungen verwendet wurden, die mit Funktionsplan-Schaltzeichen dokumentiert waren, gab es hier einen großen Anwenderkreis, der an der Weiterverwendung dieser Schaltzeichen im Steuerungsbau interessiert war.
- die Anweisungsliste (**AWL**). Hier gibt es eine entfernte Ähnlichkeit zu Assemblersprachen; d.h. Automatisierer, die im Bereich der hardwarenahen Programmierung Erfahrungen besitzen, finden verschiedene bekannte Befehle wieder.
- Strukturierter Text (**ST**). Diese SPS-Programmiersprachenversion wurde in die neue IEC-Norm 1131 aufgenommen und zielt auf den Anwenderkreis der Hochsprachenprogrammierer, die gewohnt sind, mit strukturierten Hochsprachen wie Pascal oder C umzugehen.
- die grafische Ablaufsprache (**AS** bzw. **GRAFCET**). Hier finden Projekteure von Fertigungsanlagen ihre Funktions-Beschreibungen eines sequentiellen Vorgangs wieder.

Nachfolgend soll ein Zusammenhang zwischen diesen Programmiersprachen hergestellt werden:

- Alle SPS-Programme für die gleiche SPS, unabhängig davon, in welcher Programmiersprache erstellt, werden letztlich in eine Folge von Maschinenbefehlen (Maschinenprogramm) für das Automatisierungsgerät umgewandelt („compiliert"). Oft sind solche Maschinenprogramme rückdarstellbar, d.h. es gibt eine Art „Discompiler", der aus einer Folge von Maschinenbefehlen eine Programmdarstellung in einer der obengenannten Programmiersprachen erzeugen kann. Damit ist eine Umwandlung von einem KOP- in ein FUP-Programm und umgekehrt möglich. Auch andere Wandlungsrichtungen sind mit bestimmten Grenzen durchführbar.
- Der Zwischenschritt der verschiedenen Programmiersprachen wurde deshalb eingefügt, um den Programmierer von den mühseligen Details der Maschinenbefehle zu entlasten und ihm eine Darstellungsform zu bieten, die er aus seiner früheren Tätigkeit gewohnt ist.
- Die Programmiersprachen KOP, FUP und AS (GRAFCET) sind grafische Programmiersprachen, bei denen man einen Programmteil aus einem begrenzten Vorrat von grafischen Symbolen „komponiert". AWL und ST sind dagegen textlich orientierte Programmiermöglichkeiten, bei denen der Programmierer eine bestimmte Syntax bei der

Formulierung der Anweisungen selbst beachten muß.

Wir wollen nun etwas näher auf die neben der Kontaktplanprogrammierung hauptsächlich benutzten Sprachformen FUP und AWL eingehen.

Funktionsplansprache (FUP)

In Teil 1 der Artikelreihe wurden bereits die wesentlichen FUP-Schaltzeichen Negation, UND sowie ODER eingeführt. Aus derartigen Symbolen kann man nun Netzwerke zusammenstellen, die logische Verknüpfungen realisieren. Als Beispiel betrachten wir zwei immer wieder vorkommende zusammengesetzte logische Grund-Verknüpfungen in der Stromlaufplandarstellung: die UND-vor-ODER-Verknüpfung (Bild 17) und die ODER-vor-UND-Verknüpfung (Bild 18).

Mit ein wenig Vorstellungsvermögen können wir bereits ableiten (und Sie können mit Hilfe selbst erstellter Wahrheitstabellen nachweisen), wie diese Schaltungen wirken:

- bei der UND-vor-ODER-Verknüpfung (Bild 17) muß lediglich einer der drei vertikalen Strompfade stromführend sein, damit die Lampe A5.0 leuchtet. Es reicht z.B. aus, wenn E1.2 durchschaltet. Der Name dieser Verknüpfungsart leitet sich daraus ab, daß die UND-Verknüpfungen der beiden vertikalen Pfade mit Vorrang bewertet werden und erst anschließend die ODER-Verknüpfung aus den drei Parallelpfaden bestimmt wird.
- bei der ODER-vor-UND-Schaltung (Bild 18) müssen sowohl (E1.0 v E1.1 v E1.2) als auch (E1.3 v E1.4) sowie E1.5 jeweils log. 1 für das Ansteuern der Lampe A5.0 aufweisen. Der Name der Verknüpfung weist darauf hin, daß entgegen den Regeln der Booleschen Algebra erst die ODER-Verknüpfungen bestimmt werden müssen und dann die Reihenschaltung (UND) der Ergebnisse erfolgt. Dies wird durch geeignete Zusammenfassung erreicht.

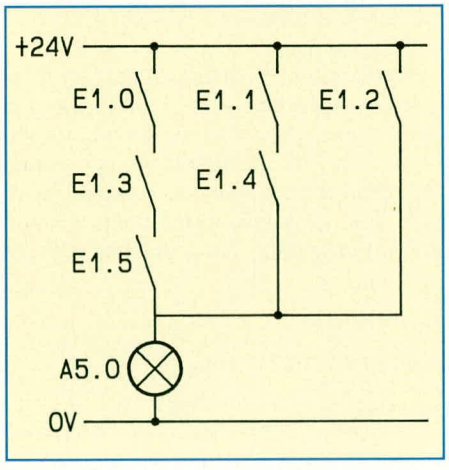


Bild 17: UND-vor-ODER-Verknüpfung

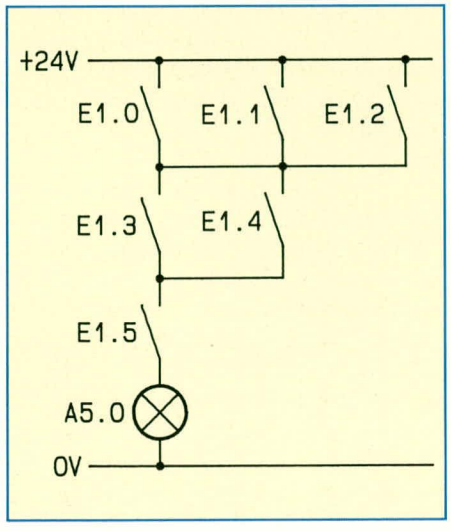


Bild 18: ODER-vor-UND-Verknüpfung

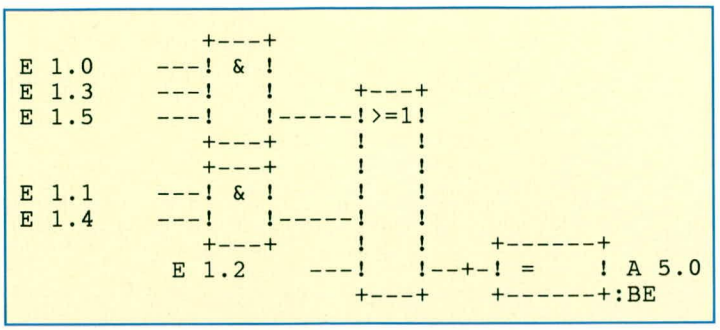


Bild 19: UND-vor-ODER-Verknüpfung in FUP

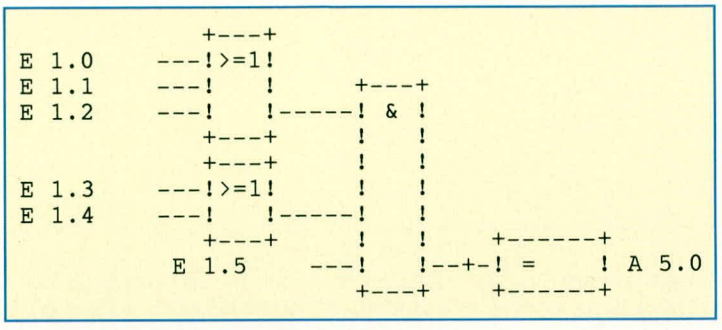


Bild 20: ODER-vor-UND-Schaltung in FUP

In den Abbildungen 19 und 20 sind diese beiden Strompfade als FUP-Darstellung zweier SPS-Netzwerke zu sehen.

Neben den drei Standardelementen NEGATION, UND sowie ODER gibt es auch in FUP ergänzende Sprachelemente wie z.B. den Timer, der hier ganz ähnlich wie in KOP erscheint. Aus der im zweiten Artikelteil gezeigten Stern-Dreieckschaltung ergibt sich demzufolge die in Abbildung 21 gezeigte FUP-Darstellung.

Hier sollen auch die in den meisten SPS eingebauten Zähler-Bausteine eingeführt werden. Sie dienen dem Zählen von Ereignissen in einem Prozeß, z.B. zur Erfassung der Anzahl in einer Anlage gefertigter Teile oder des Vorrats an Teilen in einer Förderstrecke. Gerade die Teilezählung macht in der Regel das Vorwärts- und Rückwärtszählen notwendig, daher werden in einem Zählerbaustein oft beide Möglichkeiten zusammen angeboten.

Bild 22 zeigt einen Vorwärts-/ Rückwärtszähler, der inkrementiert wird, wenn die Endschalter E17.6 und E18.0 gleichzeitig betätigt sind (ZV = zählen vorwärts), und der beim Betätigen von E3.2 vermindert wird (ZR = zählen rückwärts).

Für manche Anwendungen ist das Vorbesetzen des Zählers mit einem Anfangswert sinnvoll. Zu diesem Zweck ist ein Setzeingang S vorgesehen. Wird S durch E17.0 (positive Flanke) betätigt, dann wird der Zählwert 25 (KZ 025) in den Zähler übernommen, der Zählvorgang läuft mit diesem Wert weiter. Auch ein Rücksetzen des Zählers auf 0 kann durch ein Signal bewirkt werden (E17.1 statisch am R-Eingang). Der Zählerausgang Q ist „wahr", wenn der Zählwert ungleich 0 ist, und sonst ausgeschaltet.

Anweisungsliste (AWL)

Die AWL ist die umfassendste Sprachform bei der SPS-Programmierung, da sich sämtliche möglichen SPS-Befehle hier benutzen lassen, während die grafischen Sprachen FUP und KOP (u.a. wegen des grafischen Editors) bestimmten Einschränkungen unterliegen. Als Regel gilt, daß

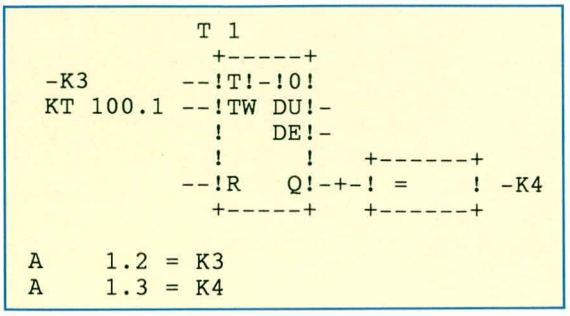
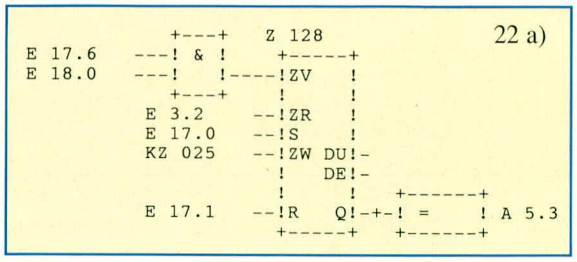


Bild 21: Netzwerk 2 Stern/ Dreieckschaltung, FUP



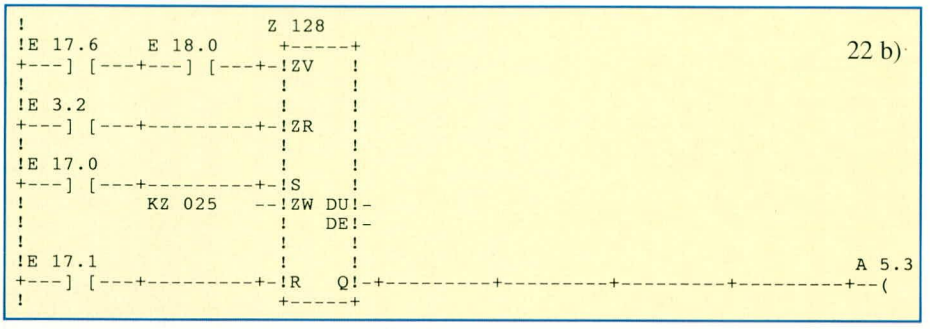


Bild 22: Vorwärts-/ Rückwärtszähler a.) in FUP-Darstellung b.) in KOP

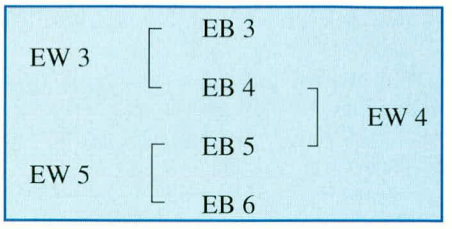


Bild 24: Zusammenhang zwischen Eingangsbytes und -worten

alle grafisch erzeugten Programme (KOP, FUP, AS) in AWL-Darstellung umwandelbar sind. Die Regel ist jedoch nicht umkehrbar, da bestimmte Anweisungen (z.B. Lade- und Transferbefehle) bei verschiedenen Programmierpaketen nicht in Grafik darstellbar sind.

Die AWL-Programmierung baut Netzwerke aus aneinandergereihten Anweisungen auf. Die Anweisungen haben den in Bild 23 abgebildeten Aufbau (DIN 19239):

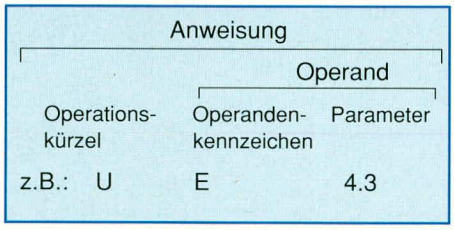


Bild 23: Aufbau einer SPS-Programmanweisung

In Tabelle 6 sind einige Beispiele für Operationskürzel zu sehen.

Einige Informationen zum Parameteraufbau haben wir im zweiten Teil bei unseren bisherigen KOP-Beispielen bereits behandelt. Dort wurde klar, daß die Parameter den Hardwareaufbau der Eingangs-/ Ausgangs-Peripherie der SPS widerspiegeln: acht einzelne Eingangsbits der SPS sind in Bytes zusammengefaßt, die Eingangsbits E 15.0 bis E 15.7 bilden das Eingangsbyte EB 15. Zwei benachbarte Bytes kann man nun gemeinsam als Wort ansprechen. Die beiden Eingangsbytes EB 3 und EB 4 z.B. bilden zusammen das Eingangswort EW 3, die Bytes EB 5 und EB 6 bilden das Wort EW 5 usw. Es kann auch eine Überschneidung von Worten geben, wenn man beispielsweise EW 3 und EW 4 verwendet, die sich im Byte EB 4 überdecken (Bild 24). Auch Doppelworte können bei vielen Steuerungen benutzt werden: Sie fassen 4 benachbarte Bytes zusammen.

Wir wollen uns nun als Anweisungslisten-Beispiel das Netzwerk 1 der Stern/ Dreieckschaltung einmal in AWL-Form ansehen (Bild 25).

Hinweise: Die Doppelpunkte vor den Anweisungen werden durch das Programmiergerät vorgegeben. Sowohl die absoluten als auch die symbolischen Adressen werden aufgeführt, die Bindestriche vor den Variablennamen (z.B. -F1) teilen dem Programmiersystem mit, daß es sich hier um symbolische Namen handelt. Klammerungen („U(..)") sind notwendig, um die beabsichtigte Bewertungsreihenfolge sicherzustellen. Die 01-Einträge kennzeichnen die erste Klammerebene. Klammern sorgen dafür, daß zunächst die Ausdrükke in den Klammern (3. und 4. Element) bewertet werden. Erst die Ergebnisse dieser Bewertung fließen in die UND-Verknüpfung der obersten Ebene ein. Ganz rechts findet sich ein Anweisungskommentar.

Um auch eine Timer-Programmierung in AWL-Darstellung zu sehen, betrachten wir das Netzwerk 2 unserer Stern-/Dreieckschaltung (Bild 26). Als Besonderheit ist hier zu erkennen, daß auch ein Timer einen logischen Zustand aufweist: UT 1 hat den Wahrheitswert 1, wenn der Timer abgelaufen ist.

Nach diesem Überfliegen der SPS-Programmiersprachen ist es nun notwendig, einen Aspekt der SPS-Programmierung zu behandeln, der insbesondere bei größeren SPS-Programmen entscheidend wird: wie kann es gelingen, einen großen Programmumfang noch übersichtlich zu gestalten?

Tabelle 6: Beispiele für Operationskürzel

Verwendung	Kürzel	Erläuterungen
Verknüpfungen	U, O, UN, ON	(UND, ODER, UND NICHT, ODER NICHT)
Zuweisung	=	
Speicheroperation	S, R	(Setzen, Rücksetzen)
Zeitoperation	SE, SS, SA,..	(Einschaltverzögerung, speichernde Einschaltverzögerung, Ausschaltverzögerung..)
Zähloperation	ZV, ZR, S, R	(Vorwärtszählen, Rückwärtszählen, Setzen, Rücksetzen)
Lade-,Transferop.	L, T	(Laden, Transferieren)
Vergleichs-, arithmet. Operation	=F,..,><F,+F,-F..	(Vergleiche, Addition, Subtraktion,..)
Bausteinaufrufe	SPA,.., BE	(abs. Bausteinaufruf, .., Bausteinende)

Als Operanden-Kennzeichen dienen je nach Operation :

- E, A, M , T, Z	bei Verknüpfungen (Eingang, Ausgang, Merker, Timer, Zähler)
- A, M	bei Zuweisungen und Speicheroperationen
- T	bei Zeitoperationen
- Z	bei Zähloperationen
- EB, EW, AB, AW, MB,..	bei Lade- und Transferoperationen (Eingangsbyte, Eingangswort, Ausgangsbyte, Ausgangswort,..)

```
:UN   E    1.0  -F1                  1. Element
:UN   E    1.1  -S1                  2. Element
:U(                                  3. El.:Parallelschaltg. aus:
:O    E    1.2  -S2        01          Einzelelement und
:O                         01          Reihenschaltung aus:
:U    A    1.0  -K1        01             1. Element und
:U    A    1.2  -K3        01             2. Element
:)                         01          Ende 3. Element
:U(                                  4. El.: Parallelschaltg aus:
:ON   A    1.0  -K1        01          Einzelelement und
:O    A    1.2  -K3        01          Einzelement
:)                         01          Ende 4. Element
:UN   A    1.1  -K2                  5. Element
```

Bild 25: Netzwerk 1 (Stern-/Dreieckschaltung) in AWL-Darstellung

```
:U    A    1.2  -K3
:L    KT 100.1       Zeitwert u. Zeitraster laden
:SE   T    1          Zeitfkt. Einschaltverzg. (SE) laden
:NOP  0               nur f. Rckdarstellung in KOP/FUP (no operation)
:NOP  0                          "
:NOP  0                          "
:U    T    1          Zeitfunktion abfragen
:=    A    1.3  -K4   bei abgelaufener Zeit Ausgang setzen
:***                  Ende-Kennung dieses Netzwerkes
```

Bild 26: Netzwerk 2 (Stern-/Dreieckschaltung) in AWL-Darstellung

9. Strukturierung von SPS-Programmen

Frühe SPS-Systeme wiesen einen linearen Programmaufbau auf. Anweisung für Anweisung wurde in den Speicher hineinprogrammiert, bis dieser voll war. Aber: Wächst der Umfang eines linearen, d.h. nicht untergliederten SPS-Programmes auf mehrere Druckseiten an, dann hat der Betrachter schon Probleme, dessen Gesamtfunktion zu überschauen. Sehr bald gingen SPS-Hersteller daher dazu über, Methoden zur Strukturierung der SPS-Programme zu erarbeiten. Der folgende Weg, der Ähnlichkeit mit Lösungen im Bereich der Hochsprachenprogrammierung aufweist, hat sich dabei als Quasi-Standard herausgebildet: die Bildung und Nutzung von Bausteinen.

SPS-Programmnetzwerke werden entsprechend ihren Funktionen im zu steuernden Prozeß zu Bausteinen zusammengefaßt. Weist z.B. eine zu automatisierende Fertigungsanlage 3 Fertigungsstationen und eine Fördereinrichtung auf, so würde es sich u.U. anbieten, für jede Station und für den Förderer je einen Baustein zu erstellen, der alle zugehörigen SPS-Netzwerke enthält.

Derartige Bausteine können separat programmiert, geändert, gespeichert und geladen werden. Bausteine können in gewissen Grenzen andere Bausteine nutzen, d.h. sie können diese bei Bedarf (auch abhängig von Bedingungen) aufrufen. So läßt sich eine Art Unterprogrammtechnik wie bei Pascal realisieren.

Es gibt verschiedene Typen von Bausteinen (die folgenden Erläuterungen beziehen sich auf die Siemens-SIMATIC-Familie):

Organisationsbausteine(OB):

Diese dienen im wesentlichen dazu, die Bearbeitung der anderen Bausteine zu organisieren, sowohl im Normalbetrieb als auch in Sondersituationen, wie z.B. nach dem Einschalten der SPS oder nach Netzausfall.

Programmbausteine (PB):

Sie enthalten den Großteil der Programmlogik. In einem Programmbaustein werden alle Netzwerke zusammengefaßt, die in bezug auf die zu steuernde Maschinenfunktion zusammengehören (z.B. für eine Station).

Funktionsbaustein(FB):

Auch sie fassen zusammengehörige Netzwerke zusammen. Hier können jedoch - anders als bei PBs - neben Standardanweisungen auch alle Sonder-Anweisungen verwendet werden (sog. ergänzende Funktionen wie z.B. Sprungfunktionen). Weiterhin ermöglichen die FBs die Verwendung von Parametern und Argumenten, wie es in der Unterprogrammtechnik, etwa bei Pascal, üblich ist: ein FB kann einmal programmiert, aber mit verschiedenen Variablen mehrfach genutzt (aufgerufen) werden.

Beispiel.: Eine Aufzugsteuerung soll 4 identische Türsteuerungen bedienen. Die Netzwerke für jede Tür sind identisch bis auf die Unterschiede in den Eingangs- und Ausgangsadressen: Etage 1 enthält die Endschalter E13.1 und E13.2, Etage 2 die Endschalter E17.0 und E17.3 usw. Man erstellt nun einen einzigen Funktionsbaustein, der die notwendigen Netzwerke enthält. Diese enthalten jedoch symbolische Adressen (Parameter), z.B. =ZU oder =AUF, die noch keinen Bezug zu irgendeiner absoluten Adresse haben. Erst beim Aufruf dieses Funktionsbausteins durch einen anderen Baustein werden wirkliche Ein-/Ausgabeadressen übergeben (Argumente), mit denen die im FB enthaltenen Netzwerke dann abgearbeitet werden: Beim Aufruf des Tür-Funktionsbausteins für die Etage 1 wird =AUF in den Netzwerken dann durch E13.1 und =ZU durch E13.2 ersetzt, bei Etage 2 wird aus =AUF die Adresse E17.0 und aus =ZU die Adresse E17.3.

Schrittbausteine:

Sie übernehmen organisatorische Aufgaben im Zusammenhang mit der Schrittketten- oder Ablaufprogrammierung mit der Ablaufsprache. Sie werden hier nicht weiter behandelt.

Datenbausteine:

Hier werden Daten abgelegt, die vom Anwenderprogramm erzeugt oder genutzt werden können. Sie dienen z.B. als Zwischenspeicher für Rechenergebnisse. Die anderen Bausteinarten können auf Datenworte in Datenbausteinen lesend oder schreibend zugreifen.

Will man ein SPS-Programm strukturieren, dann steht zunächst der wichtige OB1 zur Verfügung. Er ist der Baustein, der nach dem Aktualisieren der Eingangssignale vom Betriebssystem der SPS aufgerufen wird. Oft programmiert man in den OB1 ausschließlich Aufrufe der anderen Bausteine hinein:

```
OB1:
  : SPA PB 1     absoluter Sprung
                 zum PB 1
  : SPA PB 2     absoluter Sprung
                 zum PB 2
  : SPA PB 3     absoluter Sprung
                 zum PB 3
  : BE           Bausteinende
```

Diese Sprünge werden der Reihe nach abgearbeitet. Dabei werden die angesprungenen Bausteine komplett bearbeitet. Auch aufgerufene Programm- oder Funktionsbausteine können mit derartigen absoluten oder bedingten Sprüngen ausgestattet sein, so daß man auf diese Weise eine ganze Aufruf-Hierarchie zwischen den Bausteinen herstellen kann. Diese Gliederungsmöglichkeiten erlauben die Aufteilung einer großen Automatisierungsaufgabe in übersichtliche Programmabschnitte.

Ausblick

In der nächsten Folge werden wir uns mit der Wortverarbeitung und den Codes im SPS-Bereich befassen, Darauf folgen Überlegungen zum Entwurf von SPS-Programmen. ELV