## 2.7 Grundsätzliches zu speicherprogrammierbaren Steuerungen

Mit dem Vorstellen der Komponenten einer SPS, dem Erläutern ihrer prinzipiellen Arbeitsweise, dem Gegenüberstellen der Möglichkeiten zum Projektieren und Programmieren, dem Erklären einiger technischer Begriffe und dem Beschreiben der Maßnahmen gegen Funktionsstörungen sollen nun speicherprogrammierbare Steuerung zunächst grundsätzlich behandelt werden. Danach wird ein reales System vorgestellt. Allgemeine technische Ausführungen zum mechanischen Aufbau, (Flachbaugruppen, Baugruppenträger, Schaltschränke) und zur Verbindungstechnik erfolgen hier nicht.

### 2.7.1 Grundbausteine einer SPS

Bild 2.8 zeigt die Struktur einer SPS. Die wichtigsten Funktionselemente einer speicherprogrammierbaren Steuerung sind das Steuerwerk mit einem Mikroprozessor, manchmal sind es auch mehrere sowie die zugehörigen Speicher für Daten (Speicherbereiche für Zeiten, Zähler, Merker, Prozeßabbilder) und Programme (Programmspeicher). In SPSen werden als Speicher fast ausschließlich folgende programmierbare Speicher verwendet:

**RAM** (Schreib-Lese-Speicher) bieten höchste Flexibilität, erfordern jedoch als flüchtige Speicher Pufferbatterien zur Sicherung der gespeicherten Informationen gegen Ausfall der Spannungsversorgung.

**EPROM** (lösch- und programmierbare Nur-Lese-Speicher) müssen für Programmänderungen aus dem Gerät herausgenommen, können inhaltlich nur insgesamt gelöscht und müssen anschließend völlig neu - mit geändertem Programm -beschrieben werden. Sie verlieren als nicht flüchtige Speicher zwar ihre Informationen bei Spannungsausfall nicht, sind wegen der umständlichen Neuprogrammierung aber wenig flexibel.

**EEPROM** (elektrisch lösch- und programmierbare Nur-Lese-Speicher) sind zwar wie RAM-Speicher zeIlenweise zu beschreiben, jedoch wesentlich langsamer und wegen der begrenzten Zahl von Schreibvorgängen nur zur Daten- und Programmsicherung als Hintergrundspeicher von RAM-Speichern zu verwenden.

Die Realisierung interner, vom Prozessor bearbeiteter Zeit- und Zählfunktionen erfordert auf jeden Fall immer einen Mindestaufwand an RAM-Speichern.

Steuerwerk, Speicher sowie Ein- und Ausgabeglieder (Peripherie) müssen zum Datenaustausch untereinander Verbindung haben. Hier hat sich eine Verbindung über Sammelleitungen durchgesetzt, die als Bus bezeichnet wird. Parallel über den Bus werden Daten übertragen, und zwar gleichzeitig mit Adreßsignalen, die anzeigen, welche Funktionseinheit der Adressat der Daten ist. Das Steuerwerk verkehrt über Signaleingabe- und -ausgabeeinheiten in ähnlicher Weise mit dem Prozeß und der Prozeßführung.

Dieser Aufbau entspricht im Prinzip dem eines digitalen Einzelreglers, den wir in der Kurseinheit 1 kennen gelernt haben. Für Steuerungen sind jedoch mehr binäre und weniger analoge Ein- und Ausgänge erforderlich, in den Speichern sind mehr Zeitfunktionen und Zähler zu realisieren und bei einfachen Steuergeräten nur binäre und oft keine arithmetischen Funktionen zur Verfügung zu stellen.

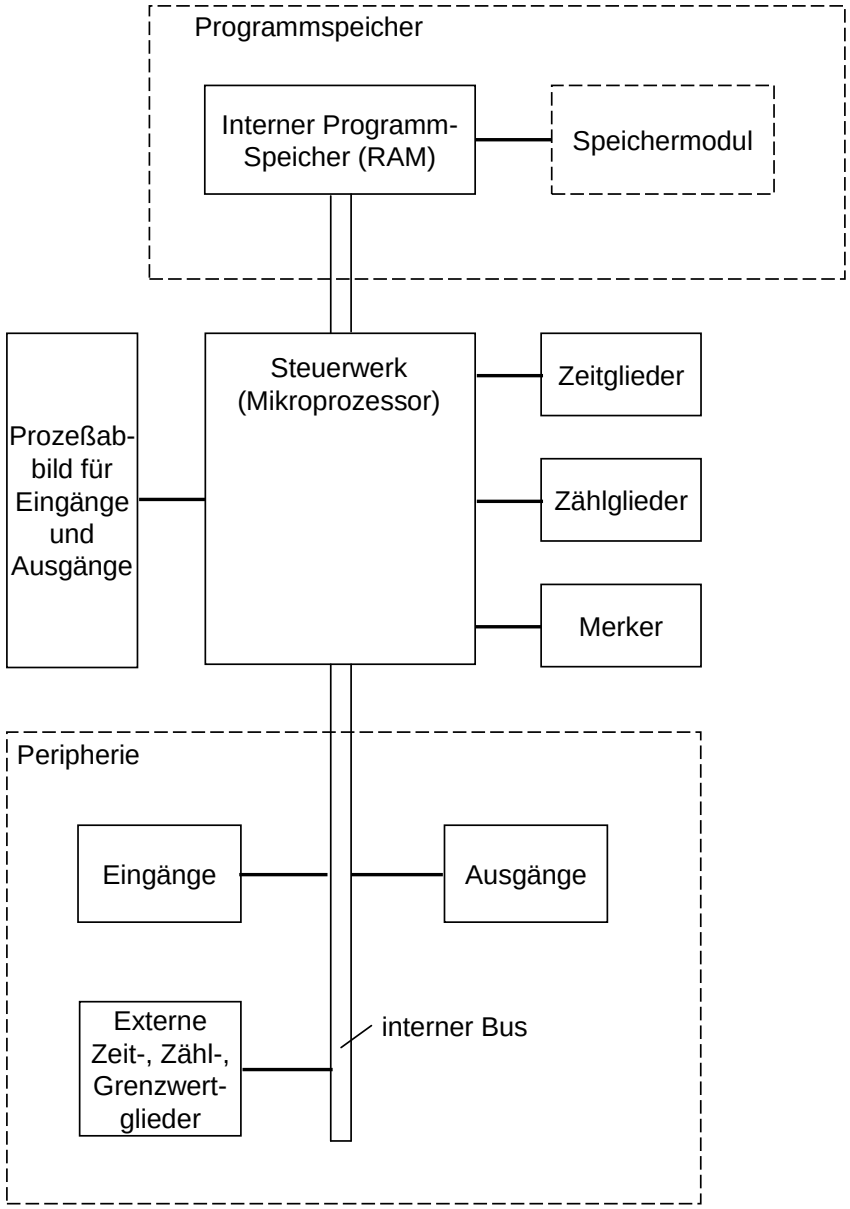



Bild 2.8: Grundbaustein einer SPS (Siemens).
Kern ist das Steuerwerk mit dem Mikroprozessor und den Speicherbereichen für Prozeßabbilder, Zeiten, Zähler und Merker. Der Programmspeicher enthält das zu bearbeitende Programm. Die Peripheriekomponenten stellen die Verbindung zum Prozeß her.

#### 2.7.1.1 Art der Prozessoren in SPSen

Ein Bitprozessor kann einzelne Bits -Binärsignale -verarbeiten, z.B. einzelne Binärsignale der Reihe nach abfragen und entsprechend eines vorgegebenen Programms (bei Binärsignalen meist Verknüpfungsfunktionen) verarbeiten. Ein Wortprozessor kann dagegen nur Worte, also mehrstellige Binärmuster (z.B. 16-stellige), abfragen und verarbeiten. Um ein einzelnes Binärsignal zu verarbeiten, muß ein solcher Prozessor mittels Verschiebebefehlen oder Maskenbildung auf das in einem Wort befindliche betreffende Binärsignal zugreifen. Ein Wortprozessor ist deshalb besser für Wortoperationen (z.B. addieren) geeignet, Binärfunktionen führt er nur unter hohem Zeitaufwand durch. So haben leistungsfähige Steuerungen heute oft sowohl Wort- als auch Bitprozessoren, wobei erstere die Wortoperationen und die anderen die Binärfunktionen übernehmen.

### 2.7.2 Arbeitsweise speicherprogrammierbarer Steuerungen

Die vier Teilbilder a bis d des Bildes 2.9 zeigen die wesentlichen Ablaufschritte eines Programmdurchlaufs, auch Zyklus genannt. Zu Beginn eines jeden Zyklusses fragt das Steuerwerk die Signalzustände an den Eingängen der Steuerung sehr schnell hintereinander ab und setzt im Prozeßabbild für Eingänge (PAE) die jedem Eingang zugeordnete Speicherzelle so auf 0 oder 1, daß diese Speicherzellen nach Ablauf dieses Bearbeitungsschrittes ein Abbild der Signalzustände der Eingänge enthalten (Bild 2.9a).

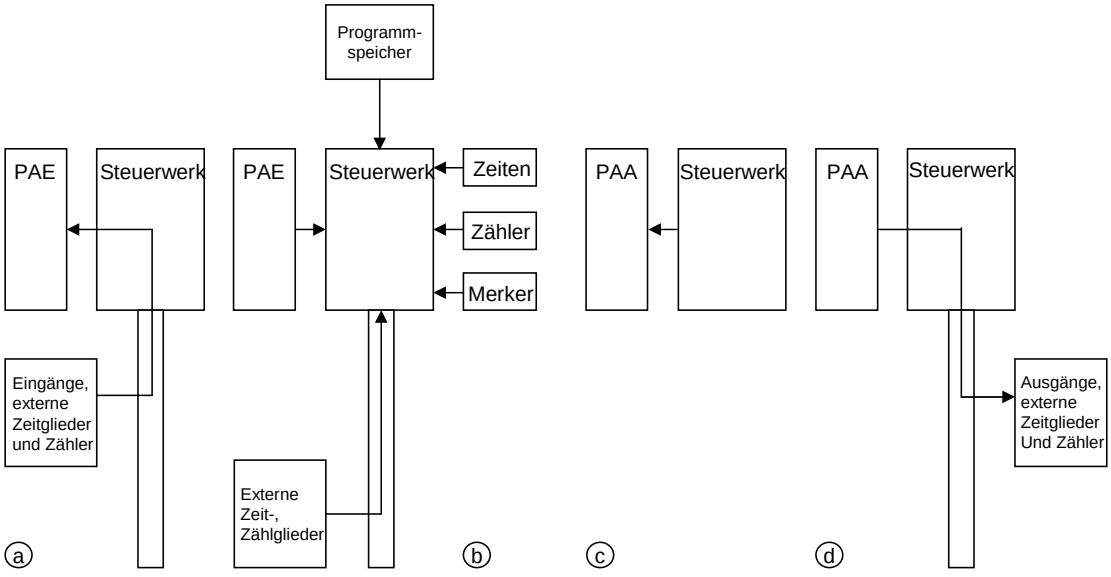



Bild 2.9: Zur Arbeitsweise einer SPS (Siemens).
Den Bearbeitungszyklus kann man grob unterteilen in die Phasen
a) Aktualisieren des Prozeßabbildes,
b) Bearbeiten des Programms,
c) Übertragen der Ergebnisse dieser Bearbeitung in das Prozeßabbild der Ausgänge und
d) anschließendes Übertragen dieses Abbildes auf die Ausgänge des Steuergerätes.

Während der nun folgenden Programmbearbeitung greift der Prozessor des Steuerwerkes auf dieses Prozeßbild und nicht direkt auf die Eingänge zurück und bearbeitet abhängig davon die im Programmspeicher stehenden Steuerungsanweisungen (Bild 2.9b). Diese sagen, sehr vereinfacht ausgedrückt, aus, welcher Eingang mit welchem anderen wie zu einem bestimmten Ausgangssignal zu verknüpfen ist. Diese Operationen werden mitunter auch Befehle genannt. Nach DIN19239 steht z.B. der Buchstabe „U“ für die Binäroperationen „UND“, der Buchstabe „0“ steht für „ODER“, „ON“ für „ODER-NICHT“ und das Gleichheitszeichen „=“ bedeutet die Zuweisung eines erarbeiteten Verknüpfungsergebnisses zu einem bestimmten Ausgang der Steuerung. Die Operandenteile der Anweisungen (Bild 2.10a) sind nach Operandenkennzeichen (Eingänge E, Ausgänge A, Merker M usw.) und Parametern (Adressen) unterteilt. Üblicherweise werden die Parameter byte-weise, d.h. in Gruppen zu 8, strukturiert. Die Beschreibung „Eingang E 2.3“ ist so zu interpretieren: die Ziffer vor dem Punkt ist die Byte-Nummer, die Ziffer dahinter gibt einen der acht Eingangskanäle 0 bis 7 dieses Bytes an. Im Verlauf dieses Arbeitsganges muß das Steuerwerk auch, wie im Bild dargestellt, Verbindung zu externen Zeit- und Zählgliedern haben.

Die Verknüpfungsergebnisse, also die Sollsignalzustände von Ausgängen, werden während der Programmbearbeitung in den Speicherzellen des Prozeßabbildes der Ausgänge (PAA) hinterlegt (Bild 2.9c). So entsteht Schritt für Schritt ein Prozeßabbild für den neuen Sollzustand der Ausgänge der Steuerung.

Erst nach Abschluß der Programmbearbeitung, also am Ende des Bearbeitungszyklus', überträgt das Steuerwerk den Inhalt (die Sollsignalzustände) des Prozeßabbilds der Ausgänge zu den Ausgabebaugruppen (Bild 2.9d). Danach beginnt das Steuerwerk den nächsten Bearbeitungszyklus mit der Übernahme der Signalzustände der Eingänge, anschließender Programmbearbeitung und nachfolgender Ausgabe des Prozeßabbildes der Ausgänge an die Ausgaben. Diese periodische Bearbeitung geht nun mit typischen Zykluszeiten von 1 bis 10msec bei Ausführung von 1.000 Anweisungen für Binäroperationen so schnell vor sich, daß sequentiell arbeitende speicherprogrammierbare Steuerungen nach außen wie parallel arbeitende verbindungsprogrammierte Steuerung wirken.

Die permanent zyklische Bearbeitung ist eine Eigenart der meisten Steuerungen. Obwohl die Zykluszeiten extrem kurz sein können, stellen sich doch bei der Prozeßbearbeitung unterschiedliche Zykluszeiten ein. Abhängig vom Prozeßzustand durchlaufen Steuerungen zum einen unterschiedliche Bearbeitungszweige und zum anderen können die Prozeßantwortzeiten unterschiedlich sein, z.B. beim Öffnen oder Schließen von Stellgeräten. Im Gegensatz dazu wird die Aktivität verteilter Prozeßleitsysteme (Kurseinheit 3) von festen Zeittakten gesteuert.

### 2.7.3 Kommunikationsschnittstellen

Speicherprogrammierbare Steuerungsgeräte sind heute allgemein mit Schnittstellenmodulen für Datenendverbindungen zur Programmierung und Rückdokumentation, zur Integration in Prozeßführungssysteme sowie zur Ankopplung von intelligenten Feldgeräten ausrüstbar. Dabei kommen serielle Schnittstellen, Feldbusse und in zunehmenden Maße Ethernet-basierte Systeme zum Einsatz.

### 2.7.4 Programmierung und Projektierung

Die Prozessoren speicherprogrammierbarer Steuerungen haben wie alle Digitalrechner die von-Neumann-Architektur, d.h. Programme werden in Speichern abgelegt. Wie wir wissen

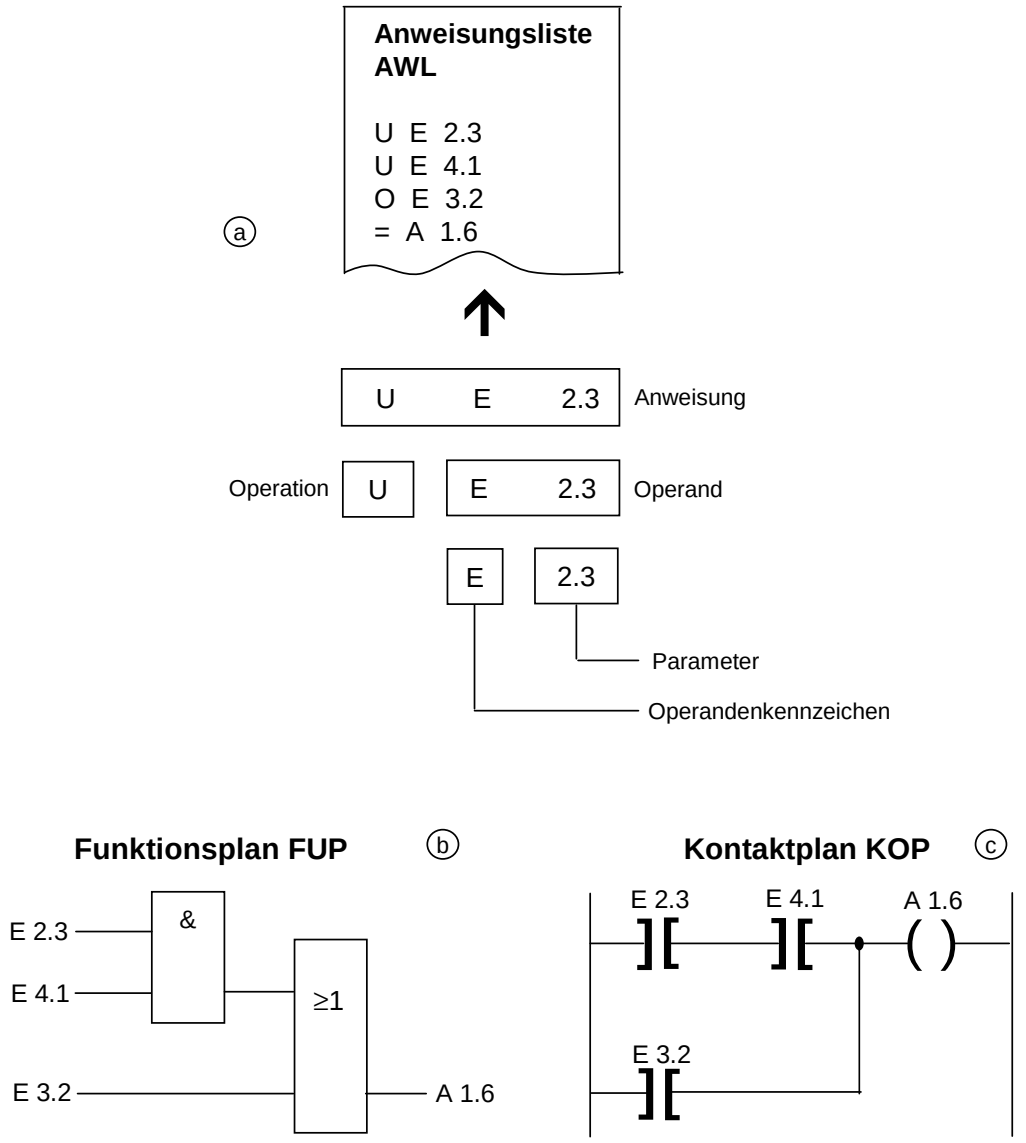



Bild 2.10: SPS-Programmiersprachen (Siemens).
Steuerungsaufgaben lassen sich in verschiedenen „Sprachen“ darstellen. Eine Anweisungsliste (a) läßt sich auch mit einfachen Programmiergeräten in die einer Steuerung verständlich Maschinensprache übersetzen. Der für die Verständigung unter den Ingenieuren besser geeignete Funktionsplan (b) und der in Amerika beliebte Kontaktplan (c) stellen höhere Ansprüche an Programmiergeräte.

und wie auch schon der Name sagt, ist ein Programm eine Folge der Anweisungen, bei SPSen in Form von Bitmustern, die eine Steuerung zu bearbeiten haben. Man kann sich das so vorstellen, daß die z.B. 64 Anweisungen oder Befehle eines Prozessors durch unterschiedliche Kombinationen von sechs Schaltern eingestellt werden, wobei ein bestimmtes Bit-Muster einen bestimmten Befehl repräsentiert. Auch Operandenadressen müssen Prozessoren in Form von Bit-Mustern mitgeteilt werden.

Nun wird niemand eine SPS mit Bit-Mustern programmieren, also für eine UND-Verknüpfung z.B. jedesmal 011001 einschreiben. Gerätehersteller werden vielmehr andere Möglichkeiten der Programmierung und Adressierung bieten z.B. in der Art, für die UND-Verknüpfung ein U auf einem alphanumerischem Datensichtgerät einzugeben oder Kontaktsymbole in einem Kontaktplan bzw. Funktionsplanelemente in einem Funktionsplan auf einem graphischen Bildschirm zu kombinieren. Adressen können bei komfortablen Systemen symbolisch vorgegeben werden, z.B. als ISTWERT 1 bis ISTWERT n, zumindest aber als Dezimalzahlen. Das Übersetzen der Informationen auf einer graphischen Programmieroberfläche in die Maschinensprache einer SPS besorgt ein Programmiergerät. (Es wäre technisch möglich, auch diese Aufgaben direkt der SPS zuzuordnen. Besonders kleine SPSen müßten dafür aber sehr aufwendig aufgerüstet werden, was Einfluß auf ihr Zeitverhalten haben kann und besonders deshalb unwirtschaftlich ist, da sich i.a. mit einem einzigen Programmiergerät sehr viele SPSen programmieren lassen, die nur bei Bedarf mit dem PG zu verbinden sind.) So erfolgt die Programmierung der meisten speicherprogrammierbaren Steuerungen mit Programmiergeräten, was eines ihrer kennzeichnenden Merkmale ist.

#### 2.7.4.1 Programmiergeräte

Programmiergeräte sind Mikrorechner spezieller Ausführung, die in den mittleren und oberen Preisklassen auch mit großflächigen LCD-Displays oder mit Bildschirmen ausgerüstet werden. Zunehmend arbeiten sie aber auch auf der Basis von Personalcomputern (PC) oder Arbeitsplatzrechnern (Workstations). Sie können die vom Anwender in einer steuerungsspezifischen höheren Programmiersprache, z.B. als Anweisungsliste (Bild 2.10a), Funktionsplan (Bild 2.10b) oder Kontaktplan (Bild 2.10c), eingegebenen Funktionen direkt in den Maschinencode von Steuerungsgeräten übersetzen, aber auch aus Maschinencode in eine höherwertige Darstellung rückübersetzen. Solche Geräte sind in direkter Kopplung mit Steuergeräten einsetzbar. Programme werden in diesem Falle meistens direkt in die Programmspeicher der Steuergeräte eingegeben bzw. zur Darstellung aus diesen Speichern gelesen. Im „on-line“-Betrieb sind aber auch Testfunktionen, Fehlersuchen und Programmkorrekturen leicht durchführbar.

Die Betriebsart „off-line“ setzt in Programmiergeräten RAM-Speicher zur Aufnahme der Programme voraus. Aus diesen Speichern lassen sich dann Programme

- auf externe Datenträger (z.B. Disketten) kopieren oder archivieren,
- in EPROM-Module übertragen, mit deren Hilfe dann die „off-line“ erstellten Programme in Steuergeräte „eingesteckt“ werden, oder
- „on-line“ in direkter Kopplung in Steuergeräte übertragen.

#### 2.7.4.2 Bausteine zur Rationalisierung des Programmierens

Zur Rationalisierung des Programmierens können häufiger vorkommende Folgen von Steuerungsanweisungen zu Funktionsbausteinen zusammengefaßt werden. So gibt es z.B. Funkti-

onsbausteine zur Ablaufsteuerung, PID-Schrittregelung, Extremwertauswahl, Protokollierung oder Codeumsetzung. Aber der Anwender kann auch selbst Steuerungsanweisungen zu Funktionsbausteinen zusammenstellen, z.B. für die Ansteuerung eines Stellgerätes mit zugehörigen Verriegelungen und Überwachung von Laufzeiten.

Ein Funktionsbaustein entsteht, indem man die Steueranweisungen der zu programmierenden Funktion nicht mit absoluten Parametern angibt, sondern diese allgemein bezeichnet, z.B. mit $X_{1...n}$, $Y_{1...n}$. Immer dann, wenn man vom Anwenderprogramm her einen Funktionsbaustein in Anspruch nehmen will, wird nach dem Aufruf des Bausteins nur noch angegeben, welche aktuellen Parameter den formalen zugeordnet sind, d.h. beispielsweise, mit welchen Ein- und Ausgängen, Merkern und Zählern der Funktionsbaustein an dieser Stelle des Programms arbeiten soll.

Der Anwender braucht deshalb auch die Befehlsfolge innerhalb eines Standardbausteins nicht zu kennen und wird sie bei komplizierten Bausteinen i.a. auch nicht verstehen. Ein Baustein wird von ihm als „schwarzer Kasten“ betrachtet. Die Hersteller von SPSen liefern Datenträger, z.B. Disketten, mit Funktionsbausteinen für ihre Systeme, die dann mit Hilfe von Programmiergeräten wie oben beschrieben in Programme eingebunden werden können.

#### 2.7.4.3 Hardware statt Software

Fallende Hardware- und steigende Software-Kosten haben zu einer für Verfügbarkeit und Übersichtlichkeit positiven Entwicklung der Struktur digitaler Systeme geführt. Während in den Anfängen alle Berechnungen, Steuerungsfunktionen und Regelungsfunktionen für eine gesamte Verfahrensanlage an einer einzigen zentralen Stelle, dem Prozeßrechner, bearbeitet wurden, wird jetzt mehr und mehr „die Intelligenz verteilt“. Mikrorechner in peripheren Geräten, z.B. in Meßumformern, arbeiten Signale auf und entlasten damit zentrale Geräte, sparen Rechenzeit und erleichtern die Programmierung.

Bei SPSen wird diese Entwicklung durch das steigende Angebot sogenannter „intelligenter“ Peripheriebaugruppen sichtbar. Dies sind eigenständige Mikrorechner, in Hardware und Firmware zugeschnitten auf die Lösung bestimmter Aufgaben. Kommunikationsprozessorbaugruppen ermöglichen z.B. die Kopplung mehrerer SPSen untereinander oder mit überlagerten Rechnern, wickeln also selbsttätig, d.h. parallel zu diesen, alle Aufgaben der Datenkommunikation mit den Prozeßgeräten ab. Regelungsbaugruppen übernehmen die Bearbeitung aufwendiger und schneller Regelungen. Positionierbaugruppen ermöglichen numerisch gesteuerte oder geregelte Positionierung von Antrieben an Werkzeugmaschinen. Ziel dieser Strategie ist die Verlagerung umfangreicher, häufig vorkommender und zeitkritischer Funktionen in eigenständige Einheiten zur softwaremäßigen und zeitlichen Entlastung zentraler Prozessoren. Auf diese Weise werden SPSen zu Multiprozessorgeräten. Nicht selten besitzen SPSen heute 10 bis 20 Prozessoren und erreichen damit sehr hohe Leistungen.

#### 2.7.4.4 Schaltalgebra

Nur um einen Begriff von der Booleschen Algebra zu geben, von der die meisten Studierenden sicher schon gehört haben, wollen wir wenigstens einige, für die Projektierung und Programmierung wichtige Grundzüge kennen lernen. Diese Algebra wird wohl in der Prozeßautomatisierung künftig nicht mehr die Rolle spielen, die ihr einmal zugemessen wurde.

Als der englische Mathematiker G. Boole im 19. Jahrhundert die Algebra der Logik formulierte, hatte er zunächst sicher weniger an elektrische Schaltungen gedacht. Wie sich später herausstellte, läßt sich diese allgemein gültige Darstellung der Logik binärer Variabler mit Erfolg auch auf elektrische und elektronische Binärglieder anwenden. Mit ihren Gleichungen kann man, wie in der Algebra kontinuierlich veränderlicher Variabler, Klammern bilden und auflösen. Unter Beachtung bestimmter, der Booleschen Algebra eigentümlichen Gesetzmäßigkeiten, lassen sich dadurch Schaltungen vereinfachen.

Das soll nun beispielhaft Bild 2.11 zeigen. Ein Relais X soll über vier parallele Strompfade ansprechen, wenn mindestens eine der Bedingungen A UND C, C UND A, B UND C oder D UND B erfüllt ist. In der Booleschen Algebra sieht die dieser Aufgabe entsprechende Gleichung unter Verwendung der Zeichen $\wedge$ und $\vee$ für die UND- bzw. die ODER-Verknüpfung dann so aus:

$$X = A \wedge C \vee D \wedge A \vee B \wedge C \vee D \wedge B$$

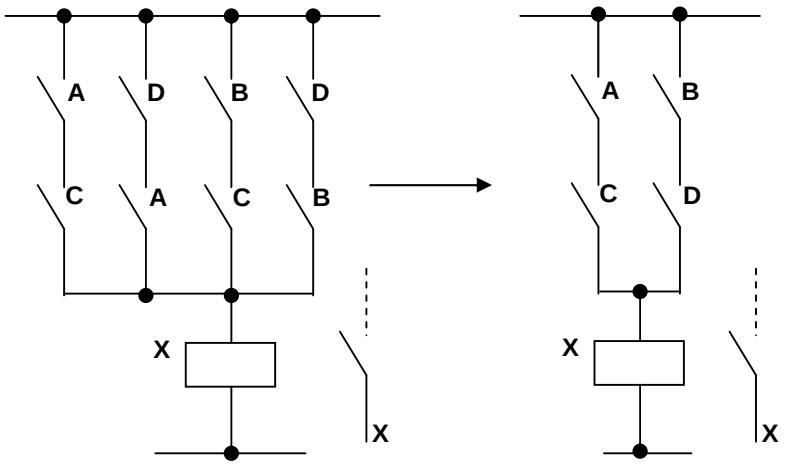



Bild 2.11: Schaltungsvereinfachung mit Hilfe Boolescher Algebra.
Durch Anwenden der Booleschen Algebra lassen sich Schaltungen oft vereinfachen. Das geschieht manchmal auf Kosten der Übersichtlichkeit.

Genau wie eine arithmetische Gleichung läßt sich obige Gleichung nun durch Bilden von Klammerausdrücken umformen:

$$X = A \wedge (C \vee D) \vee B \wedge (C \vee D) = (A \vee B) \wedge (C \vee D)$$

Aus den ursprünglichen Schaltungen der Bilder 2.11 und 2.12 entstehen durch solche Umwandlungen die jeweils rechts daneben stehenden.

Damit ist es möglich, Schaltglieder einzusparen. Das mag bei den dargestellten Kontakten oder den Anweisungen einer Anweisungsliste keine Bedeutung haben. Wenn aber die Schaltglieder teuere Stellgeräte sind, dann kann sich eine derartige Einsparung sehr lohnen, wie es Bild 2.12 als Anwendung auf die Zwei-von-Drei-Bewertung zeigt. Ein weiterer Vorteil dieser Anwendung ist, daß daraus eindeutig hervorgeht, daß eine weitere Einsparung von Stellgeräten nicht möglich ist, denn die Gleichung läßt sich nicht weiter zusammenfassen.

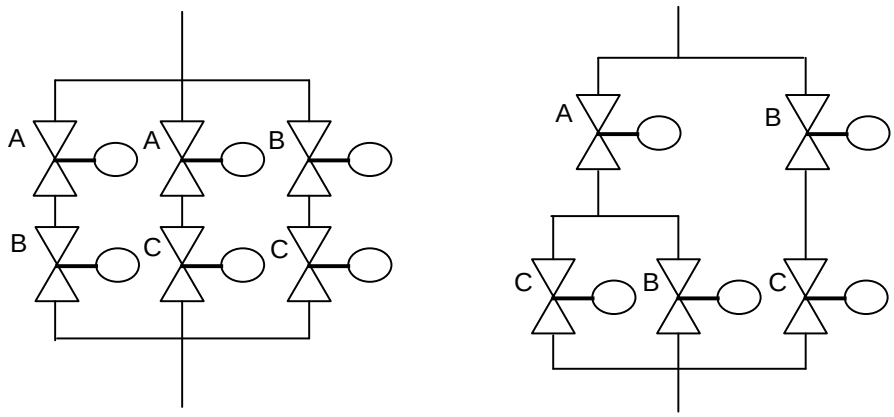



Bild 2.12: Einsparen eines Stellgeräts einer Zwei-von-Drei-Bewertung durch Anwenden der Booleschen Algebra.
Es läßt sich nicht nur ein möglicherweise sehr teueres Stellgerät einsparen, sondern auch rechnerisch nachweisen, daß eine weitere Einsparung von Stellgeräten nicht möglich ist.

Schon diese ganz einfachen Beispiele lassen erkennen, daß eine ursprüngliche Anordnung nach konsequenter Anwendung der Booleschen Algebra nur sehr schwer wiedererkannt werden kann. Die Möglichkeiten der Klammerbildungen sind mit entsprechenden Symbolen übrigens auch bei der Programmierung in Anweisungslisten möglich.

## 2.8 Programmiersprachen für SPS

Programmiersprachen und - allgemein - rechnerunterstützte Software-Entwicklungswerkzeuge für SPSen sind gegenwärtig nur in herstellerspezifischen Versionen verfügbar. Die allgemein gebräuchlichen Programmiermethoden lassen sich in zwei Hauptgruppen unterteilen, in textuelle wie Anweisungslisten, die Assembler-Sprachen ähneln, und in semigraphische, wie Funktionsplan- oder Kontaktplandarstellungen. Letztere sind eine Formalisierung elektrischer Schaltpläne für binäre Relaissteuerungen.

Die Internationale Elektrotechnische Kommission (IEC) hat eine sehr detaillierte Norm erarbeitet, die Sprachen zur Formulierung von Automatisierungsaufgaben definiert [13], [11]. Die Sprachen sind für alle Arten von SPSen geeignet. Sie überdecken einen über die klassischen SPS-Anwendungsgebiete - nämlich binäre Steuerungen - hinausgehenden Einsatzbereich bis hin zur Programmierung prozeßnaher Komponenten (PNK) verteilter Prozeßleitsysteme. Das gilt allerdings nicht für deren Bedienfunktionen und nicht für die Anwenderschnittstellen.

Diese Norm IEC61131-3, mit der wir uns vorzugsweise befassen wollen, definiert eine Familie von vier unabhängigen Sprachen, zwei textuelle und zwei graphische:

| | | | |
|---|---|---|---|
| IL | Instruction List | AWL | Anweisungsliste |
| LD | Ladder Diagramm | KOP | Kontaktplan |
| FBD | Function Block Diagramm | FUP | Funktionsplan |
| ST | Structured Text | ST | Strukturierter Text |

Die vier Sprachen sind weitgehend äquivalent und lassen sich zu großen Teilen von einer in die andere transformieren. Es ist das Ziel dieser Standardisierung, die Programmierung in Maschinen-, Assembler- oder prozeduralen Sprachen durch objektorientierte Sprachen mit graphischen Nutzerschnittstellen zu ersetzen. Obwohl das Schwergewicht der Norm auf den höheren graphischen Sprachen mit Elementen auch für die wichtigsten analogen Funktionen liegt, enthält sie noch die maschinennahe AWL und die für binäre Steuerungen geeignete Kontaktplandarstellung.

Der Grund für diesen Ansatz war, die Akzeptanz der Norm zu verbessern. Die Mehrheit der potentiellen Nutzer sind Elektroingenieure und Elektriker. Diese werden anfangs die niedrigen Programmiersprachen nutzen, welche den jetzt gängigen Sprachen sehr ähnlich sind. Kontaktpläne sind in Amerika sehr gebräuchlich bei der Auslegung von Stromversorgungskomponenten. Sie abstrahieren und formalisieren elektrische Stromlaufpläne und haben, obwohl bisher nicht genormt, allgemein von Hersteller zu Hersteller nur wenig unterschiedliche Formen. Im Gegensatz dazu sind Anweisungslisten bisher noch unterschiedlich, weil sie im Grunde nichts anderes als Assembler-Sprachen für die in SPSen eingesetzten Prozessoren sind. Um den Übergang zu Standards zu erreichen, sieht die IEC-Norm die Sprache AWL als Assembler-Sprache eines hypothetischen Prozessors mit einen Akkumulator und symbolischen Adressen vor.

### 2.8.1 Wesentliche gemeinsame Gesichtspunkte der vier IEC-Sprachen

Die vier Sprachen haben einheitliche Formen für Namen, Einzel- und Multielementvariable, Schlüsselwörter und Literale, und das sowohl für elementare als auch für abgeleitete Formen, welche die verschiedenen Formate für logische, ganzzahlige, reellwertige, Zeichenketten- und Zeitgrößen ebenso umfassen wie aufgezählte Typen, Unterbereiche und Felder. Auch die Deklaration, Spezifikation und Initialisierung von Objekten in den verschiedenen Programmorganisationseinheiten kann nur mit standardisierten Sprachkonstruktionen ausgeführt werden. Als Programmorganisationseinheiten sind Funktion, Funktionsblock und Programm vorgesehen.

Die Ausführung einer Funktion liefert genau in ein Datenelement, das mehrwertig sein kann. Funktionen enthalten keine internen Zustandsinformationen, d.h. die Ausführung einer Funktion mit den gleichen Argumenten führt immer zum gleichen Ergebnis. Funktionen und ihre Aufrufe können entweder graphisch oder textuell angegeben werden. Die Norm definiert eine große Zahl von Standardfunktionen vor, die für alle vier Sprachen einheitlich sind. Einige sind erweiterbar, d.h. die Anzahl ihrer Eingänge ist variabel. Zu dieser Gruppe gehören die logischen Funktionen UND und ODER. Die verfügbaren Standardfunktionen lassen sich in numerische Funktionen einschließlich logarithmischer und trigonometrischer Operatoren, in Typkonversions-, Bit- und Zeichenkettenmanipulations-, in Auswahl- und Vergleichsfunktionen sowie schließlich in Zeitoperationen unterteilen.

Funktionsböcke, der zweite Typ von Programmorganisationseinheiten, liefern nach ihrer Ausführung einen oder mehrere Werte. Funktionsblöcke lassen sich kopieren. Jede Instanz hat eine Identifikation und einen Datenblock mit den Werten der Ein- und Ausgänge sowie der internen Variablen. Alle Werte der Ausgangsvariablen und der internen Variablen sind von einer Ausführung eines Funktionsblockes bis zur nächsten gespeichert. Deshalb werden Aufrufe eines Funktionsblockes mit den gleichen Argumenten nicht notwendig zu gleichen Ausgangswerten führen. Es sind nämlich nach außen nur die Ein- und Ausgangswerte eines Blocks zugänglich, jedoch nicht die inneren Parameter, die z.B. internes

Speicher- oder Rückführverhalten ausdrücken. Funktionsblöcke lassen sich entweder textuell oder graphisch generieren. Analog zu den Funktionen sieht die Norm eine Anzahl von Standardfunktionsblöcken, wie bistabile Elemente, Taktdetektoren, Zähler, Zeitgeber und Datenübertragungs- und Synchronisationsoperatoren vor.

Programmeinheiten können als parallele Prozesse oder unter Kontrolle von Steuergraphen (s.u.) ablaufen. Parallelen Prozessen können zur Auflösung von Konfliktfällen beim Betriebsmittelgebrauch Prioritäten zugewiesen werden. Sie lassen sich entweder taktgesteuert oder ereignisgesteuert durch Wertänderungen logischer Variablen aktivieren.

### 2.8.2 Programmieren mit Steuerungsanweisungen

Wir wollen uns zunächst mit der für Steuerungen heute wohl noch gebräuchlichsten Art der Programmierung, der durch Auflisten einzelner Steuerungsanweisungen, beschäftigen. Diese Programmdarstellung heißt Anweisungsliste. Es handelt sich dabei um auf Steuerungsaufgaben zugeschnittene Assembler-Programmierung, die funktional über die graphischen Sprachen hinausgeht, aber die für Assembler typischen Vor- und Nachteile, wie einerseits hohe Effektivität und Flexibilität und andererseits schwerere Lesbarkeit, Fehleranfälligkeit und hohen Erstellungsaufwand, aufweist.

Anweisungslisten sind in Deutschland schon soweit Allgemeingut geworden, daß durch Fixierung von Sprachkonstruktionen in den folgenden Normen und Richtlinien eine für den Anwender untragbare Vielfalt eingeschränkt werden konnte:

- DIN19239: Steuerungstechnik, Speicherprogrammierte Steuerungen, Programmierung.
- VDI2880, Blatt 4, Entwurf 1982: Speicherprogrammierbare Steuerungsgeräte, Programmiersprachen.

Beide Werke gehen auch auf die Funktions- und Kontaktplandarstellung ein. Sie beschreiben weniger das Programmieren, sondern ordnen den zu programmierenden Funktionen mnemotechnische Abkürzungen (Anweisungsliste) oder graphische Symbole (Funktions- oder Kontaktplan) zu.

Eine Folge von Steuerungsanweisungen stellt ein Programm einer SPS dar. Steuerungsanweisungen, oder allgemein Befehle, sind charakteristisch für alle Digitalrechner. Sie bestehen aus einem Operationsteil und einem Operandenteil. Dabei gibt die Operation an, was zu tun ist, und der Operand, womit es zu tun ist. Wie sich Operations- und Operandenteil weiter untergliedern, zeigt Bild 2.13. Bei den Operationen kann zwischen solchen zur Signalverarbeitung, wie logischen Verknüpfungen, Zählen, Addieren oder Vergleichen, aber auch Setzen und Rücksetzen und Operationen zur Programmorganisation wie Laden, bedingte und unbedingte Sprünge oder Bausteinaufrufe unterschieden werden. Bild 2.14 zeigt einige Operationen aus der an DIN19239 angelehnten Programmiersprache STEP 5 von Siemens.

Die Operanden teilen sich auf in

- Operandenkennzeichen, wie Eingänge (E), Ausgänge (A), Konstante (K), Zähler (Z), Programmbausteine (PB), mit einer ergänzenden Angabe, wie z.B. Byte, Wort, Impuls, Ein- oder Ausschaltverzögerung (bei Zeitfunktionen), sowie in
- Parameter, wie beispielsweise die Adressen (Nummern) der Eingänge, Ausgänge, Merker oder Zähler.

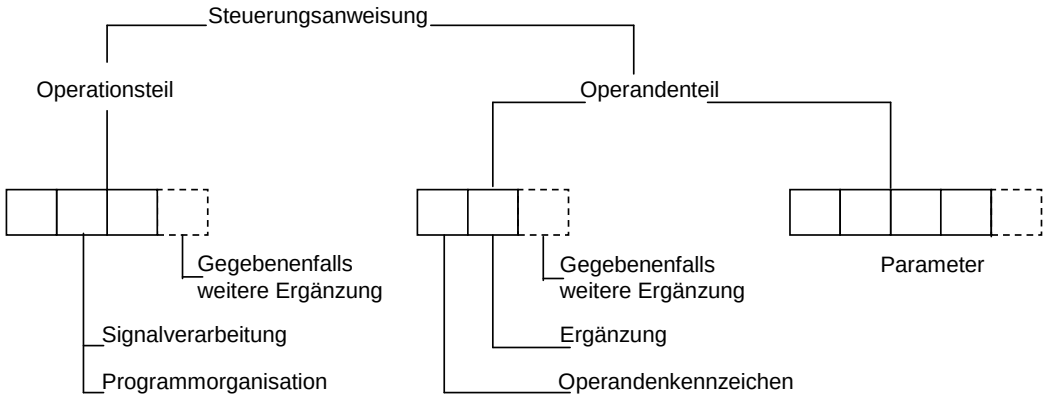



Bild 2.13: Aufbau einer Steuerungsanweisung.
In DIN19239 ist festgelegt, welche Bedeutung den Stellen einer Steuerungsanweisung zuzuordnen ist.

Operationen und Operandenkennzeichen sind in Anweisungslisten mnemotechnisch abgekürzt: U, O, N und S stehen für UND, ODER, NICHT und Setzen; ZV, ADD und MUL stehen für Zählen vorwärts, Addieren und Multiplizieren; SP, SPB und BAB stehen für Sprung unbedingt, Sprung bedingt und Bausteinaufruf.

Mit diesen Abkürzungen werden die Steuerungsanweisungen aufgelistet, die der Prozessor eines Steuergerätes schrittweise abarbeitet, wie auch aus den Bildern zu ersehen war. Besonders bei komplizierten und umfangreichen Aufgaben ist eine solche lineare Programmierung (Bild 2.15a) nicht zu empfehlen, da umfangreiche Programme nicht nur wegen der Vielzahl programmierter Funktionen, sondern vor allem durch zahlreiche Sprünge sehr unübersichtlich werden und so letztendlich nur noch vom „geistigen Vater“ eines solchen Programms zu verstehen und zu betreuen sind. Deshalb ist eine strukturierte Programmierung zu bevorzugen, wie sie als Beispiel Bild 2.15b zeigt. Das Gesamtprogramm wird unterteilt in Programmbausteine (PB), deren Bearbeitungsfolge in einem Organisationsbaustein (hier OB 1) durch Eintragen von Bausteinaufrufen festgelegt wird. Ordnet man bestimmte, gut abgrenzbare Funktionsbereiche einer zu automatisierenden Anlage Programmbausteinen zu, so wird das Programm ein transparentes Abbild der Technologie der Anlage. Solche Programmbausteine können nicht nur von einem, sondern auch von mehreren Organisationsbausteinen her angesprochen, d.h. aufgerufen und für die Bearbeitung freigegeben oder blockiert werden. Auf diese Weise lassen sich Struktur und Ablauf eines Programms transparent gestalten.

Wie schon erwähnt, sieht auch die IEC-Norm eine herstellerneutrale Programmierung in der Sprache Instruction List (IL) vor.

### 2.8.3 Höhere Programmiersprachen

Die Norm IEC61131-3 definiert eine zweite textuelle Sprache, Structured Text (ST), die eine der Programmiersprache Pascal ähnliche Syntax und Funktionalität hat. Sie bietet mathematische Ausdrücke einschließlich Funktionsaufrufen und folgende Anweisungstypen: Zuweisungen, Funktions- und -blockaufrufe, vorzeitiges Verlassen von Funktionen, Funktionsblöcken und Iterationen, Bedingungs- und Fallauswahl sowie bedingte und iterative Schleifen.

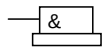
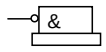
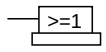
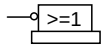
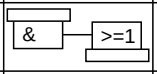
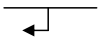
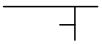
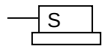
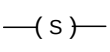
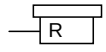
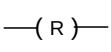
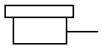
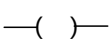
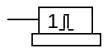
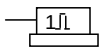
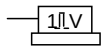
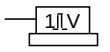
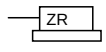
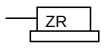
| **Operation** | | | **Beschreibung** | |
|---|---|---|---|---|
| Darstellung als | | | | |
| Anweisungsliste | Funktionsplan | Kontaktplan | | |
| AWL | FUP | KOP | AWL FUP | KOP |
| U |  |  | Abfrage auf Signalzustand „1“ und Verknüpfung nach UND | Reihenschaltung von „Schließern“ |
| UN |  |  | Abfrage auf Signalzustand „0“ und Verknüpfung nach UND | Reihenschaltung von „Öffnern“ |
| O |  |  | Abfrage auf Signalzustand „1“ und Verknüpfung nach ODER | Parallelschaltung von „Schließern“ |
| ON |  |  | Abfrage auf Signalzustand „0“ und Verknüpfung nach ODER | Parallelschaltung von „Öffnern“ |
| O |  |  | ODER-Verknüpfung von UND-Funktionen | Rückführung zur Parallelschaltung von Strompfaden |
| U( | |  | UND-Verknüpfung von Klammerausdrücken (1 Klammerebene) | Öffnen eines Abzweiges (1 Klammerebene) |
| ) | |  | Klammer zu (Abschluß eines Klammerausdruckes) | Schließen eines Abzweiges |
| S |  |  | Setzen bei Verknüpfungsergebnis „1“, bei „0“ keine Wirkung | |
| R |  |  | Rücksetzen bei Verknüpfungsergebnis „1“, bei „0“ keine Wirkung | |
| - |  |  | Zuweisen Signal „1“ bei Verknüpfungsergebnis „1“, Zuweisen Signal „0“ bei Verknüpfungsergebnis „0“ | |
| SI |  |  | Starten einer Zeile als Impuls (Signalbegrenzung) | |
| SV |  |  | Starten einer Zeit als verlängerter Impuls (Signalbegrenzung und -verlängerung | |
| ZR |  |  | Zählen rückwärts eines Zählers um 1 bei positivem Signalwechsel des Verknüpfungsergebnisses | |
| L | | | Laden des Operandenwertes in den Akku | |
| T | | | Transferieren des Akkuinhalts zu dem Operanden | |
| **BE** | **BE** | **BE** | Programmende; Sprung zum Programmanfang | |
| **BEB** | **BEB** | **BEB** | Programmende bedingt; bei Verknüpfungsergebnis „1“ Sprung zum Programmanfang, bei „0“ keine Wirkung. Beim Aufbau eines Programms ist zu berücksichtigen, dass BEB nicht wie BE die Zyklusüberwachung steuert. | |

Bild 2.14: Auflistung der Operationsteile wichtiger Steuerungsanweisungen der Sprache STEP 5 von Siemens.

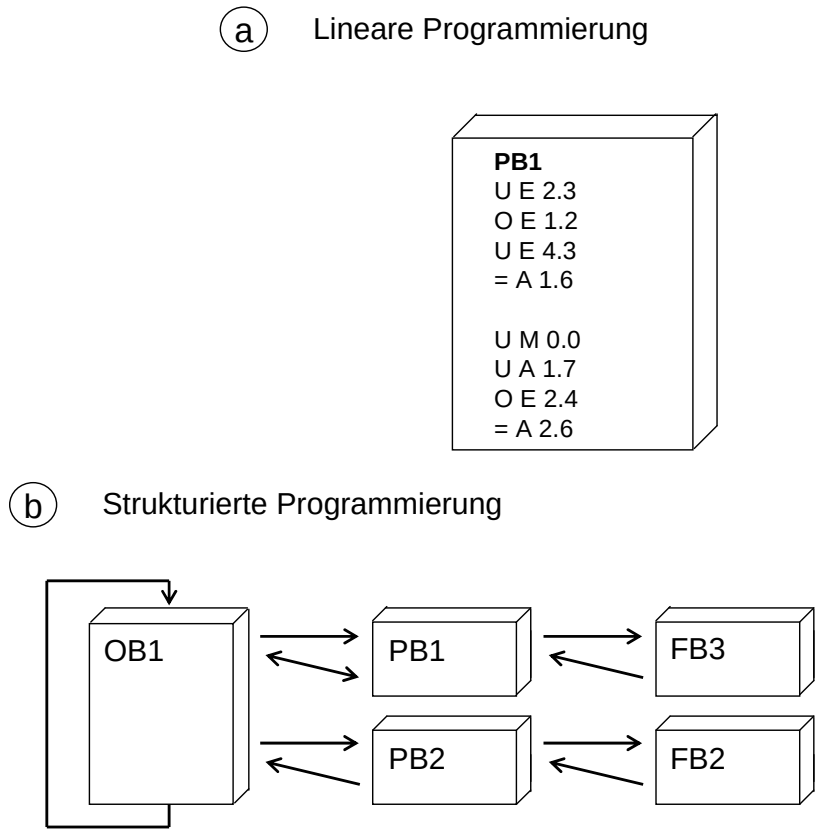



Bild 2.15: Lineare und strukturierte Programmierung (Siemens).
Lineare Programmierung (a) erreicht bei steigenden Programmlängen schnell die Grenzen der Übersichtlichkeit. Deshalb ist strukturierte Programmierung (b) mit modularem Programmaufbau zu empfehlen: OB Organisationsbausteine zur Verwaltung des Anwendungsprogramms bei zyklischer, alarm- oder zeitgesteuerter Programmbearbeitung, PB Programmbausteine zur Strukturierung der technologischen Steuerungsaufgabe, FB Funktionsbausteine für häufig wiederkehrende Funktionen.

Nach [5] gibt es interpretativ arbeitende SPS-BASIC-Versionen, die in Ansätzen Kontrollstrukturen haben, wie sie von höheren Programmiersprachen her bekannt sind, Bausteinarchitekturen und lokale Unterprogramme unterstützen.

### 2.8.4 Kontaktplandarstellung

Angelehnt an die in der Elektrotechnik üblichen Stromlaufpläne erlaubt die Kontaktplandarstellung die graphische Formulierung von Steuerungsprogrammen. Hauptelemente von Kontaktplänen sind Verbindungen, statische und transiente direkte und negierende Kontakte sowie direktbeaufschlagte, verriegelte, gehaltene und transiente Ausgänge oder Merker, die als Spulen symbolisiert werden. Diese Symbole liegen wie Sprossen in einem Verdrahtungsplan, rechts und links mit zwei Spannungsschienen verbunden. (Im Gegensatz dazu liegen bei Stromlaufplänen die Spannungsschienen oben und unten.) Die Programmierung erfolgt über Bildschirme durch aufgabengemäße Anordnung von Elementen und deren Vernetzung, also durch Hintereinander- oder Parallelschaltung von Kontakt- und Spulensymbolen. Ein KOP befähigt eine SPS, Daten mittels standardisierter graphischer Symbole zu testen und zu modifizieren.

### 2.8.5 Funktionsplandarstellung

Während AWL Anwender anspricht, die gelernt hatten, listenförmigen Programmdarstellungen zu lesen, und die Kontaktplantechnik jene, die in der Relais- und Schütztechnik zu Hause sind, ist die Funktionsplandarstellung (Bild 2.10b) sowohl für Ablauf- als auch für Verknüpfungssteuerungen geeignet. Außerdem, und das ist besonders wichtig, unterscheidet sie sich als geräteneutrale Darstellung nur wenig von dem zur Verständigung zwischen Betreibern, Verfahrens- und Sicherheitsingenieuren sowie Automatisierungsfachleuten üblichen Sprachgebrauch. Damit kann die Funktionsplandarstellung mit nur wenigen ergänzenden Erläuterungen von allen Projektbeteiligten und - wenn erforderlich - auch von Genehmigungsbehörden verstanden werden. Ist also eine derartige Darstellung das Ergebnis einer Projektierung, dann kann die Übertragung der erarbeiteten Lösung in Maschinencode sowie das Erstellen einer Programmdokumentation mit Hilfe von Programmiergeräten in einfacher Weise und (weitgehend) frei von Interpretationsfehlern erledigt werden.

Mit Funktionsplänen können Automatisierungsaufgaben unabhängig von ihrer gerätetechnischen Implementation prozeßorientiert dargestellt werden. Der Ansatz ist vom Entwurf digitaler Schaltkreise abgeleitet, bei dem jeder Chip einen bestimmten Teil einer Gesamtschaltung repräsentiert. Die direkte Umsetzung dieses Konzeptes führt zu Funktionen, die in Form von Rechtecken dargestellt und mit Ein- und Ausgängen beliebigen Datentyps versehen in Schaltungen integriert werden. Die schematische zeichnerische Repräsentation logischer und funktioneller Abhängigkeiten bietet zwar leichtere Allgemeinverständlichkeit als alphanumerische Darstellungen, die Übersichtlichkeit wird aber mit steigender Komplexität eingeschränkt. Funktionsplandarstellungen sind auch Gegenstand der deutschen Norm DIN40719, Teil 6.

Systemunabhängige Programmerstellung für SPSen wird in zwei Schritten durchgeführt:

1. Aufstellen einer Bibliothek von Funktionen und Funktionsblöcken,
2. Verbinden ausgewählter Funktionen und Funktionsblöcke.

Der zweite Schritt entspricht der Aufstellung eines Funktionsplans zur Lösung einer Automatisierungsaufgabe. Dazu muß der Anwender geeignete Funktionen und Funktionsblöcke

aus seiner Bibliothek auswählen, sie plazieren und aufgabengerecht miteinander verbinden. Dies geschieht i.a. in einfacher Weise mit Hilfe eines rechnergestützten CAD- Werkzeuges.

Eine solche Bibliothek besteht aus Standardfunktionen, die für Automatisierungsaufgaben universell anwendbar sind und die der Systemlieferant vorhält, sowie aus projektspezifischen Komponenten, die der Anwender selbst generieren muß. Letzteres kann entweder durch Definieren einer neuen Funktion geschehen oder durch Modifizieren einer schon vorhandenen. Wenn eine Bibliothek eine gewisse Größe erreicht hat, wird die zweite Möglichkeit die häufigere sein.

### 2.8.6 Steuergraphen

Die Funktionsplandarstellung wird in der IEC-Norm durch eine andere graphische Sprache ergänzt, Steuergraphen oder sequentielle Ablaufpläne (Sequential Function Charts, SFC), die als industrielle Anwendung von Petri-Netzen betrachtet werden kann und zur Formulierung der Koordination und Kooperation asynchroner Ablaufsteuerungen genutzt werden.

Mit Elementen der SFC-Sprache können die Programmorganisationseinheiten Programm und Funktionsblock in Gruppen von Schritten und Transitionen (Fortschaltbedingungen) eingeteilt werden, die mit gerichteten Verbindungen verknüpft werden. Jeder Schritt besteht aus einzelnen Aktionen und zu jeder Transition gehört eine Transitionsbedingung. Ein Schritt ist entweder aktiv oder inaktiv. Der Zustand der Programmorganisationseinheiten ist definiert durch die aktiven Schritte und die Werte ihrer internen und Ausgangsgrößen. Jeder Schritt hat zusätzliche Ausgänge, einen logischen, der angibt, ob der Schritt aktiv ist oder nicht, und einen weiteren zur Zeitüberwachung. Die Ausführung einer Programmorganisationseinheit beginnt mit dem als solchen ausgezeichneten Initialschritt. Er bestimmt die Anfangswerte der internen und Ausgangsgrößen und die Gruppe der im Anfangszustand aktiven Schritte.

Jeder Schritt besteht aus einer Anzahl von Aktionen. Ein Schritt ohne Aktion hat die Aufgabe zu warten, bis eine nachfolgende Transitionsbedingung erfüllt ist. Eine Aktion kann eine logische Variable, ein in einer der vier Programmiersprachen abgefaßter Programmteil oder wiederum ein Steuergraph sein. Schließlich kann eine Aktion eine logische Rückführgröße setzen, die angibt, ob die Aktion durchgeführt ist. Verbunden mit jeder Aktion ist eine logische Variable, deren Wert von der Überwachungszeit abhängt. Eine Aktion wird solange durchgeführt wie der Wert dieser Variablen „wahr“ ist. Zu jeder Aktion gehört eine Angabe über ihre Art, z.B. ob sie gespeichert ist oder nicht, ob sie verzögert oder zeitlich begrenzt ist.

Eine Transition stellt die Bedingung dar, bei der von einem oder mehreren Schritten (Alternativzusammenführung oder Synchronisation, Bild 2.16) zu einem oder mehreren folgenden, durch den Programmablauf vorgegebenen Schritten (Alternativverzweigung oder Simultanverzweigung, Bild 2.16) übergegangen wird. Jede Transition hat eine zugeordnete Transitionsbedingung in Form einer Booleschen Verknüpfung. Eine Transition wird aktiviert, wenn alle im Ablaufplan direkt vorhergehenden Schritte aktiv sind. Die Transition ist erfüllt, wenn sie aktiviert ist und die zugeordneten Transitionsbedingungen gesetzt sind. Die Erfüllung einer Transition führt zur Aktivierung aller im Ablaufplan unmittelbar folgenden Schritte und zur Deaktivierung aller der Transition unmittelbar vorgelagerten Schritte.

Auf einen Schritt muß immer eine Transition folgen und auf eine Transition ein Schritt – es dürfen also keine Schritte aufeinander folgen.

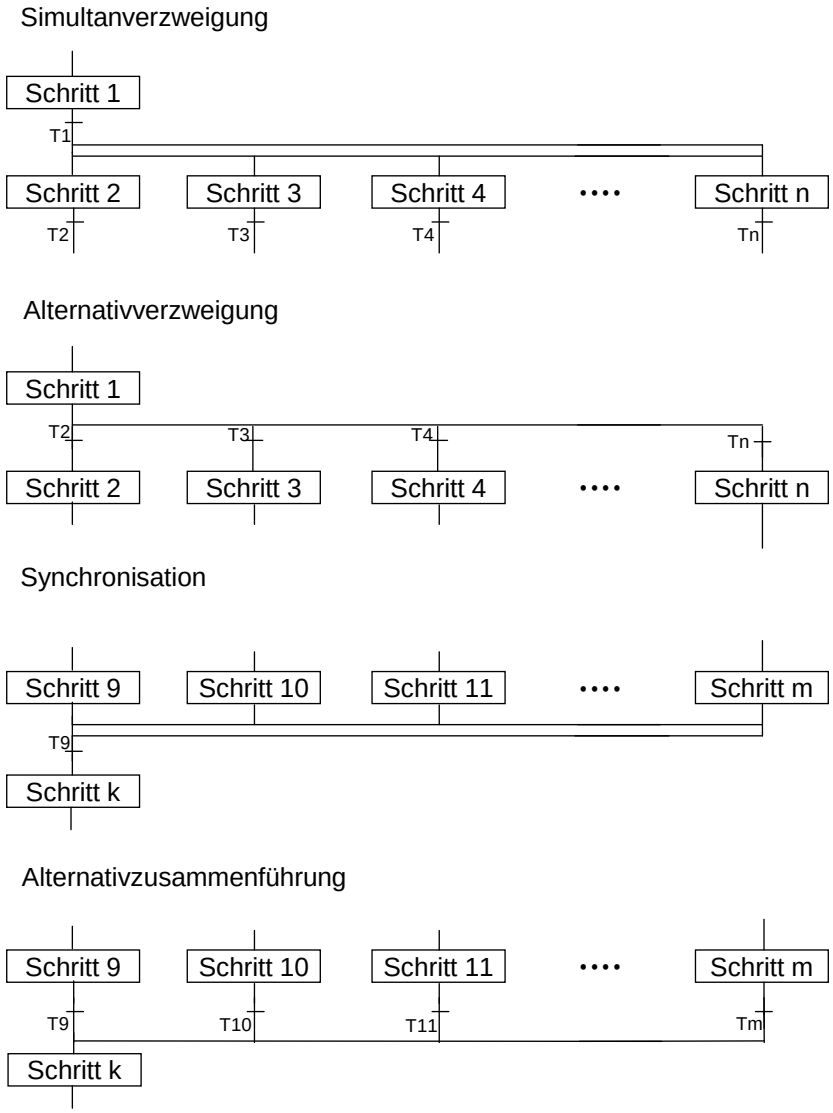



Bild 2.16: Sprachelemente zur Verzweigung und Zusammenführung von Richtungsgraphen.
Bei einer Simultanverzweigung (oder UND-Verzweigung) aktiviert die Steuerung nach Erfüllung der Transition T1 alle darauffolgenden Schritte 2 bis n. Bei einer Alternativverzweigung (oder ODER-Verzweigung) wird nur ein Zweig durchlaufen, und zwar der, dessen Transitionsbedingung (zuerst) erfüllt ist. Sind mehrere Transitionsbedingungen gleichzeitig erfüllt, so aktiviert die Steuerung davon den am weitesten links stehenden Schritt. Bei der Synchronisation wartet die Steuerung bis alle parallelen Schritte abgearbeitet sind, bei der Alternativzusammenführung auf einen der vorhergehenden Schritte.

Die Methode der Steuergraphen erlaubt die Beschreibung von Ablaufsteuerungen auf zwei Abstraktionsebenen:

- der Ablauflogik in der Übersichtsebene in Form von SFCs und
- der Details von Aktionen und Weiterschaltbedingungen in der Lupenebene (oder Detailebene) ausgedrückt in den Sprachen AWL, KOP oder FUP.

In der Übersichtsebene wird beschrieben, wie Schritte aufeinanderfolgen, welche alternativ und welche simultan durchlaufen werden. Außerdem ist zu ersehen, welche Bedingungen (Transitionen) vor dem Beginn eines Schrittes erfüllt sein müssen. Nicht zu sehen ist, welche Aktionen in einem Schritt tatsächlich auszuführen sind, z.B. welcher Ausgang auf „1“ geschaltet werden soll und von welchen Eingängen, Merkern oder Zählern die Weiterschaltbedingungen abhängen. Um dies zu erkennen, muß man die Lupenebene betrachten, die es zu jedem Schritt und zu jeder Transition gibt.

Der Vorteil dieser Strukturierung kommt besonders bei der Programmerstellung, der Visualisierung und der Diagnose zur Geltung. Der Anlagenfahrer kann aus der Übersichtsebene ersehen, welcher Schritt gestört ist, und muß sich nicht mit Details des Automatisierungsprogramms befassen, wozu er in der Regel auch nicht ausgebildet ist. Die Trennung der Ebenen hat auch beim Programmentwurf Vorteile. Die Übersichtsebene kann zunächst entworfen werden, ohne die für die Lupenebene erforderlichen Details kennen zu müssen. Das kann dann nach und nach für jeden Schritt nachgeholt werden. Sehr komplexe Steuerungsaufgaben können auch in mehreren Detailebenen strukturiert werden.

Aus dem anfangs dargestellten Bild 2.1, der Funktionsplandarstellung der steuerungstechnischen Bearbeitung der Grundoperation Dosieren nach DIN40719, Teil 6, sind, obwohl die Darstellung in dieser Form zunächst nicht zur Programmierung geeignet ist, Strukturen in Form von Richtungsgraphen mit Schritten, Aktionen und Transitionen sowie mit Verzweigungen und Zusammenführungen zu erkennen. Aus diesem realistischen Beispiel läßt sich auch erkennen, wie viel Detailarbeit der Entwurf einer Steuerung erfordern kann, insbesondere, wenn man bedenkt, daß manche Schritte und Transitionen erst nach längeren Diskussionen mit den verschiedenen, an der Projektbearbeitung beteiligten Fachleuten festgelegt werden können.

### 2.8.7 CAE-Werkzeuge für die SPS-Software-Entwicklung

Es wurden an vielen Stellen Werkzeuge entwickelt, die systemunabhängige graphische Programmierung von SPSen zu unterstützen. Sie basieren auf den textuellen und graphischen Hochsprachen, die in der IEC-Norm definiert sind. Da der Grad der in den graphischen Sprachen FBD/SFC möglichen Detaillierung limitiert ist, wird die Sprache ST für die Formulierung projektspezifischer Software-Module in Form von Funktionsblöcken genutzt, die alle Implementierungsdetails enthalten. Diese Module werden dann in den graphischen Sprachen FBD /SFC zur Lösung von Automatisierungs- und Steuerungsaufgaben ausgewählt und verknüpft. So werden die Vorteile graphischer Sprachen, nämlich Orientierung an der Denkweise des Ingenieurs, Klarheit, leichte Verständlichkeit, übersichtliche Dokumentation, kombiniert mit den Vorteilen textuellen Programmierens, und zwar uneingeschränkte Ausdrucksmöglichkeit syntaktischer Details, von Steuerungsstrukturen, Regelalgorithmen und des Zeitverhalten.

Im folgenden wollen wir uns einen Überblick über solche Werkzeuge verschaffen. Sie sollen einen methodischen und strukturierten top-down-Entwurf, in der Planungsphase das Automatisierungskonzept sowie die Erstellung von Bibliotheken gut ausgetesteter Standard-

module für Einzelprojekte oder Projektklassen unterstützen. Gleichzeitig repräsentieren sie sehr leistungsfähige Werkzeuge zur schnellen Prototyperstellung.

Zur Entwicklung einer neuen Software kann der Programmierer ein CAD- Werkzeug zum Erstellen seiner Zeichnungen von Funktionsblöcken und Richtungsgraphen nutzen. Im einzelnen sucht er sich geeignete graphische Elemente aus seiner Bibliothek, plaziert sie in seinem Entwurf und verbindet sie gemäß des geforderten logischen Ablaufs. Anschließend wird der Entwurf gespeichert und mit einem Hilfsprogramm des CAD-Systems in Listenform überführt. Diese Netzlisten stellen eine dem Logikplan voll äquivalente Form des Programms dar. Sie werden dann von einem zweiten Postprozessor, einem Übersetzer, in die Sprache ST überführt. Der Übersetzer erstellt komplette Programmorganisationseinheiten mit der Definition von Eingangs-, Ausgangs- und lokalen Variablen, von parallelen Prozessen, von instantiierten Funktionsblöcken, mit dem Aufrufen von Ablaufsequenzen und mit der Beschreibung von Schritten, Transitionen und Aktionen. Die Quelltexte dieser Programmeinheiten werden dann in einer Bibliothek gespeichert, die natürlich auch mit einem Texteditor mit handcodierten Modulen gefüllt werden kann. Sie dienen als Eingabe für den letzten Postprozessor, einen Binder, der ablauffähige Programme durch Einfügen der Quellentexte aller Funktionen und Funktionsblöcke aus der Bibliothek, die in Modulen aufgerufen werden sowie aller Programmeinheiten, die vom Anwender als Hauptprogramme spezifiziert werden, erzeugt. Solche Programme sind lauffähig in dem Sinne, daß sie ein Steuerungsproblem vollständig lösen. Sie müssen dann natürlich noch in den Maschinencode einer speziellen SPS übersetzt werden.

## 2.9 Maßnahmen gegen Funktionsstörungen in Steuerungen

Auch bei gewissenhafter Erarbeitung von Aufgabenstellungen, auch beim Einsatz Programmierfehler weitgehend vermeidender höherer Programmiersprachen und auch bei sorgfältiger Auswahl von Hardware-Komponenten lassen sich Fehler in Steuerungen nicht ganz ausschließen. Als Steuerung wird hier ein Gesamtsystem betrachtet. Besonders muß die Eingangs- und Ausgangsperipherie, einschließlich der Signalgeber und Stellgeräte, einbezogen werden, denn erfahrungsgemäß treten nur ungefähr 5% der Fehler und Ausfälle im Steuerungsgerät selbst, der Rest aber in der Peripherie auf.

Die Hauptforderungen, die an Prozeßleiteinrichtungen, und damit auch an Steuerungen gestellt werden, sind:

**Zuverlässigkeit:** Eine Steuerung soll möglichst lange ungestört laufen, einen hohen mittleren Ausfallabstand (MTBF), also eine große mittlere fehlerfreie Betriebszeit haben.

**Verfügbarkeit:** Eine Steuerung soll in einem möglichst hohen Prozentsatz der Betriebszeit einwandfrei arbeiten.

**Sicherheit:** Fehler in einer Steuerung sollen, wenn nicht ganz vermeidbar, den Prozeß in einen sicheren Betriebszustand führen.

### 2.9.1 Zuverlässigkeit von SPSen

Steuerungen greifen häufig in größere Betriebsbereiche ein, deren Ausfälle bereichsübergreifend und damit schwerwiegender als die Ausfälle einzelner Regelkreise sind. Da hilft es meistens auch nicht, eine defekte Steuerung sofort wieder instandsetzen zu können. Der betroffene Betrieb muß bei Ausfall einer Steuerung abgestellt werden und läßt sich oft nur mit umständlichen Maßnahmen wieder anfahren.