SPS-Technik - KOMPAKT

1. Aufbau und Arbeitsweise
1.1 Einleitung

Steuergeräte können als VPS (verbindungsprogrammierte Steuerungen) oder **SPS** (**speicherprogrammierbare Steuerungen**) ausgeführt werden. Die SPS ersetzt Hilfsschütze und Relais einer VPS durch ein von einem Mikrorechner ausgeführtes **Steuerprogramm**. Dieses kann mit Hilfe eines Computers, eines speziellen Programmiergerätes oder einem programmierbaren Speicherbaustein, der in die SPS gesteckt wird eingegeben werden.

Wie eine VPS arbeitet auch eine SPS nach dem **EVA-Prinzip**:

Eingabeebene	**V**erarbeitungsebene	**A**usgabeebene
- Sensoren (Geber) - Bedienelemente	- SPS	- Aktoren - Anzeigeelemente

Die Vorteile einer SPS liegen im Wesentlichen in ihrer kompakten Bauweise und ihrer guten Wartbarkeit (Fehlersuche und -korrektur, Anpassung an neue Erfordernisse). Des weiteren lassen sich analoge Größen erfassen und verarbeiten.

1.2 Baugruppen der SPS

Speicherprogrammierbare Steuerungen werden als **Kompaktgerät** oder als **modulares Stecksystem** geliefert. Beim Stecksystem wird die SPS aus einzelnen Baugruppen je nach Anforderung der zu realisierenden Steuerung zusammengesteckt. Betrachtet werden hier beispielhaft einige Baugruppen der Simatic S7-300 von Siemens:

Netzteil PS (Power Supply): Liefert als **Versorgungsspannung** für alle Baugruppen der SPS und gegebenenfalls für die Sensoren eine stabilisierte Gleichspannung von 24 Volt.

Zentralbaugruppe CPU (Central Processing Unit): Enthält den **Arbeitsspeicher** und den eigentlichen **Prozessor**. Dieser führt das im Arbeitsspeicher befindliche Steuerprogramm aus. Die CPU bestimmt im Wesentlichen die Leistungsfähigkeit der SPS.

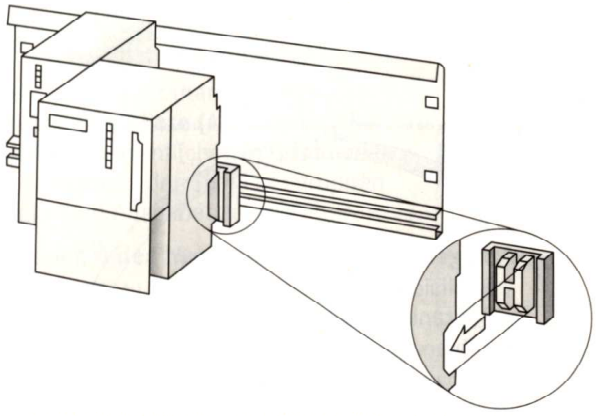
Eingabebaugruppe DI (Digital Input) bzw. AI (Analog Input): Stellt **digitale** / **analoge Eingänge** für binäre / analoge Sensoren (Geber) zur Verfügung. Bei den digitalen Eingängen findet eine **Signalanpassung**, bei analogen Eingängen eine **AD-Wandlung** statt.

Ausgabebaugruppe DO (Digital Output) bzw. AO (Analog Output): Stellt **digitale** / **analoge Ausgänge** für binäre / analoge Aktoren zur Verfügung.

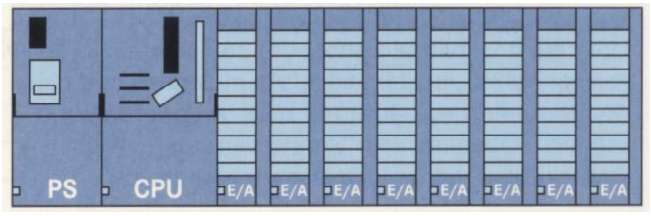
Kombinierte Ein- Ausgabebaugruppe DIO (Digital Input Output) bzw. AIO (Analog Input Output)

1.3 Aufbaurichtlinien

Beim modularen Aufbau werden die Baugruppen im Elektronikschrank auf einer Profilschiene befestigt.



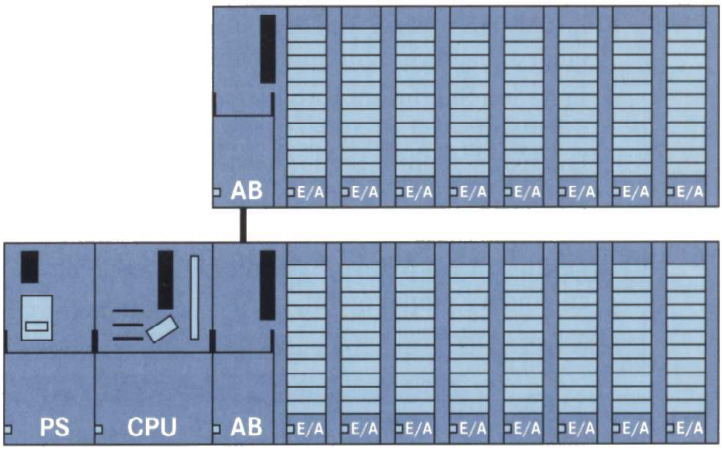
Steckplatz 1: Netzteil PS
Steckplatz 2: Zentralbaugruppe CPU
Steckplatz 3 - 10: Alle sonstigen Baugruppen (bspw. IO)



einzeiliger Aufbau

Die Verbindung zwischen den einzelnen Baugruppen erfolgt auf der Rückseite der Baugruppen durch Aufstecken von **Busverbindern**. Die Energieversorgung und die Datenübertragung erfolgen über den so entstehenden **Rückwandbus** (**back-plane bus**).

Beim einzeiligen Aufbau können rechts neben der CPU nur maximal acht Baugruppen aufgesteckt werden. Die Anzahl der Ein- und Ausgänge kann durch Verwendung von **Anschaltbaugruppen** AB und dem damit verbundenen mehrzeiligen Aufbau erweitert werden.

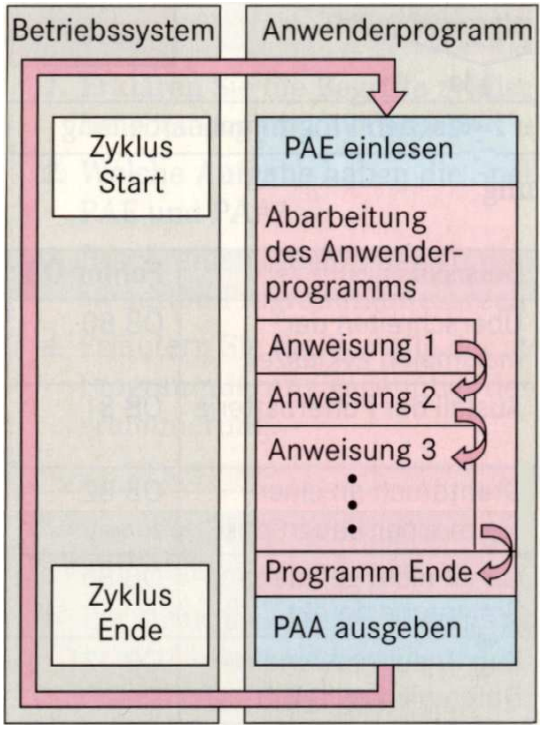


zweizeiliger Aufbau

2. Grundlagen der SPS-Programmierung
2.1 Programmabarbeitung durch die SPS

Die einzelnen Anweisungen des Steuerprogramms werden in einer **zyklischen Abarbeitung** immer wieder neu durchlaufen. Der Ablauf lässt sich wie rechts abgebildet darstellen.

Nach Zyklusstart werden zunächst alle Eingangszustände der Anlage abgefragt und im **Prozessabbild der Eingänge (PAE)** gespeichert. Die CPU beginnt danach mit der **seriellen** (= schrittweisen) Abarbeitung des Steuerprogramms. Ausgaberelevante Bearbeitungsergebnisse werden im **Prozessabbild der Ausgänge (PAA)** gespeichert. Änderungen der Eingangssignale während der Programmabarbeitung bleiben unberücksichtigt. Nach Programm-Ende werden die im PAA gespeicherten Signalzustände über den Rückwandbus an die Ausgabebaugruppen und von dort an die Aktoren der Anlage übergeben.



Die für das Einlesen der Eingänge, das Abarbeiten des Programms und das Ausgeben der Ausgänge von eines SPS benötigte Zeit ist die **Zykluszeit**. Sie beträgt im Regelfalle wenige Millisekunden.

Die Zykluszeit bestimmt die **Reaktionszeit** der SPS auf Eingangsänderungen. Sie ist mindestens so groß wie die Zykluszeit. Im Extremfall (wenn sich ein Eingang direkt nach dem Einlesen ändert) kann sie die doppelte Zykluszeit betragen: $t_{Zyklus} < t_{Reaktion} < 2 \cdot t_{Zyklus}$

Über die Programmiersoftware (**TIA-Portal** o.a.) lässt sich die **maximale Zykluszeit** einstellen. Wird diese während der Programmausführung überschritten, betrachtet die SPS dies als Fehlfunktion. Das Programm wird unmittelbar unterbrochen und ein entsprechendes - vom SPS-Programmierer entwickeltes und in einem sogenannten **Fehler-Organisationsbaustein** (Fehler-OB) abgelegtes - **Fehlerbehandlungsprogramm** gestartet. Man spricht von einer **ereignisorientierten Programmunterbrechung**.

Für unterschiedliche mögliche Systemstörungen innerhalb der SPS können entsprechende Fehlerbehandlungsprogramme in eigene Fehler-OBs abgelegt werden. Einige Beispiele sind rechts aufgeführt.

Fehlerart	Beispiel	Fehler-OB
Zeitfehler	Überschreiten der maximalen Zykluszeit	OB 80
Stromversorgungsfehler	Ausfall der Pufferbatterie	OB 81
Diagnosealarm	Drahtbruch an einer Baugruppe	OB 82
Ziehen/ Stecken-Alarm	Entfernen oder Hinzufügen einer Baugruppe	OB 83
Programmablauffehler	OB ist nicht geladen, Baugruppe defekt	OB 85
Programmierfehler	Zugriff auf notwendige Daten nicht möglich	OB 121

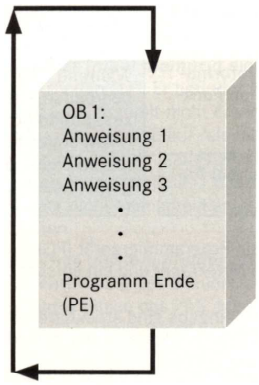
Lässt sich durch die Abarbeitung des Fehler-OBs der Fehler beheben, wird die zyklische Programmabarbeitung an der Stelle, an der das Steuerprogramm unterbrochen wurde, fortgesetzt. Lässt sich der Fehler nicht beheben oder wurde der entsprechende Fehler-OB nicht programmiert, schaltet die SPS in den Betriebszustand STOP.

2.2 Strukturierung des Steuerprogramms

Eine übersichtliche Programmiertechnik ermöglicht eine einfachere Programmwartung. Bei der SPS-Programmierung wird zwischen **linearer** und **strukturierter Programmierung** unterschieden.

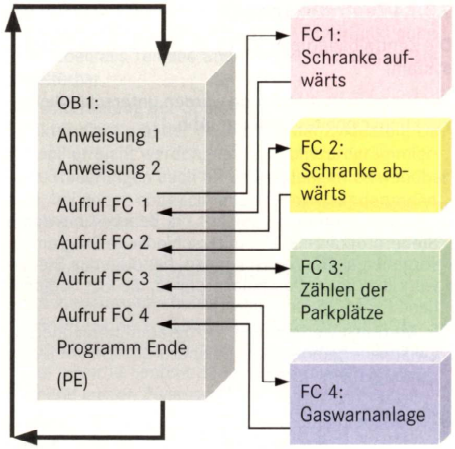
Lineare Programmierung:

- Gesamtes Steuerprogramm befindet sich im Organisationsbaustein OB1
- Alle Anweisungen werden vom Prozessor zyklisch abgearbeitet
- Geeignet für einfache, unkomplexe Steuerungsaufgaben
  (komplexe Steuerungsaufgaben werden schnell zu unübersichtlich)



Strukturierte Programmierung:

- Unterteilung des Gesamtprogramms in einzelne technologisch zusammenhängende Unterprogramme
- Ablegen der Unterprogramme in Codebausteinen (FC)
- Bedingter Aufruf über Sprungbefehle (bedingt = an Bedingung geknüpft)



Vorteile der strukturierten Programmierung:

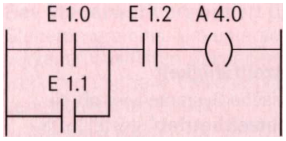
- Ermöglichung der parallelen Programmentwicklung durch mehrere Programmierer
- Verkürzung der Zykluszeit durch Nichtbeachtung eines Unterprogramms bei Nichteintreten der Aufrufbedingung
- Mehrfachnutzung einzelner Programmteile an unterschiedlichen Stellen des Hauptprogramms möglich
- Vereinfachung der Programmwartung (Fehlerbehebung oder Anpassung nur einmal im Unterprogramm)

2.3 Programm-Darstellungsarten

Die Erstellung eines Steuerprogramms kann in drei verschiedenen **Darstellungsarten** (= Programmiersprachen) erfolgen. In allen dreien lassen sich prinzipiell miteinander gleichrangige Programme erstellen:

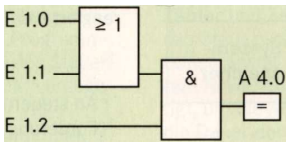
Kontaktplan KOP:

ähnliche Form wie Stromlaufplan



Funktionsplan FUP:

Verwendung logischer Verknüpfungssymbole



Anweisungsliste AWL:

Textuelle statt graphischer Programmierung

```
U (
O E 1.0
O E 1.1
)
U E 1.2
= A 4.0
```

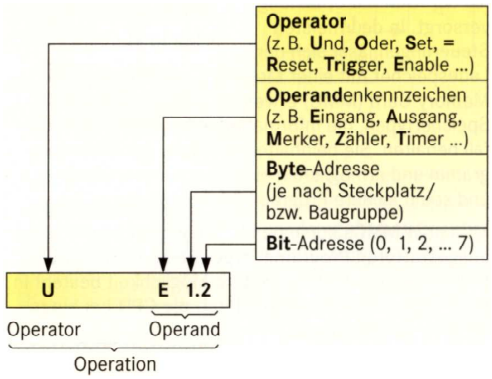
Aufgrund des allgemein hohen Bekanntheitsgrades des Funktionsplans hat sich dieser in den letzten Jahren in vielen Bereichen als SPS-Programmiersprache duchgesetzt.

Funktionspläne benötigen im Vergleich zu AWL sehr viel Platz. Um dieses Skript kompakt zuhalten, sind einige Programmbeispiele daher als AWL angegeben.

2.4 Operatoren, Operanden und Operationen

Befehle setzen sich aus **Operator** (was soll getan werden?) und **Operand** (womit soll etwas getan werden?) zusammen.

Der Operand kann z.B. der aktuelle Zustand eines Eingangs oder Ausgangs oder eines Merkers, Zählers oder Timers sein. Bei Ein- oder Ausgängen setzt sich die Adresse aus Byte- (=Steckplatz des Bausteins) und Bitadresse (= Nummer des Bits auf dem Baustein) zusammen. Bei Merkern, Zählern oder Timern ist die Adresse gleich der Nummer des entsprechend gewünschten Merkers, Zählers oder Timers.



3. Einführung in die SPS-Programmierung

3.1 Schalten eines oder mehrerer Ausgänge

Aufgabe 1: Bei Betätigung des Tasters S1 (NO) soll der Leuchtmelder P1 leuchten, ansonsten nicht.

Lösung:

AWL:

```
U    E     0.1          // Lade Eingang 0.1 in den 1-Bit-Speicher „VKE“ (Verknüpfungsergebnis)
=    A     0.0          // Gib den Inhalt des 1-Bit-Speichers „VKE“ auf Ausgang 0.0 aus
BE
```

KOP:

```
E 0.1                                          A 0.0
---] [---+---------+---------+---------+---(    )-
```

FUP:

```
                +---+     +-------+
E 0.1      ---! & !--+-! =     ! A 0.0
                +---+     +-------+:BE
```

Aufgabe 2: Bei Betätigung des Tasters S1 (NC) soll der Leuchtmelder P1 leuchten, ansonsten nicht.

Lösung:

AWL:
```
UN   E    0.1          // Lade den Eingang 0.1, negiere ihn und speichere das Ergebnis im „VKE“
=    A    0.0          // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
BE
```

KOP:
```
E 0.1                                          A 0.0
---]/[---+---------+---------+---------+--(    )-
```

FUP:
```
                +---+    +-------+
E 0.1      --0! & !--+-! =     ! A 0.0
                +---+    +-------+:BE
```

Aufgabe 3: Bei Betätigung des Tasters S1 (NO) sollen die Leuchtmelder P1, P2 und P3 leuchten, ansonsten nicht.

Lösung:

AWL:
```
U    E    0.1          // Lade den Eingang 0.1 in das „VKE“
=    A    0.0          // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
=    A    0.1          // Gib den Inhalt des „VKE“ auf Ausgang 0.1 aus
=    A    0.2          // Gib den Inhalt des „VKE“ auf Ausgang 0.2 aus
BE
```

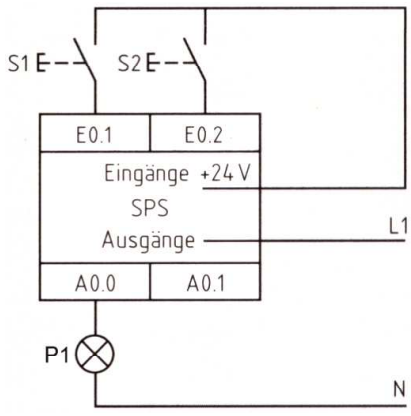
KOP:
```
E 0.1                                          A 0.0
---] [---+---------+---------+---------+--(    )-
                                          !
                                          !   A 0.1
                                          +--(    )-
                                          !
                                          !   A 0.2
                                          +--(    )-


                                             :BE
```

FUP:
```
                +---+    +-------+
E 0.1       ---! & !--+-! =     ! A 0.0
                +---+   ! +-------+
                        ! +-------+
                        +-! =     ! A 0.1
                        ! +-------+
                        ! +-------+
                        +-! =     ! A 0.2
                          +-------+:BE
```

3.2 Grundverknüpfungen

Aufgabe 1: Bei Betätigung der Taster S1 (NO) UND S2 (NO) soll der Leuchtmelder P1 leuchten, ansonsten nicht.



SPS-Klemmenanschlussplan

Lösung:

AWL:
```
U    E    0.1          // Lade den Eingang 0.1 in das „VKE“
U    E    0.2          // VerUNDe das „VKE“ mit dem Eingang 0.2 und speichere das Ergebnis im „VKE“
=    A    0.0          // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
BE
```

KOP:
```
E 0.1     E 0.2                                          A 0.0
---] [---+---] [---+----------+----------+----------+--(    )-
```

FUP:
```
                +---+
E 0.1      ---! & !    +-------+
E 0.2      ---!   !--+-! =     ! A 0.0
                +---+    +-------+:BE
```

Aufgabe 2: Bei Betätigung des Tasters S1 (NO) ODER des Tasters S2 (NO) soll der Leuchtmelder P1 leuchten, ansonsten nicht.

→ SPS-Klemmenanschlussplan wie bei UND-Schaltung

Lösung:

AWL:
```
O    E    0.1          // Lade den Eingang 0.1 in das „VKE“
O    E    0.2          // VerODERe das „VKE“ mit dem Eingang 0.2 und speichere das Ergebnis im „VKE“
=    A    0.0          // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
BE
```

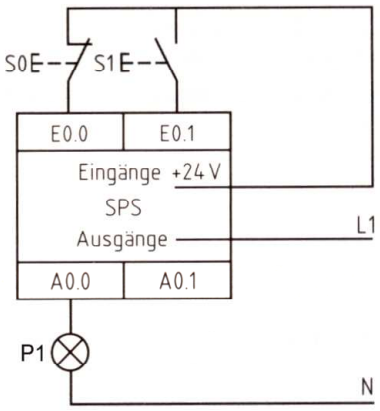
KOP:
```
E 0.1                                                   A 0.0
---] [---+----------+----------+----------+----------+--(    )-
         !
E 0.2    !
---] [---+                                             :BE
```

FUP:
```
                +---+
E 0.1      ---!>=1!    +-------+
E 0.2      ---!   !--+-! =     ! A 0.0
                +---+    +-------+:BE
```

3.3 Schalten speichernder Ausgänge

Aufgabe 1:
Nach Betätigung des Tasters S1 soll der Leuchtmelder P1 leuchten, auch wenn der Geber nicht mehr betätigt wird. Nach Betätigung des Tasters S0 soll der Leuchtmelder P1 nicht mehr leuchten.

Eingangsbeschaltung:
Ein Drahtbruch darf nicht zu einem ungewollten Einschaltvorgang oder zur Verhinderung eines möglichen Ausschaltvorganges führen. Signalgeber mit Einschaltfunktion sind daher als Schließer, Signalgeber mit Ausschaltfunktion als Öffner realisieren → **Drahtbruchsicherheit**



SPS-Klemmenanschlussplan

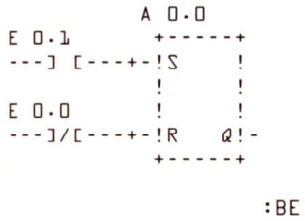
Lösung 1:

AWL:

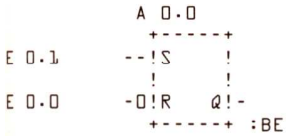
```
U   E   0.1
S   A   0.0
UN  E   0.0
R   A   0.0
BE
```

// Lade den Eingang 0.1 in das „VKE“
// **S**etze Ausgang 0.0 speichernd falls „VKE“ =1
// Lade den Eingang 0.0, negiere ihn und speichere das Ergebnis im „VKE“
// **R**ücksetze Ausgang 0.0 speichernd falls „VKE“ = 1

KOP:



FUP:



Bei gleichzeitigem Anliegen von S- und R-Signal wird der Ausgang ständig rückgesetzt (→ **R-dominant**).

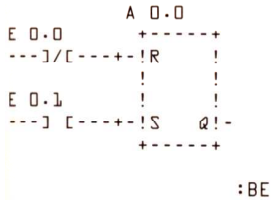
Lösung 2:

AWL:

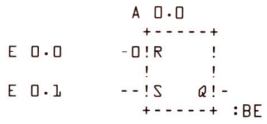
```
UN  E   0.0
R   A   0.0
U   E   0.1
S   A   0.0
BE
```

// Lade den Eingang 0.0, negiere ihn und speichere das Ergebnis im „VKE“
// **R**ücksetze Ausgang 0.0 speichernd falls „VKE“ = 1
// Lade den Eingang 0.1 in das „VKE“
// **S**etze Ausgang 0.0 speichernd falls „VKE“ =1

KOP:



FUP:



Bei gleichzeitigem Anliegen von R- und S-Signal wird der Ausgang ständig gesetzt (→ **S-dominant**).

Merker:

Die Anzahl der verfügbaren Ausgänge einer SPS ist begrenzt. Für Verknüpfungen innerhalb der SPS, bei denen keine Signalisierung außerhalb der SPS erforderlich ist, werden deshalb **Merker** eingesetzt:

- in einer größeren Anzahl in der SPS vorhanden
- entsprechen den Hilfsschützen der Schütztechnik
- werden wie Ausgänge programmiert
- werden durch Speicherzellen des RAM-Speichers der SPS gebildet

Bei Ausfall der Betriebsspannung der SPS geht der Speicherinhalt der **Normalmerker** verloren. Besteht die Notwendigkeit, Anlagenzustände in Merkern zu speichern, die nach einem Spannungsausfall noch erhalten bleiben, können nullspannungssichere **Haftmerker** (= **remanente Merker**) eingesetzt werden.

Netzwerke:

Programmbausteine können in einzelne **Netzwerke** unterteilt werden. Ein Netzwerk ist ein abgeschlossener Teil eines SPS-Programms, der im Wesentlichen einen Schritt (→ Schrittketten) einer Steuerung realisiert:

- Programmierung eines Speichers
- Ansteuerung eines Ausgangs

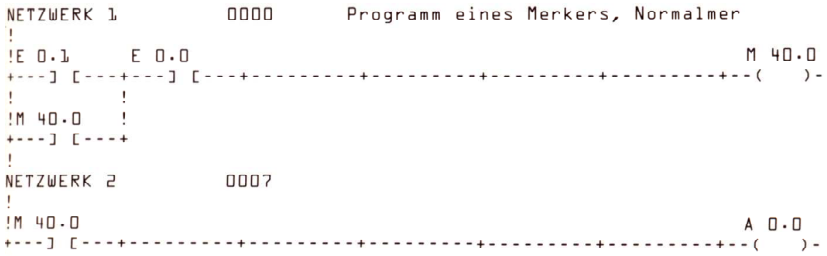
Lösung 3:

AWL:

```
NETZWERK 1                0000
0000        :U(
0001        :O    E     0.1
0002        :O    M    40.0      // M 40.0 ist Normalmerker
0003        :)                   // M 0.0 wäre Haftmerker
0004        :U    E     0.0
0005        :=    M    40.0
0006        :***

NETZWERK 2                0007
0007        :U    M    40.0
0008        :=    A     0.0
0009        :BE
```

KOP:



FUP:

3.4 Zusammenfassung der wesentlichen Programmierregeln

- Jedes Netzwerk beginnt mit einer **U**ND- bzw. **O**DER-Anweisung
- Jedes Netzwerk wird mit einer Zuweisung (**=**), einer **S**etz- oder einer **R**ücksetzanweisung abgeschlossen
- Mit einem Eingang können beliebig viele Ausgänge verknüpft sein
- Jeder Ausgang darf nur einmal im Programm zugewiesen, gesetzt oder rückgesetzt werden
- Zur Verkürzung der Zykluszeit werden alle Programmbausteine mit der **BE**-Anweisung (Bausteinende) abgeschlossen

4. Anwendung von Zeitstufen und Zählern

Zeitstufen:

Aufgabe 1:
Wird der Taster S1 mindestens 3 Sekunden lang betätigt, soll der Leuchtmelder P1 leuchten bis der Taster nicht mehr betätigt wird.

Lösung:

```
AWL:  U    E  0.1      // Lade den Eingang 0.1 in das „VKE“
      L    S5T#3S      // Lade Timerkonstante „3 Sekunden“ (in Sekunden S, Minuten MIN oder Stunden H)
      SE   T  5        // Starte Einschaltverzögerung mit Timerbaustein 5 (nur bei 01-Flanke des „VKE“ !!!)
      U    T  5        // Lade aktuellen Inhalt von Timerbaustein 5 in das „VKE“
      =    A  0.0      // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
      BE
```

Aufgabe 2:
Wird der Taster S1 betätigt, soll der Leuchtmelder P1 sofort leuchten. Wird der Taster nicht mehr betätigt, soll der Leuchtmelder noch 3 Sekunden weiter leuchten.

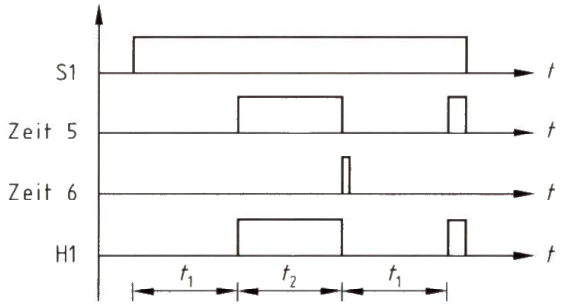
Lösung:

```
AWL:  U    E  0.1      // Lade den Eingang 0.1 in das „VKE“
      L    S5T#3S      // Lade Timerkonstante „3 Sekunden“ (in Sekunden S, Minuten MIN oder Stunden H)
      SA   T  5        // Starte Ausschaltverzögerung mit Timerbaustein 5 (nur bei 10-Flanke des „VKE“ !!!)
      U    T  5        // Lade aktuellen Inhalt von Timerbaustein 5
      =    A  0.0      // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus
      BE
```

Aufgabe 3:
Solange der Taster S1 betätigt wird, soll der Leuchtmelder P1 mit einer Frequenz von f = 0,5 Hz blinken.

Lösung:

```
AWL:  U    E  0.1
      UN   T  6
      L    S5T#1S
      SE   T  5
      U    T  5
      =    A  0.0
      ***
      U    T  5
      L    S5T#1S
      SE   T  6
      BE
```



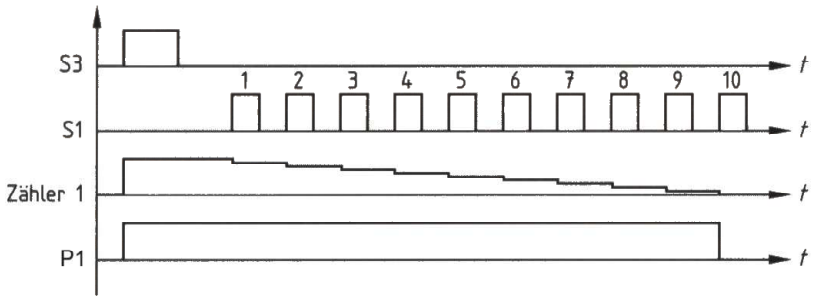
Diese Lösung soll nur als Programmierbeispiel dienen. Bei modernen Speicherprogrammierbaren Steuerungen benutzt man zur Lösung dieser Aufgabe einen frei programmierbaren sogenannten Blinkmerker.

Zähler:

Aufgabe 4:
Es sollen freie Parkplätze in einer Tiefgarage gezählt werden. Es stehen insgesamt 100 Parkplätze zur Verfügung. Sensor 1 registriert die einfahrenden Fahrzeuge, Sensor 2 die ausfahrenden. Mit Taster 1 wird die Anlage initialisiert (alle Parkplätze sind frei). Mit Taster 2 wird die Einfahrt in die Tiefgarage gesperrt (kein Parkplatz ist frei). Wenn alle Parkplätze belegt sind, soll eine rote ansonsten eine grüne Meldeleuchte leuchten.

Lösung:

```
AWL:  U    E  0.1      // Lade den Eingang 0.1 (Geber 1) in das „VKE“
      ZR   Z  1        // Zähle rückwärts: Zähler 1 (nur bei 01-Flanke des „VKE“ !!!)
      U    E  0.2      // Lade den Eingang 0.2 (Geber 2) in das „VKE“
      ZV   Z  1        // Zähle vorwärts: Zähler 1 (nur bei 01-Flanke des „VKE“ !!!)
      U    E  0.3      // Lade den Eingang 0.3 (Initialisieren) in das „VKE“
      L    C#100       // Lade Zählerkonstante 100
      S    Z  1        // Setze Zähler 1 (auf aktuellen Inhalt von C#)
      U    E  0.4      // Lade den Eingang 0.4 (Sperren) in das „VKE“
      R    Z  1        // Rücksetze Zähler 1 (auf Null setzen / nur bei 01-Flanke des „VKE“ !!!)
      ***
      U    Z  1        // Lade den Zustand von Zähler 1 (> 0 ergibt eine 1, ansonsten 0) in das „VKE“
      =    A  0.0      // Gib den Inhalt des „VKE“ auf Ausgang 0.0 aus (Grüne Meldeleuchte)
      UN   Z  1        // Lade den Zustand von Zähler 1 negiert in das „VKE“
      =    A  0.1      // Gib den Inhalt des „VKE“ auf Ausgang 0.1 aus (Rote Meldeleuchte)
      BE
```



5. Programmierung von Ablaufsteuerungen

5.1 Einleitung

Eine **Ablaufsteuerung** läuft zwangsweise in Schritten ab, wobei das Weiterschalten von einem Schritt zum nächsten von den **Weiterschaltbedingungen** abhängig ist.

zeitgeführte Ablaufsteuerungen:	**prozessgeführte** Ablaufsteuerungen:
Weiterschalten ist nur von der Zeit abhängig	Weiterschalten ist abhängig von Signalen der Anlage

In der Praxis kommen häufig Mischformen aus zeit- und prozessgeführten Ablaufsteuerungen vor.

Zur grafischen Darstellung wurden bis März 2005 **Funktionspläne** nach DIN 40719 verwendet. April 2005 wurde die Norm durch einige Änderungen zur DIN 60848 **Grafcet**. Im Folgenden wird zunächst auf die alte Norm eingegangen um im Anschluss die Änderungen durch Grafcet aufzugreifen.

Da Ablaufsteuerungen schrittweise arbeiten, spricht man auch von **Schrittketten**.

5.2 Darstellung von Schrittketten mittels FUP

Ein Schritt entspricht einem definierten Anlagenzustand mit

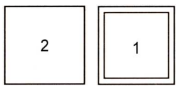
- eindeutiger Startbedingung
- eindeutiger Aufgabenstellung (einem oder mehreren Befehlen)
- eindeutiger Abschlussbedingung.

Beispiel:
Wenn Taster S1 betätigt wird, schließt Magnetventil Q1 und es öffnet Magnetventil Q2 bis Taster S2 betätigt wird.

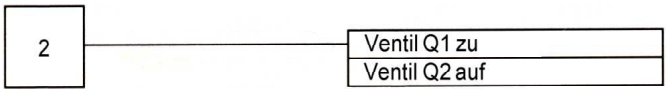
Startbedingung: Betätigung von Taster S1
Aufgabenstellung: Magnetventil Q1 schließen / Magnetventil Q2 öffnen
Abschlussbedingung: Betätigung von Taster S2

Darstellung:

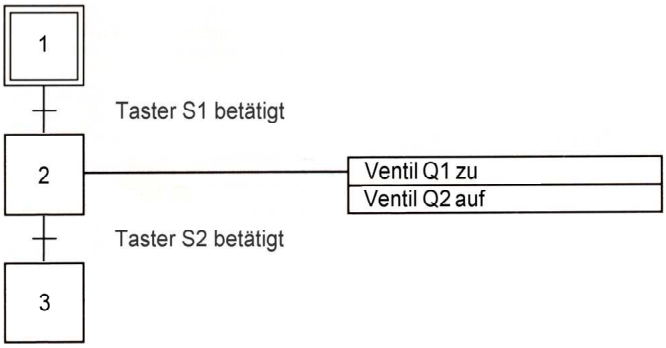
Ein **Schritt** wird als Quadrat mit (alpha)numerischer Kennzeichnung des Schrittes gezeichnet. Der Anfangsschritt (= Grundschritt, Initialschritt) wird mit einer Doppellinie gekennzeichnet:



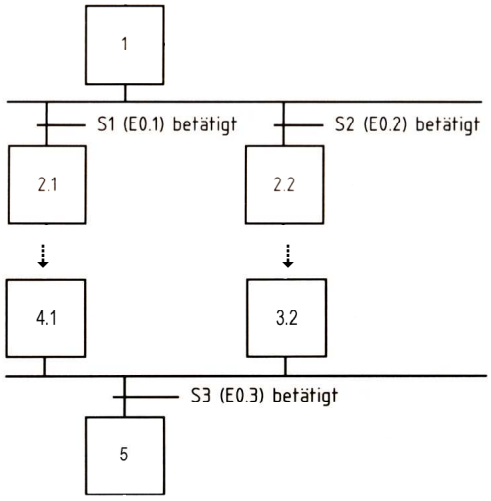
Ein **Befehl** wird als rechteckiges Kästchen mit Verbindungslinie neben den entsprechenden Schritt gezeichnet. Wenn mehrere Befehle in einem Schritt auszuführen sind, werden diese in einzelnen Kästchen untereinander aufgeführt:



Die **Einschalt-** bzw. **Weiterschaltbedingung** wird mit einem kurzen Querstrich an der Verbindungslinie zwischen zwei Schrittsymbolen gekennzeichnet, neben dem die Schaltbedingung in textueller Form oder als boolescher Ausdruck angegeben wird:



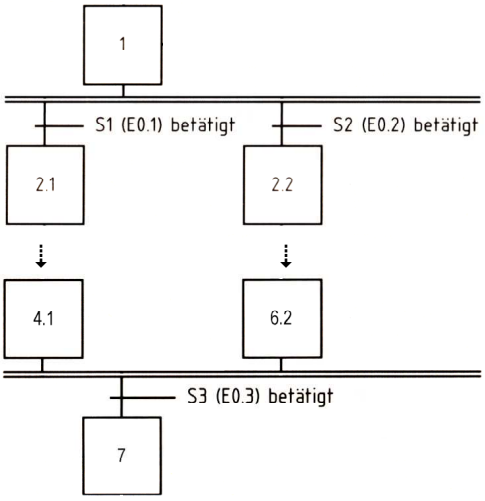
Verzweigt ein Prozess je nach Anlagensituation in unterschiedliche Teilprozesse, von denen nur einer ausgeführt wird (**ODER**), wird dies mit einem einfachen Strich zwischen dem Vorgängerschritt und den möglichen Folgeschritten gekennzeichnet. Die Zusammenführung nach einer ODER-Verzweigung zu gemeinsamen Folgeschritten erfolgt entsprechend.



Wenn S1 vor S2 betätigt wird, folgen auf Schritt 1 Schritt 2.1 und die darauf folgenden Schritte. Schritt 2.2 wird nicht ausgeführt.

Zum Weiterschalten muss entweder 4.1 oder 3.2 aktiv und S3 betätigt sein.

Teilt sich ein Prozess in parallel ablaufende (voneinander zeitlich unabhängige) Teilprozesse auf, die alle ausgeführt werden (**UND**), erfolgt die Darstellung entsprechend mit einem doppelten Strich. Die Zusammenführung nach einer UND-Verzweigung (Synchronisation der Teilprozesse) zu gemeinsamen Folgeschritten erfolgt entsprechend.



Sobald S1 betätigt wird, wird Schritt 2.1 gestartet. Sobald S2 betätigt wird, wird Schritt 2.2 gestartet. Erst wenn alle auf das UND folgenden Teilprozesse gestartet wurden (die Reihenfolge ist egal), wird Schritt 1 beendet.

Zum Weiterschalten müssen sowohl 4.1 als auch 6.2 aktiv und S3 betätigt sein.

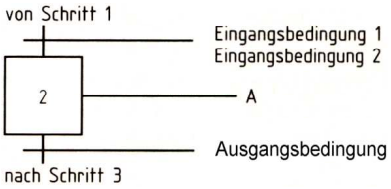
5.3 Abwicklung und Umsetzung von Schrittketten

Schaltregeln:

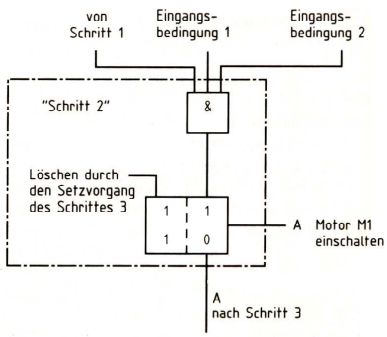
- Die Schritte einer Kette werden nacheinander durchlaufen
- In einer Kette ist immer nur ein Schritt gesetzt (aktiv).
  Bei UND-Verzweigungen gilt dies für jede Teilkette
- Wird ein Schritt gesetzt, wird der davor liegende Schritt gelöscht
- Die einzelnen Schritte können Befehle an die Stellglieder ausgeben
- Die Weiterschaltbedingung eines Schrittes kann prozessabhängig oder zeitabhängig sein

In der Realisierung einer Schrittkette in einem SPS-Programm ist ein Schritt eine **Speicherschaltung**:

Makrodarstellung:



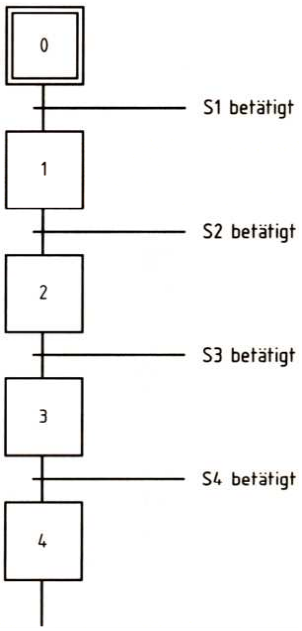
Ausführliche Darstellung:



Umsetzung einer Schrittkette in einem SPS-Programm:

Eine Schrittkette mit 4 Schritten soll umgesetzt werden. Die Schritte 1 bis 4 sollen durch die LEDs der Ausgänge A2.1 bis A2.4 angezeigt werden (Verwendung der Ausgänge als Schrittmerker).

Schritt1 soll gesetzt werden, wenn Taster S1 betätigt wird. Ebenso sollen die folgenden Schritte 2, 3 bzw. 4 durch Betätigung der Taster S2, S3 respektive S4 gesetzt werden.




```
                Schritt 1
U    E     0.1
S    A     2.1          Setzen von Schritt 1
U    A     2.2
R    A     2.1          Ruecksetzen von Schritt 1
***                     Ende Netzwerk

                Schritt 2
U    E     0.2
U    A     2.1
S    A     2.2          Setzen von Schritt 2
U    A     2.3
R    A     2.2          Ruecksetzen von Schritt 2
***                     Ende Netzwerk

                Schritt 3
U    E     0.3
U    A     2.2
S    A     2.3          Setzen von Schritt 3
U    A     2.4
R    A     2.3          Ruecksetzen von Schritt 3
***                     Ende Netzwerk

                Schritt 4
U    E     0.4
U    A     2.3
S    A     2.4          Setzen von Schritt 4
U    A     2.5
R    A     2.4          Ruecksetzen von Schritt 4
BE                      Programm Ende
```

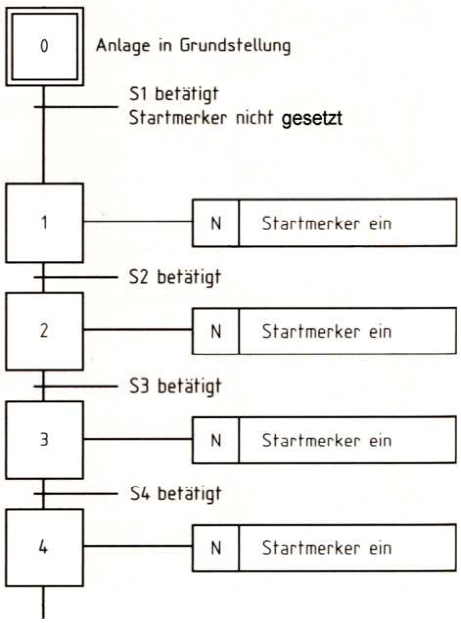
5.4 Verwendung von Startmerkern

Die Schrittkette aus Abschnitt 5.3 weist einen Fehler auf:

→ Schritt 1 kann jederzeit gesetzt werden, wenn seine Eingangsbedingung erfüllt ist, unabhängig davon, ob der Initialzustand 0 gesetzt ist oder nicht.

Zur Abhilfe wird ein **Startmerker** eingesetzt, der darüber informiert, ob die Schrittkette bereits gestartet wurde.

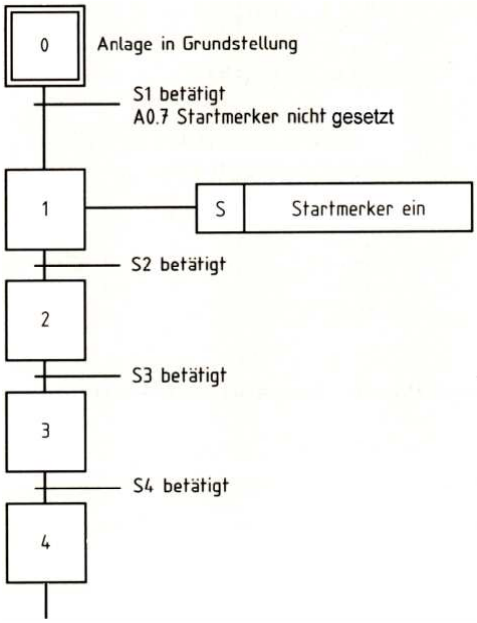
Nicht speichernder Startmerker:



// Schritt 1

```
U     E 0.1
UN    A 2.7     // Startmerker
S     A 2.1
U     A 2.2
R     A 2.1
```

Speichernder Startmerker:



// Schritt 1

```
U     E 0.1
UN    A 2.7     // Startmerker
S     A 2.1
U     A 2.2
R     A 2.1
```

```
// Abarbeiten der anderen Schritte
...
// Startmerker
O     A 2.1
O     A 2.2
O     A 2.3
O     A 2.4
=     A 2.7
```

```
// Abarbeiten der anderen Schritte
...
// Startmerker
U     A 2.1
S     A 2.7
```

Befehlsarten:

Im FUP wird links neben der allgemeinen Bezeichnung eines Befehls auch die Befehlsart angegeben, bspw.

- S (stored): gespeichert
- D (delayed): verzögert
- L (time limited): zeitlich begrenzt
- N (not stored): nicht gespeichert

In der Automatisierungstechnik spielen Schrittketten, die irgendwann automatisch beendet werden, keine Rolle. Tatsächlich kehrt die Anlage nach dem letzten auszuführenden Schritt wieder in die Grundstellung zurück. Im FUP wird dies durch eine Rückführung zum Initialschritt 0 gekennzeichnet.

Bei speichernden Startmerkern muss nach dem letzten Schritt der eigentlichen Schrittkette ein zusätzlicher Schritt eingeführt werden, in dem der Startmerker zurückgesetzt wird. Nach dem nächsten Zyklus wird dann ohne Weiterschaltbedingung zum Initialschritt 0 zurückgeschaltet.

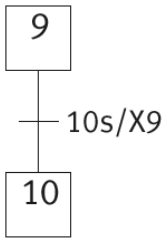
5.5 Vom Funktionsplan zu Grafcet

GRAFCET = GRAphe Fonctionnel de Commande Etape Transition

(sinngemäß: Darstellung der Steuerungsfunktion mit Schritten und Weiterschaltbedingungen)

Die grundsätzliche Darstellung der Schritte (Quadrate) und der Übergänge (Trennstrich zwischen zwei Schritten) bleibt erhalten. Die Weiterschaltbedingung wird aber nur noch in Textform oder als boolescher Ausdruck rechts neben den Trennstrich geschrieben.

Alle zeitlichen Ereignisse werden durch eine vorgestellte Zeit beschrieben. Soll beispielsweise nach Ablauf einer festgelegen Zeit in den nächsten Schritt weitergeschaltet werden, so ist als Weiterschaltbedingung die Zeit und der boolesche Zustand des aktiven Schritts (TRUE), getrennt durch einen Schrägstrich, anzugeben (X = Bool). Im folgenden Beispiel schaltet die Steuerung 10 Sekunden nach Aktivierung von Schritt 9 in den Schritt 10 weiter:

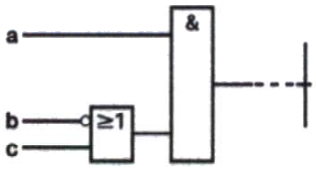


Auf der folgenden Seite werden die wesentlichen Unterschiede der alten und der neuen Norm einander gegenübergestellt.

DIN 40719 Teil 6 | **DIN EN 60848 "GRAFCET"**

Beschreibung einer Weiterschaltbedingung

Tür geschlossen (a) und (kein Druck ($\overline{b}$) oder Werkstück vorhanden (c))

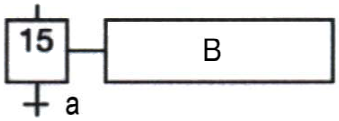


Tür geschlossen (a) und (kein Druck ($\overline{b}$) oder Werkstück vorhanden (c))

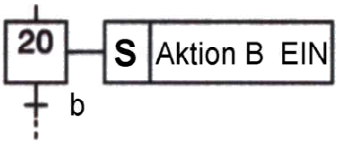
oder als andere Möglichkeit:

$a \cdot (\overline{b} + c)$

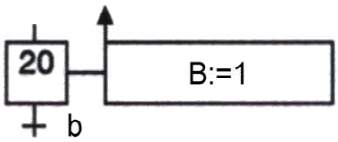
Nicht speichernde Aktion (not saving)



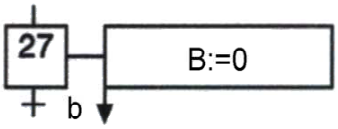
Speichernde Aktion (saving)



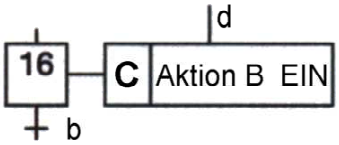
- bei Aktivierung des Schritts

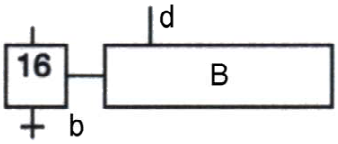


- bei Deaktivierung des Schritts

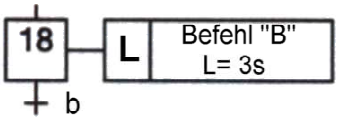


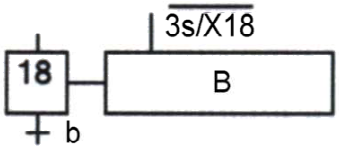
Bedingte Aktion (conditioned)



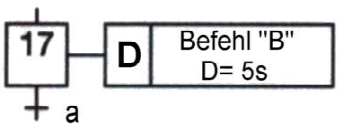


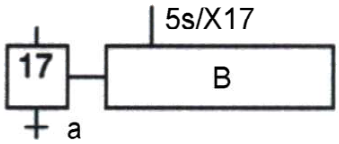
Zeitbegrenzte Aktion (time limited)





Verzögerte Aktion (delayed)



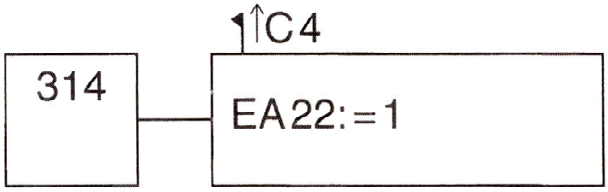


Sonderfälle bei speichernden Aktionen:

- Speichernde Aktion bei Ereignis

Bei einer speichernden Aktion bei Ereignis wird der Variablen EA 22 nur dann der Wert „ 1 “ zugewiesen, wenn bei aktiviertem Funktionsschritt 314 die Zuweisungsbedingung C4 ihren Wert ändert. Die Werteveränderung der Zuweisungsvariablen kann sowohl von 0 nach 1 - „steigende Flanke“, dargestellt wie im vorliegenden Beispiel durch einen Pfeil nach oben links neben der Zuweisungsbedingung - als auch von 1 nach 0 - „fallende Flanke“, dargestellt durch einen Pfeil nach unten - erfolgen.

Der Wert von EA 22 bleibt über die Aktivität des Funktionsschrittes 314 hinaus gespeichert.



- Zeitabhängige, speichernde Aktion

Eine zeitabhängige, speichernde Aktion wird nicht explizit in der Norm erwähnt. Die Zeit lässt sich jedoch als Ereignis definieren. In diesem Fall wird der Variablen EA 22 6 Sekunden nach der Aktivierung des Funktionsschritts 314 der Wert „ 1 “ zugewiesen.

Auch hier bleibt der Wert von EA 22 über die Aktivität des Funktionsschritts 314 hinaus gespeichert.