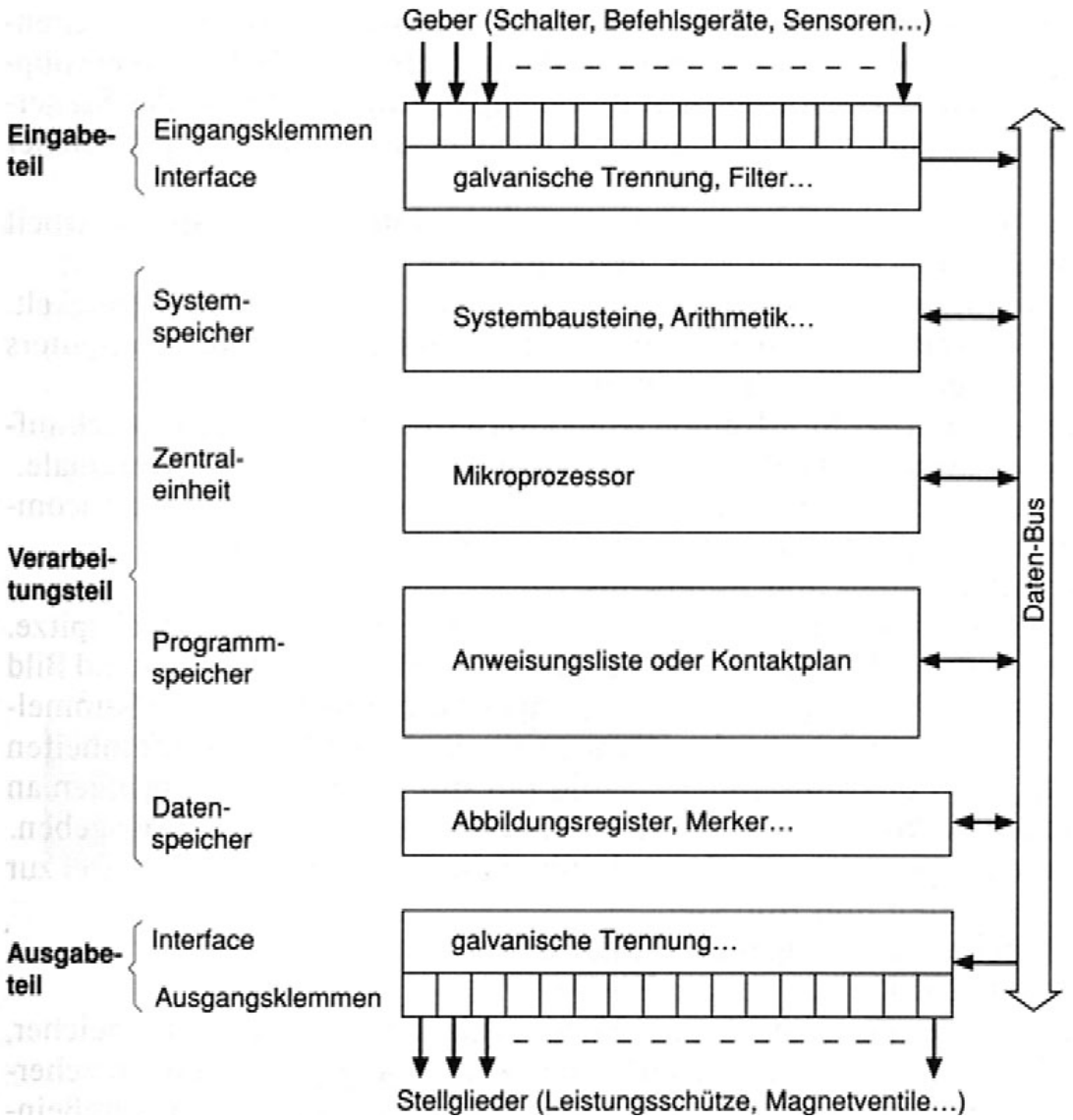
Speicherprogrammierbare Steuerung (SPS)

Definition nach DIN EN 61131-1

„Ein digital arbeitendes **elektronisches System** für den Einsatz in industrieller Umgebung mit einem **programmierbaren Speicher** zur internen Speicherung der anwenderorientierten Steuerungsanweisungen zur **Implementierung** spezifischer Funktionen wie z.B. **Verknüpfungssteuerung, Ablaufsteuerung, Zeit-, Zählfunktion und aritmetische Funktionen**, um durch digitale oder analoge **Eingangs- und Ausgangssignale** verschiedene Arten von Maschinen oder **Prozessen zu steuern**.

Die speicherprogrammierbare Steuerung und die zugehörige Peripherie (das SPS-System) sind so konzipiert, dass sie sich leicht in ein industrielles Steuerungssystem integrieren und in allen ihren beabsichtigten Funktionen einfach einsetzen lassen.“

Aufbau einer SPS



Abarbeitung eines SPS-Programms

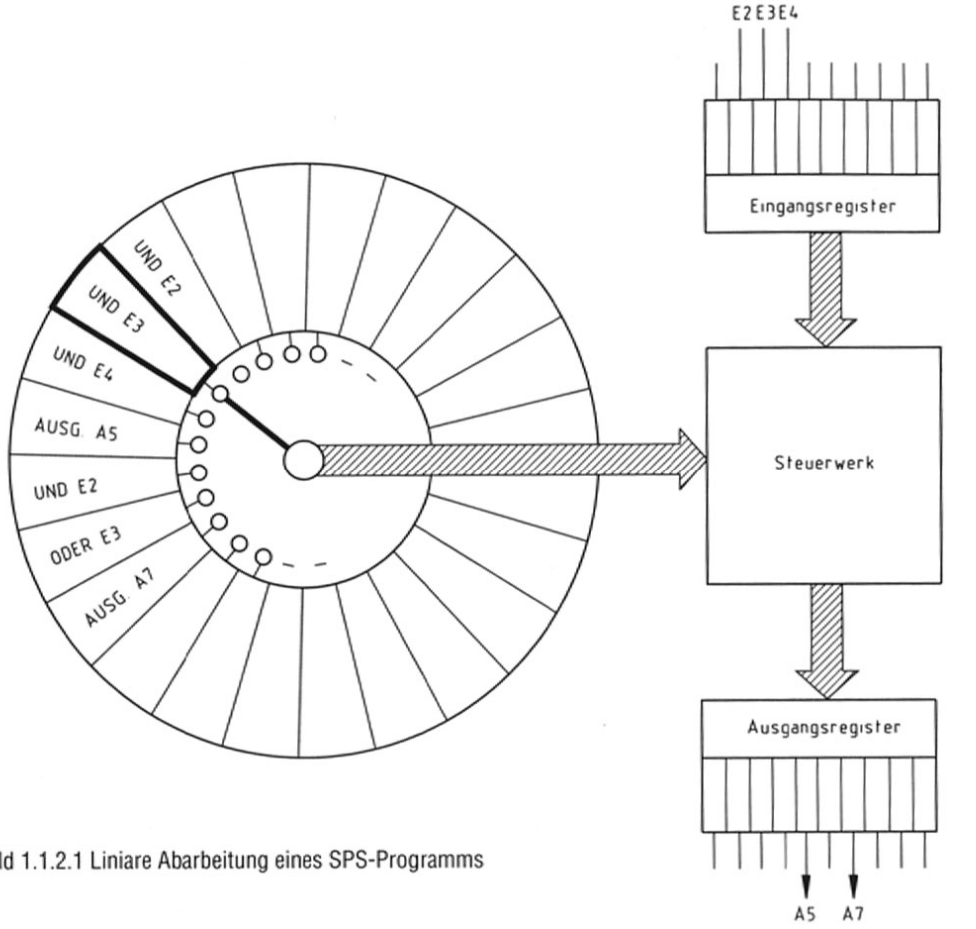


Bild 1.1.2.1 Liniare Abarbeitung eines SPS-Programms

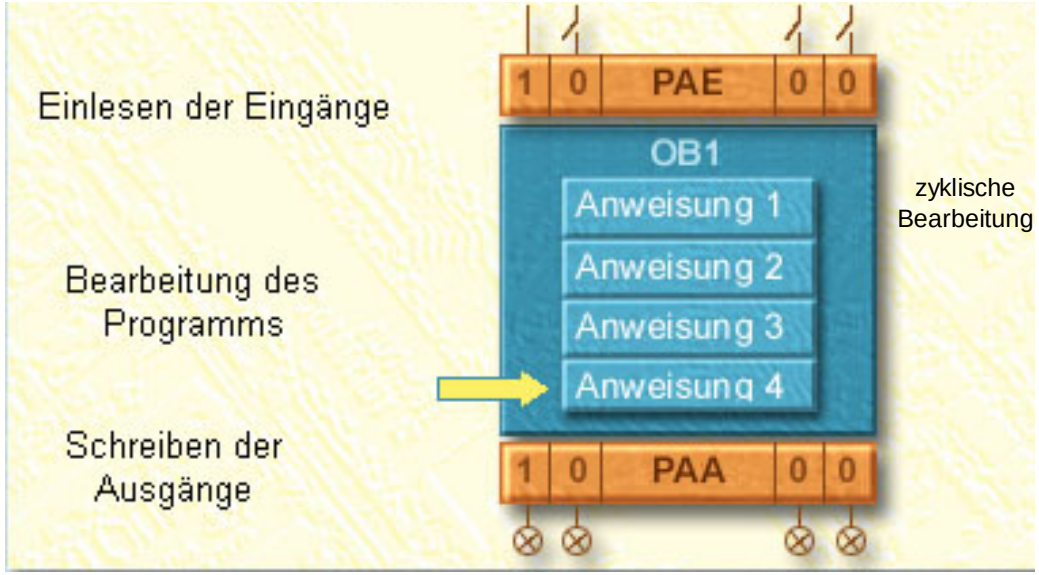
Eingangsregister

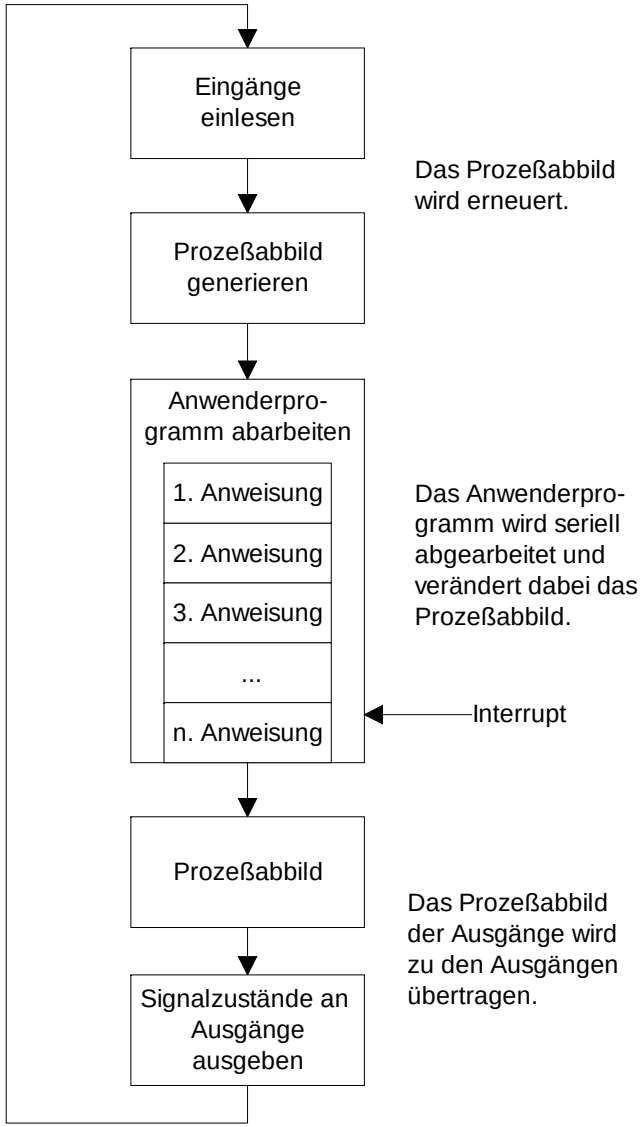
Steuerwerk

Ausgangsregister

Schlüsselwort:

ZYKLISCHE BEARBEITUNG



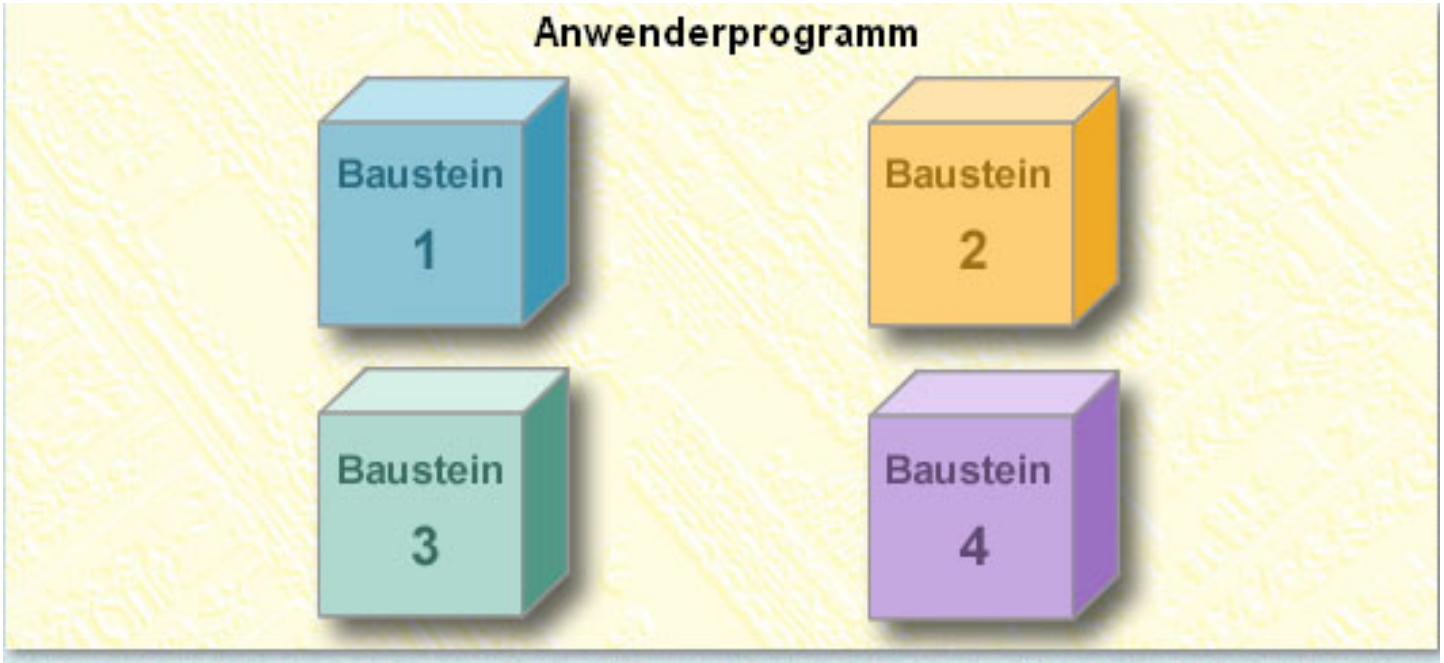


Programmverabeitung einer SPS

Die SPS bearbeitet das Programm **zyklisch**. Am Anfang eines Zyklus steht das **Einlesen der Signalzustände** (*Prozessabbild*) und am Ende das **Ausgeben der Stellwerte**, unabhängig davon, ob sie sich verändert haben oder nicht.

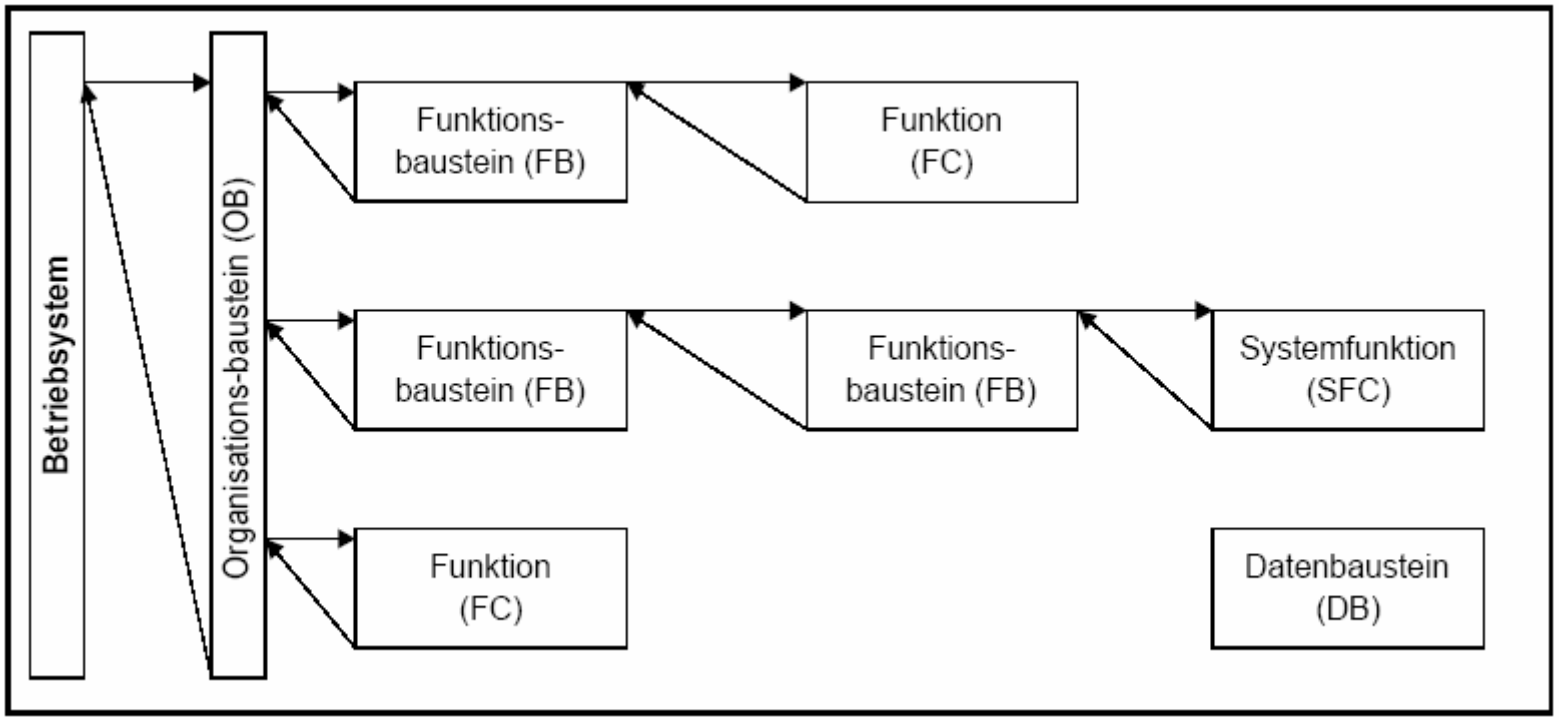
Die Zeitdauer vom Beginn des Einlesens der Eingänge bis zum Abschluss des Schreibvorgangs der Ausgänge wird ***Zykluszeit*** genannt. Die Dauer eines Zyklus ist nicht konstant, sondern hängt von der Anzahl abgearbeiteter Befehle sowie der Art und Anzahl der Ein- und Ausgänge ab.

Wünscht der Anwender, dass sein Programm auf Signale des Prozesses reagiert, die innerhalb des Bearbeitungszeitraums stattfinden, so kann er Interruptsignale festlegen, die den Bearbeitungszyklus unterbrechen.



Zur Automatisierung verfahrenstechnischer Prozesse werden häufig umfangreiche Anwenderprogramme benötigt. Deshalb werden Anwenderprogramme **strukturiert programmiert**, d.h. in einzelne, in sich geschlossene Programmabschnitte aufgeteilt. Diese Programmabschnitte entsprechen den einzelnen Teilfunktionen des Prozesses und werden als **Bausteine** eines Programms bezeichnet.

Aufteilung in Bausteine



Hierbei enthalten OB, FBs und FCs das vom Benutzer eingegebene Programm, DBs dienen der Datenspeicherung. SFBs und SFCs sind vom SPS-Hersteller fest vorgegeben und nicht editierbar.

Kurzübersicht über die Bausteintypen

Organisationsbausteine (OB) regeln den Programmablauf.
OBs sind, abhängig vom auslösenden Ereignis, in Klassen eingeteilt (z.B. zeitgesteuert, alarmgesteuert), die verschiedene Prioritäten aufweisen. Je nach Priorität können sie sich gegenseitig unterbrechen.
Beim Start eines OB wird eine detaillierte Startinformation über das auslösende Ereignis mitgeliefert. Diese Information kann im Anwenderprogramm ausgewertet werden.

Funktionsbausteine (FB) enthalten das eigentliche Anwenderprogramm. Funktionsbausteine können bei jedem Aufruf (der so genannten Instanz) mit unterschiedlichen Daten versorgt werden. Diese Daten sowie interne Variable (z.B. für Zwischenwerte) und Ergebnisse werden im zugeordneten Instanz-Datenbaustein hinterlegt und vom System automatisch verwaltet.

Instanz-Datenbausteine (Instanz-DB) werden bei Aufruf eines FB/SFB dem Baustein zugeordnet. Beim Übersetzen werden sie automatisch generiert. Der Anwender kann von jeder Stelle seines Anwenderprogramms oder auch von einem Bedien- und Beobachtungs-System auf diese Instanzdaten (natürlich auch symbolisch) zugreifen.

Funktionen (FC) enthalten Programmroutinen für häufig verwendete Funktionen. Jede Funktion hat einen festen Funktionswert (als Ergänzung zur IEC-Norm sind mehrere Ausgangsparameter möglich). Alle Ausgangsparameter müssen unmittelbar nach dem Aufruf weiterverarbeitet werden. Funktionen benötigen deshalb keinen Instanz-Datenbaustein.

Datenbausteine (DB) sind Datenbereiche zur Speicherung von Anwenderdaten. Zusätzlich zu den Daten, die jeweils einem Funktionsbaustein zugeordnet sind (Instanz-Daten), können globale Daten definiert und von beliebigen Bausteinen genutzt werden (z.B. für Rezepturen). Den Komponenten eines Datenbausteines kann ein elementarer oder strukturierter Datentyp zugewiesen werden. Elementare Datentypen sind z.B. BOOL, REAL oder INTEGER. Strukturierte Datentypen (Felder und Strukturen) setzen sich aus elementaren Datentypen zusammen (z.B. ein Rezept). Die Daten eines Datenbausteins können symbolisch adressiert werden. Dies erleichtert die Programmierung und die Lesbarkeit des Programms.

Systemfunktionsbausteine (SFB) sind Funktionsbausteine, die im Betriebssystem der CPU integriert sind, z.B. SEND, RECEIVE, Regler. Die Variablen der SFBs werden ebenfalls in Instanz-DBs abgelegt.

Systemfunktionen (SFC) sind Funktionen, die im Betriebssystem der CPU integriert sind, z.B. Zeitfunktionen, Blocktransfer

Befehlsklassen

- **Bitoperationen**: UND, ODER, Exkl.ODER, Negation, Klammersetzung, Setzen/Rücksetzen, Flankendetektion, Rotation
- **Zähler/Zeitgeber, Speicher**: Zähler vor/rück zählen und initialisieren, Zeit als (verlängerter) Impuls, Ein-/Ausschaltverzögerung, speichernde Ein-/Ausschaltverzögerung,
- **Lade-/Transferbefehle**
- **Programmorganisation**: Bausteinaufruf/-ende (un)bedingt, (un)bedingte Sprünge sowie NOP
- **Arithmetik-Operationen:** + - * / sowie > < = <>

SPS-Programmiersprachen

- Anweisungsliste (AWL)
- Kontaktplan (KOP)
- Funktionsplan (FUP)
- Strukturierter Text (ST)*
- Ablaufsprache (AS)*

* In diesem Skript nicht weiter behandelt

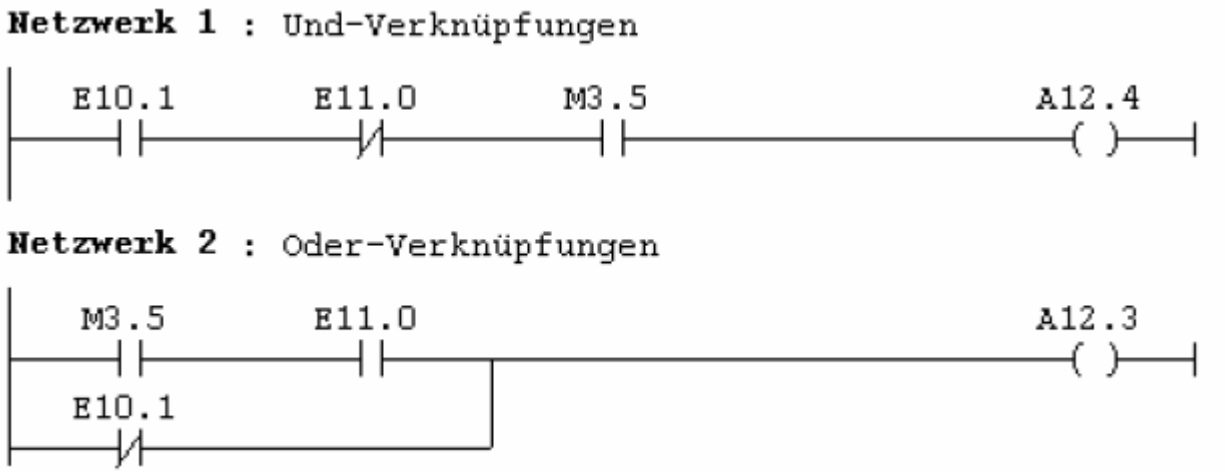
Anweisungsliste (AWL)

AWL ist an Assemblerprogrammiersprachen angelehnt; logische Verknüpfungen sind durch mnemotechnische Abkürzungen realisierbar. Im Gegensatz zur „klassischen“ Programmiersprache ist hier allerdings die Reihenfolge der Befehle von eklatanter Bedeutung, da nach jeder Zeile das **VKE-Bit (Verknüpfungsergebnis)** zur Auswertung der nächsten herangezogen wird. Diese Eigenart macht zum Teil „Umwege“ nötig, wenn komplexere Verknüpfungen mit verschalteten UND/ODER umgesetzt werden sollen.

```
Netzwerk 1 : Und-Verknüpfungen

      U     E     10.1
      UN    E     11.0
      U     M      3.5
      =     A     12.4


Netzwerk 2 : Oder-Verknüpfungen

      U     M      3.5
      U     E     11.0
      ON    E     10.1
      =     A     12.3
```

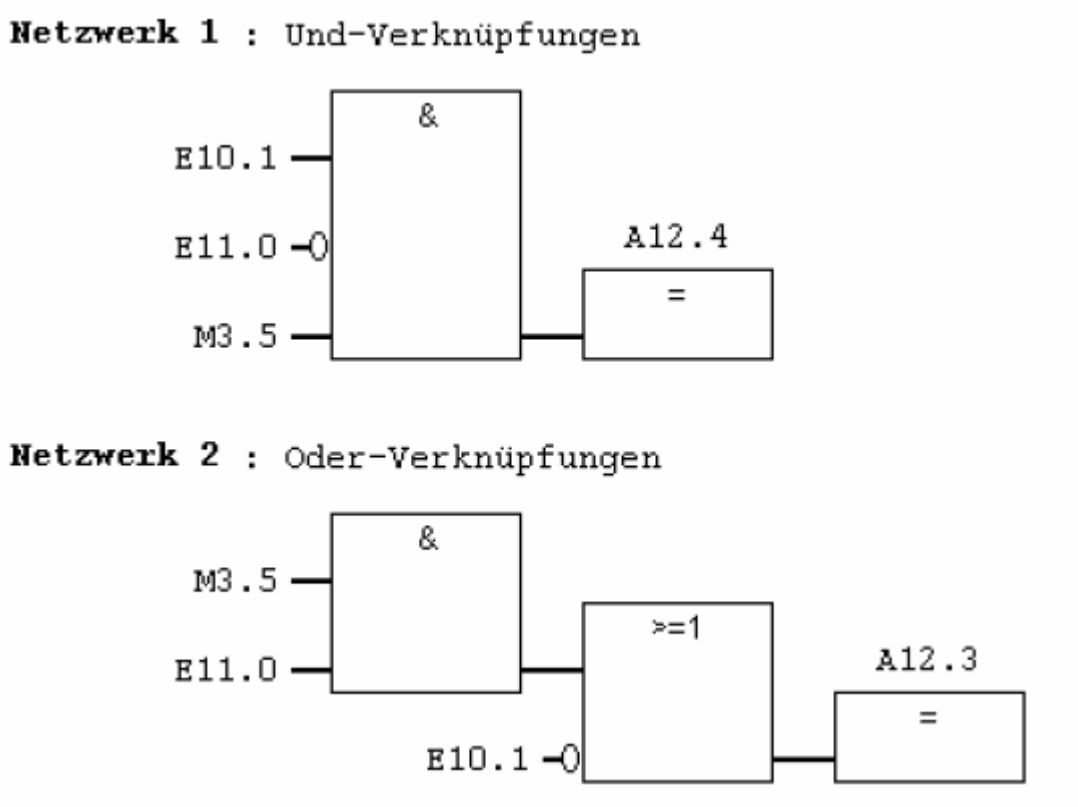
Kontaktplan (KOP)

Kontaktplan ist an Stromlaufplan angelehnt und erleichtert den Umstieg von Relais-Schaltungen zur SPS. Die senkrechten Linien links und rechts kann man als die Pole eines Stromkreises interpretieren. Eingänge werden als Relaiskontakte (Taster) dargestellt, ein **Schließer** erfüllt hierbei die logische Funktion **UND**, ein **Öffner UND NICHT**. Logisches **ODER** lässt sich durch parallele Strompfade modellieren. Der zu schaltende Ausgang wird als Relaisspule dargestellt, analog z.B. zu einer Glühbirne.



Funktionsplan (FUP)

Der Funktionsplan ist eine grafische Darstellung der Verknüpfung und stellt – zumindest für relativ simple logische Funktionen – eine recht intuitive Programmiervariante dar. Die elementaren Verknüpfungen sind jeweils durch ein spezielles Symbol dargestellt, Verbindungen zwischen ihnen werden einfach als Linien gezogen.

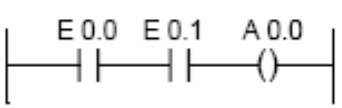


UND-Verknüpfung

UND - VERKNÜPFUNG

Die UND - Verknüpfung entspricht einer Reihenschaltung von Kontakten im Stromlaufplan. Am Ausgang A 0.0 erscheint Signalzustand 1, wenn alle Eingänge gleichzeitig den Signalzustand 1 aufweisen. Wenn einer der Eingänge den Signalzustand 0 aufweist, bleibt der Ausgang im Signalzustand 0.

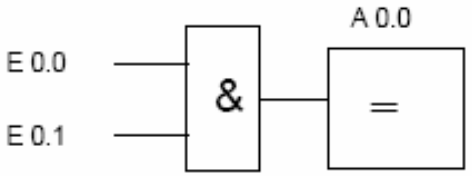
KOP



AWL

```
U  E 0.0
U  E 0.1
=  A 0.0
```

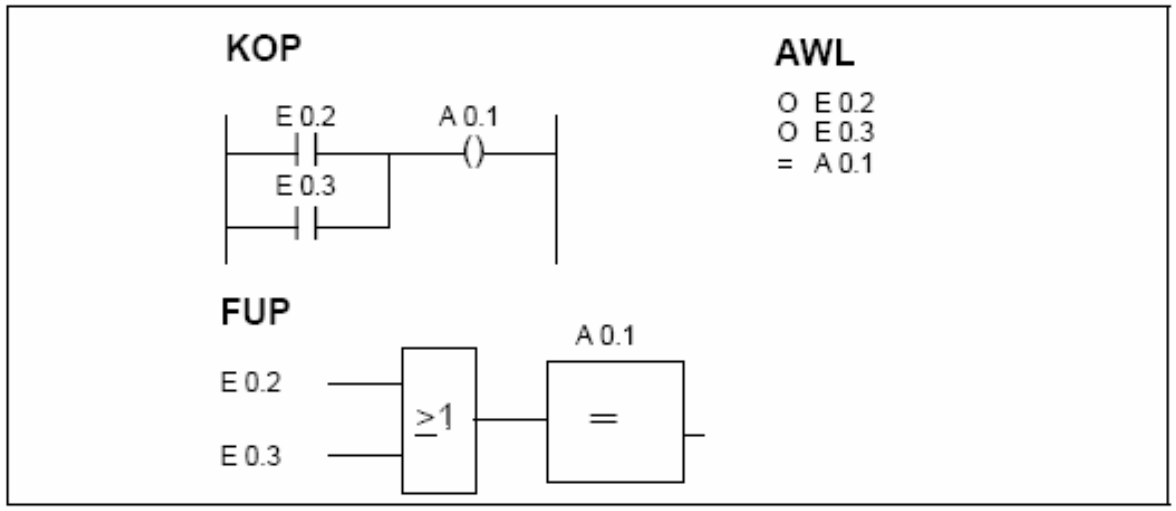
FUP



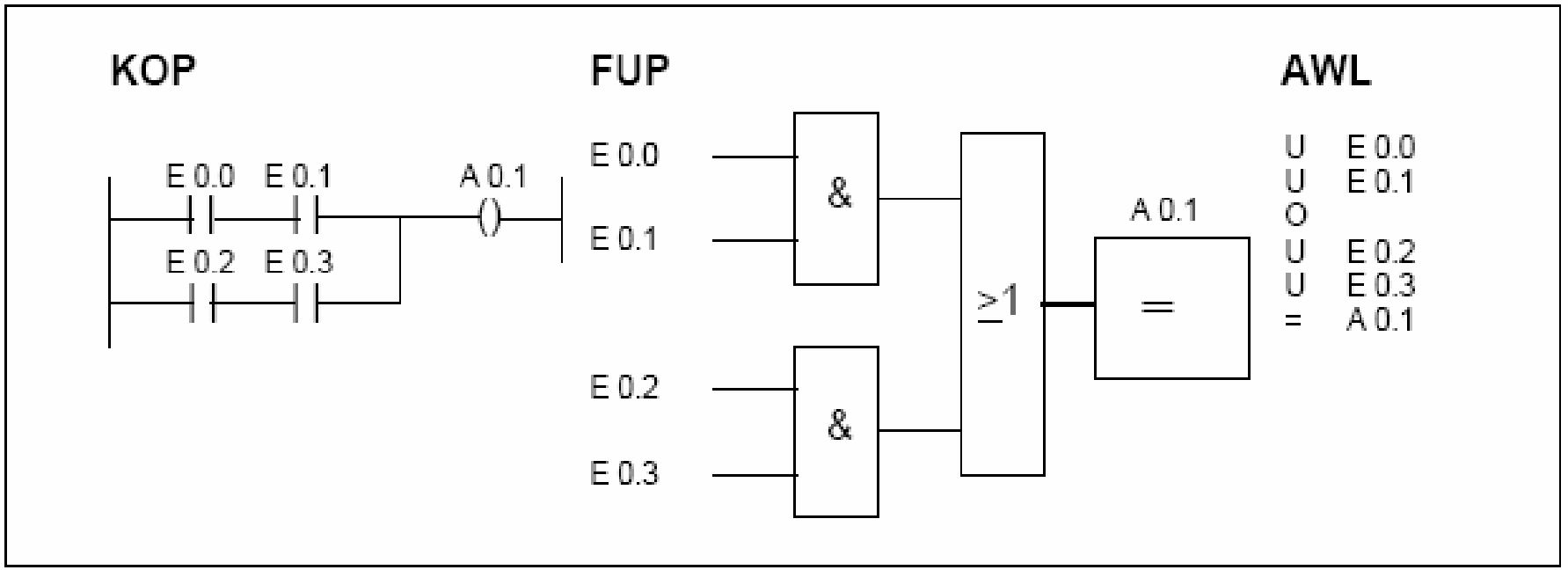
ODER-Verknüpfung

ODER - VERKNÜPFUNG

Die ODER - Verknüpfung entspricht einer Parallelschaltung von Kontakten im Stromlaufplan. Am Ausgang A 0.1 erscheint der Signalzustand 1, wenn mindestens einer der Eingänge den Signalzustand 1 aufweist. Nur wenn alle Eingänge den Signalzustand 0 aufweisen, bleibt der Signalzustand am Ausgang auf 0.

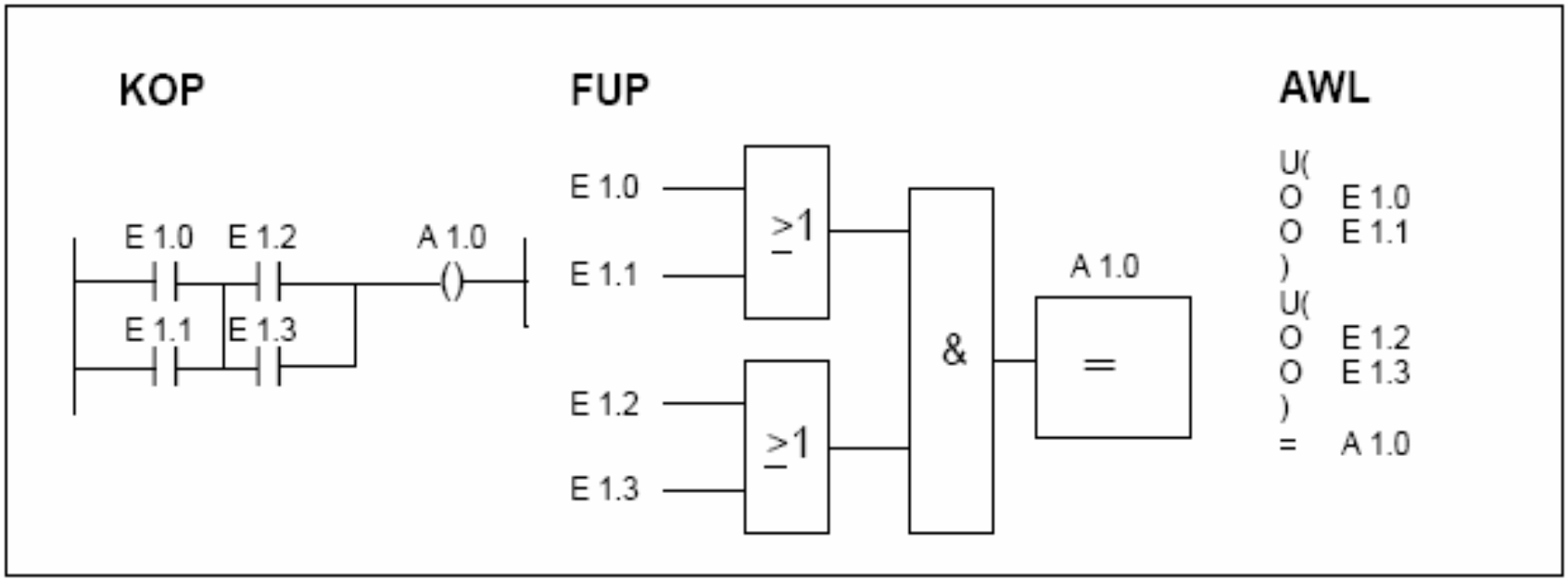


UND vor ODER



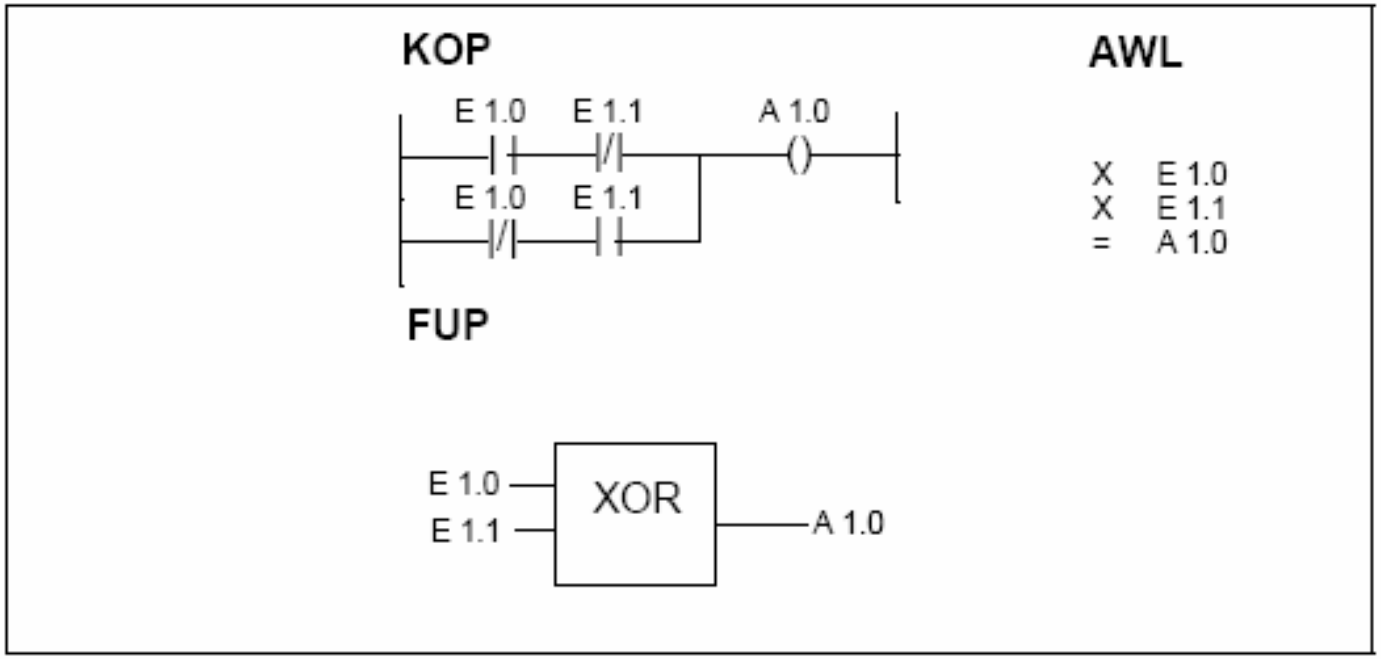
UND Verknüpfungen haben Vorrang und werden **IMMER** vor den **ODER** Verknüpfungen bearbeitet.

ODER vor UND



Damit **ODER**-Verknüpfungen Vorrang vor **UND**-Verknüpfungen haben, müssen sie durch **Klammern** zusammengefasst werden.

Exklusives ODER (XOR)



Die **Exklusiv-ODER**-Verknüpfung darf nur mit genau zwei Eingängen benutzt werden.

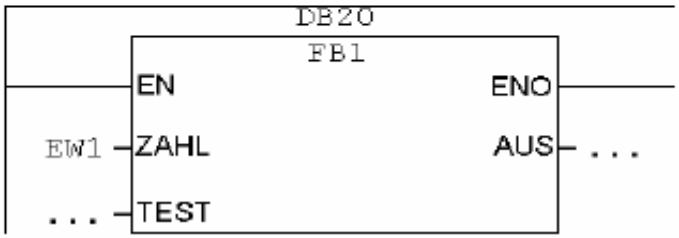
Bausteinaufruf

Mit **CALL** können Funktionen (FC), Funktionsbausteine (FB), Systemfunktionen (SFC) und Systemfunktionsbausteine (SFB) aufgerufen und parametriert oder mit zugehörigem lokalen DB geöffnet werden. Bei FCs muss für jeden der Formalparameter ein Operand angegeben werden, bei FBs ist dies nicht zwingend nötig; ausgelassene Parameter werden stattdessen dem Instanz-DB des aufgerufenen FB entnommen. Es handelt sich bei CALL um einen unbedingten Befehl (bei Verzicht auf Parametrierung entspricht CALL dem Befehl UC).

CALL FB1, DB20

ZAHL :=	EW 1	ZAHL (Formalparameter) wird EW 1 (Aktualparameter) zugeordnet.
AUS :=		AUS (Formalparameter) wird kein Parameter zugeordnet.
TEST :=		TEST (Formalparameter) wird kein Parameter zugeordnet.

KOP/FUP



Unkonditionierter Bausteinaufruf

UC entspricht im wesentlichen CALL, mit dem Unterschied, dass keine Parametrisierung möglich ist. Als unbedingter Befehl wird UC ebenso wie CALL stets unabhängig vom VKE ausgeführt → dies macht oft softwareseitige Verriegelungen nötig, um Sicherheitsaspekte zu berücksichtigen.

UC FB3

Konditionierter Bausteinauf

CC ist ein vom VKE abhängiger Bausteinaufruf, bei dem wie bei UC keine Parametrierung vorgenommen werden kann.

U E3.1
CC FB25

Bausteinende

Das Ende eines Bausteins kann bedingt oder unbedingt erfolgen.

BEB

Bausteinende abhängig vom VKE, z.B.

UN E2.1

BEB

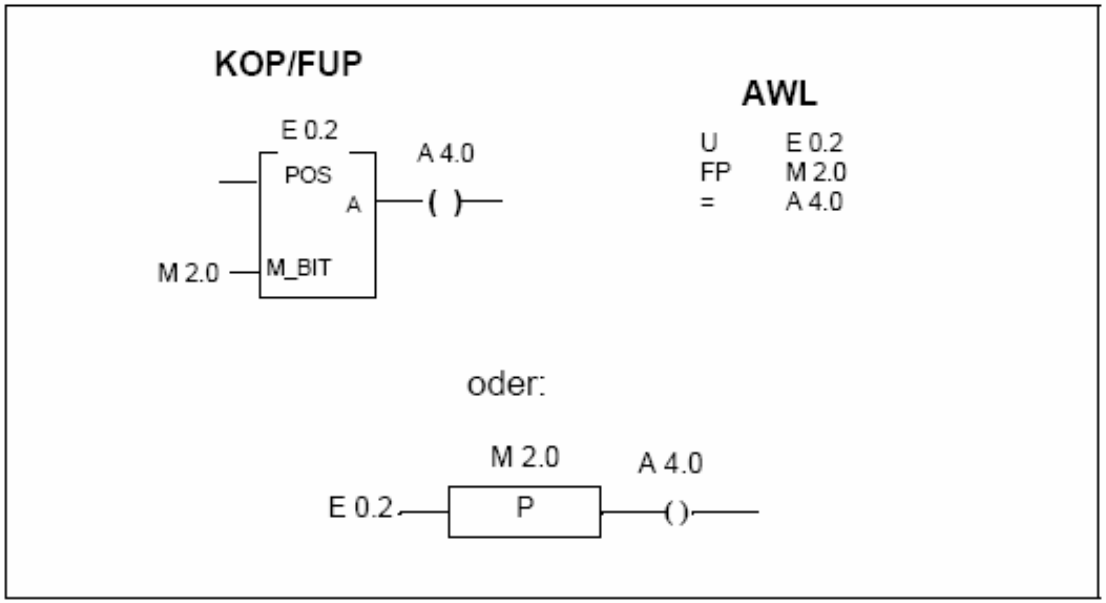
BEA

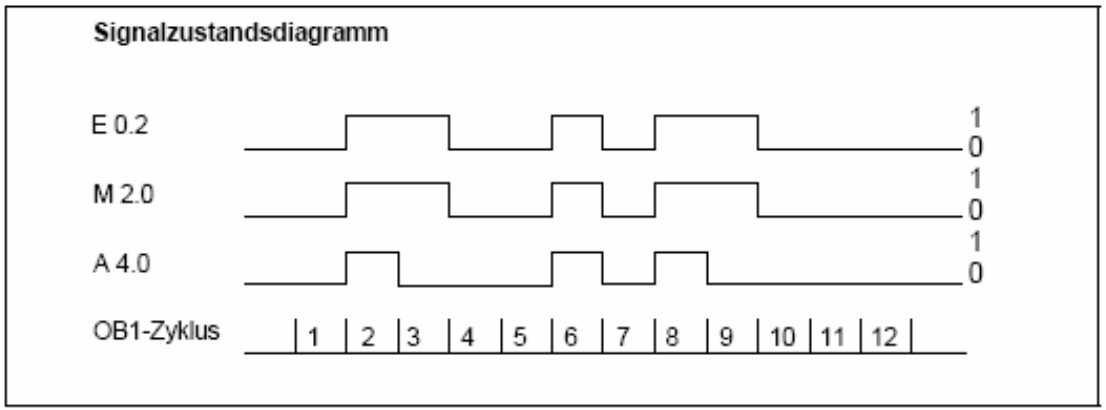
Bausteinende unabhängig vom VKE.

Flankendetektion

Zur Flankendetektion wird stets **ein Merkerbit** benötigt (hier M2.0).

Es ist unbedingt darauf zu achten, dass dieser Merker an keiner anderen Stelle des Programms genutzt wird!





Setzen (S) / Rücksetzen (R)

Durch S bzw. R wird abhängig vom VKE der gegebene Operand gesetzt bzw. zurückgesetzt (1/0).

Wichtig: Die Verknüpfungskette wird durch diesen Befehl unterbrochen und jeder nachfolgende Verknüpfungsbefehl liefert nur den Wert des eigenen Operanden. Im Beispiel wäre A4.1 daher nur von E1.2 und nicht von E1.3 abhängig.

```
U E1.3
S A4.0
O E1.2
= A4.1
```

Flip-Flops

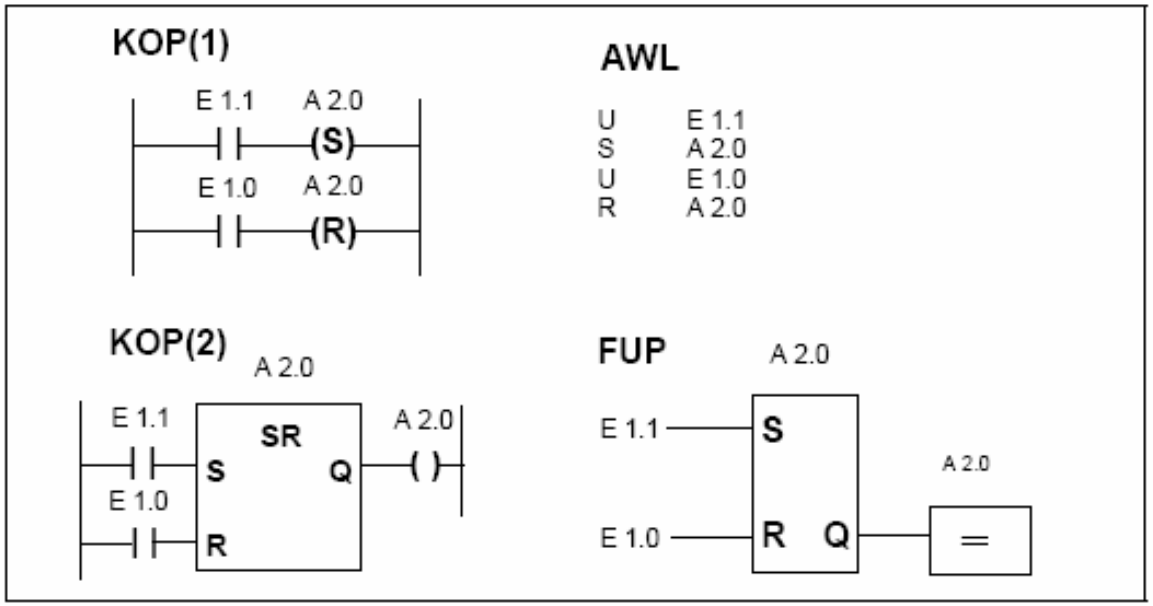
Netzwerk 1 : Flipflop-Baustein, Reset=E1.1 ist dominant

AWL zur Übersicht:

```
U E1.0
S A 4.1
U E 1.1
R A 4.1
```

Netzwerk 2 : Programmiertes Flipflop, Reset=E1.1 ist dominant

Vorrangiges Rücksetzen (RS)

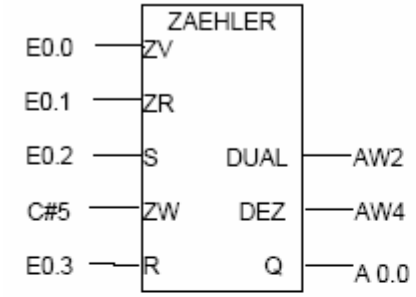


Die zuletzt programmierten Anweisungen werden von der Steuerung mit Vorrang bearbeitet. Im Beispiel wird zunächst die Setzoperation ausgeführt; der Ausgang A 2.0 wird wieder zurückgesetzt und bleibt für den Rest der Programmbearbeitung zurückgesetzt.

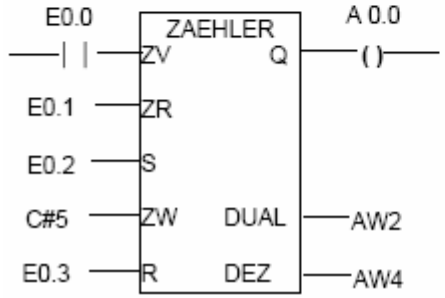
Dieses kurzzeitige Setzen des Ausgangs wird nur im Prozeßabbild durchgeführt. Der Signalzustand auf der dazugehörigen Peripheriebaugruppe wird während der Programmbearbei-tung nicht beeinflußt.

Zähler

FUP Z1

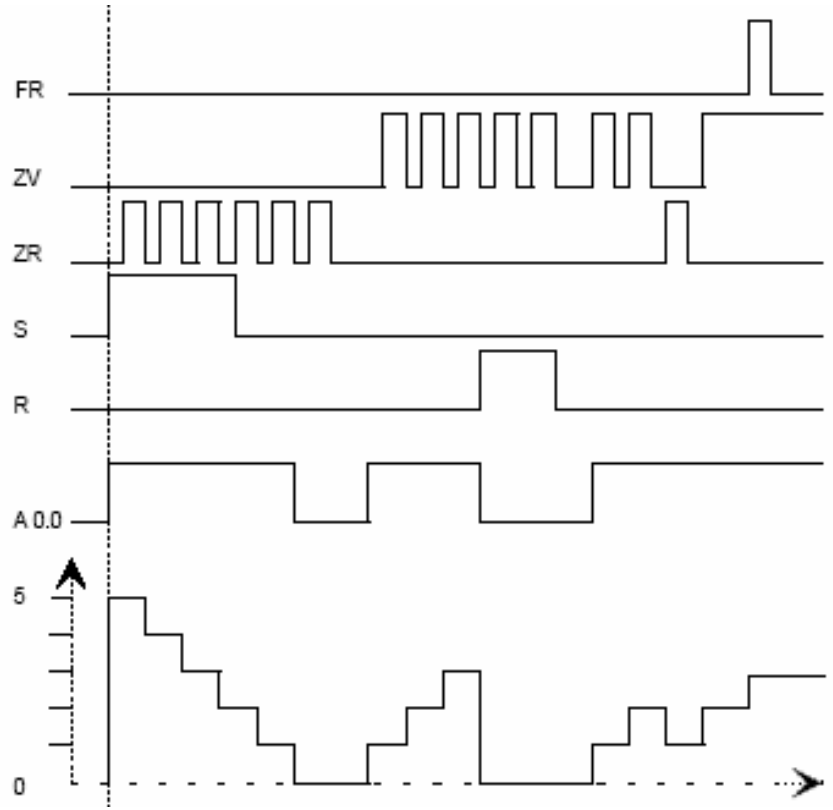


KOP Z1



AWL

```
U   E 0.7     Freigabe (nur in AWL)
FR  Z1
U   E 0.0
ZV  Z1        Vorwärts zählen
U   E 0.1
ZR  Z1        Rückwärts zählen
U   E 0.2
L   C#5       Vorgabewert für Zähler laden
S   Z1        Zähler mit Vorgabewert setzen
U   E 0.3
R   Z1        Setze Zähler Z1 zurück
L   Z1        Lade Zähler Z1 binär-codiert
T   AW2
LC  Z1        Lade Zähler Z1 BCD-codiert
T   AW4
U   Z1        Abfrage des Zählers Z1
=   A 0.0
```



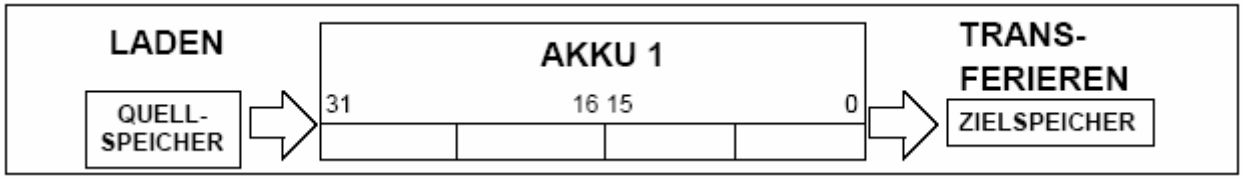
Lade-/Transferbefehle (nur AWL)

Mit dem Transferbefehl kann man Daten zwischen Merkern, E/A-Gruppen, Datenbausteinen, Registern und Prozessabbildern austauschen. Ein Transfer kann nie direkt erfolgen, sondern nur über den Akku1, wobei der Inhalt von Akku1 in Akku2 gesichert wird und der zu transferierende Wert nach dem Transfer im Akku1 zur weiteren Verfügung steht. Laden und Transferieren sind *unbedingte* Operationen die unabhängig vom VKE in jedem Zyklus ausgeführt werden!

L EW 0

T AW 3

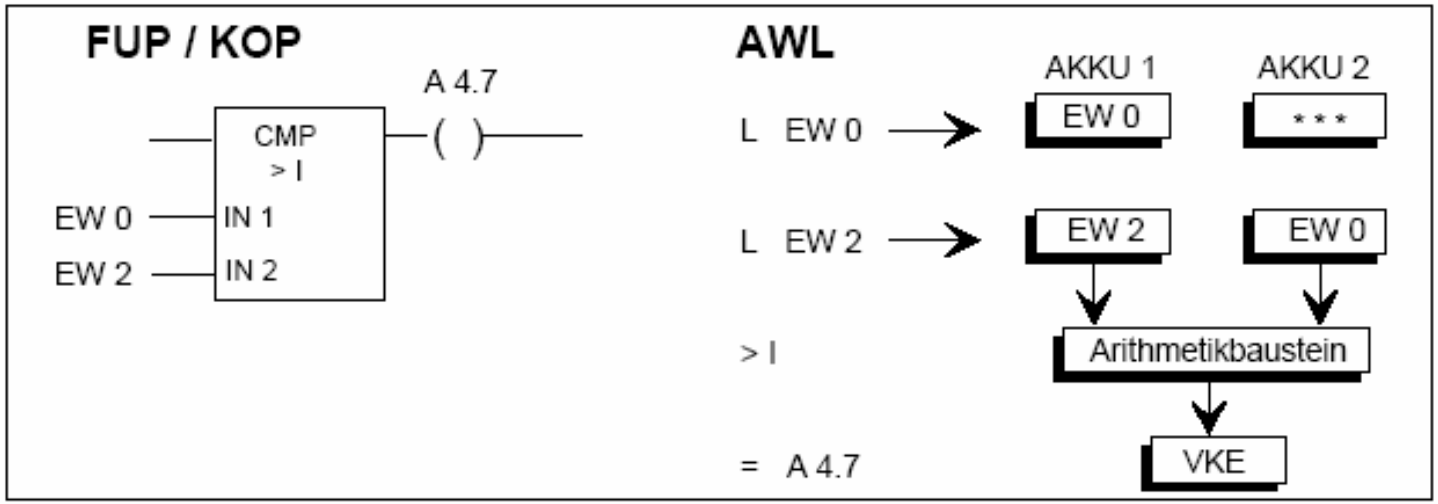
T AW 7



Vergleichsoperator

Ein Vergleich kann auf Ganz- oder Real-Zahlen gleichen Zahlenformats angewandt werden.

Das Ergebnis der Operatoren **<, >, ==, <=, >=, <>** ist dabei das VKE. Verglichen werden können nur Zahlen im Akku, daher müssen die Zahlen zuvor geladen werden.

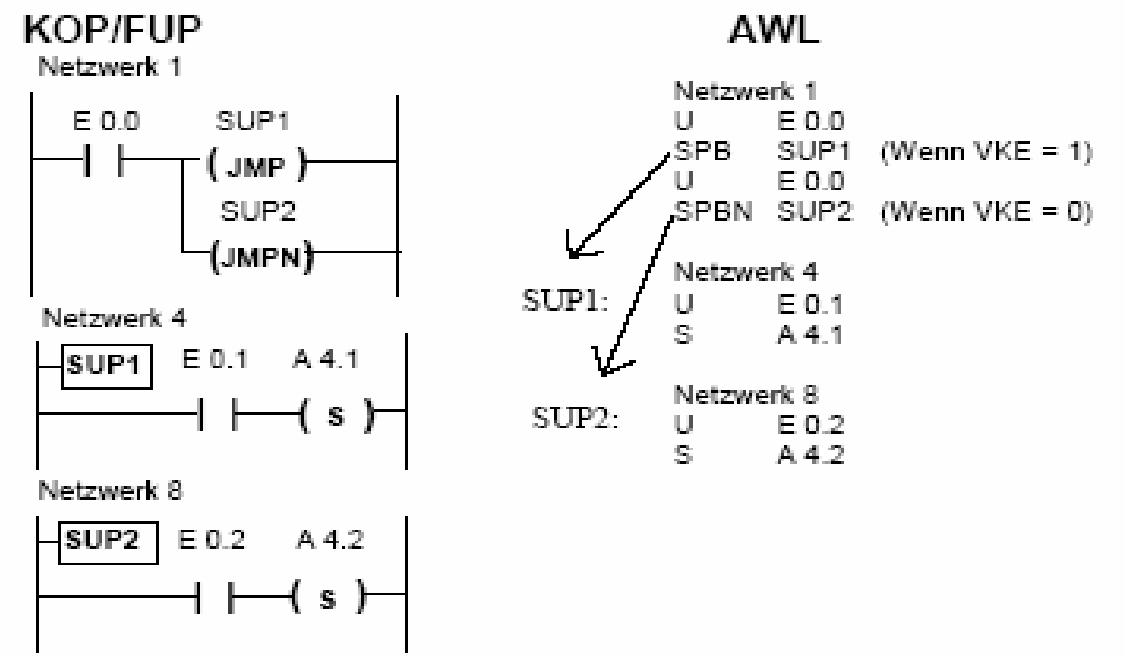


Sprungbefehle

Ein Sprung kann unbedingt oder bedingt erfolgen und bewirkt bei Ausführung, dass die Programmabarbeitung an der als Operand angegeben Markierung fortgesetzt wird.

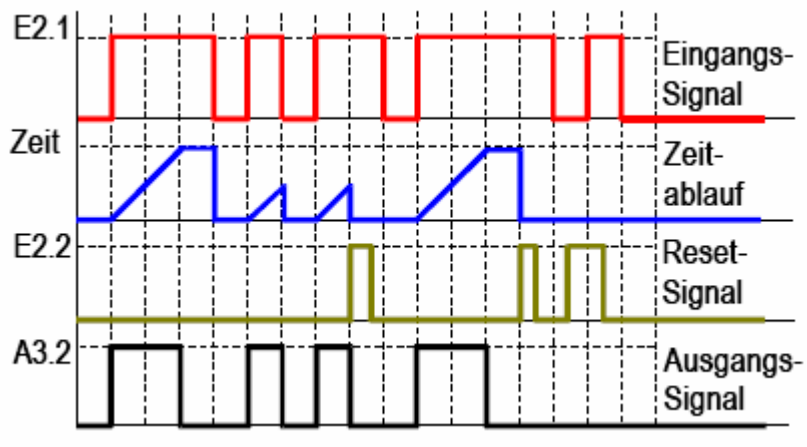
SPA unbedingter Sprung **SPB** bedingter Sprung bei VKE = 1

SPBN bei VKE = 0



Zeit als Impuls (SI)

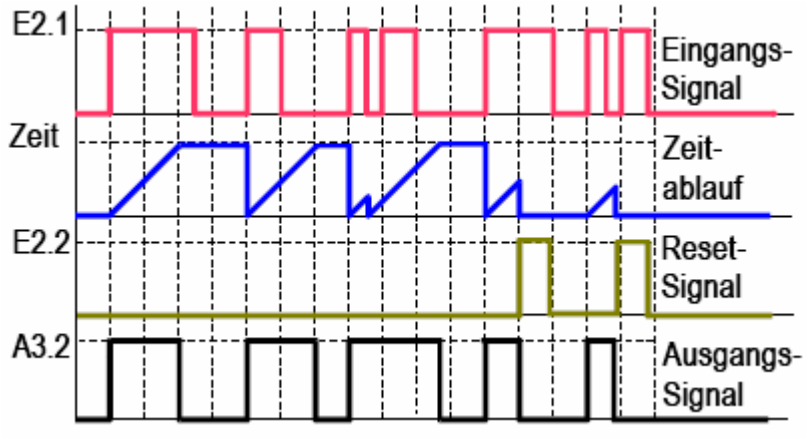
```
U E2.1
L S5T#2s
SI T1
U E2.2
R T1
U T1
= A3.2
```



SI startet einen Impuls fester Länge sobald das Eingangssignal auf H (High) wechselt. Der Timerausgang wird zurückgesetzt wenn das Eingangssignal auf L (Low) wechselt, die definierte Dauer abgelaufen ist oder das definierte Reset-Signal auf H liegt. Der Timerwert wird zurückgesetzt, wenn der Starteingang auf L wechselt oder ein Reset eintritt. Man kann diese Zeitfunktion hervorragend zum Entprellen nutzen.

Zeit als verlängerter Impuls (SV)

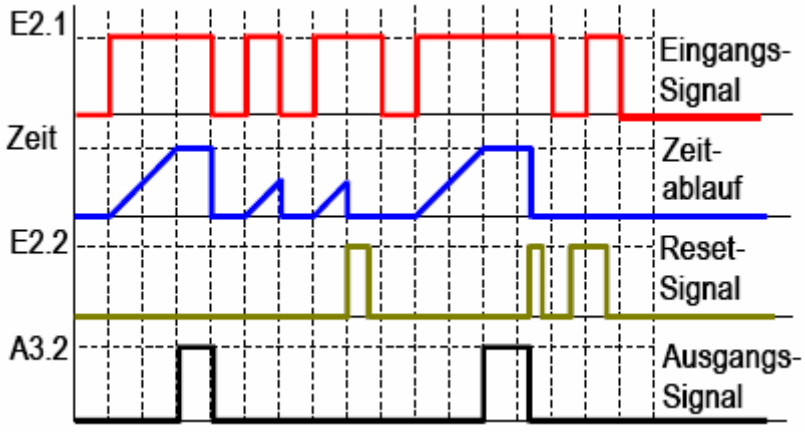
```
U E2.1
L S5T#2s
SV T1
U E2.2
R T1
U T1
= A3.2
```



Bei SV startet der Timer mit Änderung des Starteingangs auf H und hält den Timerausgang solange auf H bis die Dauer abgelaufen oder ein Reset aufgetreten ist. Ausschalten des Starteingangs hat keine Auswirkung (Selbsthaltung nach einmaligem Triggern). Der Timerwert wächst bzw. bleibt auf H solange nicht neu getriggert wird oder ein Reset eintritt. Man kann diese Funktion also zum Erzeugen längerer Impulse aus kurzen Ereignissen nutzen.

Zeit als Einschaltverzögerung (SE)

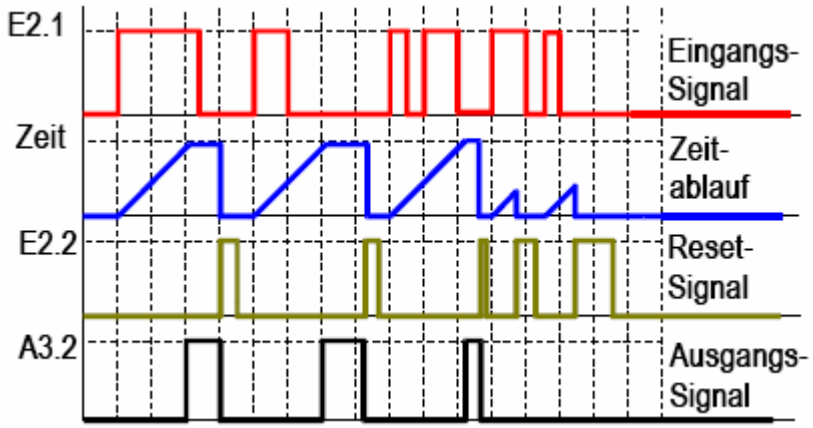
```
U E2.1
L S5T#2s
SE T1
U E2.2
R T1
U T1
= A3.2
```



Bei SE wechselt der Timerausgang erst auf H wenn der Starteingang nach Wechsel von L zu H für die gewählte Dauer durchweg auf H geblieben ist und wechselt zurück zu L bei Wechsel des Eingangs zu L oder Eintreten eines Reset. Diese Funktion kann wie ein Tiefpass zur Dämpfung benutzt werden.

Zeit als speichernde Einschaltverzögerung (SS)

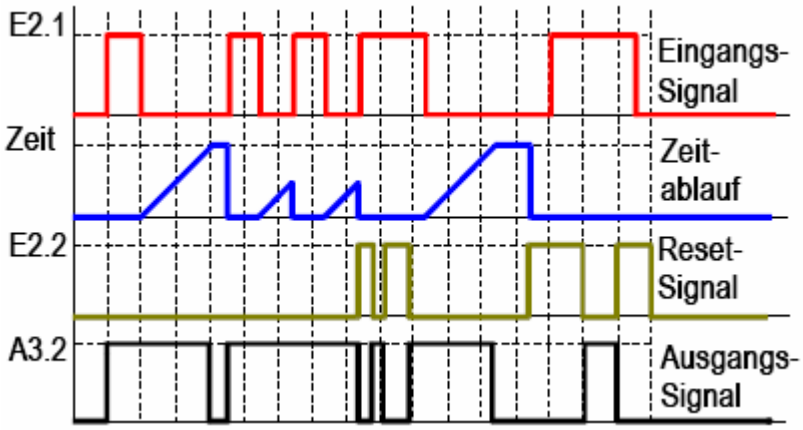
```
U E2.1
L S5T#2s
SS T1
U E2.2
R T1
U T1
= A3.2
```



Bei SS wechselt der Timerausgang mit der definierten Zeitverzögerung nach einem Triggern auf H. Er wird nur zurückgesetzt, wenn ein Reset eintrifft (in dem Fall bleibt er zunächst L) oder neu getriggert wird (in dem Fall wechselt er nach der def. Zeit wieder auf H).

Zeit als Ausschaltverzögerung (SA)

```
U E2.1
L S5T#2s
SA T1
U E2.2
R T1
U T1
= A3.2
```



Bei SA setzt ein H Eingangssignal den Timerausgang auf H. Fällt das Eingangssignal auf L, so folgt ihm der Timerausgang erst mit der vordefinierten Zeitverzögerung. Ein Reset setzt den Timerausgang für seine Dauer auf L, danach kann dieser (H am Eingang vorausgesetzt) jedoch sofort wieder auf H wechseln. Mehrfaches Wechseln des Eingangssignals triggert den Timerausgang neu. Funktion vergleichbar mit Anspringen des Bildschirmschoners.