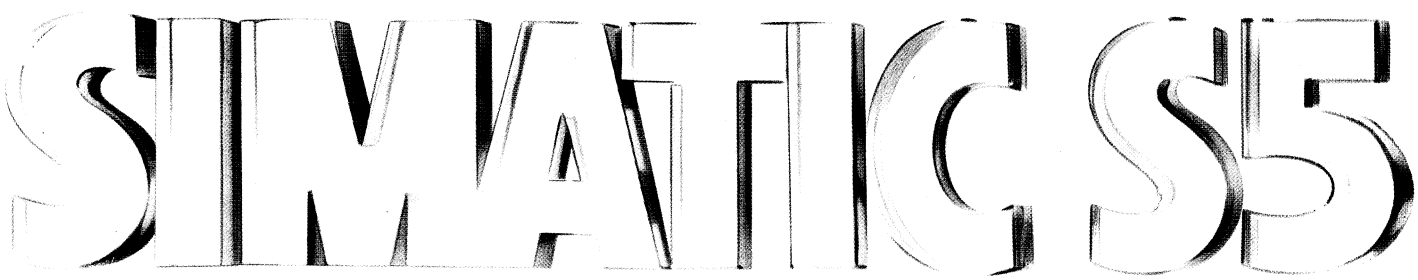
SIEMENS



Automatisierungsgerät SIMATIC S5-110 S/B

Programmieranleitung

SIEMENS

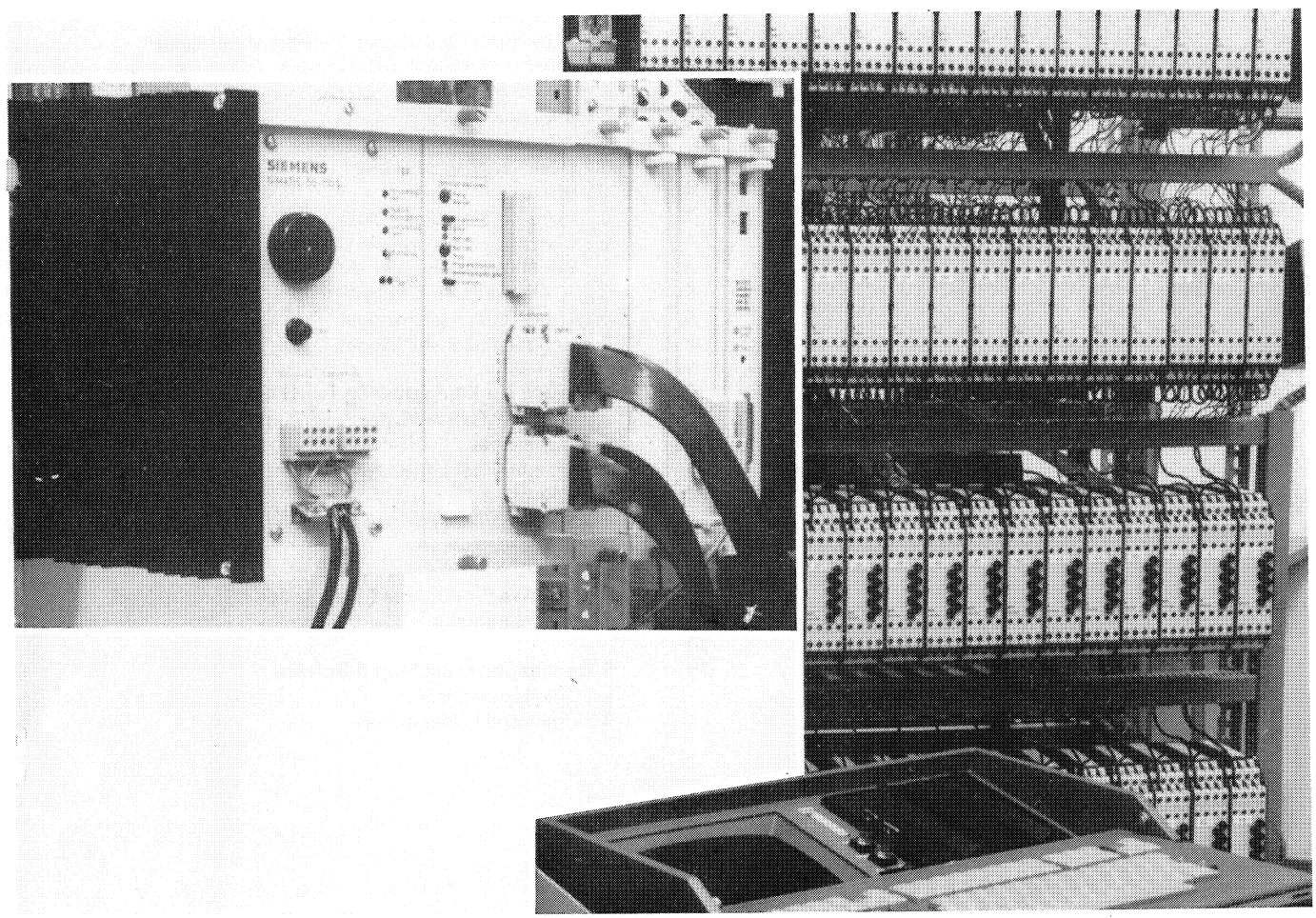
SIMATIC S5

Automatisierungsgeräte S5–110S/B

6ES5–110

Programmieranleitung

Bestell-Nr.: GWA 4NEB 807 2122-01



Automatisierungsgerät S5–110S: links Zentralgerät, rechts Peripherie und im Vordergrund Programmiergerät 670

Inhalt Seite

1. Erläuterungen

1.1 Anwendungsbereich

Das speicherprogrammierbare Automatisierungsgerät S5–110S ist ein leistungsfähiges Gerät zur Prozeßautomatisierung (Steuern, Melden, Überwachen, Protokollieren). Es ist sowohl für den Aufbau einfachster Steuerungen mit binären Signalen als auch zur Lösung umfangreicher Automatisierungsaufgaben einsetzbar. Seine Anwenderprogramme werden mit der Programmiersprache STEP-5 erstellt.

1.2 Programmiersprache STEP-5

Die Operationen der Programmiersprache STEP-5 ermöglichen die Programmierung von einfachen binären Funktionen bis hin zu komplexen digitalen Funktionen.
Bei der Programmierung sind, abhängig von dem verwendeten Programmiergerät, die drei Darstellungsarten Funktionsplan (FUP), Kontaktplan (KOP) und Anweisungsliste (AWL) (Bild 1) möglich, so daß die Programmiermethode dem jeweiligen Anwendungsfall angepaßt werden kann. Der von den Programmiergeräten erzeugte Maschinencode ist bei den drei Darstellungsarten identisch. Bei Berücksichtigung bestimmter Programmierregeln (siehe „Regeln zur Kompatibilität zwischen den Darstellungsarten KOP – FUP – AWL" Seite 42) kann das Programmiergerät (PG) 670/675 das Anweisungsprogramm von einer Darstellungsart in die jeweils andere übersetzen.

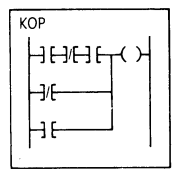
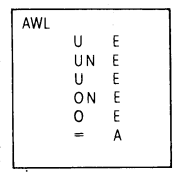
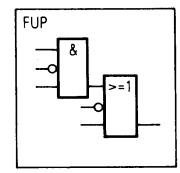
Kontaktplan	Anweisungsliste	Funktionsplan
Programmieren mit grafischen Symbolen wie Stromlaufbahn entspricht DIN 19239 (Entwurf)	Programmieren mit mnemotechnischen Abkürzungen der Funktionsbezeichnungen entspricht DIN 19239 (Entwurf)	Programmieren mit graf. Symbolen entspricht IEC 117-15 DIN 40700 DIN 40719 DIN 19239 (Entw.)
		

Bild 1 Darstellungsarten der Programmiersprache STEP 5

1.3 Programmierung

1.3.1 Programmstruktur

Das Automatisierungsgerät S5–110S zwingt aufgrund seiner Struktur den Anwender zur strukturierten Programmierung, d. h. zum Aufteilen des Gesamtprogramms in einzelne, in sich abgeschlossene Programmabschnitte (Bausteine). Dieses Verfahren bietet dem Anwender folgende Vorteile:

einfache und übersichtliche Programmierung auch großer Programme,

Möglichkeit zum Standardisieren von Programmteilen,

einfache Programmorganisation,

leichte Änderungsmöglichkeit,

einfacher Programmtest,

einfache Inbetriebnahme.

Für die Gliederung des Anwenderprogramms gibt es verschiedene Bausteintypen, die für unterschiedliche Aufgaben verwendet werden:

Programmbausteine (PB)

Sie werden zur Strukturierung des Anwenderprogramms in technologisch orientierte Programmteile eingesetzt.

Funktionsbausteine (FB)

Sie dienen zum Programmieren von häufig wiederkehrenden komplexen Funktionen (z. B. Einzelsteuerung, Melde-, Rechenfunktionen) sowie der Bearbeitung von Ablaufketten.

Datenbausteine (DB)

Sie dienen zum Abspeichern von Daten und Texten.

Die Anzahl der programmierbaren Bausteine beläuft sich auf maximal
– 128 Programmbausteine
– 48 Funktionsbausteine
– 63 Datenbausteine (ohne DB 0)
Jeder Baustein sollte 256 Anweisungen nicht überschreiten.
Alle programmierten Bausteine werden vom Programmiergerät in beliebiger Reihenfolge im Programmspeicher hinterlegt (Bild 2). Mit einem ebenfalls vom Anwender zu programmierenden Organisationsbaustein wird dann die Reihenfolge, in der die Bausteine bearbeitet werden sollen, festgelegt.

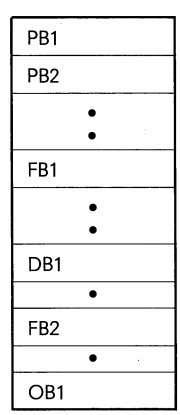


Bild 2 Ablage der Bausteine in beliebiger Reihenfolge in den Programmspeicher

1.3.2 Programmorganisation

Mit der Programmorganisation wird festgelegt, ob und in welcher Reihenfolge die Programm- und Funktionsbausteine bearbeitet werden (Bild 3). Dazu werden im Organisationsbaustein 1 (OB 1) entsprechende Aufrufe (bedingt oder unbedingt) der gewünschten Bausteine programmiert (siehe „Organisatorische Aufgaben" Seite 11).

Der Organisationsbaustein 1 wird wie die anderen Bausteine ebenfalls im Programmspeicher hinterlegt.

Eine Sonderrolle kommt den Funktionsbaustein 0 zu. Er dient der Alarmreaktion durch den Anwender (siehe Seite 15).

Von Programm- und Funktionsbausteinen können weitere Programm- und Funktionsbausteine in beliebigen Kombinationen aufgerufen werden. Die maximal zulässige Schachtelungstiefe beträgt mit dem Organisationsbaustein 8 Bausteine (Bild 4), wobei ein verwendeter Datenbaustein nicht mitgezählt wird.

OB Organisationsbaustein
PB Programmbaustein
FB Funktionsbaustein
DB Datenbaustein

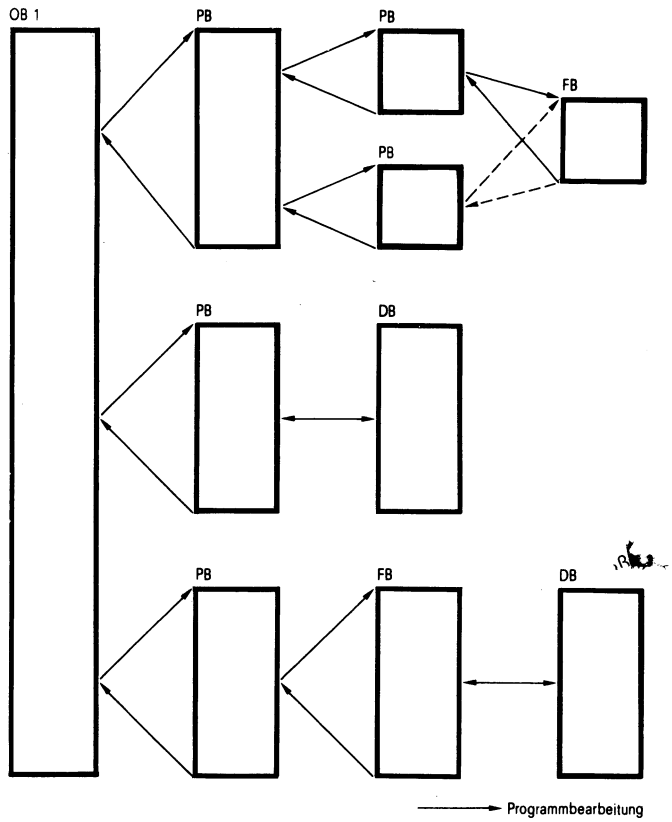


Bild 3 Programmorganisation in der Programmiersprache STEP 5

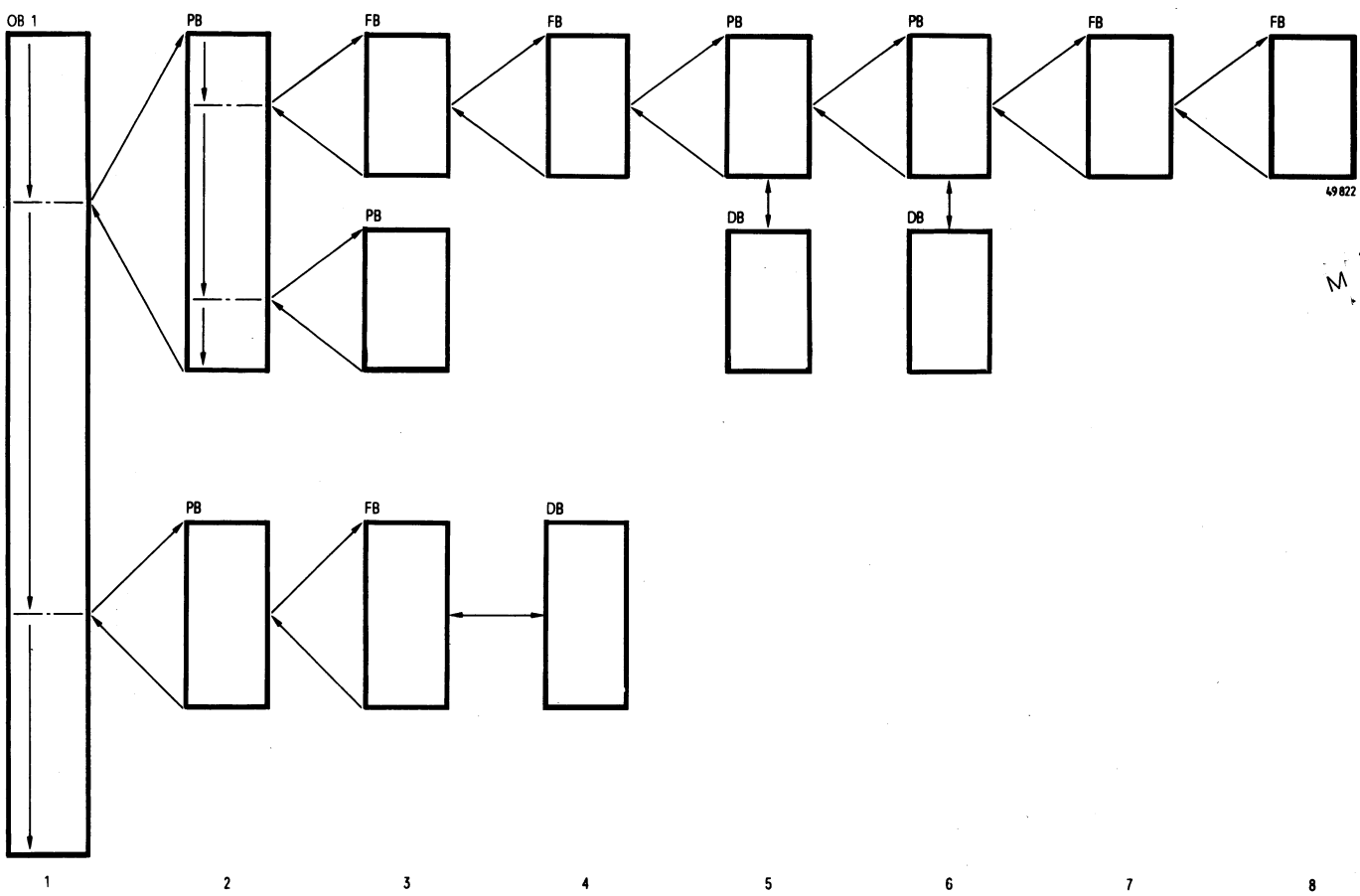


Bild 4 Beispiel für Programmstruktur bei Ausnutzung der maximalen Schachtelungstiefe

1.3.3 Programmbearbeitung

Das Anwenderprogramm kann auf drei verschiedene Arten bearbeitet werden (Bild 5):

Zyklische Programmbearbeitung

Für eine zyklische Programmbearbeitung des Anwenderprogramms ist der Organisationsbaustein 1 vorgesehen. Dieser Baustein wird zyklisch durchlaufen und ruft dabei die dort programmierten Bausteine auf. Die maximale Zykluszeit kann 270 ms betragen, danach spricht die Zykluszeitüberwachung an.

Alarmgesteuerte Programmbearbeitung

Bei dieser Art der Programmbearbeitung wird die zyklische Programmbearbeitung, abhängig von maximal 32 Eingangssignalen, jeweils bei einem Bausteinwechsel unterbrochen. Zum Aufruf des Alarmprogramms ist der Funktionsbaustein 0 (FB 0) vorgesehen.

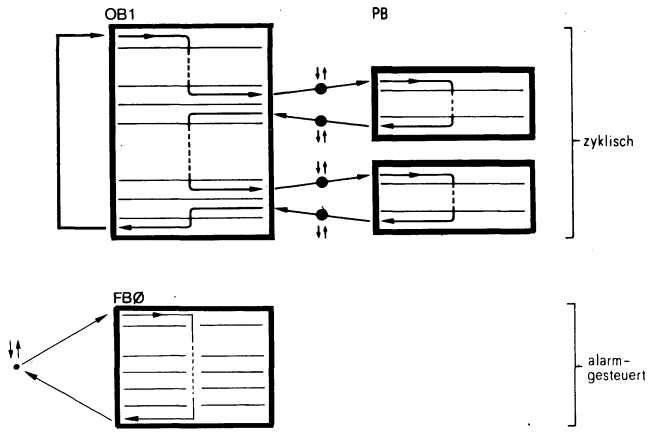
Zeitgesteuerte Programmbearbeitung

Bei dieser Art der Programmbearbeitung werden bestimmte Programm- oder Funktionsbausteine in einem wählbaren Zeitraster in die zyklische Programmbearbeitung eingeschoben. Durch diese Programmiermöglichkeit kann die mittlere Zykluszeit für ein Anwenderprogramm gesenkt werden.

Der Aufruf zeitgesteuerter Bausteine wird nicht wie bei AG 150 A/K automatisch, sondern vom Anwender frei wählbar durchgeführt.

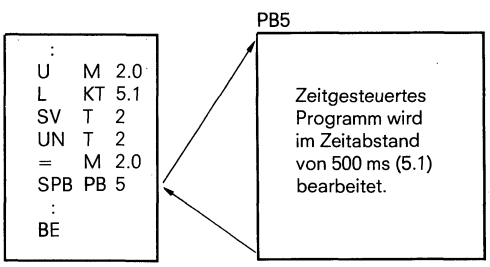
Hinweis:

Bei Programmieren der Zeitbasis 10 ms ist darauf zuachten, daß die Bearbeitungszeit jedes einzelnen Bausteins < 10 ms ist, da die Zeiten nur an den Bausteingrenzen bearbeitet werden.



⇅ Unterbrechungspunkt, an dem eine alarmgesteuerte oder zeitgesteuerte
● Programmbearbeitung eingeschoben wird.

Bild 5 Arten der Programmbearbeitung



Beispiel einer zeitgesteuerten Programmbearbeitung

1.4 Allgemeine Hinweise

Werden Standardfunktionsbausteine eingesetzt, so werden die Merkerbytes 200 bis 255 belegt und sind für den Anwender nicht mehr verwendbar.

Ebenso ist die Zeit 0, der Zähler 0 und der Datenbaustein 0 bereits belegt.

Wird das Servicegerät 333 C eingesetzt, so ist der Funktionsbaustein 1 und der Datenbaustein 1 belegt. Für die Anschaltung AS 512 werden die Funktionsbausteine 2, 3, 4, 5, 6 und der Datenbaustein 2 belegt. Die Datenwörter 0 der Datenbausteine müssen freigehalten werden.

Standard-Funktionsbausteine belegen die Nummern 1 bis 12 Anwender-Funktionsbausteine sind daher nur mit den Nummern 13 bis 47 zu erstellen.

Programmierbausteine können in den 3 Darstellungsarten (AWL, KOP, FUP) mit dem „Grundoperationsvorrat" der Programmiersprache STEP 5 programmiert werden.

Achtung!

Die Programmierung von Datenworten, die außerhalb des zugehörigen Bausteins liegen, können zu einem undefinierten Verhalten des Gerätes führen (Datenwort-Nr. > Datenbaustein-Länge).

Folgende Programmierfehler führen das Automatisierungsgerät in den Stop-Zustand:

1. Programmierung eines STEP-5-Befehles der nicht zum Sprachumfang des AG 110 S gehört.
2. Programmieren eines Aufrufs eines nicht zulässigen Bausteins (Nr. zu groß).
3. Anwahl eines Datenwortes (LDW,TDW) ohne Vorwahl eines Datenbausteins (ADB).
4. Überschreiten der Bausteinschachtelungstiefe 8.

2. Programmbausteine

2.1 Programmierung von Programmbausteinen

Die Programmierung eines Programmbausteins (PB) beginnt mit der Angabe einer Programmbaustein-Nummer zwischen 0 und 127 (Beispiel: PB 25). Danach folgt das eigentliche Steuerungsprogramm, das mit der Anweisung „BE" abgeschlossen wird.

Ein Programmbaustein soll einschließlich „BE" aus maximal 256 Anweisungen bestehen (Bild 6). Der Bausteinkopf, den das Programmiergerät automatisch zum Programmbaustein generiert, belegt weitere 5 Wörter im Programmspeicher.

Ein Programmbaustein soll immer ein abgeschlossenes Programm beinhalten. Verknüpfungen über Bausteingrenzen hinweg sind nicht sinnvoll.

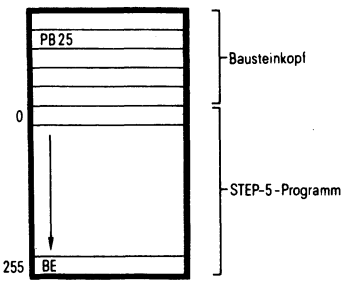


Bild 6 Aufbau eines Programmbausteins

2.2 Aufruf von Programmbausteinen

Durch Bausteinaufrufe werden die Programmbausteine zum Bearbeiten freigegeben (Bild 7). Diese Bausteinaufrufe können innerhalb eines Organisations-, Programm- oder Funktionsbausteins programmiert werden. Sie sind vergleichbar mit „Sprünge in ein Unterprogramm" und können sowohl unbedingt (SPA PBx) als auch bedingt (SPB PBx) ausgeführt werden.

Nach der Anweisung „BE" wird in den Baustein zurückgesprungen, in dem der Bausteinaufruf programmiert wurde. Sowohl nach einem Bausteinaufruf, als auch nach „BE" kann das Verknüpfungsergebnis nicht mehr weiter verknüpft werden. Das Verknüpfungsergebnis wird jedoch in den „neuen Baustein" mitgenommen und kann ausgewertet werden (siehe Alarmbearbeitung S. 15).

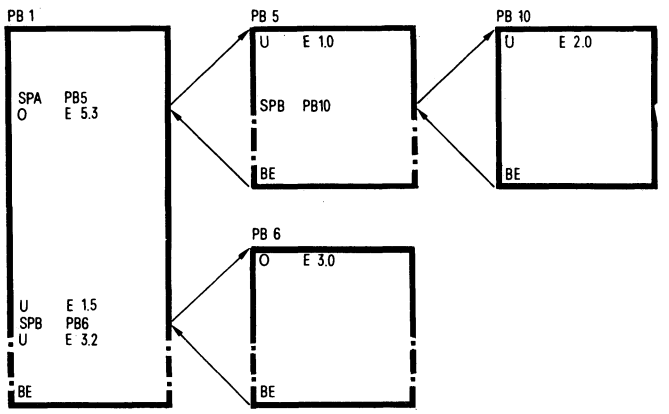


Bild 7 Bausteinaufrufe, die die Bearbeitung eines Programmbausteins freigeben

Unbedingter Aufruf:

Der angesprochene Programmbaustein wird unabhängig vom vorherigen Verknüpfungsergebnis bearbeitet.

Bedingter Aufruf:

Der angesprochene Programmbaustein wird abhängig vom vorherigen Verknüpfungsergebnis bearbeitet.

3. Datenbausteine

3.1 Programmierung von Datenbausteinen

In Datenbausteinen (DB) werden die Daten abgespeichert, die innerhalb des Anwenderprogramms erforderlich sind.

Daten können sein:

beliebige Bitmuster; z. B. für Anlagenzustände

Zahlen (Hexa, Dual, Dezimal); z. B. für Zeitwerk, Rechenergebnisse

alphanumerische Zeichen; z. B. für Meldetexte

Datenbausteine sind wie Programmbausteine aufgebaut. Die Programmierung beginnt mit der Angabe einer Datenbaustein-Nummer zwischen 1 und 63 (Beispiel: DB 25). Jeder Datenbaustein kann aus bis zu 256 Datenwörtern (16 Bit) bestehen (Bild 8). Das Eingeben von Daten muß in aufsteigender Reihenfolge der Datenwörter, beginnend mit Datenwort 0, erfolgen, wobei das Datenwort 0 (DW 0) vom Anwender nicht verwendet werden sollte, da es als Zwischenspeicher für bestimmte Funktionsbausteine vorgesehen ist.

Pro Datenwort wird im Programmspeicher ein Speicherwort belegt. Vom Programmiergerät wird außerdem zu jedem Datenbaustein ein Bausteinkopf generiert, der weitere 5 Wörter im Programmspeicher belegt.

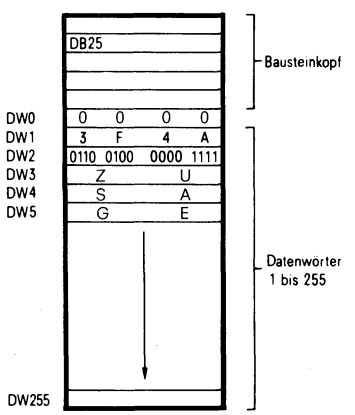


Bild 8 Aufbau eines Datenbausteins

3.2 Aufruf von Datenbausteinen

Datenbausteine (DB) können nur unbedingt aufgerufen werden. Der Aufruf bleibt im Baustein solange gültig, bis ein neuer Aufruf erfolgt.

Der Aufruf eines Datenbausteins kann innerhalb eines Programmbausteins oder eines Funktionsbausteins programmiert werden.

Beispiel

Es soll der Inhalt des Datenwortes 1 vom Datenbaustein 10 in das Datenwort 1 des Datenbausteins 20 transferiert werden (Bild 9).

```
:A   DB 10
:L   DW1
:A   DB 20
:T   DW1
```

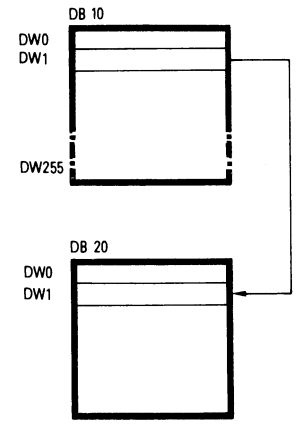


Bild 9 Aufruf eines Datenbausteins

Wird von einem Programmbaustein, in dem bereits ein Datenbaustein adressiert wurde, ein weiterer Programmbaustein aufgerufen und in diesem Baustein ein anderer Datenbaustein adressiert, so ist dieser Datenbaustein nur in dem aufgerufenen Programmbaustein gültig. Nach dem Rücksprung in den aufrufenden Programmbaustein gilt wieder der alte Datenbaustein (Bild 10).

Beispiel

Im Programmbaustein 7 wird der Datenbaustein 10 aufgerufen. In der folgenden Bearbeitung werden die Daten dieses Datenbausteins bearbeitet.

Nach dem Aufruf wird der Programmbaustein 20 bearbeitet. Der Datenbaustein 10 ist jedoch nach wie vor gültig. Erst mit dem Aufruf von Datenbaustein 11 wird der Datenbereich gewechselt. Bis zum Ende von Programmbaustein 20 ist nun Datenbaustein 11 gültig.

Nach dem Bausteinwechsel zurück in Programmbaustein 7 ist wieder der Datenbaustein 10 gültig.

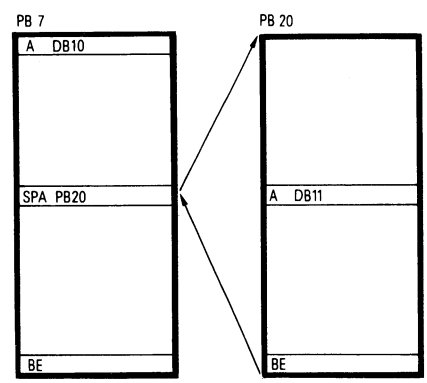


Bild 10 Aufruf eines Datenbausteins innerhalb eines anderen Datenbausteins

Achtung!

Wird ein Datenlade- oder Transferbefehl mit einer Datenwort-Nummer > Bausteinlänge programmiert, so kann das Automatisierungsgerät in einen undefinierten Zustand geraten!

Beispiel

Es wird der Datenbaustein 10 mit 10 Datenworten programmiert. Durch den Befehl TDW 10 wird ein Speicherwort **außerhalb des Bausteins überschrieben!** (Bild 11).

Die Bearbeitung eines Datenlade- oder Transferbefehls, **dem kein Aufruf** eines Datenbausteins **vorangegangen** ist, führt das Automatisierungsgerät in den Stopzustand.

```
:A   DB  10
:L   KH  10
:T   DW 10
```

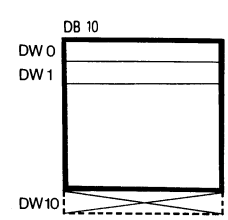


Bild 11 **Fataler** Programmierfehler

4. Funktionsbausteine

4.1 Allgemeines

Funktionsbausteine sind ebenso Teile des Anwenderprogramms wie z. B. Programmbausteine. Sie weisen jedoch gegenüber Programmbausteinen vier wesentliche Unterschiede auf:

Funktionsbausteine lassen sich parametrieren, d. h. es können die Aktualoparanden, mit denen ein Funktionsbaustein arbeiten soll, vorgegeben werden.

Funktionsbausteine können mit einem gegenüber den Programmbausteinen erweiterten Operationsvorrat programmiert werden.

Das Programm eines Funktionsbausteines läßt sich nur als Anweisungsliste erstellen und dokumentieren.

Der Aufruf eines Funktionsbausteins wird graphisch als „schwarzer Kasten" dargestellt.

Funktionsbausteine stellen innerhalb des Anwenderprogramms eine komplexe, abgeschlossene Funktion dar. Ein Funktionsbaustein kann entweder als Softwareprodukt von Siemens bezogen werden („Standard-Funktionsbausteine" auf Mini-Floppy-Disk) oder vom Anwender selbst programmiert werden. Die ergänzenden Operationen, die es zusätzlich zu den Grundoperationen (Seite 20) gibt, können nur in Funktionsbausteinen programmiert werden.

4.2 Aufbau

Ein Funktionsbaustein besteht aus dem Bausteinkopf und dem Bausteinrumpf (Bild 12).

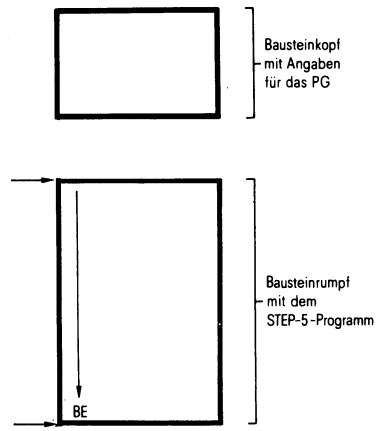


Bild 12 Aufbau eines Funktionsbausteins

Bausteinkopf

Der Bausteinkopf enthält alle Angaben, die das Programmiergerät benötigt, um den Funktionsbaustein graphisch darstellen zu können, und um die Operanden bei der Parametrierung des Funktionsbausteins prüfen zu können. Vor der Programmierung des Funktionsbausteins wird dieser Bausteinkopf (mit Unterstützung des Programmiergeräts) vom Anwender eingegeben (siehe „Erstellung eines Funktionsbausteins" nächste Seite).

Bausteinrumpf

Der Bausteinrumpf enthält das eigentliche Programm des Funktionsbausteins. Im Bausteinrumpf ist die auszuführende Funktion mit der Programmiersprache STEP 5 beschrieben und niedergelegt. Bei einem Aufruf des Funktionsbausteins wird nur der Bausteinrumpf bearbeitet. Zum Programmieren der Funktionsbausteine ist gegenüber den Programmbausteinen ein erweiterter Operationsvorrat vorhanden (siehe „Ergänzende Operationen" Seite 34).

4.3 Aufruf zur Parametrierung

Mit Funktionsbausteinen (FB) werden häufig wiederkehrende oder sehr komplexe Funktionen realisiert. Sie stehen nur einmal im Programmspeicher und werden von einem übergeordneten Baustein einmal oder mehrfach aufgerufen, wobei bei jedem Aufruf andere Parameter verwendet werden können.

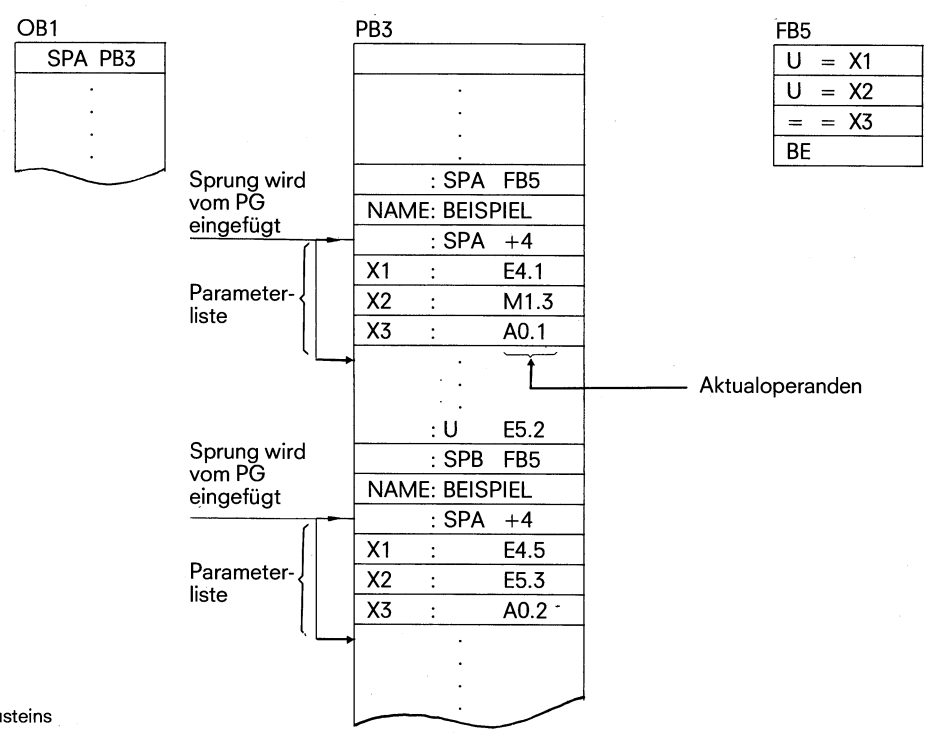


Bild 13 Parametrierung eines Funktionsbausteins

Die Funktionsbausteine werden wie die Programm- und Datenbausteine unter einer bestimmten Bezeichnung (FB 1 bis FB 47) im Programmspeicher hinterlegt. Anwender-Funktionsbausteine sollen von FB 47 abfallend adressiert werden, um nicht mit dem Standard-Funktionsbausteinen, die von FB 1 bis FB 12 adressiert sind, zu kollidieren.

Der Aufruf eines Funktionsbausteins kann innerhalb eines Programmbausteins oder eines anderen Funktionsbausteins programmiert werden. Der Aufruf setzt sich aus der Aufrufanweisung und der Parameterliste zusammen.

Aufrufanweisung

SPA FBn unbedingter Aufruf
SPB FBn bedingter Aufruf

Unbedingter Aufruf:

Der angesprochene Funktionsbaustein wird unabhängig vom vorherigen Verknüpfungsergebnis bearbeitet.

Bedingter Aufruf:

Der angesprochene Funktionsbaustein wird nur dann bearbeitet, wenn das vorherige Verknüpfungsergebnis „VKE" = 1 ist.

Parameterliste

Die Parameterliste steht direkt nach der Aufrufanweisung (Bild 13). In ihr werden die Eingangs-, Ausgangsvariablen und Daten definiert (siehe „Art der Bausteinparameter"). Die Parameterliste kann maximal **40 Variable** enthalten.

Bei der Bearbeitung des Funktionsbausteins werden anstelle der formalen Parameter die Varaiblen aus der Parameterliste verwendet. Die Reihenfolge der Variablen in der Parameterliste wird durch das Programmiergerät überwacht.

Die Sprunganweisung hinter dem FB-Aufruf wird vom Programmiergerät automatisch eingefügt, beim Auslesen jedoch nicht angezeigt. Der FB-Aufruf belegt im Programmspeicher zwei Wörter, jeder Parameter ein weiteres Speicherwort.
Die erforderliche Speicherlänge der Standard-Funktionsbausteine sowie die Laufzeit werden im Katalog ST 56 angegeben.

Die bei der Programmierung am Programmiergerät erscheinenden Bezeichner für die Ein- und Ausgänge des Funktionsbausteins, sowie der Name, sind im Funktionsbaustein selbst abgelegt. Deshalb müssen, bevor mit der Programmierung am Programmiergerät begonnen wird, alle erforderlichen Funktionsbausteine auf die Programmdiskette überspielt bzw. direkt in den Programmspeicher des Automatisierungsgerätes eingegeben werden.

4.4 Erstellung von Funktionsbausteinen

Entsprechend dem Aufbau eines Funktionsbausteins gliedert sich die Erstellung in zwei Teile:

Eingabe des Bausteinkopfes und Eingabe des Bausteinrumpfes.

Vor der Eingabe des Bausteinrumpfes (STEP-5-Programm) wird der Bausteinkopf eingegeben. Der Bausteinkopf enthält:

die Bibliotheksnummer,

den Namen des Funktionsbausteins,

Formaloperanden (die Namen der Bausteinparameter),

die Art des Bausteinparameters,

den Typ des Bausteinparameters.

Bibliotheksnummer

Es kann eine Nummer von 0 bis 65535 vorgegeben werden. Der Funktionsbaustein erhält diese Nummer zugesprochen, unabhängig von seinem symbolischen oder absoluten Parameter.

Eine Bibliotheksnummer sollte nur einmal vorgegeben werden, um einen bestimmtem Funktionsbaustein eindeutig identifizieren zu können. Standard-Funktionsbausteine haben eine Produktnummer.

Name des Funktionsbausteins

Der Name, der den Funktionsbaustein bezeichnet, kann maximal 8 Zeichen lang sein. Er ist nicht mit dem symbolischen Anlagenkennzeichen identisch.

Formaloperand (Name des Bausteinparameters)

Der Formaloperand kann maximal 4 Zeichen lang sein und muß mit einem Buchstaben beginnen.

Es können pro Funktionsbaustein max. 40 Parameter programmiert werden.

Art der Bausteinparameter

Als Art des Bausteinparameters kann „E", „A", „D", „B", „T" oder „Z" eingegeben werden.

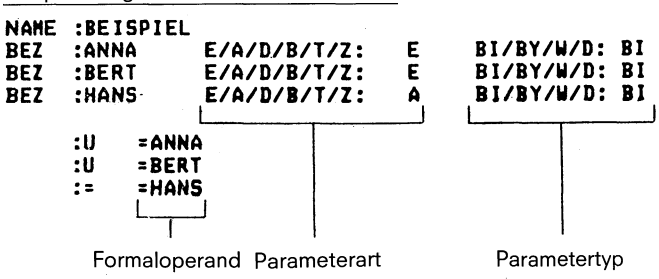
E = Eingangsparameter
A = Ausgangsparameter
D = Datum
B = Befehl
T = Zeit (Timer)
Z = Zähler

„E, D, B, T" oder „Z" sind Parameter, die bei graphischer Darstellung auf der linken Seite des Funktionssymbols gezeichnet werden. Mit „A" gekennzeichnete Parameter werden bei der graphischen Darstellung auf der rechten Seite des Funktionssymbols gezeichnet.

Parametrierung

Operationen (Subtitutionsbefehle), die parametrierbar sein sollen, werden im Funktionsbaustein mit dem Formaloperanden programmiert (formal). Dabei können die Formaloperanden auch mehrmals an verschiedenen Stellen im Funktionsbaustein angesprochen werden.

Beispiel: Programm im Funktionsbaustein



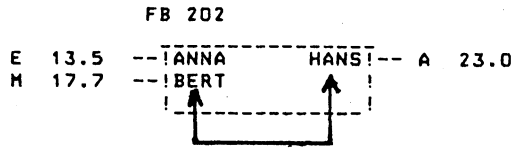
Aufruf des Funktionsbausteins

AWL

```
      :SPA FB202
NAME  :BEISPIEL
ANNA  :     E 13.5
BERT  :     M 17.7
HANS  :     A 23.0
```

Formeloperanden

KOP/FUP



Ausgeführtes Programm

```
:U   E 13.5
:U   M 17.7
:=   A 23.0
```

Aktualoperand

Art und Typ des Bausteinparameters mit zugelassenen Aktualoperanden

Art des Parameters	Typ des Parameters		Zugelassene Aktualoperanden	
E, A	BI	für einen Operanden mit Bitadresse	E n.m A n.m M n.m	Eingänge Ausgänge Merker
	BY	für einen Operanden mit Byteadresse	EB n AB n MB n DL n DR n PB n	Eingangsbytes Ausgangsbytes Merkerbytes Datenbytes links Datenbytes rechts Peripheriebytes
	W	für einen Operanden mit Wortadresse	EW n AW n MW n DW n PW n	Eingangswörter Ausgangswörter Merkerwörter Datenwörter Peripheriewörter
D	KM KY KH KC KT KZ KF	für ein Binärmuster (16 Stellen) für zwei byteweise Betragszahlen im Bereich jeweils von 0 bis 255 für ein Hexadezimalmuster (max. 4 Stellen) für ein Zeichen (max. 2 alphanumerische Zeichen) für einen Zeitwert (BCD-codierter Zeitwert) mit Zeitraster 1.0 bis 999.3 für einen Zählerwert (BCD-codiert) 0 bis 999 für eine Festpunktzahl im Bereich von −32768 bis +32767	Konstanten	
B	keine Typanzeige zulässig		DB n FB n PB n	Datenbausteine, ausgeführt wird der Befehl ADBn. Funktionsbausteine (nur ohne Parameter zulässig) werden unbedingt (SPA ..n) aufgerufen Programmbausteine werden unbedingt (SPA ..n) aufgerufen
T	keine Typangabe zulässig		T	Zeit; der Zeitwert ist als Datum zu parametrieren oder als Konstante im Funktionsbaustein zu programmieren.
Z	keine Typangabe zulässig		Z	Zähler; der Zählwert ist als Datum zu parametrieren od als Konstante im Funktionsbaustein zu programmieren.

4.5 Standard-Funktionsbausteine

Für die Verwendung der Standard-Funktionsbausteine aus dem Katalog ST 56 an dem AG 110S gelten folgende Einschränkungen:
Die Standard-FBs 30, 35 bzw. 36 für das Servicegerät 333 C sind an dem AG 110S nicht ablauffähig. Es muß weiterhin der speziell für das AG 110S bzw. AG 130W entwickelte Standard-Funktionsbaustein verwendet werden.

Die Standard-FBs für Ablaufsteuerungen (FB 70 – FB 75) können beim AG 110S nicht eingesetzt werden, da die Programmierung von Schrittbausteinen an diesem AG nicht möglich ist.
Bei den Meldefunktionen sind nur die FB 50 – FB 56 für die Meldung an die Prozeßperipherie einsetzbar. Die FB 64 – FB 69 für Meldungen an die Standardperipherie sind nicht verwendbar.
Die Versorgung der Standardschnittstelle (Anschaltung 512C) mit den Funktionsbausteinen FB 120 – FB 129 ist nicht möglich. Es müssen weiterhin die speziellen Schnittstellenbausteine für das AG 110S bzw. AG 130W verwendet werden.
Für die Regelung mit dem AG 110S wird ein gesondertes Softwarepaket erstellt.
Bei Verwendung von Standard-FBs ist darauf zu achten, daß bei Bausteinnummer >47 diese FBs mit einer anderen Bausteinnummer (≤47) geladen werden müssen, da an dem AG 110S nur max. 47 Funktionsbausteine verwendet werden können (FB1 – FB47).

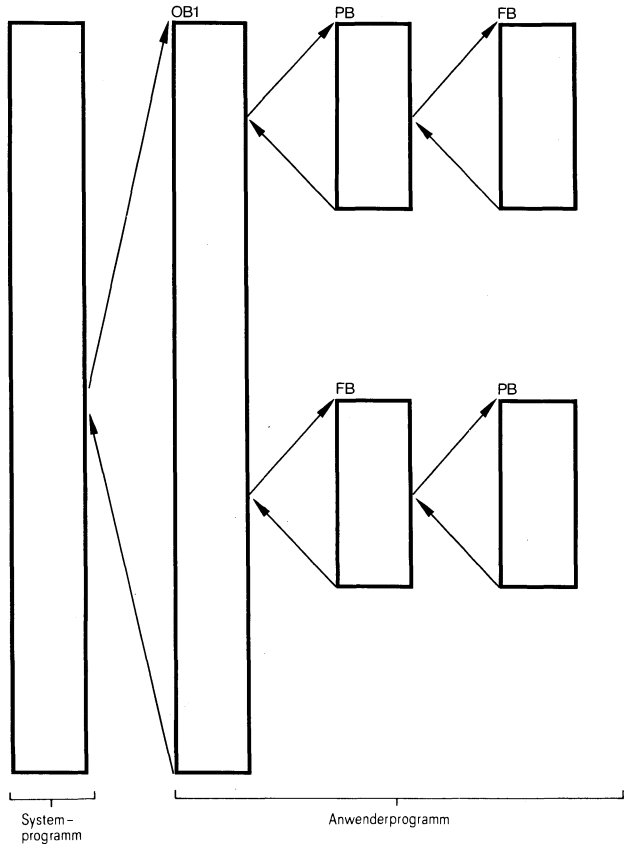
5. Organisatorische Aufgaben

5.1 Allgemeines

Das Gesamtprogramm eines Automatisierungsgeräts besteht aus dem Systemprogramm und dem Anwenderprogramm (Bild 14). Das Systemprogramm ist die Gesamtheit aller Anweisungen und Vereinbarungen geräteinterner Betriebsfunktionen (z. B. Sicherstellen von Daten bei Ausfall der Versorgungsspannung). Dieses Programm ist ein fester Bestandteil des Automatisierungsgerätes (EPROM) und darf vom Anwender nicht verändert werden. Das Anwenderprogramm ist die Gesamtheit aller vom Anwender programmierten Anweisungen und Vereinbarungen für die Signalverarbeitung, durch die eine zu steuernde Anlage (Prozeß) gemäß der Steuerungsaufgabe beeinflußt wird. Die Schnittstelle zwischen dem Systemprogramm und dem Anwenderprogramm ist der Organisationsbaustein 1.

Der Organisationsbaustein 1 (OB 1) ist Teil des Anwenderprogramms genauso wie Programmbausteine oder Funktionsbausteine. Der Organisationsbaustein 1 wird jedoch nur vom Systemprogramm aufgerufen. Als Anwender kann man den Organisationsbaustein 1 nicht aufrufen. Der OB 1 dient der zyklischen Bearbeitung des Anwenderprogramms.

Eine programmierte Reaktion des Anwenders auf Gerätefehler bzw. die Steuerung des Bearbeitungsmodus des Anwenderprogramms durch weitere Organisationsbausteine wie bei AG 150 A/K ist nicht möglich. Sie ist im System festgelegt.



OB1 Organisationsbaustein 1
PB Programmbaustein
FB Funktionsbaustein

Bild 14: Gesamtprogramm eines Automatisierungsgeräts

Bearbeitungsmodus des Anwenderprogramms

- Zyklische Bearbeitung durch Programmierung des OB 1 (siehe Seite 12).
- Alarmgesteuerte Bearbeitung durch Programmieren des FB 0 (siehe Seite 12 und 15).
- Zeitgesteuerte Bearbeitung direkt im Anwenderprogramm programmiert (siehe Seite 5).

Neustart- und Anlaufbetrieb

Festgelegt durch Bedienelemente an der Zentralbaugruppe (Betriebsschalter, Rücksetz-Taste).

- Neustart manuell (siehe Seite 18).
- Neustart manuell mit Rücksetzen (siehe Seite 18).
- Neustart automatisch (siehe Seite 18).

Gerätefehlerbehandlung

Gerätefehler führen das AG in den Stopzustand.

- Speicherfehler
- Batterieausfall bei Neustart
- Quittungsverzug
- Zykluszeitüberschreitung

Programmlaufbesonderheiten

Programmlaufbesonderheiten führen das AG in den Stopzustand.

- Nicht dekodierbarer Befehl (Seite 19).
- Nicht zulässiger Baustein (Seite 19).
- Nicht vorhandener Datenbaustein (Seite 19).
- Bausteinstacküberlauf (Seite 19).

5.2 Übersicht

Bearbeitungs-programm	Bezeichnung bzw. Bearbeitungsanstoß	Seite
OB für zyklische Bearbeitung		
OB 1	Programmanfang	12
FB für alarmgesteuerte Bearbeitung		
FB 0	Signalzustandswechsel an E 0.0 E 16.0 E 32.0 E 48.0 E 0.1 E 16.1 E 32.1 E 48.1 E 0.2 E 16.2 E 32.2 E 48.2 E 0.3 E 16.3 E 32.3 E 48.3 E 0.4 E 16.4 E 32.4 E 48.4 E 0.5 E 16.5 E 32.5 E 48.5 E 0.6 E 16.6 E 32.6 E 48.6 E 0.7 E 16.7 E 32.7 E 48.7	15
Zeitgesteuerte Bearbeitung		
Im Anwender-programm festgelegt	Zeitraster beliebig wählen	5

Bearbeitungs-programm	Bezeichnung bzw. Bearbeitungsanstoß	Seite
Neustart und Anlaufbetrieb		
Im Betriebs-system festgelegt	Neustart manuell Neustart manuell mit Rücksetzen Neustart automatisch nach Netzspannungs-ausfall	18
Behandlung von Gerätefehlern und Programmlaufbesonderheiten		
Betriebs-system	Speicherfehler Betriebsausfall bei Neustart Quittungsverzug Zykluszeitüberschreitung Nicht dekodierbarer Befehl Nicht zulässiger Baustein Nicht vorhandener Datenbaustein Bausteinstack-Überlauf	19

5.3 Programmierung der zyklischen Bearbeitung

Die zyklische Bearbeitung ist die „normale" Bearbeitung bei speicherprogrammierbaren Steuerungen (Bild 15). Der Prozessor beginnt mit der Programmbearbeitung am Anfang des STEP-5-Programms, arbeitet die STEP-5-Anweisungen der Reihe nach bis zum Ende des Programms ab und beginnt dann wieder mit der Bearbeitung am Programmanfang.

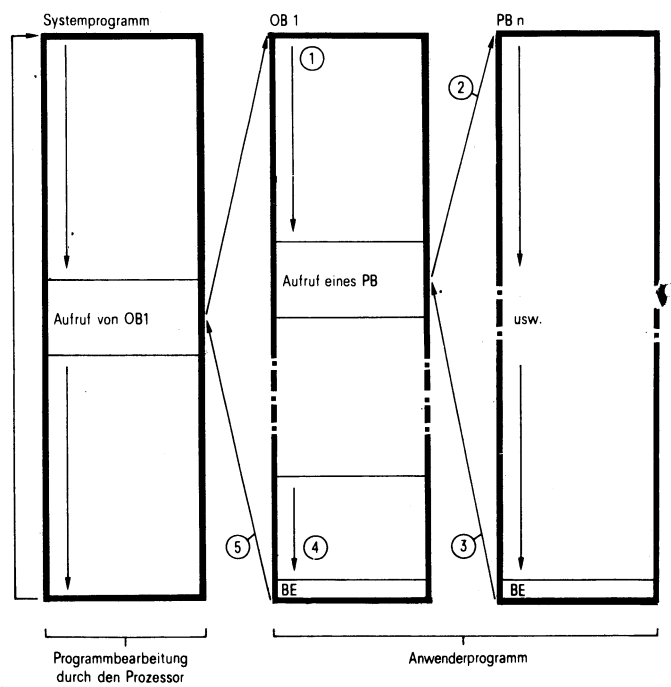
5.3.1 Schnittstelle zwischen Systemprogramm und zyklischer Bearbeitung

Der Organisationsbaustein 1 ist die Schnittstelle zwischen dem Systemprogramm und der zyklischen Bearbeitung des Anwenderprogramms. Die erste STEP-5-Anweisung im Organisationsbaustein 1 ist gleichzeitig die erste Anweisung des Anwenderprogramms, also gleichbedeutend mit dem Programmanfang.

Im Organisationsbaustein 1 werden die Programm- und Funktionsbausteine des zyklischen Programms aufgerufen. In diesen aufgerufenen Bausteinen können wieder Bausteinaufrufe stehen, d. h. die Bausteine können geschachtelt werden (siehe „1.3.2 Programmorganisation" Seite 4).

Die Laufzeit des Anwenderprogrammes ergibt sich aus der Summe aller Laufzeiten der aufgerufenen Bausteine. Wird ein Baustein „n" mal aufgerufen, so muß seine Laufzeit „n" mal bei der Summenbildung berücksichtigt werden.

Die Zykluszeit kann max. 270 ms betragen.



① Erste Anweisung des STEP-5-Programms.
② Erster Aufruf eines Programmbausteins. In diesem Baustein können auch weitere Aufrufe stehen (siehe „Programmorganisation" Seite 4).
③ Rücksprung vom letzten bearbeiteten Programm- oder Funktionsbaustein.
④ Der Organisationsbaustein wird mit BE abgeschlossen.
⑤ Rücksprung ins Systemprogramm.

Bild 15: Zyklische Programmbearbeitung

5.3.2 Grobgliederung des Programms

Im Organisationsbaustein OB 1 steht eine Grobgliederung des Anwenderprogramms. Die Dokumentation dieses Bausteins soll auf den ersten Blick die wesentlichen Programmstrukturen zeigen (Bild 16) bzw. programmtechnisch zusammenhängende Anlagenteile hervorheben (Bild 17).

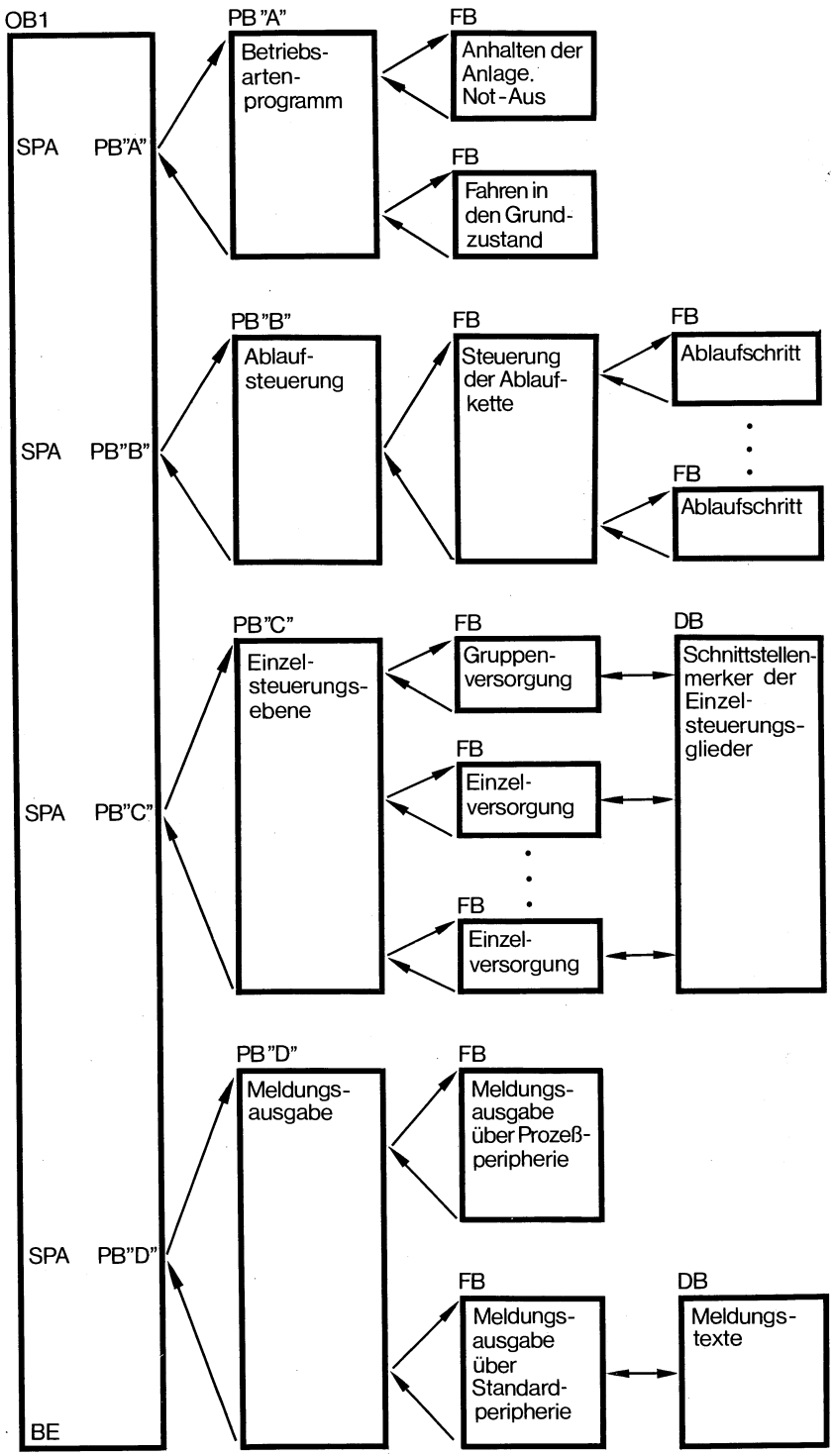


Bild 16: Gliederung des Anwenderprogramms nach Programmstruktur

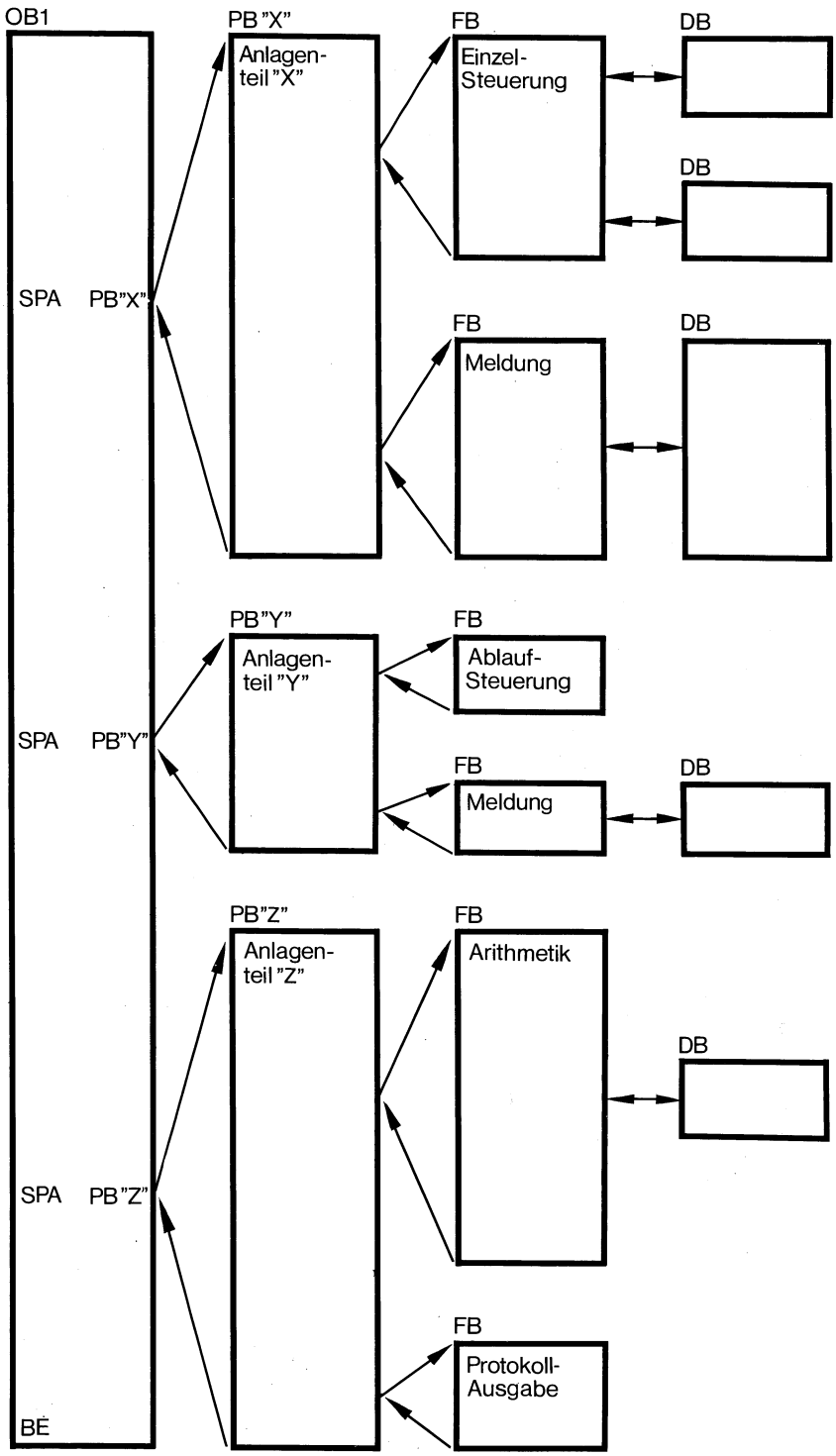


Bild 17: Gliederung des Anwenderprogramms nach Anlagenstruktur

5.4 Programmierung der alarmgesteuerten Bearbeitung (Bearbeitung eines Prozeßalarms)

Mit dem Automatisierungsgerät S5-110 S kann eine „alarmgesteuerte" Bearbeitung durchgeführt werden. Eine alarmgesteuerte Bearbeitung liegt vor, wenn ein vom Prozeß kommendes Signal den Prozessor im Automatisierungsgerät veranlaßt, die zyklische Bearbeitung zu unterbrechen und ein spezifisches Programm zu bearbeiten. Nach der Bearbeitung dieses Programms kehrt der Prozessor zur Unterbrechungsstelle im zyklischen Programm zurück und setzt dort seine Bearbeitung fort (Bilder 19 und 20).

Schnittstelle zwischen Systemprogramm und alarmgesteuerter Bearbeitung

Der Funktionsbaustein 0 (FB 0) ist die Schnittstelle zwischen Systemprogramm und alarmgesteuerter Bearbeitung. Der Anwender nimmt im FB 0 die Auswertung der Flankenwechsel der Alarmeingabebytes 0, 16, 32 und 48 vor.

Für die Reaktion auf Alarme an den Ausgabebaugruppen sind die Ausgangsbytes 0, 16, 32, 48 der Peripherie 110 vorgesehen. Nur die Verwendung dieser Ein-/Ausgänge sichert eine minimale Reaktionszeit.

Anwenderprogramm	Alarm-Eingänge	Alarm-Reaktion
FB 0	E 0.0/16.0/32.0/48.0 E 0.1/16.1/32.1/48.1 E 0.2/16.2/32.2/48.2 E 0.3/16.3/32.3/48.3 E 0.4/16.4/32.4/48.4 E 0.5/16.5/32.5/48.5 E 0.6/16.6/32.6/48.6 E 0.7/16.7/32.7/48.7	A 0.0/16.0/32.0/48.0 A 0.1/16.1/32.1/48.1 A 0.2/16.2/32.2/48.2 A 0.3/16.3/32.3/48.3 A 0.4/16.4/32.4/48.4 A 0.5/16.5/32.5/48.5 A 0.6/16.6/32.6/48.6 A 0.7/16.7/32.7/48.7

Unterbrechungsstellen

Das zyklisch bearbeitete Programm kann nicht an jeder beliebigen Stelle durch eine alarmgesteuerte Bearbeitung unterbrochen werden. Dies ist nur an den Bausteingrenzen möglich (Bild 18). Nur dann, wenn von einem Baustein auf einen anderen gewechselt wird – sei es durch den Aufruf eines neuen Bausteins oder durch die Rückkehr zum übergeordneten Baustein nach einer Bausteinende-Anweisung – kann das Systemprogramm den Funktionsbaustein 0 für die alarmgesteuerte Bearbeitung aufrufen.

Priortierung von Prozeßalarmen

Die Reihenfolge der Bearbeitung bei gleichzeitigem Auftreten von Signalzustandswechseln in den Alarmeingabebaugruppen (Priorisierung der Alarme) nimmt der Anwender im FB 0 vor.

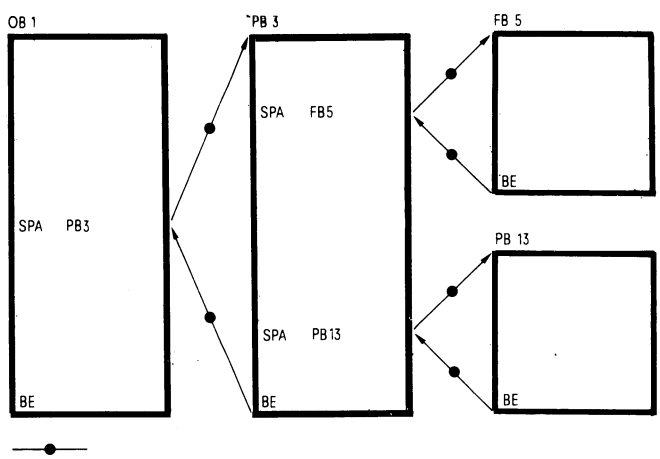
Reaktionszeit

Während der Bearbeitung eines Bausteins kann keine alarmgesteuerte Bearbeitung stattfinden. Ein auftretender Alarm wird erst bei einem Bausteinwechsel bearbeitet, also wenn ein Baustein aufgerufen oder beendet wird. Die maximale Reaktionszeit zwischen dem Auftreten eines Alarms und seiner Bearbeitung entspricht der Bearbeitungszeit eines Bausteins zuzüglich den Transferzeiten für Alarmeingänge und Reaktionszeit des Anwenders auf Alarme an den Ausgängen (< 2ms).

Um die Transferzeiten der Alarmeingänge und Ausgänge zur Peripherie 110 minimal zu halten, sind folgende Vorschriften zu berücksichtigen.

1. Die Alarmeingabebaugruppen der Peripherie 110 sind an den Steckplätzen 0, 16, 32 oder 48 zu stecken.
2. Der Anwender spricht als Reaktion auf Alarme im FB 0 die Alarmausgänge 0, 16, 32 oder 48 an.
3. Alarmeingänge und -ausgänge werden im FB 0 über die Lade- und Transferbefehle LPB, LPW und TPB, TPW angesprochen.

Aufrufe weiterer Bausteine im FB 0 verzögern die Reaktion auf Alarme auf der Alarmausgangsseite.



Unterbrechungsstellen, an denen eine alarmgesteuerte Bearbeitung „eingefügt" werden kann.

Bild 18: Unterbrechungsstellen im zyklisch arbeitenden Programm

5. Organisatorische Aufgaben

5.4 Programmierung der alarmgesteuerten Bearbeitung

Beispiel: Alarmgesteuerte Bearbeitung

Legende zu den Bildern 19 und 20

① Beginn der zyklischen Bearbeitung. Das Systemprogramm ruft den Organisationsbaustein OB 1 auf.

② Auftreten eines Alarms am Eingang E 0.3. Der Signalzustand des Eingangs E 0.3 ändert sich von „0" nach „1".

③ Bausteinwechsel. Der Signalzustandswechsel am Eingang E 0.3 wird registriert und ausgewertet. Die Bearbeitung des zyklischen Programms wird unterbrochen.

④ Das Systemprogramm ruft den Funtionsbaustein 0 auf . Das Programm dieses Funktionsbausteins wird bis zur Anweisung BE (Bausteinende) bearbeitet. Danach springt der Prozessor zurück in die zyklische Programmbearbeitung.

⑤ Da kein weiterer Alarm vorliegt, wird die Bearbeitung des zyklischen Programms an der unterbrochenen Stelle fortgesetzt.

⑥ Auftreten eines Alarms am Eingang E 0.6. Der Signalzustand des Eingangs E 0.6 ändert sich von „0" nach „1".

⑦ Auftreten eines Alarms am Eingang E 0.0. Der Signalzustand des Eingangs E 0.0 ändert sich von „0" nach „1".

⑧ Bausteinwechsel. Die Signalzustandswechsel an den Eingängen E 0.6 und E 0.0 werden registriert. Die Bearbeitung des zyklischen Programms wird unterbrochen.

⑨ Das Systemprogramm ruft den Funktionsbaustein FB 0 auf. Im FB 0 werden die Signalzustandswechsel an den Eingängen E 0.6 und E 0.0 ausgewertet.

⑩ Auftreten eines Alarms am Eingang E 0.4. Der Signalzustand des Einganges E 0.4 ändert sich von „0" auf „1".

⑪ Der Signalzustandswechsel am Eingang E 0.4 wird registriert. Die zyklische Programmbearbeitung bleibt unterbrochen.

⑫ Das Systemprogramm ruft den Funktionsbaustein FB 0 erneut auf. Im FB 0 wird der Signalzustandswechsel am E 0.4 ausgewertet.

⑬ Da kein weiterer Alarm vorliegt, wird die Bearbeitung des zyklischen Programms an der unterbrochenen Stelle fortgesetzt.

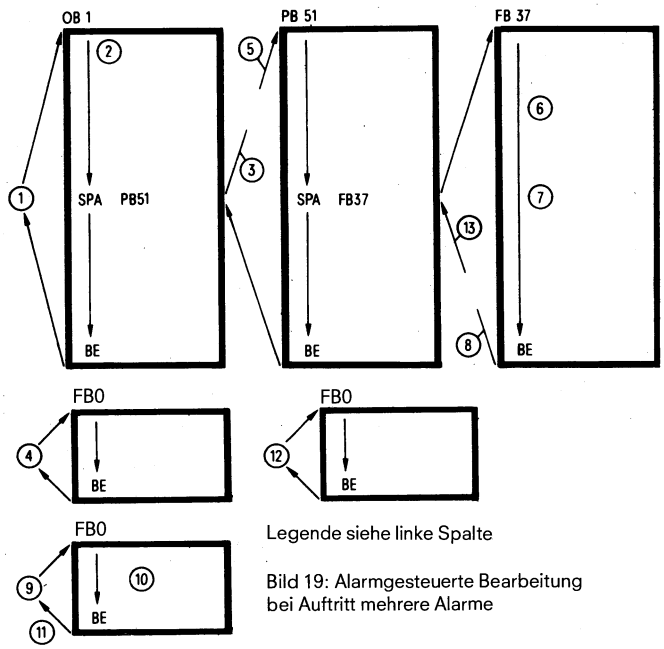


Bild 19: Alarmgesteuerte Bearbeitung bei Auftritt mehrere Alarme

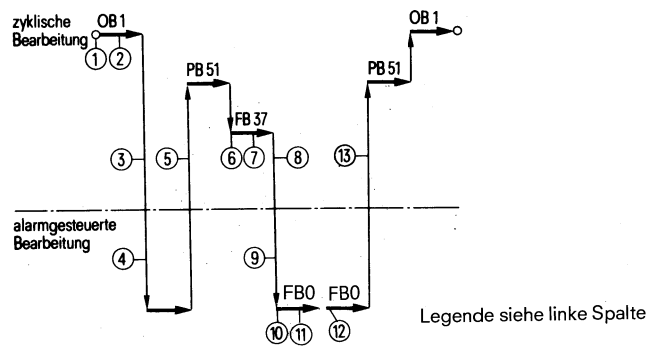


Bild 20: Darstellung der Bausteine des vorhergehenden Beispiels in der Reihenfolge ihrer Bearbeitung in einem Diagramm

Sperren der alarmgesteuerten Bearbeitung

Ein alarmgesteuertes Programm wird an einer Bausteingrenze in das zyklische Programm „eingeschoben". An dieser Stelle wird das zyklische Programm unterbrochen. Diese Unterbrechung kann sich negativ auswirken, wenn ein zyklischer Programmteil in einer bestimmten Zeit bearbeitet werden muß, um z. B. eine bestimmte Reaktionszeit zu erreichen.

Wenn ein Programmteil durch eine alarmgesteuerte Bearbeitung nicht unterbrochen werden darf, kommen folgende Programmiermöglichkeiten in Frage:

Das Programm enthält keinen Bausteinwechsel. Dadurch kann es auch nicht unterbrochen werden.

Das Programm steht selbst in einem alarmgesteuerten Programm. Hier kann es auch bei einen Bausteinwechsel nicht von einem weiteren Alarm unterbrochen werden.

Man programmiert die Operation „Alarme sperren" AS und hebt die sperrende Wirkung mit der Operation „Alarme freigeben" AF wieder auf (nur in Funktionsbausteinen möglich; siehe „Ergänzende Operationen" Seite 38). Zwischen den Operationen AS und AF wird keine alarmgesteuerte Bearbeitung durchgeführt. (Die zeitgesteuerte Bearbeitung wird damit nicht gesperrt).

Starten der alarmgesteuerten Bearbeitung

Die Eingabe der Alarmsignale erfolgt über Eingabebaugruppen mit Sammelinterrupt der Peripherie 110. Diese Alarmeingabebaugruppen sind unter den Steckplätzen 0 und/oder 16/32/48 zu stecken.

Programmierung des Funktionsbausteins FB 0

Bei Auftreten eines Alarms wird der Funktionsbaustein FB 0 aufgerufen. In diesem Baustein muß der Anwender die einzelnen Eingänge abfragen, die erforderlichen Verknüpfungen durchführen und die Reaktionen an den Ausgängen auf die Ausgabebaugruppen 0 und/oder 16/32/48 transferieren (STEP-5-Befehle TPB/TPW).

Werden im FB 0 weitere Bausteine aufgerufen, so verzögert sich die Alarmreaktion an den Ausgängen 0/16/32/48.

Sollen auch kurze Alarmimpulse (z. B. Zählimpulse) erfaßt werden, so sind im Alarmprogramm die Alarmeingaben in bestimmten Abständen auf Flankenwechsel zu untersuchen. Das kann z. B. mit folgender Befehlsfolge erreicht werden (siehe Seite 17).:

Beispiel: Erfassung von Alarmimpulsen im Alarmbaustein FBO.

Kommentar	Befehle			Befehlserläuterung
	⋮			
	:L	PB	0	aktuelles Peripheriebyte 0 in Akku 1 laden
	:T	MB	2	Inhalt von Akku 1 wird ins Merkerbyte 2 transferiert (akt. PB 0)
Abfrage der Peripherie-eingänge 0.0 bis 0.7 auf Flankenwechsel. Liegt bei irgend einen Eingang ein Flankenwechsel vor, so wird zu den entsprechenden Marken gesprungen	:L	MB	1	Merkerbyte 1 (altes PB 0) wird in Akku 1 geladen. Inhalt von Akku 1 (akt. PB 0) wurde in den Akku 2 geschoben.
	:XOW			Bitmuster von Akku 1 wird mit Akku 2 auf Gleichheit überprüft und das Ergebnis in Akku 1 hinterlegt
	:T	MB	0	Inhalt von Akku 1 (Ergebnis) wird ins Merkerbyte 0 transferiert
	:L	MB	2	Merkerbyte 2 (akt. PB 0) in Akku 1 laden
	:T	MB	1	Inhalt von Akku 1 (akt. PB 0) ins Merkerbyte 1 transferieren
	:T	EB	0	Inhalt von Akku 1 (akt. PB 0) in das Prozeßbild des Eingangsbyte 0 tranferieren
	:U	M	0.0	Merbit 0 des Merkerbytes 0 auf log 1 abfragen
	:SPB	=E 00		Ist Merkerbit 0.0 log 1 (VKE = 1) so wird zu der Marke E00 gesprungen
	:U	M	0.1	Merkerbit 1 des Merkerbytes 0 auf log 1 abfragen
	:SPB	=E 01		Ist Merkerbit 0.1 log 1 (VKE = 1) so wird zu der Marke E01 gesprungen
	⋮			Abarbeiten des Merkerbytes 0
	:SPB	=E 07		siehe oben
	⋮			Bearbeiten von weiteren alarmgesteuerten Befehlen
Nochmalige Abfrage der Peripherieeingänge 0.0 bis 0.7 auf Flankenwechsel.	:L	PB	0	aktuelles Peripheriebyte 0 in Akku 1 laden
	:T	MB	2	Inhalt von Akku 1 wird in Merkerbyte 2 transferiert (akt. PB 0)
	:L	MB	1	Merkerbyte 1 (altes PB 0) wird in Akku 1 geladen: Inhalt von Akku 1 (akt. PB 0) wurde in den Akku 2 geschoben
	:XOW			Bitmuster von Akku 1 wird mit Akku 2 auf Gleichheit überprüft und das Ergebnis in Akku 1 hinterlegt
	:T	MB	3	Inhalt von Akku 1 (Ergebnis) wird ins Merkerbyte 3 transferiert
	:L	MB	2	Merkerbyte 2 (akt. PB 0) in Akku 1 laden
	:T	MB	1	Inhalt von Akku 1 (akt. PB 0) in Merkerbyte 1 transferieren
	⋮			Bearbeiten von weiteren alarmgesteuerten Befehlen
	:L	MB	3	Merkerbyte 3 (Ergebnis) in Akku 1 laden
	:T	MB	0	Inhalt von Akku 1 (Ergebnis) wird in Merkerbyte 0 transferiert
Tritt bei der nochmaligen Abfrage der Peripherie-eingänge ein Flankenwechsel auf, so wird zum Anfang des FB 0 zurückgesprungen	:L	KF	0	Festpunktkonstante 0 wird in Akku 1 geladen: Inhalt von Akku1 (Ergebnis) wurde in den Akku 2 geschoben
	:!=F			Akku 1 wird mit Akku 2 auf Gleichheit überprüft, bei Gleichheit VKE = 1 sonst VKE = 0
	:=	M	4.0	Bei Gleichheit der Akkus (VKE = 1) wird Merker 4.0 gesetzt
	:UN	M	4.0	Merkerbit 4.0 auf log 0 abfragen
	:SPB	FB	0	Ist Merkerbit 4.0 log 0 (VKE = 1) so wird zum Anfang des Funktionsbausteines 0 gesprungen
	:L	KF	0	Festpunktkonstante 0 wird in Akku 1 geladen
	:T	MB	3	Inhalt von Akku 1 (Festpunktkonstante 0) wird ins Merkerbyte 3 transferiert
	⋮			Bearbeiten von weiteren alarmgesteuerten Befehlen
	E 00 :			Bearbeiten von Befehlen wenn das Merkerbit 0.0 gesetzt ist
Bearbeitung der Flankenwechsel	⋮			
	E 07 :			Bearbeiten von Befehlen wenn das Merkerbit 0.7 gesetzt ist
	:BE			Programmende

Beispiel für eine Alarmbearbeitung

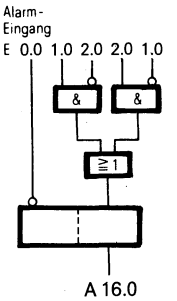
Genaues Positionieren mit einem Endschalter

Funktionsbeschreibung:
Ein Ausgang wird über eine Verriegelungsbedingung Taster E1.0 oder E2.0 eingeschaltet und soll nach Ansprechen eines Endtasters E0.0 mit möglichst kurzer und gleichbleibender Verzögerungszeit wieder ausgeschaltet werden (Bild 21).

Programmierung:
Die Setzbedingung für den Ausgang ist in dem Programmbaustein 2 programmiert.

Der Rücksetzeingang ist auf den Alarmeingang E 0.0 gelegt. Der FB 0 wird entsprechend der Einstellung in der „Eingabebaugruppe mit Sammelinterrupt" 0 bei steigender oder fallender Flanke des Anschlusses 0.0 aufgerufen. Es sei nur 1 Eingang an Eingangsbyte 0 angeschlossen E 0.0.

Programmierung des FB 0:
Um den aktuellen Zustand des Eingangs E 0.0 abfragen zu können, wird zunächst das Eingangsbild des Bytes „0" mit den Operationen L PB und T EB aktualisiert. Bei einer fallenden Flanke am Alarmeingang E 0.0 und Rücksetzen des Ausgangs A 16.0 wird der Ausgang mit den Operationen L AB und T PB direkt zur Ausgangsbaugruppe transferiert. Beim Transferieren zu den Peripheriebytes 0 bis 63 wird das Ausgangsprozessabbild automatisch mit nachgeführt.




```
FB 0                    PB2
:L    PB0       :U      E1.0
:T    EB0       :UN     E2.0
:UN   E 0.0     :O
:R    A 16.0    :U      E2.0
:L    AB16      :UN     E1.0
:T    PB16      :S      A16.0
:BE
```

Bild 21: Beispiel für eine Alarmbearbeitung

5.5 Programmierung des Anlaufverhaltens

Das Systemprogramm unterscheidet drei verschiedene Anlaufarten des Automatisierungsgeräts:

manueller Neustart

manueller Neustart mit Rücksetzen

automatischer Neustart

Die Anlaufart ist im Betriebssystem festgelegt und kann vom Anwender lediglich über die Taste „Rücksetzen" beeinflußt werden.

Manueller Neustart

Ein Neustart wird durch Handbedienung ausgelöst, indem der Stop-Schalter auf der Zentralbaugruppe von der Stellung „Stop" in die Stellung „Betrieb" gebracht wird.

Dabei führt das System folgende Tätigkeiten durch:

Rücksetzen der nicht remanenter Merker (M 128.0 – M 255.7)

Laden des Eingangs-Prozess-Abbilds

Löschen des Ausgangs-Prozess-Abbilds

Rücksetzen aller Ausgänge der Peripherie

Aufbau der Baustein-Adreßliste

Manueller Neustart mit Rücksetzen

Ein Neustart mit Rücksetzen wird durch Handbedienung ausgelöst, indem die Taste „Rücksetzen" gedrückt wird und gleichzeitig der Stop-Schalter auf der Zentralbaugruppe von der Stellung „Stop" in die Stellung „Betrieb" gebracht wird.

Dabei führt das System folgende Tätigkeiten durch:

Löschen aller Zeitwerte

Löschen aller Zählerwerte

Rücksetzen aller Merker

Laden des Eingangs-Prozess-Abbilds

Löschen des Ausgangs-Prozess-Abbilds

Rücksetzen aller Ausgänge der Peripherie

Aufbau der Baustein-Adreßliste

Automatischer Neustart

Bei Netzspannungsausfall und anschließende Netzspannungswiederkehr versucht das Automatisierungsgerät automatisch einen Neustart durchzuführen. Die Funktion des automatischen Neustarts ist identisch mit der des manuellen Neustarts.

Wünscht der Anwender keinen automatischen Neustart bei Spannungswiederkehr nach Spannungsausfall, so kann er einen nicht remanenten Merker am Anfang des OB 1 abprüfen, und mit dem STEP-5-Befehl „STP" das System in den Stopzustand führen.

5.6 Auswertung von Gerätefehlern und Programmlaufbesonderheiten

Das Systemprogramm kann fehlerhaftes Arbeiten des Zentralprozessors, Fehler im Systemprogramm oder Auswirkungen einer fehlerhaften Programmierung durch den Anwender feststellen. Bei einigen dieser Fehler arbeitet der Zentralprozessor nicht mehr einwandfrei. Das AG geht dann in den Stop-Zustand.

Die Reaktion auf Gerätefehler und Programmlaufbesonderheiten sind im System festgelegt. Folgende Ereignisse werden ausgewertet.

Speicherfehler

Batterieausfall (bei Neustart)

Quittungsverzug bei Speicherzugriff

Zykluszeitüberschreitung

Nicht dekodierbarer Befehl

Nicht zulässiger Baustein

Nicht vorhandener Datenbaustein

Baustein-Stack-Überlauf

Speicherfehler

Bei einem Neustart erkennt das Systemprogramm fehlerhaft adressierte Anwenderspeichermodule und geht in den Stop-Zustand. (falsche Brückenbelegung).

Darüberhinaus wird bei jedem Neustart das Betriebssystem überprüft. Wird ein Fehler vom Betriebssystem erkannt, so geht das AG in den Stop-Zustand.

Wurde die Funktion „Speicher komprimieren", welche das AG im Auftrag des Programmiergerätes durchführt, etwa durch einen Netzspannungsausfall unterbrochen, so verzweigt das AG bei Netzspannungswiederkehr in den Stop-Zustand.

Batterieausfall

Wird bei einem Neustart ein Batterieausfall (Unterschreiten der minimal zulässigen Batteriespannung) erkannt, so verzweigt das AG in den Stop-Zustand.

Quittungsverzug

Ein Quittungsverzug tritt auf, wenn ein nicht vorhandener Speicherbereich angesprochen wird. Die Ursache eines Quittungsverzugs kann ein Defekt der Speicherbaugruppe sein, oder das Entfernen der Baugruppe im Betrieb.

Zykluszeitüberschreitung

Die Zykluszeit umfaßt die gesamte Zeitdauer einer Bearbeitung des zyklischen Programms. Darin enthalten sind der Aufruf und die Bearbeitung des Organisationsbausteins OB 1 und die in diesem Organisationsbaustein aufgerufenen Programm- und Funktionsbausteine mit ihren Schachtelungen, sowie alle in diesem Zyklus bearbeiteten zeit- und alarmgesteuerten Programmteile. Das zyklische Programm endet mit einer Bausteinende-Anweisung im Organisationsbaustein (OB 1). Überschreitet die Bearbeitungszeit eine bestimmte Zeitdauer (die im Prozessor eingestellte „Zykluszeit"), erkennt das Systemprogramm den Fehler „Zykluszeitüberschreitung".

Die Zykluszeitüberschreitung kann z. B. durch fehlerhafte Programmierung ausgelöst werden, wenn bei einem bestimmten Prozeßstand der Prozessor in einer Programmschleife läuft, oder durch Ausfall des Taktgenerators.

Tritt eine Zykluszeitüberschreitung auf, unterbricht das Systemprogramm die Bearbeitung des STEP-5-Programms und geht in den Stop-Zustand.

Nicht dekodierbarer Befehl

Bearbeitet das AG einen STEP-5-Befehl, der nicht zum Sprachumfang des Automatisierungsgerätes SIMATIC S5-110S gehört, so geht es in den Stop-Zustand.

Nicht zulässiger Baustein

Im Anwenderprogramm steht ein Bausteinaufrufbefehl mit einer Bausteinnummer, die größer ist als die für das AG maximal zulässig (127 bei PB's, 63 DB's, 47 bei FB's). Das AG geht bei der Bearbeitung des Befehls in den Stop-Zustand.

Nicht vorhandener Datenbaustein

Im Anwenderprogramm wird ein Datenwort-Lade- oder Transferbefehl bearbeitet, ohne daß ein zugehöriger Datenbaustein zuvor aufgerufen wurde (Adressbereichszuordnung fehlt für DB). Das AG geht in den Stop-Zustand.

Baustein-Stack-Überlauf

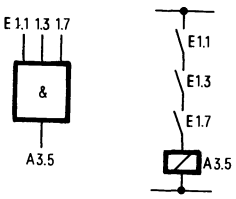
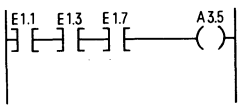
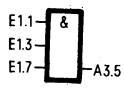
Werden im Anwenderprogramm hintereinander mehr als 7 Bausteine aufgerufen, ohne daß zwischenzeitlich ein BE-Befehl durchlaufen wurde, so wird die maximal zulässige Bausteinverschachtelungstiefe 8 überschritten. Das AG geht in den Stop-Zustand.

6. Programmierbeispiele

6.1 Grundoperationen

6.1.1 Verknüpfungsfunktionen

UND-Verknüpfung

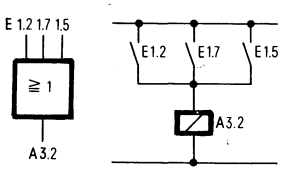
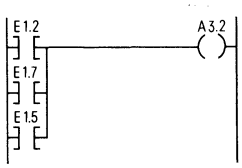
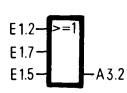
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 1.1 U E 1.3 U E 1.7 = A 3.5		

Am Ausgang A 3.5 erscheint Signalzustand „1", wenn alle Eingänge gleichzeitig den Signalzustand „1" aufweisen.

Am Ausgang A 3.5 erscheint Signalzustand „0", wenn mindestens einer der Eingänge den Signalzustand „0" aufweist.

Die Anzahl der Abfragen und die Reihenfolge der Programmierung ist beliebig.

ODER-Verknüpfung

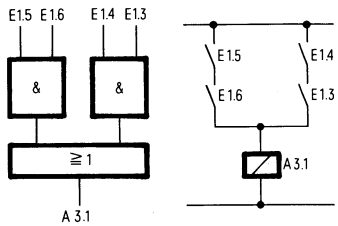
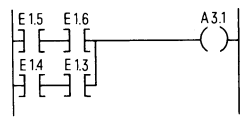
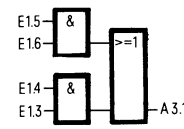
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	O E 1.2 O E 1.7 O E 1.5 = A 3.2		

Am Ausgang A 3.2 erscheint Signalzustand „1", wenn mindestens einer der Eingänge den Signalzustand „1" aufweist.

Am Ausgang A 3.2 erscheint Signalzustand „0", wenn alle Eingänge gleichzeitig den Signalzustand „0" aufweisen.

Die Anzahl der Abfragen und die Reihenfolge der Programmierung ist beliebig.

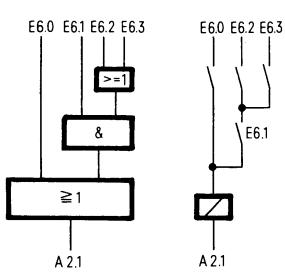
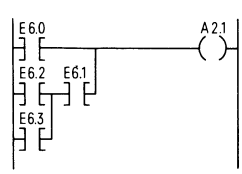
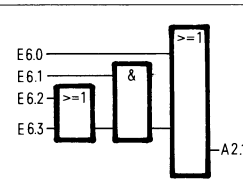
UND-vor-ODER-Verknüpfung

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 1.5 U E 1.6 O U E 1.4 U E 1.3 = A 3.1		

Am Ausgang A 3.1 erscheint Signalzustand „1", wenn mindestens eine UND-Verknüpfung erfüllt ist.

Am Ausgang A 3.1 erscheint Signalzustand „0", wenn keine UND-Verknüpfung erfüllt ist.

ODER-vor-UND-Verknüpfung

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	O E 6.0 O U E 6.1 U(O E 6.2 O E 6.3) = A 2.1		

Am Ausgang A 2.1 erscheint Signalzustand „1", wenn Eingang E 6.0 oder Eingang E 6.1 und einer der Eingänge E 6.2 bzw. E 6.3 Signal „1" führen.

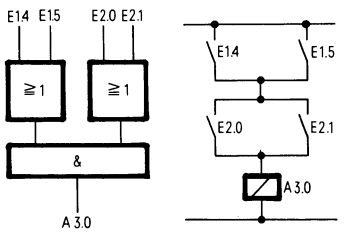
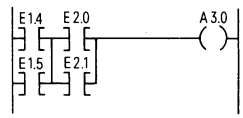
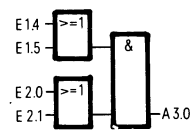
Am Ausgang A 2.1 erscheint Signalzustand „0", wenn Eingang E 6.0 Signal „0" führt und die UND-Verknüpfung nicht erfüllt ist.

6. Programmierbeispiele

6.1 Grundoperationen

6.1.1 Verknüpfungsfunktionen

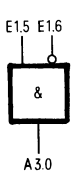
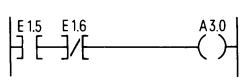
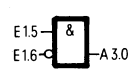
ODER-vor-UND-Verknüpfung

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U(O E 1.4 O E 1.5) U(O E 2.0 O E 2.1) = A 3.0		

Am Ausgang A 3.0 erscheint Signalzustand „1", wenn beide ODER-Verknüpfungen erfüllt sind.

Am Ausgang A 3.0 erscheint Signalzustand „0", wenn mindestens eine ODER-Verknüpfung nicht erfüllt ist.

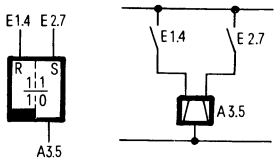
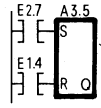
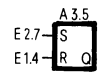
Abfrage auf Signalzustand „0"

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 1.5 UN E 1.6 = A 3.0		

Am Ausgang A 3.0 erscheint Signalzustand „1" nur dann, wenn der Eingang E 1.5 den Signalzustand „1" und der Eingang E 1.6 den Signalzustand „0" führt.

6.1.2 Speicherfunktionen

RS-Speicherglied für Speichernde Signalausgabe

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 2.7 S A 3.5 U E 1.4 R A 3.5 NOP 0		

Signalzustand „1" am Eingang E 2.7 bewirkt das Setzen des Speicherglieds.

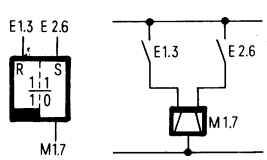
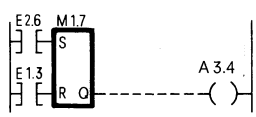
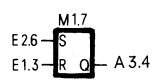
Wechselt der Signalzustand am Eingang E 2.7 nach „0", so bleibt dieser Zustand erhalten, d. h. das Signal wird gespeichert.

Signalzustand „1" am Eingang E 1.4 bewirkt das Rücksetzen des Speicherglieds.

Wechselt der Signalzustand am Eingang E 1.4 nach „0", so bleibt dieser Zustand erhalten.

Bei gleichzeitigem Anliegen des Setzsignals (Eingang E 2.7) und des Rücksetzsignals (Eingang E 1.4) ist die zuletzt programmierte Abfrage (hier U E 1.4) während der Bearbeitung des übrigen Programms wirksam.

RS-Speicherglied mit Merkern

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 2.6 S M 1.7 U E 1.3 R M 1.7 U M 1.7 = A 3.4		

Signalzustand „1" am Eingang E 2.6 bewirkt das Setzen des Speicherglieds.

Wechselt der Signalzustand am Eingang E 2.6 nach „0", so bleibt dieser Zustand erhalten, d. h. das Signal wird gespeichert.

Signalzustand „1" am Eingang E 1.3 bewirkt das Rücksetzen des Speicherglieds.

Wechselt der Signalzustand am Eingang E 1.3 nach „0", so bleibt dieser Zustand erhalten.

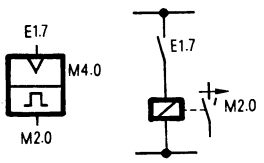
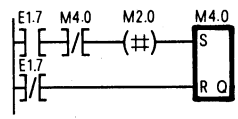
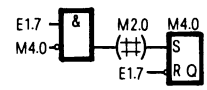
Bei gleichzeitigem Anliegen des Setzsignals (Eingang E 2.6) und des Rücksetzsignals (Eingang E 1.3) ist die zuletzt programmierte Abfrage (hier U E 1.3) während der Bearbeitung des übrigen Programms wirksam.

6. Programmierbeispiele

6.1 Grundoperationen

6.1.2 Speicherfunktionen

Nachbildung eines Wischrelais

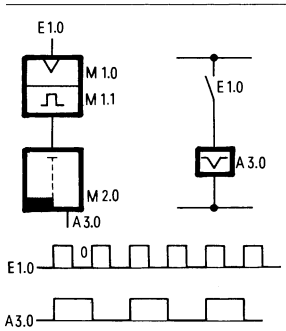
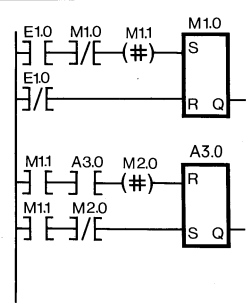
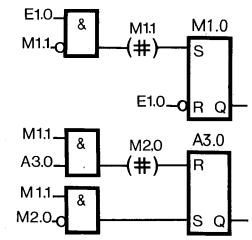
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 1.7 UN M 4.0 = M 2.0 U M 2.0 S M 4.0 UN E 1.7 R M 4.0 NOP 0		

Bei jeder ansteigenden Flanke des Eingangs E 1.7 ist die UND-Verknüpfung (U E 1.7 und UN M 4.0) erfüllt und mit VKE = „1" werden die Merker M 4.0 und M 2.0 („Flankenmerker") gesetzt.

Beim nächsten Bearbeitungszyklus ist die UND-Verknüpfung U E 1.7 und UN M 4.0 nicht erfüllt, da der Merker M 4.0 gesetzt worden ist.

Der Merker M 2.0 wird zurückgesetzt.
Der Merker M 2.0 führt also während eines einzigen Programmdurchlaufs Signalzustand „1".

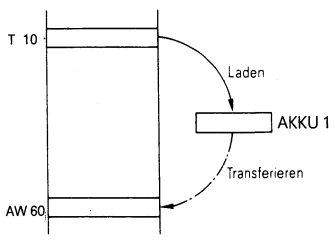
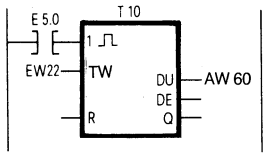
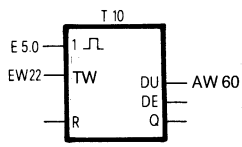
Binäruntersetzer (T-Kippglied)

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 1.0 UN M 1.0 = M 1.1 U M 1.1 S M 1.0 UN E 1.0 R M 1.0 NOP 0 U M 1.1 U A 3.0 = M 2.0 U M 2.0 R A 3.0 U M 1.1 UN M 2.0 S A 3.0 NOP 0		

Der Binäruntersetzer (Ausgang A 3.0) wechselt bei jedem Signalzustandswechsel von „0" nach „1" (ansteigende Flanke) des Einganges E 1.0 seinen Zustand. Am Ausgang des Speicherglieds erscheint deshalb die halbe Eingangsfrequenz.

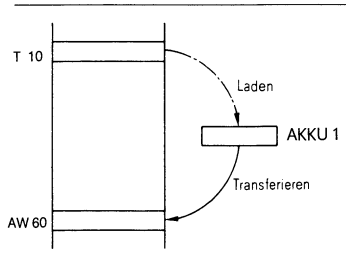
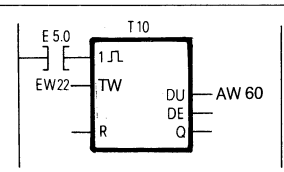
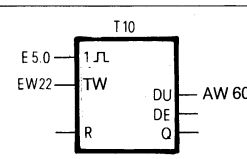
6.1.3 Lade- und Transferfunktionen

Laden eines Zeitwertes

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 5.0 L EW 22 SI T 10 L T 10		

Der Inhalt der mit T 10 adressierten Speicherzelle wird in den Akkumulator 1 geladen.

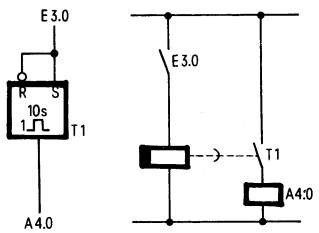
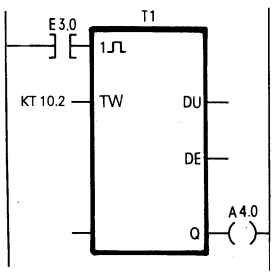
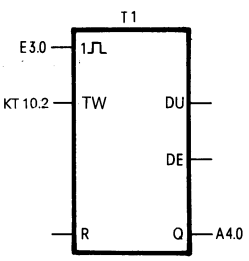
Transferieren

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 5.0 L EW 22 SI T 10 L T 10 T AW60		

Der Inhalt des Akkumulator 1 wird in das mit AW60 adressierte Prozeßabbild transferiert. Am AW60 sieht man in diesem Beispiel die Zeit T 10 dual-codiert mitlaufen.

6.1.4 Zeitfunktionen

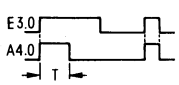
Impuls

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 3.0 L KT 10.2 SI T 1 U T 1 = A 4.0		

Bei Verknüpfungsergebnis „1" und erstmaliger Bearbeitung wird das Zeitglied gestartet. Bei wiederholter Bearbeitung mit Verknüpfungsergebnis „1" bleibt das Zeitglied unbeeinflußt.

Bei Verknüpfungsergebnis „0" wird das Zeitglied auf Null gesetzt (gelöscht).

Die Abfragen U T bzw. O T liefern Signalzustand „1", solange die Zeit läuft.

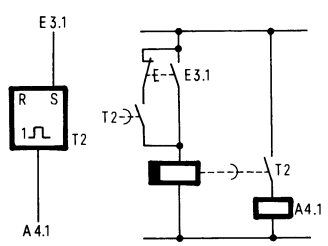
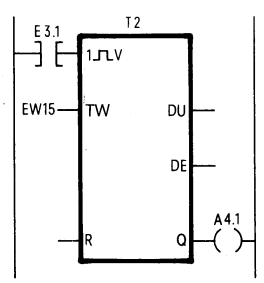
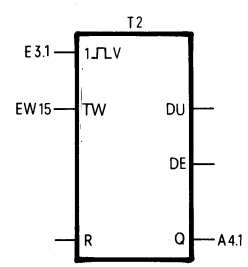


KT 10.2≙ 10·1s = 10s

Das Zeitglied wird mit dem angegebenen Wert (10) geladen. Die Zahl rechts vom Punkt gibt das Zeitraster an:
0 = 0.01s 2 = 1s
1 = 0.1s 3 = 10s

Die Ausgänge D U und D E sind digitale Ausgänge. Am Ausgang D U steht der Zeitwert dual-codiert am Ausgang D E BCD-codiert an.

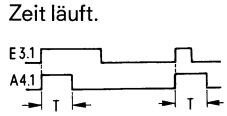
Verlängerter Impuls

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	U E 3.1 L EW 15 SV T 2 U T 2 = A 4.1		

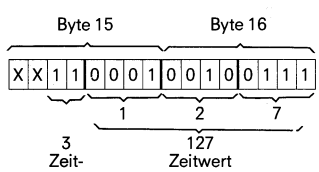
Bei Verknüpfungsergebnis „1" und erstmaliger Bearbeitung wird das Zeitglied gestartet.

Bei Verknüpfungsergebnis „0" bleibt das Zeitglied unbeeinflußt.

Die Abfragen U T oder O T liefern Signalzustand „1", solange die Zeit läuft.



EW 15:

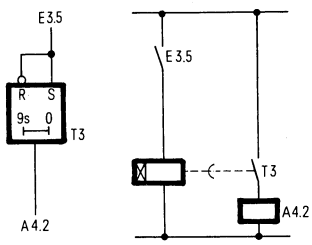


Setzen des Zeitwerts mit dem im BCD-Code vorliegenden Wert der Operanden E, A, M oder D (im Beispiel Eingangswort 15)
EW 15:
In dem obigen Beispiel beträgt die Zeit T des verlängerten Impulses 1270s, die durch das Eingangswort 15 (EW 15) bestimmt wird.

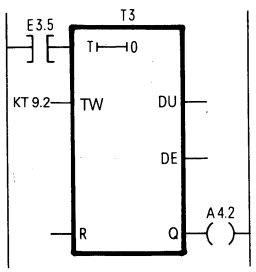
EW 15≙ 127·10s = 1270s
Zeitraster (Multiplikator für Zeitwert):
0 ≙ 0.01s 2 ≙ 1s
1 ≙ 0.1s 3 ≙ 10s

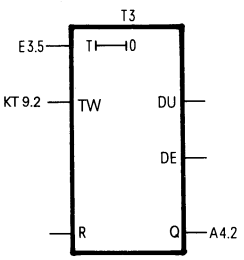
Einschaltverzögerung

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan




```
U  E  3.5
L  KT 9.2
SE T  3
U  T  3
=  A  4.2
```

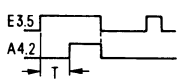




Bei Verknüpfungsergebnis „1" und erstmaliger Bearbeitung wird das Zeitglied gestartet. Bei wiederholter Bearbeitung mit Verknüpfungsergebnis „1" bleibt das Zeitglied unbeeinflußt.

Bei Verknüpfungsergebnis „0" wird das Zeitglied auf Null gesetzt (gelöscht).

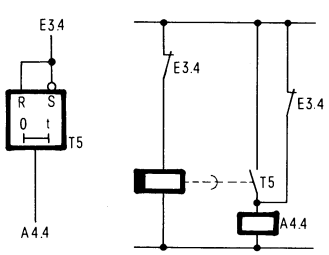
Die Abfrage von U T bzw. O T liefert Signalzustand „1", wenn die Zeit abgelaufen und das Verknüpfungsergebnis am Eingang noch ansteht.



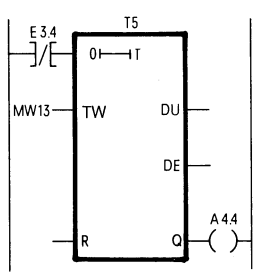
KT 9.2:
Das Zeitglied wird mit dem angegebenen Wert (9) geladen. Die Zahl rechts vom Punkt gibt das Zeitraster an:
0 ≙ 0.01s 2 ≙ 1s
1 ≙ 0.1s 3 ≙ 10s

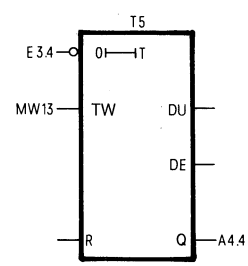
Ausschaltverzögerung

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan




```
UNE  3.4
L    MW 13
SA T  5
U  T  5
=  A  4.4
```

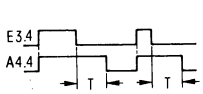


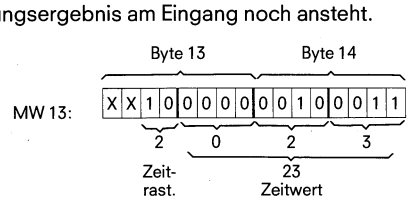


Bei Verknüpfungsergebnis „0" und erstmaliger Bearbeitung wird das Zeitglied gestartet. Bei wiederholter Bearbeitung mit Verknüpfungsergebnis „0" bleibt das Zeitglied unbeeinflußt.

Bei Verknüpfungsergebnis „1" wird das Zeitglied auf Null gesetzt (gelöscht).

Die Abfragen U T bzw. O T liefern Signalzustand „1", wenn die Zeit läuft oder das Verknüpfungsergebnis am Eingang noch ansteht.





Setzen des Zeitwertes mit dem im BCD-Code vorliegenden Wert der Operanden E, A, M oder D (im Beispiel Merkerwort 13)

MW 13:
In dem obigen Beispiel beträgt die Zeit T der Ausschaltverzögerung 23s, die durch das Merkerwort 13 (MW 13) bestimmt wird.
MW 13 ≙ 23 · 1s = 23s
Zeitraster:
0 ≙ 0.01s 2 ≙ 1s
1 ≙ 0.1s 3 ≙ 10s

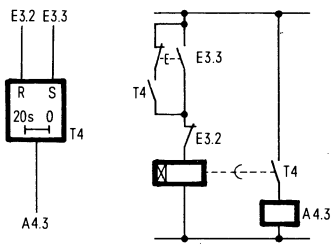
6. Programmierbeispiele

6.1 Grundoperationen

6.1.4 Zeitfunktionen

Speichernde Einschaltverzögerung

Vorlage

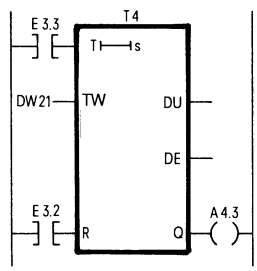


STEP-5-Darstellung

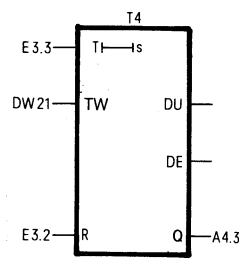
Anweisungsliste

```
U  E  3.3
L  DW 21
SS T  4
U  E  3.2
R  T  4
U  T  4
=  A  4.3
```

Kontaktplan



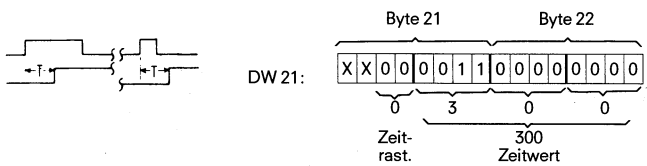
Funktionsplan



Bei Verknüpfungsergebnis „1" und erstmaliger Bearbeitung wird das Zeitglied gestartet.

Bei Verknüpfungsergebnis „0" bleibt das Zeitglied unbeeinflußt.

Die Abfragen U T bzw. O T liefern Signalzustand „1", wenn die Zeit abgelaufen ist. Der Signalzustand wird erst dann „0", wenn das Zeitglied mit der Funktion R T zurückgesetzt wurde.



Setzen des Zeitwertes mit dem im BCD-Code vorliegenden Wert der Operanden E, A, M oder D (im Beispiel Datenwort 21)

DW 21:

In dem obigen Beispiel beträgt die Zeit T der Einschaltverzögerung 3s, die durch das Datenwort 21 (DW 21) bestimmt wird.

DW 21 ≙ 300 · 0,01s = 3s

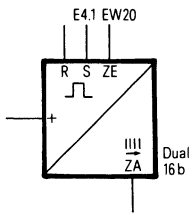
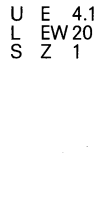
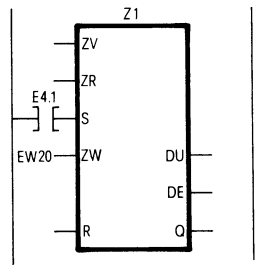
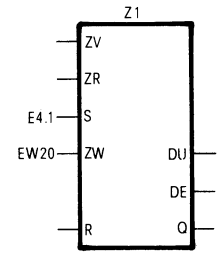
Zeitraster:

0 ≙ 0.01s 2 ≙ 1s

1 ≙ 0.1s 3 ≙ 10s

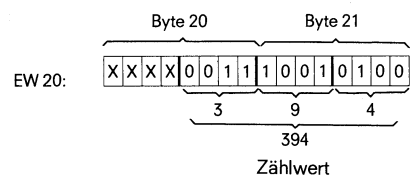
6.1.5 Zählfunktionen

Zähler setzen

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
			

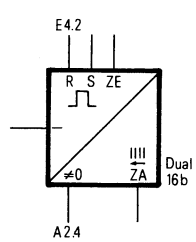
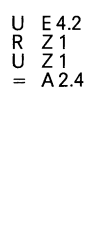
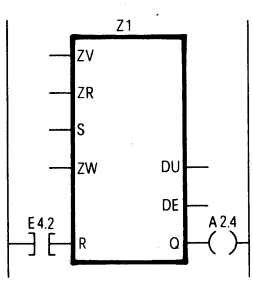
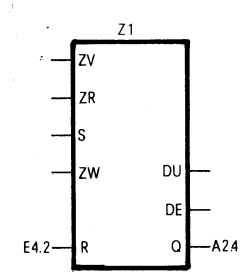
Bei Verknüpfungsergebnis „1" und erstmaliger Bearbeitung wird der Zähler gesetzt. Bei wiederholter Bearbeitung bleibt der Zähler unbeeinflußt (unabhängig davon, ob das Verknüpfungsergebnis „1" oder „0" ist). Bei erneuter erstmaliger Bearbeitung mit Verknüpfungsergebnis „1" wird die Zähler wieder gesetzt (Flankenauswertung).

Der für die Flankenauswertung des Setzeingangs erforderliche Merker ist im Zählwort mitgeführt. Das Zählwort kann sein: EW, AW, MW, DW.
In diesen Zählworten wird der Zählwert des Zähler angegeben. Der Zählwert wird als 16 Bit Wort BCD codiert vorgegeben, wobei die ersten 4 Bits des Zählwortes nicht verarbeitet werden.



In dem obigen Beispiel beträgt der Anfangswert der Zähler 394.
Die Ausgänge D U und D E sind digitale Ausgänge. Am Ausgang D U steht der Zählwert dual-codiert, am Ausgang D E BCD codiert an.

Zähler zurücksetzen

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
			

Bei Verknüpfungsergebnis „1" wird der Zähler auf Null gesetzt (rückgesetzt).

Bei Verknüpfungsergebnis „0" bleibt der Zähler unbeeinflußt.

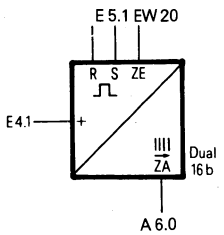
6. Programmierbeispiele

6.1 Grundoperationen

6.1.5 Zählfunktionen

Vorwärts zählen

Vorlage

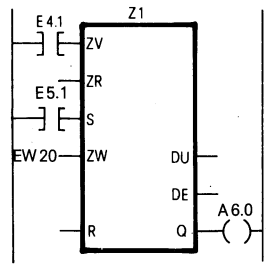


STEP-5-Darstellung

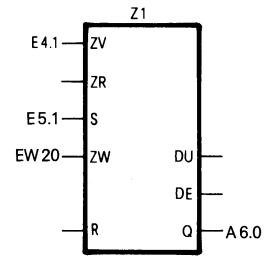
Anweisungsliste

```
U   E   4.1
ZV  Z   1
NOP 0
U   E   5.1
L   EW20
S   Z   1
NOP 0
NOP 0
NOP 0
U   Z   1
=   A   6.0
```

Kontaktplan



Funktionsplan

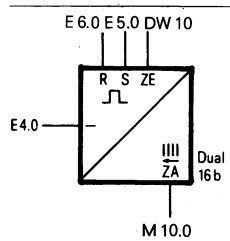


Der Wert des adressierten Zählers wird nur um 1 erhöht, wenn am ZV-Eingang des Zählers ein Flankenwechsel von „0" nach „1" auftritt. Die für Flankenauswertung der Zähleingänge erforderlichen Merker sind im Zählwort mitgeführt.

Durch die zwei getrennten Flankenmerker für ZV und ZR kann ein Zähler mit zwei verschiedenen Eingängen als Vorwärts-/Rückwärtszähler verwendet werden.
Der Q-Ausgang des Zählers ist „1", solange der aktuelle Zählwert >0 ist.

Rückwärts zählen

Vorlage

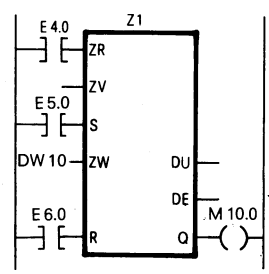


STEP-5-Darstellung

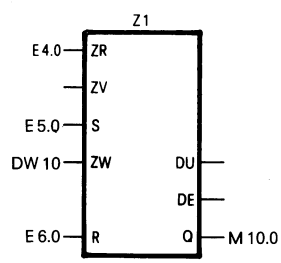
Anweisungsliste

```
U   E   4.0
ZR  Z   1
NOP 0
U   E   5.0
L   DW10
S   Z   1
U   E   6.0
R   Z   1
NOP 0
NOP 0
U   Z   1
=   M   10.0
```

Kontaktplan



Funktionsplan

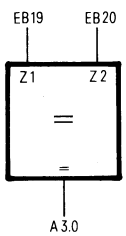
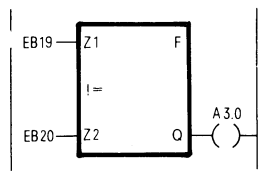
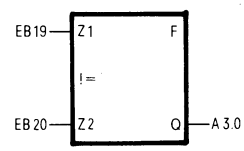


Der Wert des adressierten Zählers wird nur um 1 erniedrigt, wenn am ZR-Eingang des Zählers ein Flankenwechsel von „0" nach „1" auftritt. Die für Flankenauswertung der Zähleingänge erforderlichen Merker sind im Zählwort mitgeführt.

Durch die zwei getrennten Flankenmerker für ZV und ZR kann ein Zähler mit zwei verschiedenen Eingängen als Vorwärts-/Rückwärtszähler verwendet werden.
Der Q-Ausgang des Zählers ist „1", solange der aktuelle Zählwert >0 ist.

6.1.6 Vergleichsfunktionen

Vergleich auf gleich

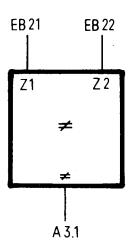
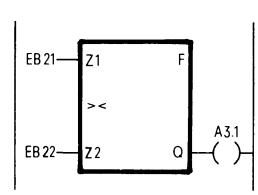
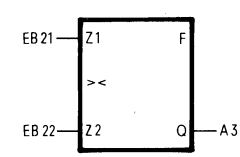
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 19 L EB 20 !=F = A 3.0		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktion verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 19 = EB 20	0	0	1
EB 19 < EB 20	0	1	0
EB 19 > EB 20	1	0	0

Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf gleich kann mit der Sprungfunktion (bei VKE = 1) SPZ = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

Vergleich auf ungleich

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 21 L EB 22 ><F = A 3.1		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktionen verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 21 = EB 22	0	0	0
EB 21 < EB 22	0	1	1
EB 21 > EB 22	1	0	1

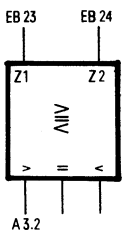
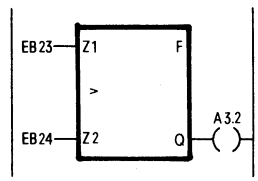
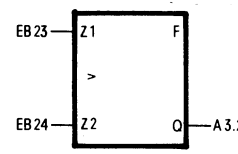
Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf ungleich kann mit der Sprungfunktion (bei VKE = 1) SPN = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

6. Programmierbeispiele

6.1 Grundoperationen

6.1.6 Vergleichsfunktionen

Vergleich auf größer

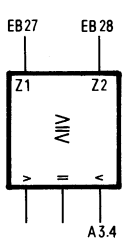
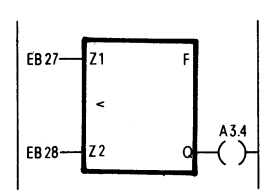
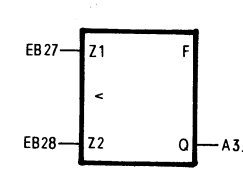
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 23 L EB 24 >F = A 3.2		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktion verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 23 = EB 24	0	0	0
EB 23 < EB 24	0	1	0
EB 23 > EB 24	1	0	1

Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf größer kann mit der Sprungfunktion (bei VKE = 1) SPP = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

Vergleich auf kleiner

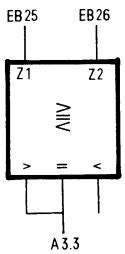
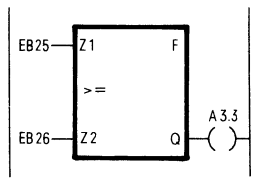
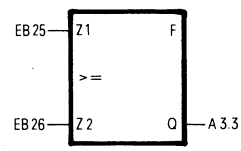
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 27 L EB 28 <F = A 3.4		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktion verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 27 = EB 28	0	0	0
EB 27 < EB 28	0	1	1
EB 27 > EB 28	1	0	0

Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf kleiner kann mit der Sprungfunktion (VKE = 1) SPM = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

Vergleich auf größer-gleich

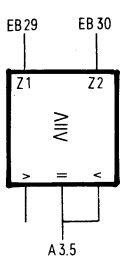
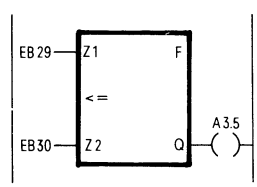
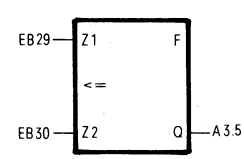
Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 25 L EB 26 >=F = A 3.3		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktion verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 25 = EB 26	0	0	1
EB 25 < EB 26	0	1	0
EB 25 > EB 26	1	0	1

Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf größer-gleich kann mit der Sprungfunktion (VKE = 1) SPB = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

Vergleich auf kleiner-gleich

Vorlage	STEP-5-Darstellung		
	Anweisungsliste	Kontaktplan	Funktionsplan
	L EB 29 L EB 30 <=F = A 3.5		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleicherfunktion verglichen. Das Vergleichsergebnis wird durch die Anzeigenbits ANZ0 und ANZ1 angezeigt.

für	ANZ1	ANZ0	VKE
EB 29 = EB 30	0	0	1
EB 29 < EB 30	0	1	1
EB 29 > EB 30	1	0	0

Beim Vergleich wird die Zahlendarstellung der Operanden (Festpunktrechnung) berücksichtigt.
Nach einem Vergleich auf kleiner-gleich kann mit der Sprungfunktion (VKE = 1) SPB = . . . zu einer „Marke" (± 127 Wörter) gesprungen werden.

6.2 Ergänzende Operationen

Funktionsbausteine können gegenüber den Programmbausteinen mit einem erweitertem Operationsvorrat programmiert werden. Der Gesamtoperationsvorrat für Funktionsbausteine besteht aus den Grundoperationen und den ergänzenden Operationen.

Bei den Funktionsbausteinen werden die Operationen nur in Anweisungsliste AWL dargestellt. Die Programme der Funktionsbausteine können also nicht in graphischer Form (FUP oder KOP) programmiert werden.

Im folgenden sind die Operationen beschrieben, die nur bei Funktionsbausteinen verwendet werden können. Zusätzlich sind die Kombinationsmöglichkeiten der Substitutionsbefehle mit den Aktualoperanden angegeben.

6.2.1 Binäre Verknüpfungsfunktionen

Beispiel	AWL	Erläuterung
Am Eingang E2.0 ist eine Lichtschranke installiert die Stückgut zählt. Nach jeweils 100 Stück soll entweder in den Funktionsbaustein FB5 oder in den FB6 verzweigt werden. Nach 800 Stück soll der Zähler 10 automatisch rückgesetzt werden und vom Neuen hochzählen.	U E 2.0 ZV Z 10 U E 3.0 L KZ 0 S Z 10 O E 4.0 O M 5.2 R Z 10	Der Zählwert des Zählers Z10 wird durch den Eingang E3.0 mit der Konstanten 0 geladen. Mit jeden positiven Flankenwechsel am E2.0 wird der Zählwert um 1 erhöht. Der Zähler wird entweder durch E4.0 oder Merker M5.2 zurückgesetzt.
	LC Z 10 T DW12	Der aktuelle Zählwert des Zählers wird BCD-codiert im Datenwort abgelegt.
	PN DW12.8 SPB FB 5	Solange das 8. Datenbit des DW12 null ist, wird in den FB5 gesprungen. Dies ist bei den ersten, dritten, fünften usw. hundert Stück der Fall.
	P DW12.8 SPB FB 6	Solange das 8. Datenbit des DW12 eins ist, wird in den FB6 gesprungen. Dies ist bei den zweiten, vierten, sechsten usw. hundert Stück der Fall.
	P DW12.11 = M 5.2	Wenn das 11. Datenbit des DW12 eins wird (der Zählwert damit 800 ist), wird der Merker M5.2 bedingt gesetzt.
Am Eingang E10.0 ist eine Lichtschranke installiert die Stückgut zählt. Nach jeweils 256 Stück soll der Zähler zurückgesetzt werden und von Neuem hochzählen.	: U E 10.0 : ZV Z 20 : U E 11.0 : L KZ 0 : S Z 20	Der Zählwert des Zählers 20 wird durch den Eingang E11.0 mit der Konstanten 0 geladen. Mit jedem positiven Flankenwechsel am E10.0 wird der Zählwert um 1 erhöht.
	: P Z 20.8 : SPB= VOLL[1] : BEA	Hat der Zählwert die Zahl 256 ≙ 100_H erreicht (das Bit 8 ist „1"), so wird zu der Marke „VOLL" gesprungen, ansonsten wird der Baustein beendet.
	VOLL: RU Z 20.8 : BE	Das Bit 8 des Zählers Z 20 wird unbedingt auf „0" gesetzt, damit steht im Zählwert wieder 000_H.

Hinweis:
Die Zeit- und Zählwerte sind im Zähl-/Zeitwort Hexadezimal in den 10 niederwertigsten Bit hinterlegt (Bit 0 bis Bit 9). Die Zeitbasis (-raster) ist in Bit 12 und 13 des Zeitwortes hinterlegt.

1) siehe Sprungfunktionen 6.2.4 (Seite 36)

6.2.2 Digitale Verknüpfungsfunktionen

Beispiel	AWL	Erläuterungen
Die Hexzahl 3F84H soll mit dem Eingangswort 1 der Eingänge EW 1 UND verknüpft werden. Das Ergebnis soll im Ausgangswort 10 der Ausgänge (AW 10) gezeigt werden. 3F84H EW 1 4793H Ergebnis UW 0780H	L KH 3F84 L EW 1 UW T AW 10	Die Hexzahl 3F84 wird in den Akku 1 geladen gleichzeitig wird der alte Inhalt von Akku 1 in den Akku 2 geschoben. Das Eingangswort 1 (EW 1) wird in den Akku 1 geladen und die Hexzahl in den Akku 2 geschoben. Der Inhalt von Akku 1 wird mit Akku 2 digital UND verknüpft und das Ergebnis in Akku 1 hinterlegt. Der Inhalt (Ergebnis) von Akku 1 wird zum Ausgangswort 10 transferiert.
Die beiden Bitmuster 0101 1110 1000 1011 und 0111 0001 0111 1100 sollen miteinander ODER verknüpft werden. 0101 1110 1000 1011 0111 0001 0111 1100 Ergebnis OW 0111 1111 1111 1111	L KM 01....11 L KM 01....00 OW T MW 13	Das erste Bitmuster wird in den Akku 1 geladen, gleichzeitig wird der alte Inhalt von Akku 1 in den Akku 2 geschoben. Das zweite Bitmuster wird in den Akku 1 geladen und das erste Bitmuster in den Akku 2 geschoben. Der Inhalt von Akku 1 wird mit Akku 2 digital ODER verknüpft und das Ergebnis in Akku 1 hinterlegt. Der Inhalt (Ergebnis) von Akku 1 wird ins Merkerwort 13 (MW 13) transferiert.
Das Eingangswort 5 soll mit dem Datenwort 12 auf Gleichheit überprüft werden. Die ungleichen Bits der Worte sollen im Ausgangswort 6 gezeigt werden. DW 12 EA83H EW 5 68C5H Ergebnis XOW 8246H	L DW 12 L EW 5 XOW T AW 6	Das Datenwort 12 wird in den Akku 1 geladen, gleichzeitig wird der alte Inhalt von Akku 1 in den Akku 2 geschoben. Das Eingangswort 5 wird in den Akku 1 geladen und das Datenwort 12 in den Akku 2 geschoben. Der Inhalt von Akku 1 wird mit Akku 2 digital EXOR verknüpft und das Ergebnis in Akku 1 hinterlegt. Der Inhalt (Ergebnis) von Akku 1 wird zum Ausgangswort 6 transferiert.

6.2.3 Rechenfunktionen

Beispiel	AWL	Erläuterung
Von der Zahl+127 soll das rechte Byte des Datenwortes 85 subtrahiert werden und das Ergebnis im linken Byte des Datenwortes 85 abgespeichert werden. 127 DR 85 74 Ergebnis–F 53	L KF +127 L DR 85 –F T DL 85	Die konstante Festpunktzahl + 127 wird in den Akku 1 geladen, gleichzeitig wird der alte Inhalt von Akku 1 in den Akku 2 geschoben. Das rechte Byte des Datenwortes 85 wird in den Akku 1 geladen und die Festpunktzahl + 127 in den Akku 2 geschoben. Der Inhalt von Akku 1 wird vom Akku 2 subtrahiert, und das Ergebnis in Akku 1 hinterlegt. Der Inhalt von Akku 1 (Ergebnis) wird ins linke Byte des Datenwortes 85 transferiert.

Hinweis:
Bei Überschreitung des Zahlbereiches (– 32768 bis + 32767) ist das Ergebnis der Operation undefiniert (OV = „1")

6.2.4 Sprungfunktionen

Das Sprungziel für unbedingte und bedingte Sprünge wird symbolisch angegeben (maximal 4 Zeichen). Dabei ist der Symbolparameter des Sprungbefehls identisch mit der Symboladresse der anzuspringenden Anweisung. Bei der Programmierung muß berücksichtigt werden, daß die absolute Sprungdistanz nicht mehr als ± 127 Wörter umfaßt und eine STEP-5-Anweisung aus mehr als einem Wort bestehen kann. Sprünge dürfen nur innerhalb eines Bausteins durchgeführt werden; Sprünge über Segmente hinweg sind nicht zulässig.

Beispiel	AWL	Erläuterung
Ist kein Eingang des Eingangswort 1 gesetzt, so wird zur Marke „AN 1" gesprungen. Stimmen Eingangswort 1 und Ausgangswort 3 nicht überein, so wird zur Marke „AN 0" zurückgesprungen. Stimmen EW 1 und AW 3 überein, so wird EW 1 mit dem Datenwort 12 verglichen. Ist EW 1 größer oder kleiner DW 12 so wird zur Marke „Ziel" gesprungen.	AN0:L EW1 :SPZ=AN1 :U E1.0 ⋮	Das Eingangswort 1 wird in den Akku 1 geladen. Ist Akkuinhalt 1 gleich Null, so wird zur Marke „AN 1" gesprungen, ansonsten wird die nächste Anweisung (UE 1.0) bearbeitet.
	AN1:L EW1 :L AW3 :XOW	Vergleich von Eingangswort 1 und Ausgangswort 3. Bei Ungleicheit sind einzelne Bits im Akku 1 gesetzt.
	:SPP=AN0	Ist der Akku 1 nicht null, so wird zur Marke „AN 0" zurückgesprungen, sonst nächste Anweisungen bearbeiten.
	:L EW1 :L DW12 :><F	Das Eingangswort 1 wird mit dem Datenwort 12 auf größere oder kleinere verglichen. Bei größer oder kleiner wird VKE „1" gesetzt.
	:SPB=Ziel	Ist VKE = „1", so wird zur Marke „Ziel" gesprungen. Ist VKE = „0", so wird die nächste Anweisung bearbeitet.
	⋮ ZIEL:U E12.2 ⋮	

6.2.5 Zeit- und Zählerfunktionen

Beispiel	AWL	Erläuterung
Eine Zeit T2 wird durch E 2.5 als verlängerter Impuls gestartet, mit 5Os Impulsbreite. Diese Zeit setzt den A 4.2 für die Dauer des Impulses. ⋮	U E 2.5 L KT 5.3 SV T 2 U T 2 = A 4.2 ⋮	Starten einer Zeit T2 als verlängerter Impuls. Der Ausgang 4.2 wird für 5Os gesetzt.
Wird der A 3.4 immer wieder gesetzt so soll die Zeit immer wieder vom neuen gestartet werden.	U A 3.4 FR T 2 BE	Wird der Ausgang 3.4 während der Zeit gesetzt (positiver Flankenwechsel des VKE) in der noch der Eingang 2.5 gesetzt ist, so wird die Zeit T2 neu gestartet. Das heißt der Ausgang 4.2 bleibt um die erneut gestartet Zeit gesetzt, oder wird von neuen gesetzt. Ist der Eingang 2.5 beim Flankenwechsel von Ausgang 3.4 nicht gesetzt, so wird die Zeit nicht neu gestartet.

6.2.6 Schiebefunktionen

Beispiel	AWL	Erläuterung
Bei der Hexzahl 14AFH im Datenwort 1 sollen die letzten 4 Bit wegfallen, und die sich daraus ergebene neue Hexzahl 014AH soll im Datenwort 3 abgespeichert werden.	L DW 1 SR W 4 T DW 3	Lade Datenwort 1 in den Akku 1. Schiebe den Inhalt von Akku 1 vier Bitstellen nach rechts. Die beim Schieben frei werdenden Bitstellen werden mit Nullen aufgefüllt. Transferiere den Inhalt von Akku 1 in das Datenwort 3.

Hinweis:
Die Schiebefunktionen werden unabhängig von Bedingungen ausgeführt. Das zuletzt hinausgeschobene Bit kann mit Sprungfunktionen abgefragt werden.
Mit SPZ kann gesprungen werden, wenn das Bit „0" ist und mit SPN oder SPP wenn das Bit „1" ist.

6.2.7 Umwandlungsfunktionen

Beispiel	AWL	Erläuterung
Der Inhalt des Datenwortes 64 soll Bit für Bit invertiert werden und im Datenwort 78 abgelegt werden.	L DW 64	Lade Datenwort 64 in den Akku 1.
	KEW	Einerkomplement des Inhalt von Akku 1. Ergebnis befindet sich im Akku 1.
DW 64 EA83H DW 78 157CH	T DW 78	Transferiere den Inhalt von Akku 1 ins Datenwort 78.
Der Inhalt des Datenwortes 42 ist als Festpunktzahl zu interpretieren und mit umgekehrten Vorzeichen ins Datenwort 35 abzulegen.	L DW 42	Lade Datenwort 42 in den Akku 1.
	KZW	Zweierkomplement des Inhalt von Akku 1. Ergebnis befindet sich im Akku 1.
DW 42 +51 DW 35 −51	T DW 35	Transferiere den Inhalt von Akku 1 ins Datenwort 35.

6.2.8 Dekrementieren/Inkrementieren

Beispiel	AWL	Erläuterung
Die Hexkonstante 1010H soll um das Inkrement 16 erhöht werden und im Datenwort 8 abgelegt werden. Außerdem soll das Ergebnis vom Inkrementieren um das Dekrement 33 erniedrigt werden und im Datenwort 9 abgelegt werden.	L KH 1010	Lade Hexkonstante 1010H in den Akku 1.
	I 16	Inkrementiere das Low-Byte von Akku 1 um 16. Das Ergebnis 1020H befindet sich im Akku 1
	T DW 8	Transferiere den Inhalt von Akku1 (1020H) in das Datenwort 8. Da im Akku 1 noch das Ergebnis vom Inkrementieren steht, kann man direkt das Dekrement 33 davon bilden.
	D 33	Das Ergebnis wäre FFFH. Da aber das High-Byte des Akku 1 nicht mit dekrementiert wird steht im Akku 1 10FFH als Ergebnis.
	T DW 9	Der Inhalt von Akku 1 wird ins Datenwort 9 transferiert (10FFH).

Hinweis:
Die In- und Dekrementation erfolgt immer Dezimal, während das Ergebnis immer Hexadezimal im Akku 1 abgelegt wird.

6.2.9 Befehlsausgabe sperren/freigeben

Beispiel	AWL	Erläuterung
Wird Eingang 0.5 gesetzt sollen die darauf folgenden Befehle gesperrt werden.	U E 0.5	Abfrage des Eingangs 0.5 auf Signalzustand „1"
	BAS	Befehlsausgabe sperren bei gesetzten Eingang 0.5 (VKE ≙ „1")
	⋮	
	U E 0.6	Abfrage des Eingangs 0.6 auf Signalzustand „1"
Wird Eingang 0.6 gesetzt und Eingang 0.5 zurückgesetzt so soll die Befehlssperre aufgehoben werden.	UN E 0.5	Abfrage des Eingangs 0.5 auf Signalzustand „0"
	BAF	Befehlsausgabe freigeben, wenn Eingang 0.6 gesetzt und 0.5 nicht gesetzt ist (VKE ≙ „1").

Hinweis:
„Befehlsausgabe sperren/freigeben" kann z. B. eingesetzt werden, um eine Ablaufkette auf einen bestimmten Schritt nachzuführen, ohne die Ausgänge der durchlaufenen Schritte zu setzen oder rückzusetzen.

6.2.10 Alarme sperren/freigeben

Beispiel	AWL	Erläuterung
Alarmbearbeitung in einem bestimmten Programmteil sperren und dann wieder freigeben.	⋮	
	= A 7.5	
	AS	Alarme sperren
	U E 2.3	
	⋮	
	SPA FB 3	Tritt ein Alarm auf so wird beim Sprung in den FB 3 nicht erst in den Alarmbaustein FB 0 gesprungen sondern das Programm ganz normal abgearbeitet.
	⋮	
	AF	Alarm freigeben Liegt der Alarm jetzt noch an, so wird bei einem Bausteinwechsel zuerst in den Alarmbaustein FB 0 gesprungen.
	⋮	

Hinweis:
„Alarme sperren/freigeben" kann z. B. angewendet werden, wenn bei einer zeitgesteuerten Bearbeitung die alarmgesteuerte Bearbeitung unterdrückt werden soll (siehe 5.4 Programmierung der alarmgesteuerten Bearbeitung Seite 15).

6.2.11 Bearbeitungsfunktionen

Beispiel	AWL	Erläuterung
Im Datenwort 12 sollen immer die aktuellen Parameter der Eingänge hinterlegt werden, die man abfragt.	B DW12	Bearbeite Datenwort 12. Steht im Datenwort 12 High-Byte 6 Low-Byte 43 so wird der Eingang 43.6 auf Signalzustand „1" abgefragt.
	U E 0.0	
	⋮	
Im Datenwort 13 sollen immer die aktuellen Parameter der Zeiten hinterlegt werden, die man erneut freigeben will.	B DW13	Bearbeite Datenwort 13. Steht im Datenwort 13 High-Byte 0 Low-Byte 15 so wird die Zeit 15 durch den Eingang 43.6 zum Neustart freigegeben.
	FR T0	
Es sollen die Inhalte der Datenwörter DW20 bis DW100 auf Signalzustand „0" gesetzt werden. Das „Indexregister" für den Parameter der Datenwörter ist DW0.	:L KB20	Lade konstante Zahl 20 in Akku 1.
	:T DW1	Transferie Inhalt von Akku 1 ins Datenwort 1.
	M1 :L KH0	Lade Hexkonstante 0 in Akku 1.
	:B DW1	Bearbeite Datenwort 1
	:T DW0	Transferiere den Inhalt von Akku 1 ins Datenwort, dessen Adresse im Datenwort 1 hinterlegt ist.
	:L DW1	Lade Datenwort 1 in den Akku 1.
	:L KB1	Lade konstante Zahl 1 in Akku 1. Datenwort 1 wird in Akku 2 geschoben.
	:+F	Akku 2 und Akku 1 werden zusammenaddiert und das Ergebnis in Akku 1 hinterlegt (Erhöhung der Datenwortadresse).
	:T DW1	Transferie Inhalt von Akku 1 ins Datenwort 1 (neue Datenwortadresse).
	:L KB100	Die konstante Zahl 100 wird in Akku 1 geladen und die neue Datenwortadresse in Akku 2 geschoben.
	:<=F	Vergleich der Akku's auf kleiner gleich Akku 2 ≤ Akku 1
	:SPB=M1	Springe bedingt zur Marke M1 solange Akku 2 ≤ Akku 1 ist.

Hinweis:
Das Programmiergerät 670 prüft die Zulässigkeit der Kombination der Parameter mit den Operanden nicht ab. Zu der angegebenen Operation wird der Parameter aus dem Daten- oder Merkerwort zugeordnet. Das High-Byte des Merker- oder Datenwortes wird nur für Ein/Ausgänge und für Merker benötigt (zwischen 0 und 7), ansonsten muß es null sein.

Die Operation „UE" wird in Kombination mit „B DW" und „B MW" zur Operation „UA", wenn die Byteadresse im Daten/Merkerwort größer 127 ist.
Wird zu den Operationen, die mit dem „B DW" oder „B MW" kombiniert werden, ein Parameter ≠ 0 angegeben, so wird keine „Adressrechnung" durchgeführt, sondern die beiden Parameter werden nach einer ODER-Funktion verknüpft.

Folgende Operationen können mit dem BDW/BMW kombiniert werden:

U-, UN-, O-, ON-	Binäre Verknüpfungen
S-, R-, =-	Speicherfunktionen
FRT, RT, SA T, SE T, SI T, SS T, SV T	Zeitfunktionen
FRZ, R Z, S Z, ZR Z, ZV Z	Zählfunktionen
L-, LC-, T-	Lade- und Transferfunktionen
SPA, SPB, SPZ, SPN, SPP, SPM, SPO	Sprungfunktionen
SLW, SRW	Schiebefunktionen
D, I	Dekrementieren, Inkrementieren
A DB, SPA-, SPB-	Bausteinaufrufe

6.2.12 Substitutionsfunktionen

Das AG führt bei der Bearbeitung des Step-5 Programmes innerhalb eines Funktionsbausteins eine „Substitution" durch, wenn der Operand ein Formaloperand ist. (d. h. = HANS steht für E 1.5 siehe Seite 9).

Der Formaloperand wird beim Aufruf des Funktionsbausteins durch einen richtigen Operanden ersetzt (subtituiert).

UND-ODER Verknüpfung mit RS-Speicherglied

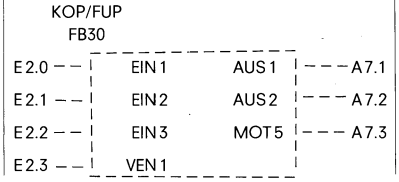
Programm im Funktionsbaustein (FB30)

```
U     =EIN 1
UN    =EIN 2
O     =EIN 3
S     =MOT 5
=     =AUS 1
U     =VEN 1
U     =EIN 2
ON    =EIN 3
RB    =MOT 5
=     =AUS 2
BE
```

Funktionsbausteinaufruf

```
AWL
      : SPA FB 30
NAME: VERKNUE.
EIN 1:     E 2.0
EIN 2:     E 2.1
EIN 3:     E 2.2
VEN 1:     E 2.3
AUS 1:     A 7.1
AUS 2:     A 7.2
MOT5:      A 7.3
      : BE
```

KOP/FUP



ausgeführtes Programm

```
:U     E 2.0
:UN    E 2.1
:0     E 2.2
:S     A 7.3
:=     A 7.1
:U     E 2.3
:U     E 2.1
:ON    E 2.2
:R     A 7.3
:=     A 7.2
:BE
```

Operation		Beschreibung
U	=....	Und-Funktion. Abfrage eines Formaloperanden auf Signalzustand „1" [Operanden E,A,M,T,Z].
UN	=....	Und-Funktion. Abfrage eines Formaloperanden auf Signalzustand „0" [Operanden E,A,M,T,Z].
O	=....	Oder-Funktion. Abfrage eines Formaloperanden auf Signalzustand „1" [Operanden E,A,M,T,Z].
ON	=....	Oder-Funktion. Abfrage eines Formaloperanden auf Signalzustand „0" [Operanden E,A,M,T,Z].
S	=....	Setzen (binär) eines Formaloperanden [Operanden E,A,M].
RB	=....	Rücksetzen (binär) eines Formaloperanden [Operanden E,A,M].
=	=....	Zuweisung des Verknüpfungsergebnisses an einen Formaloperanden [Operanden E,A,M].

Lade- und Transferfunktionen

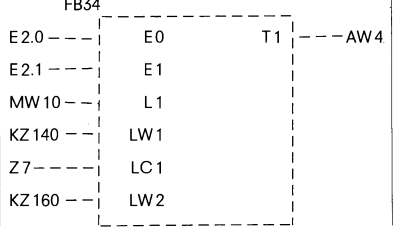
Programm im Funktionsbaustein (FB34)

```
:U     =E0
:L     =L1
:S     Z6
:U     =E1
:LW    =LW1
:S     Z7
:U     E2.2
:ZV    Z6
:ZV    Z7
:LC    =LC1
:T     =T1
:U     E2.7
:R     Z6
:R     Z7
:LW    =LW2
:LC    =LC1
:!=F
:R     Z7
:BE
```

Funktionsbausteinaufruf

```
AWL
      : SPA FB 34
NAME: LADE/TRAN
E0  :      E  2.0
E1  :      E  2.1
L1  :      MW10
LW1 :      KZ 140
LC1 :      Z  7
T1  :      AW 4
LW2 :      KZ 160
      : BE
```

KOP/FUP



ausgeführtes Programm

```
:U     E   2.0
:L     MW 10
:S     Z   6
:U     E   2.1
:L     KZ  140
:S     Z   7
:U     E   2.2
:ZV    Z   6
:ZV    Z   7
:LC    Z   7
:T     AW  4
:U     E   2.7
:R     Z   6
:R     Z   7
:L     KZ  160
:LC    Z   7
:!=F
:R     Z   7
:BE
```

Operation		Beschreibung
L	=....	Laden eines Formaloperanden. Der Wert des als Formaloperanden vorgegebenen Operanden wird in den Akku 1 geladen [Operanden EB,EW,MB,MW,AB,AW,DR,DL,DW,PB,PW].
LC	=....	Laden codiert eines Formalopranden. Der Wert der als Formaloperand vorgegebenen Zeit- oder Zählzelle wird BCD-codiert in den Akku 1 geladen [Operanden T,Z].
LW	=....	Laden des Bitmusters eines Formaloperanden. Das Bitmuster des Formaloperanden wird in den Akku 1 [Operanden KB, KC, KF, KH, KM, KY, KT, KZ].
T	=....	Transferieren zu einem Formaloperanden. Der Akku-Inhalt wird zu dem als Formalopranden vorgegebenen Operanden transferiert [Operanden EB,EW,MB,MW,AB,AW,DR,DL,DW,PB,PW].

Zeitfunktionen

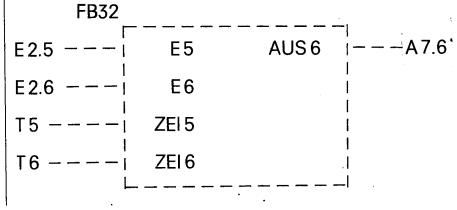
Programm im Funktionsbaustein (FB32)

```
:UN     =E5
:U      =E6
:L      KT5.2
:SAR    =ZEI5
:U      =E5
:UN     =E6
:L      KT5.2
:SSV    =ZEI6
:U      ZEI5
:O      =ZEI6
:=      =AUS6
:U      E2.7
:RD     =ZEI5
:RD     =ZEI6
:BE
```

Funktionsbausteinaufruf

```
AWL
        : SPA FB 32
   NAME: ZEIT
   E5   :     E  2.5
   E6   :     E  2.6
   ZEI  5:    T  5
   ZEI  6:    T  6
   AUS  6:    A  7.6
        : BE
```

KOP/FUP



ausgeführtes Programm

```
:UN     E  2.5
:U      E  2.6
:L      KT 5.2
:SA     T  5
:U      E  2.5
:UN     E  2.6
:L      KT 5.2
:SS     T  6
:U      T  5
:O      T  6
:=      A  7.6
:U      E  2.7
:R      T  5
:R      T  6
:BE
```

Operation		Beschreibung
SI	=....	Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Impuls [Operand T].
SVZ	=....	Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als verlängerter Impuls [Operanden T und Z (siehe Zählfunktionen)].
SE	=....	Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Einschaltverzögerung [Operand T].
SAR	=....	Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Ausschaltverzögerung [Operanden T und Z (siehe Zählfunktionen)].
SSV	=....	Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als speichernde Einschaltverzögerung [Operanden T und Z (siehe Zählfunktionen)].
FR	=....	Freigabe eines Formaloperanden für Neustart [Operanden T und Z (siehe Zählfunktionen)].
RD	=....	Rücksetzen (digital) eines Formaloperanden [Operanden T und Z].

Zählfunktionen

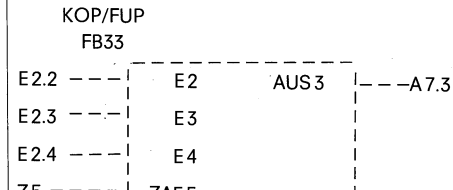
Programm im Funktionsbaustein (FB33)

```
:U      =E2
:L      KZ17
:SVZ    =ZAE5
:U      =E3
:SSV    =ZAE5
:U      =E4
:SAR    =ZAE5
:U      =ZAE5
:=      =AUS3
:U      E2.7
:RD     =ZAE5
:BE
```

Funktionsbausteinaufruf

```
AWL
        : SPA FB 33
   NAME: ZAEHL
   E2   :     E  2.2
   E3   :     E  2.3
   E4   :     E  2.4
   ZAE5 :     Z  5
   AUS3 :     A  7.3
        : BE
```

KOP/FUP

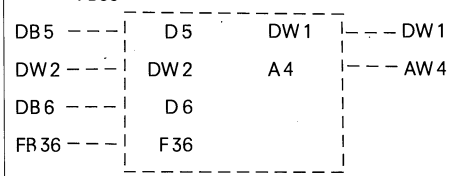


ausgeführtes Programm

```
:U      E  2.2
:L      KZ 17
:S      Z  5
:U      E  2.3
:ZV     Z  5
:U      E  2.4
:ZR     Z  5
:U      Z  5
:=      A  7.3
:U      E  2.7
:R      Z  5
:BE
```

Operation		Beschreibung
SVZ	=....	Setzen eines als Formaloperand vorgegebenen Zählers mit dem vorher geladenen Zählwert [Operanden Z und T (siehe Zeitfunktion)].
SSV	=....	Vorwärtszählen eines als Formaloprand vorgegebenen Zählers [Operanden Z und T (siehe Zeitfunktionen)].
SAR	=....	Rückwärtszählen eines als Formaloperand vorgegebenen Zählers [Operanden Z und T (siehe Zeitfunktionen)].
FR	=....	Freigabe eines Formaloperanden für Neustart [Operanden Z und T (siehe Zeitfunktionen)].
RD	=....	Rücksetzen (digital) eines Formaloperanden [Operanden Z und T].

Bearbeitungsfunktionen

Programm im Funktionsbaustein (FB35)	Funktionsbausteinaufruf	ausgeführtes Programm
:B =D5 :L =DW2 :B =D6 :T =DW1 :T =A4 :B =F36 :BE	AWL : SPA FB 35 NAME: BEARB. D5 : DB 5 DW2 : DW2 D6 : DB 6 DW1 : DW1 A4 : AW4 F36 : FB 36 : BE KOP/FUP FB35  	:A DB 5 :L DW 2 :A DB 6 :T DW 1 :T AW 4 :SPA FB 36 :BE

Operation	Beschreibung
B =....	Bearbeite Formaloprand Nur die Operationen A DB SPA PB SPA FB können substituiert.

7. Regeln zur Kompatibilität zwischen den Darstellungsarten KOP–FUP–AWL

7.1 Allgemeines

Jede Darstellungsart der Programmiersprache STEP 5 beinhaltet eigene Eigenschaften und bestimmte Grenzen.
Daraus ergibt sich, daß ein in AWL geschriebener Programmbaustein nicht ohne weiteres in KOP oder FUP ausgegeben werden kann, und daß darüber hinaus, die graphischen Darstellungsarten KOP und FUP gegebenenfalls nicht vollständig kompatibel sein können.
In anderen Worten, die Rückübersetzbarkeit ist nicht immer gegeben.
Wurde das Programm in KOP oder FUP eingegeben so ist es grundsätzlich in AWL rückübersetzbar.

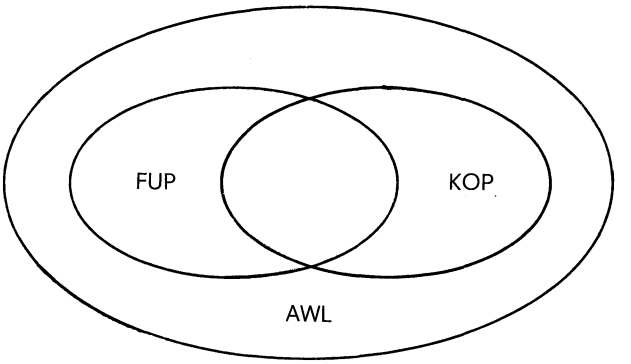


Bild 22 Umfang bzw. Begrenzungen der Darstellungsarten der Programmiersprache STEP 5.

Ziel dieses Abschnittes ist die Aufstellung einiger Regeln, deren Einhaltung eine vollständige Kompatibilität der drei Darstellungsarten gewährleistet.
Diese Regeln teilen sich folgendermaßen auf.

- Kompatibilitätsregeln zwischen den graphischen Darstellungsarten.

Die Einhaltung dieser Regeln ermöglicht die Eingabe in einer graphischen Darstellungsart und die entsprechende Ausgabe in den übrigen Darstellungsarten.

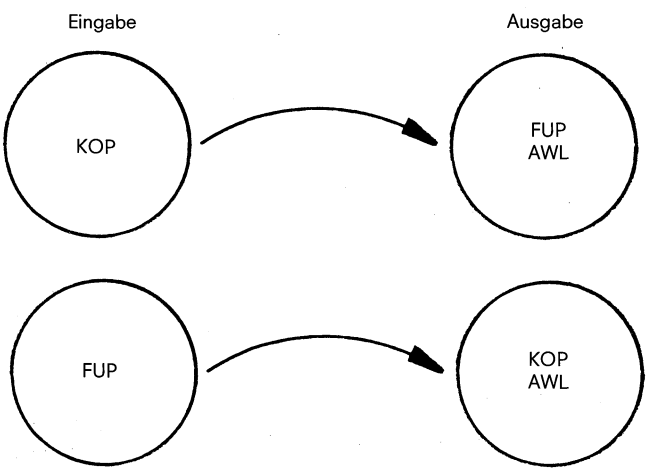


Bild 23 Graphische Eingabe

- Kompatibilitätsregeln zwischen Anweisungsliste und graphischen Darstellungsarten.

Die Einhaltung dieser Regeln gewährleistet die Eingabe in einer frei wählbaren Darstellungsart, sei sie graphisch oder nicht, und die entsprechende Ausgabe in den anderen zwei Darstellungsarten.

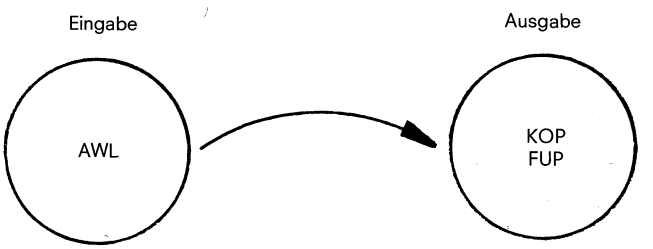
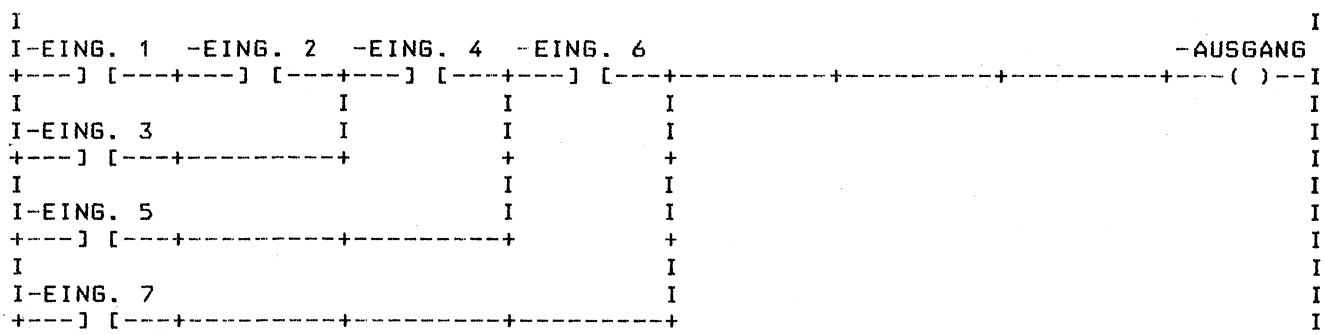


Bild 24 Eingabe in Anweisungsliste

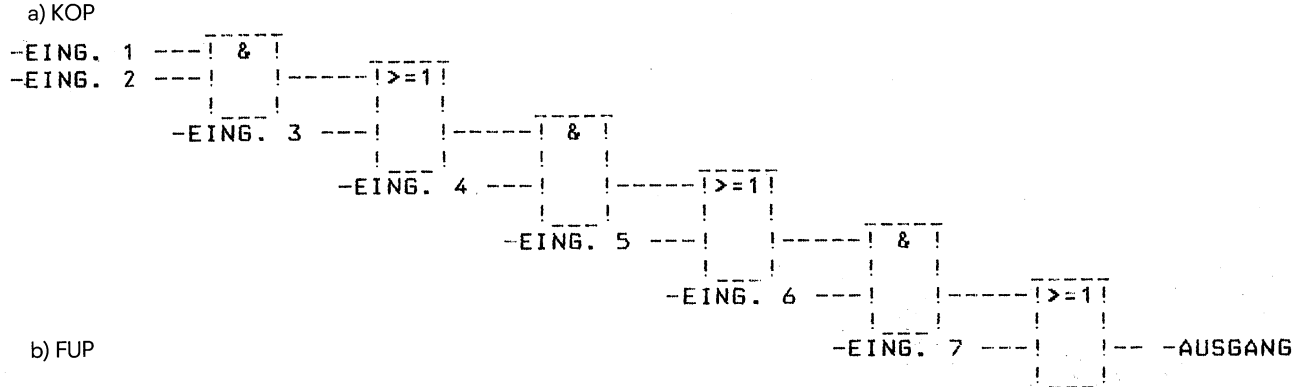
7.2 Kompatibilitätsregeln zwischen den graphischen Darstellungsarten

7.2.1 Eingabe in KOP, Ausgabe in FUP (AWL)

Regel: Bildgrenzen für FUP nicht überschreiten.
Eine zu große Schachtelung kann im FUP zur Überschreitung der Bildgrenzen (8 Ebenen) führen.



a) KOP

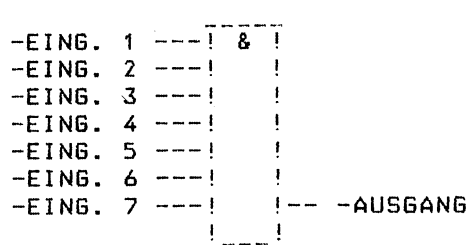


b) FUP

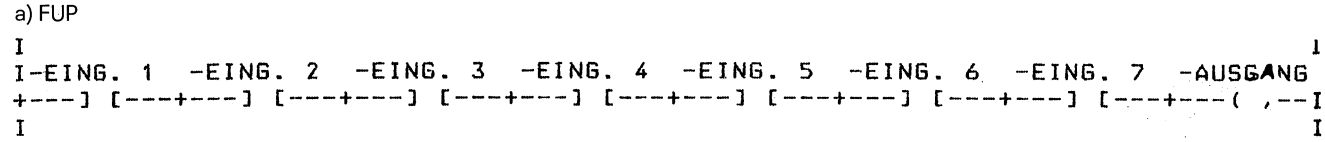
Bild 25 Beispiel einer maximalen KOP-Schachtelung zur Ausgabe in FUP

7.2.2 Eingabe in FUP; Ausgabe in KOP (AWL)

Regel 1: Bildgrenzen für KOP nicht überschreiten.
Eine zu große Anzahl von Eingängen an einem FUP-Kasten führt zur Überschreitung der KOP-Bildgrenze.



a) FUP



b) KOP

Bild 26 Beispiel einer maximalen UND-Kastenausbaues in FUP zur Ausgabe in KOP

Regel 2: Der Ausgang eines komplexen Gliedes (Speicher, Vergleicher, Zeiten und Zähler) darf nicht mit ODER weiterverknüpft werden.

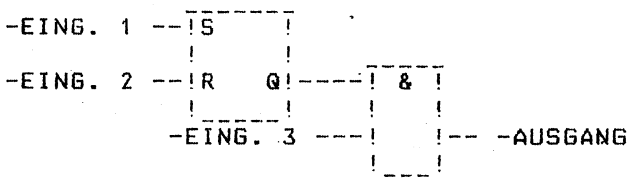


Bild 27 Nur UND-Kasten sind in FUP nach einem komplexen Glied zuläßig.

Regel 3: Konnektoren

- Konnektoren bei ODER-Kasten immer erlaubt.
- Konnektoren bei UND-Kasten nur am ersten Eingang erlaubt.

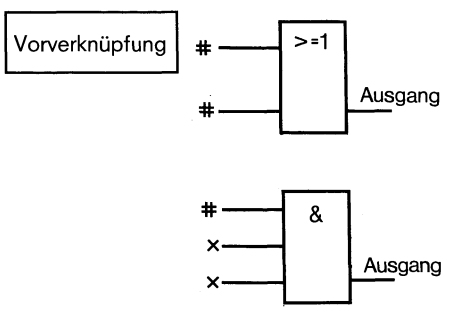


Bild 28 Beispiele zu der Zulässigkeit der Konnektoren bei den ODER- und UND-Kästen (#Konnektoren erlaubt; × Konnektoren nicht erlaubt).

Hinweis:
Konnektoren sind Zwischenmerker um immer wiederkehrende Verknüpfungen einzusparen.

7.3 Kompatibilitätsregeln zwischen AWL und graphischen Darstellungsarten

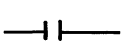
Regel 1: UND-Verknüpfung
Abfrage des Signalzustandes und Verknüpfung nach UND

KOP: Kontakt in Reihe

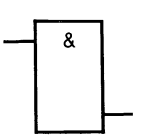
FUP: Eingang eines UND-Kastens

AWL: Anweisung U . . .

KOP:



FUP:



AWL: U . . .

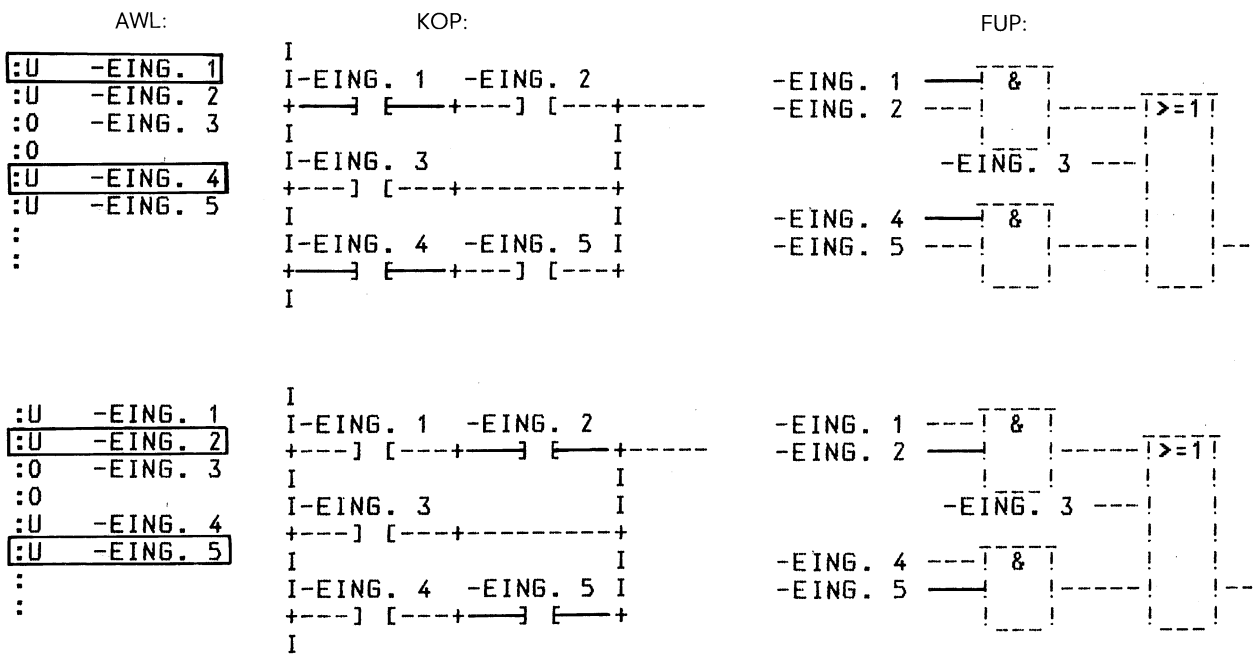


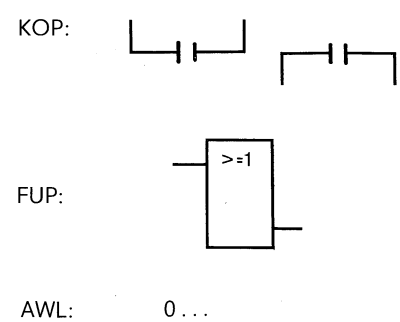
Bild 29 Erläuterung zur UND-Verknüpfung

Regel 2: ODER-Verknüpfung
Abfrage des Signalzustandes und Verknüpfung nach ODER

KOP: Nur ein Kontakt in einem Parallelzweig

FUP: Eingang eines ODER-Kasten

AWL: Anweisung 0 . . .



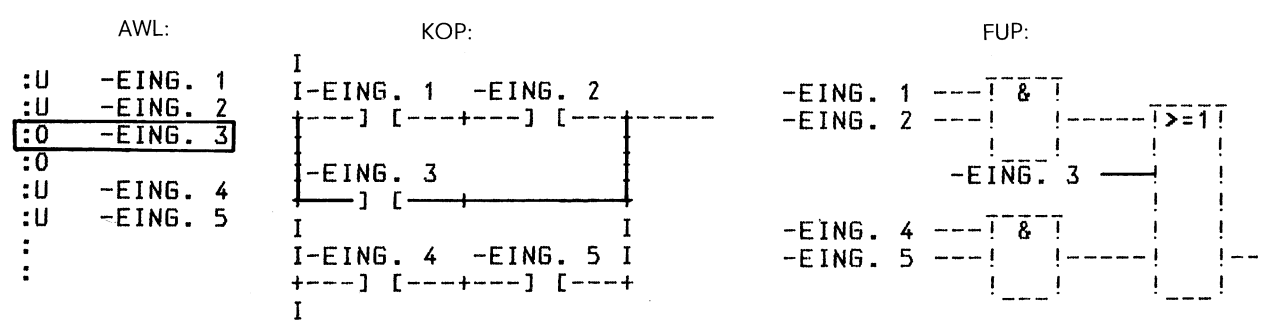


Bild 30 Erläuterung zur ODER-Verknüpfung

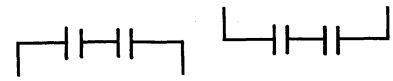
7. Regeln zur Kompatibilität zwischen den Darstellungsarten KOP–FUP–AWL

7.3 Kompatibilitätsregeln zwischen AWL und graphischen Darstellungsarten

Regel 3: UND vor ODER Verknüpfung
Oder-Verknüpfungen von Und-Funktionen

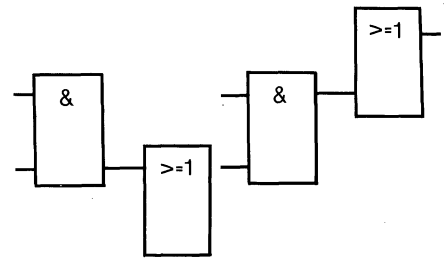
KOP: mehrere Kontakte in einem Parallelzweig

KOP:



FUP: Kasten vor ODER-Kasten

FUP:



AWL: Anweisungen O
U...
U...

AWL:

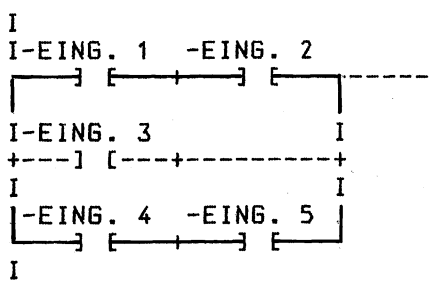
U...
U...

O
U...
U...

AWL:

```
:U   -EING. 1
:U   -EING. 2
:O   -EING. 3
:O
:U   -EING. 4
:U   -EING. 5
```

KOP:



FUP:

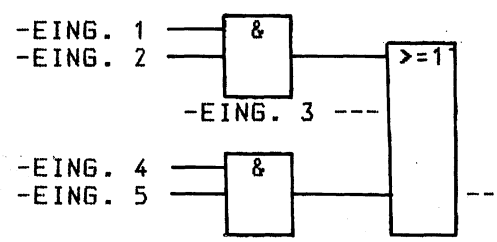


Bild 31 Erläuterungen zur UND-vor-ODER Verknüpfung

Regel 4: Klammerung
In dieser Regel wird die Klammerung von komplexen in sich abgeschlossenen binären Verknüpfungen, oder von komplexen Gliedern mit Vor- und Nachverknüpfungen behandelt.

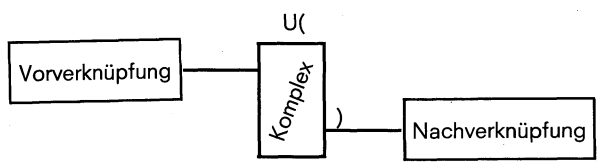


Bild 32 Erläuterung zur Klammerung

a) Komplexe binäre Verknüpfung
In diese Klasse gehört die ODER vor UND Verknüpfung dessen Regeln wie folgt lauten:

- UND verknüpfen von ODER-Funktionen
  KOP: Parallelkontakte in Serie weiterschalten
  FUP: ODER-Kasten vor UND-Kasten
  AWL: Anweisungen

```
U(
Oder-
Verknüpfung
)
.
.
.
```

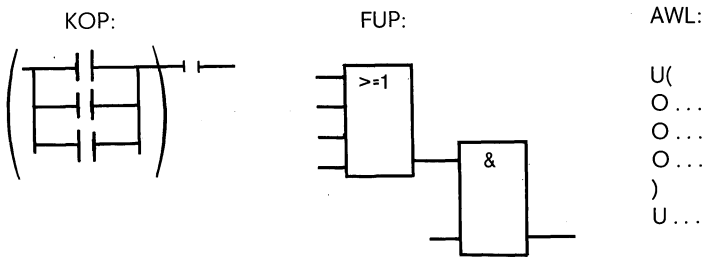


Bild 33 Erläuterung zur ODER vor UND Verknüpfung

Die ODER vor UND Verknüpfungen stellen eine Untermenge der komplexen binären Verknüpfungen dar, wobei parallele Kontakte die einfachste komplexe binäre Verknüpfung ist.

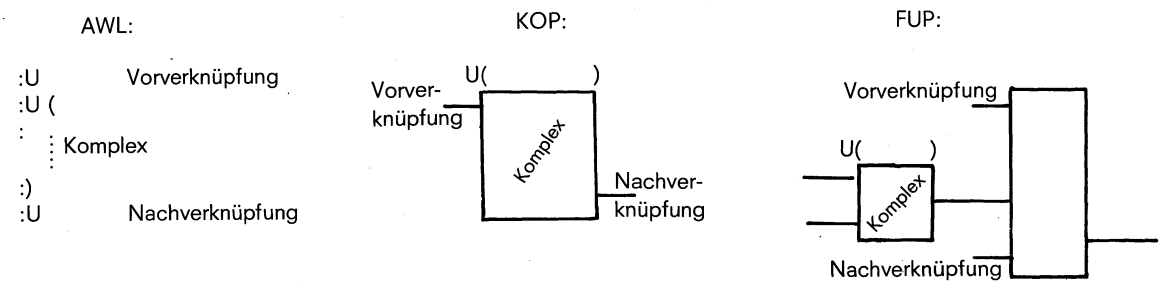


Bild 34 Erläuterungen zur Klammerung von komplexen binären Funktionen.

b) Komplexe Glieder (Speicher-, Zeit-, Vergleich- oder Zählfunktionen).

Für die komplexen Glieder müssen folgende Regeln eingehalten werden.

- Keine Nachverknüpfung vorhanden – keine Klammerung
- Nachverknüpfung UND U(. . . .)
- Nachverknüpfung ODER (nur für FUP, bei KOP nicht erlaubt) O(. . . .)
- Ein komplexes Glied kann keine Vorverknüpfung haben.

Außerdem muß jeder unbeschaltete Ein- oder Ausgang mit NOP 0 versorgt werden.

Ausnahme: S, TW bei Zeiten und S, ZW bei Zählern müssen stets gemeinsam beschaltet sein.

Bei der Programmierung in AWL sind die komplexen Glieder in derselben Reihenfolge zu programmieren, wie sie am Bildschirm in graphischer Darstellungsart parametriert werden.

Ausnahme: Zeit- und Zählwert. Der entsprechende Wert muß vorher in den Akkumulator durch einen Ladebefehl hinterlegt werden.

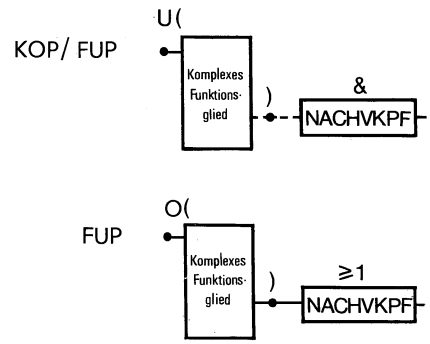


Bild 35 Erläuterungen zur Klammerung von komplexen Gliedern.

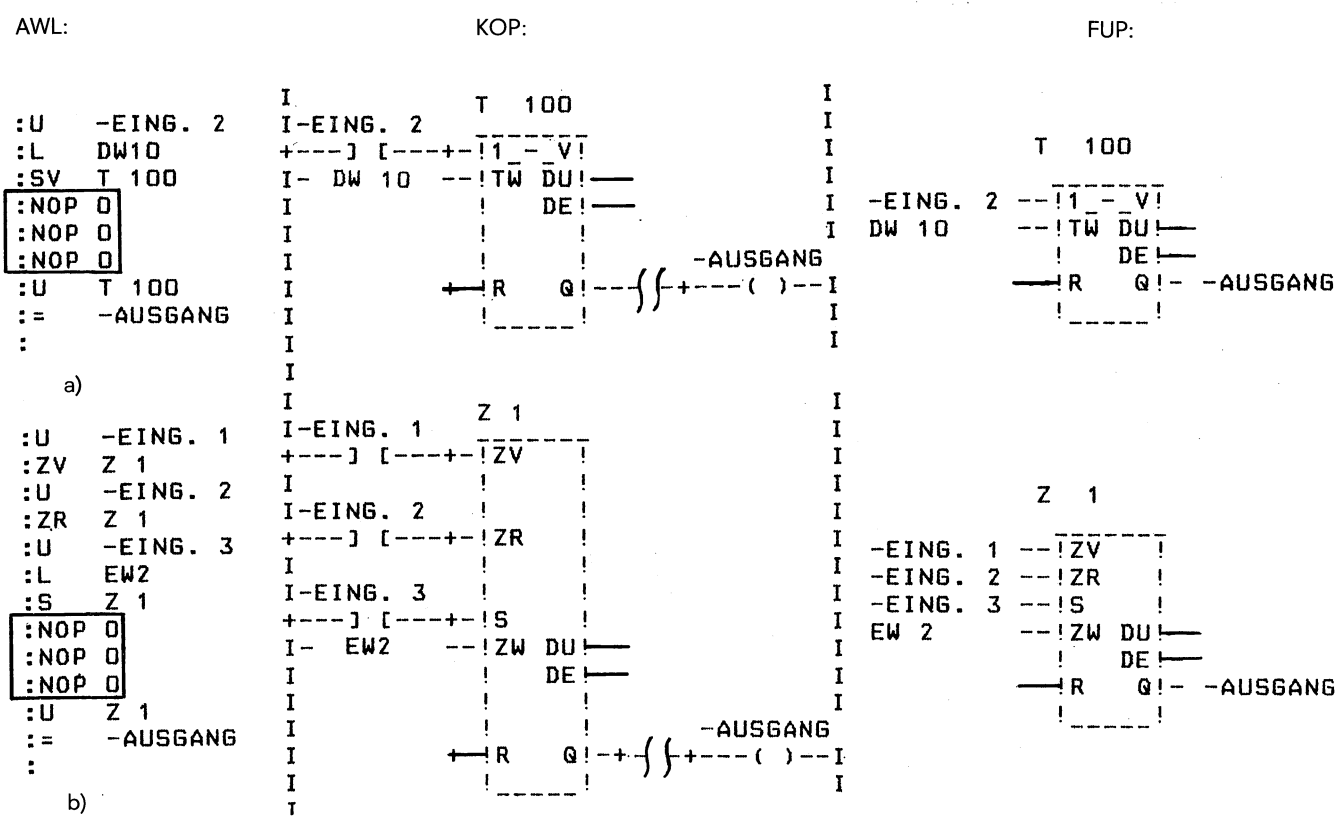


Bild 36 Beispiel zur Versorgung unbeschalteter Ein- Ausgänge a) bei Zeiten b) bei Zählern

Hinweis: Pro Segment ist nur ein komplexes Funktionsglied zulässig

Die folgenden Beispiele zeigen die vier vorgeführten Fälle in einer komplexen binären Verknüpfung einmal in den Darstellungsarten AWL und KOP und einmal in AWL und FUP.

Beispiel 1: AWL/KOP
Fall 1: UND (Kontakte in Reihe)
Fall 2: ODER (nur 1 Kontakt in einem Parallel-Zweig)
Fall 3: UND-vor-ODER (mehrere Kontakte in einem Parallel-Zweig)
Fall 4: ODER-vor-UND (Klammerung)

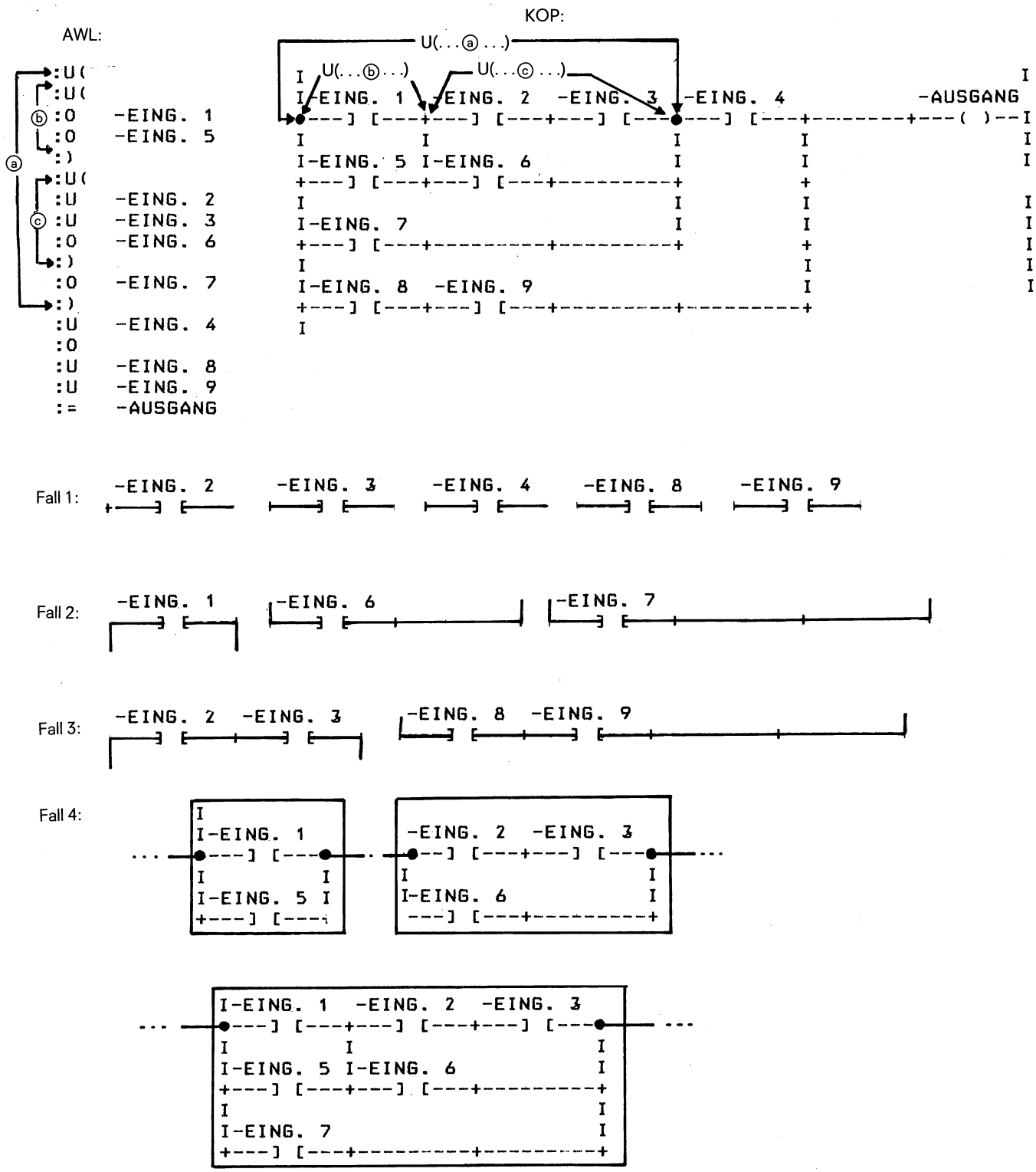


Bild 37 Beispiel 1: AWL/KOP

7. Regeln zur Kompatibilität zwischen den Darstellungsarten KOP–FUP–AWL

7.3 Kompatibilitätsregeln zwischne AWL und graphischen Darstellungsarten

Beispiel 2: AWL/FUP
Fall 1: UND (Eingang eines UND-Kasten)
Fall 2: ODER (Eingang eines ODER-Kasten)
Fall 3: UND-vor-ODER (UND-Kasten vor ODER-Kasten)
Fall 4: ODER-vor-UND (ODER-Kasten vor UND-Kasten)

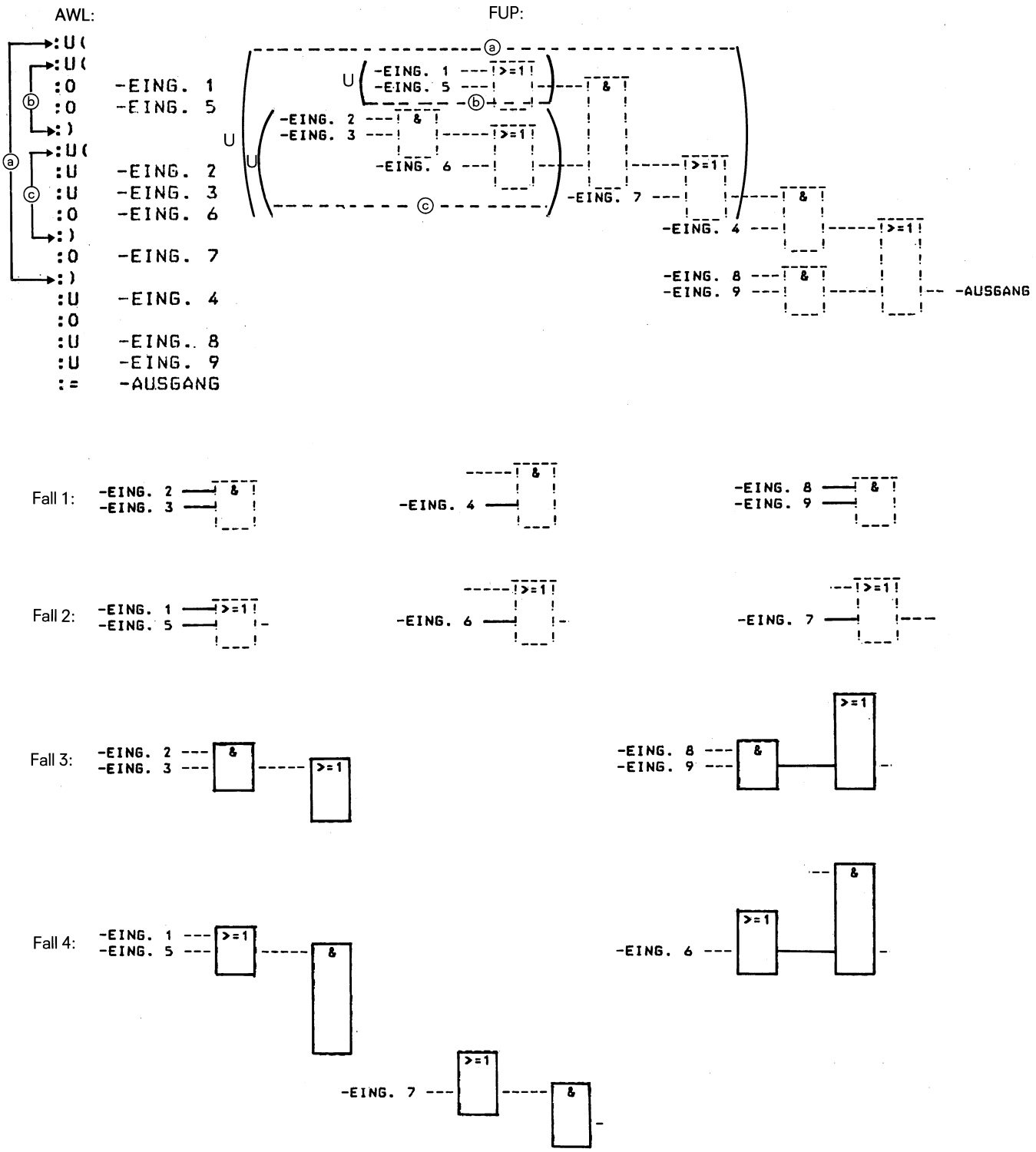


Bild 38 Beispiel 2: AWL/FUP

Regel 5: Konnektoren

Der Klarheit wegen werden die Regeln für Konnektoren getrennt für die Darstellungsarten KOP und FUP aufgelistet. Anschließend folgt ein gemeinsames Beispiel.

a) bei KOP
Ein Konnektor merkt sich das Verknüpfungsergebnis als Zwischenspeicher von den Operationen die vor ihm in der eigenen Stromschiene programmiert worden sind.
Dabei gelten folgende Regeln:

AWL:
```
= M...
U M...
```

KOP:

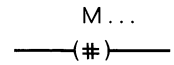


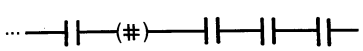
Bild 39 Der Konnektor in AWL und KOP

- Konnektor in Reihe (in Serie mit anderen Kontakten).
  Ein Konnektor wird in diesem Fall wie ein normaler Kontakt behandelt.

AWL:
```
.
.
.
U...
U...
U...
= M...
U M...
U
.
.
.
```

- Konnektor in einem Parallel-Zweig Innerhalb eines Parallelzweiges wird ein Konnektor wie ein normaler Kontakt behandelt.
  Zusätzlich muß der gesamte Parallel-Zweig in eine Klammerung des Typs O(. . .) eingeschlossen werden.

KOP:

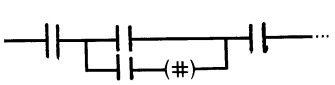


a)

- Ein Konnektor darf nie unmittelbar nach der Stromschiene (Konnektor als erster Kontakt) oder direkt nach einer Eröffnung einer Stromschiene (Konnektor als erster Kontakt in einen Parallel-Zweig) stehen.

AWL:
```
.
.
.
U...
U(
U...
O(
U...
= M...
U M...
)
)
U...
.
.
.
```

KOP:



b)

Bild 40 Konnektor-Regeln für KOP a) Konnektor in Reihe, b) Konnektor im Parallelzweig

b) bei FUP
Ein Konnektor merkt sich das Verknüpfungsergebnis als Zwischenspeicher der gesamten binären Verknüpfung vor diesem Konnektor. Dabei gelten folgende Regeln:

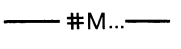
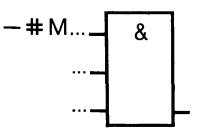
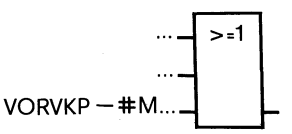
AWL:
```
= M...
U M...
```
FUP:


Bild 41 der Konnektor in AWL und FUP

- Konnektor am ersten Eingang eines UND-bzw. ODER-Kastens. Der Konnektor wird ohne Klammerung abgesetzt.

AWL:
```
= M...
U M...
U...
U...
```
FUP:


- Konnektor nicht am ersten Eingang eines ODER-Kastens. Die gesamte binäre Verknüpfung vor dem Eingang wird in eine Klammerung des Typs O(. . .) eingeschlossen.

AWL:
```
.
.
.
O...
O(
VORVKP
= M...
U M...
)
.
.
.
```
FUP:


- Konnektor nicht am ersten Eingang eines UND-Kastens. Die gesamte binäre Verknüpfung vor dem Eingang wird in eine Klammerung des Typs U(. . .) eingeschlossen. Nur bei FUP erlaubt (bei KOP graphisch nicht darstellbar).

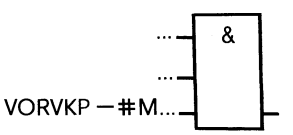
AWL:
```
.
.
.
U...
U(
VORV
= M...
U M...
)
.
.
.
```
FUP:


Bild 42 Konnektor-Regeln für FUP

Beispiele zu den Konnektoren:

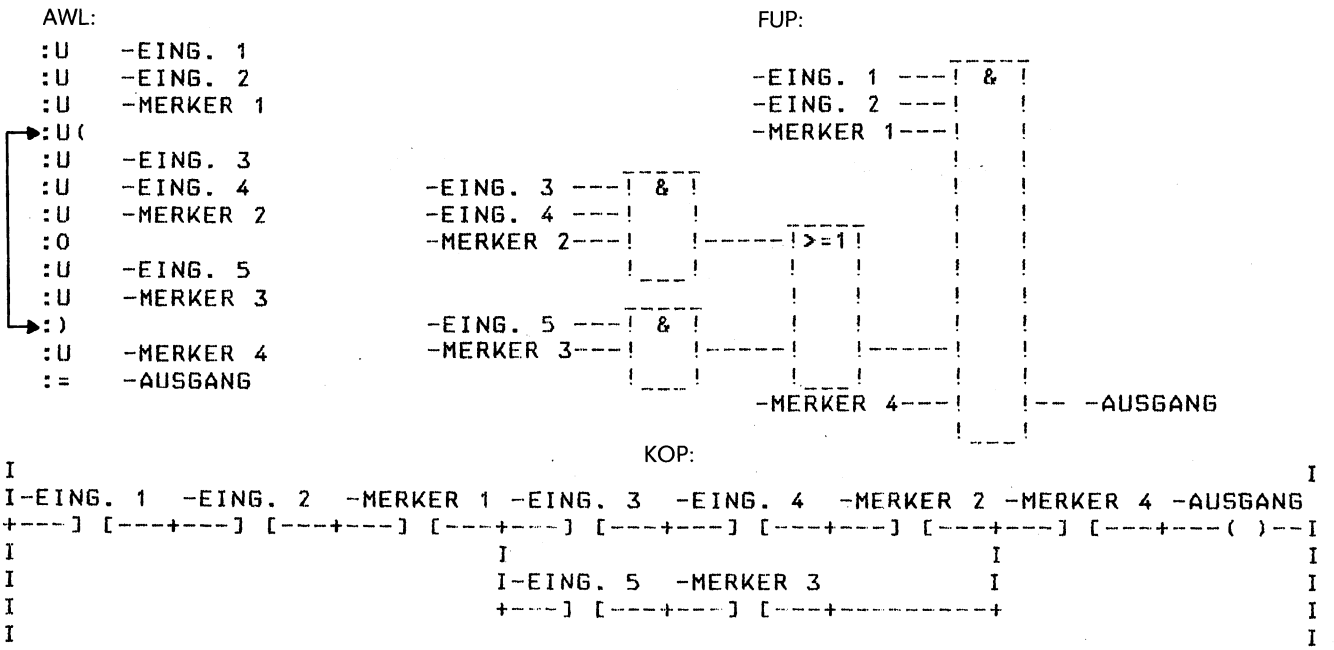


Bild 43 Beispiel 1: ohne Konnektoren

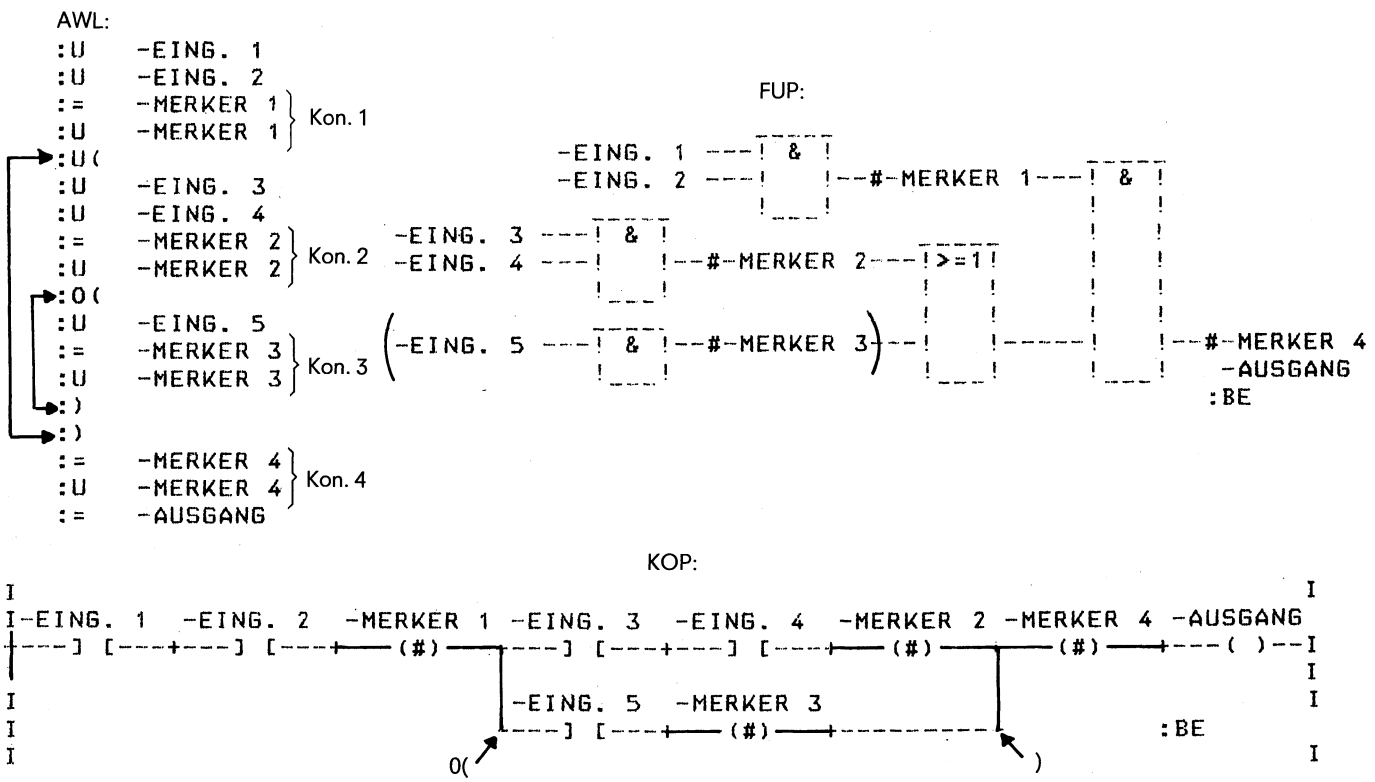


Bild 44 Beispiel 2: mit Konnektoren

Konnektor 1: Verknüpfungsergebnis von UE 1 und UE 2
Konnektor 2: Verknüpfungsergebnis von UE 3 und UE 4
Konnektor 3: Verknüpfungsergebnis von UE 5
Konnektor 4: Verknüpfungsergebnis von der gesamten binären Verknüpfung

8. Hinweise für die Abschätzung des erforderlichen Speicherplatzes

Das Automatisierungsgerät S5-110S ermöglicht einen Ausbau des Anwenderspeichers bis zu maximal 24,5 KWörter. Unabhängig vom gesteckten Speicher sind in der Zentralbaugruppe intern 0,5 KWörter Anwenderspeicher vorhanden.

Der für ein Programm erforderliche Speicherplatz kann wie folgt grob abgeschätzt werden:

Anweisungen in Programmbausteinen:

A $\Sigma(\text{PB-AW}) \approx 8 \times \Sigma(\text{E+A}) + 12$
(ΣAntriebe + ΣAblaufketten)

Anweisungen in Funktionsbausteinen:

B $\Sigma\text{FB-AW} \approx (\text{Anzahl der FB}) \times 150$

Datenwörter:

C $\Sigma\text{DW} \approx 2 \times \Sigma$Antriebe + ΣSchritte (Ablaufkette) + $10 \times \Sigma$Meldungen + 256 (für Protokollierung)

Anweisungen im Organisationbaustein:

D $\Sigma\text{OB-AW}: \frac{8 \times \Sigma\,(E + A)}{150}$

Erforderlicher Speicherplatz $\approx A + B + C + D$

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) VKE 1	ERAB 2	ANZ0 3	ANZ1 4	OVR 5 abhängig von:	beeinflußt:	Funktion
9.1.1 Verknüpfungsoperationen											
				Bit adr.	Byte adr.						UND Verknüpfung
U E	0.0 bis 127.7[1]	6,9	C 1100	0 0XXX	0 0XXX	0 XXXX			2	1,2	Abfrage eines Eingangs auf Signalzustand „1"
U A	0.0 bis 127.7[1]	6,9	C 1100	0 0XXX	8 1XXX	0 XXXX			2	1,2	Abfrage eines Ausgangs auf Signalzustand „1"
U M	0.0 bis 255.7	7,1	8 1000	0 0XXX	0 XXXX	0 XXXX			2	1,2	Abfrage eines Merkers auf Signalzustand „1"
UN E	0.0 bis 127.7[1]	7,4	E 1110	0 0XXX	0 0XXX	0 XXXX			2	1,2	Abfrage eines Eingangs auf Signalzustand „0"
UN A	0.0 bis 127.7[1]	7,4	E 1110	0 0XXX	8 1XXX	0 XXXX			2	1,2	Abfrage eines Ausgangs auf Signalzustand „0"
UN M	0.0 bis 255.7	7,1	A 1010	0 0XXX	0 XXXX	0 XXXX			2	1,2	Abfrage eines Merkers auf Signalzustand „0"
				Bit adr.	Byte adr.						ODER Verknüpfung
O E	0.0 bis 127.7[1]	7,2	C 1100	8 1XXX	0 0XXX	0 XXXX			2	1,2	Abfrage eines Eingangs auf Signalzustand „1"
O A	0.0 bis 127.7[1]	7,2	C 1100	8 1XXX	8 1XXX	0 XXXX			2	1,2	Abfrage eines Ausgangs auf Signalzustand „1"
O M	0.0 bis 255.7	7,4	8 1000	8 1XXX	0 XXXX	0 XXXX			2	1,2	Abfrage eines Merkers auf Signalzustand „1"
ON E	0.0 bis 127.7[1]	7,4	E 1110	8 1XXX	0 0XXX	0 XXXX			2	1,2	Abfrage eines Eingangs auf Signalzustand „0"
ON A	0.0 bis 127.7[1]	7,4	E 1110	8 1XXX	8 1XXX	0 XXXX			2	1,2	Abfrage eines Ausgangs auf Signalzustand „0"
ON M	0.0 bis 255.7	7,7	A 1010	8 1XXX	0 XXXX	0 XXXX			2	1,2	Abfrage eines Merkers auf Signalzustand „0"
					Wort adr.						UND Verknüpfung
U T	1 bis 127	7,4	F 1111	8 1000	0 XXXX	0 XXXX			2	1,2	Abfrage einer Zeit auf Signalzustand „1"
UN T	1 bis 127	7,7	F 1111	C 1100	0 XXXX	0 XXXX			2	1,2	Abfrage einer Zeit auf Signalzustand „0"
U Z	1 bis 127	7,7	B 1011	8 1000	0 XXXX	0 XXXX			2	1,2	Abfrage eines Zählers auf Inhalt > 0
UN Z	1 bis 127	7,7	B 1011	C 1100	0 XXXX	0 XXXX			2	1,2	Abfrage eines Zählers auf Inhalt = 0
					Wort adr.						ODER Verknüpfung
O T	1 bis 127	7,7	F 1111	9 1001	0 XXXX	0 XXXX			2	1,2	Abfrage einer Zeit auf Signalzustand „1"
ON T	1 bis 127	8,0	F 1111	D 1101	0 XXXX	0 XXXX			2	1,2	Abfrage einer Zeit auf Signalzustand „0"
O Z	1 bis 127	8,0	B 1011	9 1001	0 XXXX	0 XXXX			2	1,2	Abfrage eines Zählers auf Inhalt > 0
ON Z	1 bis 127	8,2	B 1011	D 1101	0 XXXX	0 XXXX			2	1,2	Abfrage eines Zählers auf Inhalt = 0

1) Die Ein- und Ausgangsbytes (worte) 64–127 (64–126), sowie die Peripheriebytes (worte) 64–127 (64–126) sind als zusätzliche Merkerbits, -byte und -worte verwendbar, da der mögliche Peripherieausbau 64 Ein/Ausgabebytes nicht überschreiten kann.

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) 1 VKE	2 ERAB	3 ANZ0	4 ANZ1	5 OVR	Funktion
					abhängig von:			beeinflußt:		
										UND/ODER Verknüpfung
O	———	4,7	F 1111	B 1011	0 ———			2	1,2	ODER-Verknüpfungen von UND-Funktionen
O(	———	8,5	B 1011	B 1011	0 ———			2	1,2	ODER-Verknüpfungen von Klammerausdrücken
U(	———	8,8	B 1011	A 1010	0 ———			2	1,2	UND-Verknüpfungen von Klammerausdrücken
)	———	8,5	B 1011	F 1111	0 ———			2	1,2	Klammer zu

9.1.2 Speicheroperationen

Operation		Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Bit adr.	Byte adr.		Anzeigen 2) 1	2	Funktion
S	E	0.0 bis 127.7[1]	8,0	D 1101	0 0XXX	0 0XXX	0 XXXX	1	2	Setze Eingang auf „1"
S	A	0.0 bis 127.7[1]	8,0	D 1101	0 0XXX	8 1XXX	0 XXXX	1	2	Setze Ausgang auf „1"
S	M	0.0 bis 255.7	7,1	9 1001	0 0XXX	0 XXXX	0 XXXX	1	2	Setze Merker auf „1"
R	E	0.0 bis 127.7[1]	8,0	F 1111	0 0XXX	0 0XXX	0 XXXX	1	2	Setze Eingang auf „0"
R	A	0.0 bis 127.7[1]	8,0	F 1111	0 0XXX	8 1XXX	0 XXXX	1	2	Setze Ausgang auf „0"
R	M	0.0 bis 255.7	7,4	B 1011	0 0XXX	0 XXXX	0 XXXX	1	2	Setze Merker auf „0"
=	E	0.0 bis 127.7[1]	7,7	D 1101	8 1XXX	0 0XXX	0 XXXX	1	2	Setze Eingang begrenzt auf „1"
=	A	0.0 bis 127.7[1]	7,7	D 1101	8 1XXX	8 1XXX	0 XXXX	1	2	Setze Ausgang begrenzt auf „1"
=	M	0.0 bis 255.7	6,8	9 1001	8 1XXX	0 XXXX	0 XXXX	1	2	Setze Merker begrenzt auf „1"

9.1.3 Zeit- und Zählloperationen

Operation		Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Wort adr.		Anzeigen 2) 1	2	Funktion
SI	T	1 bis 127	18,7	3 0011	4 0100	0 XXXX	0 XXXX	1	2	Starten einer Zeit als Impuls
SV	T	1 bis 127	17,9	1 0001	C 1100	0 XXXX	0 XXXX	1	2	Starten einer Zeit als verlängerter Impuls
SE	T	1 bis 127	18,1	2 0010	4 0100	0 XXXX	0 XXXX	1	2	Starten einer Zeit als Einschaltverzögerung
SS	T	1 bis 127	18,1	2 0010	C 1100	0 XXXX	0 XXXX	1	2	Starten einer Zeit als speichernde Einschaltverzögerung
SA	T	1 bis 127	17,9	1 0001	4 0100	0 XXXX	0 XXXX	1	2	Starten eineer Zeit als Ausschaltverzögerung
R	T	1 bis 127	10,4	3 0011	C 1100	0 XXXX	0 XXXX	1	2	Rücksetzen einer Zeit
S	Z	1 bis 127	19	5 0101	C 1100	0 XXXX	0 XXXX	1	2	Setzen eines Zählers
R	Z	1 bis 127	10,4	7 0111	C 1100	0 XXXX	0 XXXX	1	2	Rücksetzen eines Zählers
ZV	Z	1 bis 127	17,0	6 0110	C 1100	0 XXXX	0 XXXX	1	2	Vorwärtszählen eines Zählers
ZR	Z	1 bis 127	17,0	5 0101	4 0100	0 XXXX	0 XXXX	1	2	Rückwärtszählen eines Zählers

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Operations-Code Byte 1	Anzeigen 2) 1 VKE abhängig von:	2 ERAB	3 ANZ0	4 ANZ1 beeinflußt:	5 OVR	Funktion
9.1.4 Lade- und Transferfunktionen				Byte/Wort adr.						
L EB	0 bis 127[1)]	7,4	4 A 0100 1010	0 0 0XXX XXXX	—			—		Lade Eingangsbyte des Prozeßabbilds der Eingänge in den Akku 1
L EW	0 bis 126[1)]	9,3	5 2 0101 0010	0 0 0XXX XXXX	—			—		Lade Eingangswort des Prozeßabbilds der Eingänge in den Akku 1
L AB	0 bis 127[1)]	7,4	4 A 0100 1010	8 0 1XXX XXXX	—			—		Lade Ausgangsbyte des Prozeßabbilds der Ausgänge in den Akku 1
L AW	0 bis 126[1)]	9,3	5 2 0101 0010	8 0 1XXX XXXX	—			—		Lade Ausgangswort des Prozeßabbilds der Ausgänge in den Akku 1
L MB	0 bis 255	7,7	0 A 0000 1010	0 0 XXXX XXXX	—			—		Lade Merkerbyte in den Akku 1
L MW	0 bis 254	9,3	1 2 0001 0010	0 0 XXXX XXXX	—			—		Lade Merkerwort in den Akku 1
L DR	1 bis 255	12,1	2 A 0010 1010	0 0 XXXX XXXX	—			—		Lade Datum (rechtes Byte) des aktuellen Datenbausteins in den Akku 1
L DL	1 bis 255	13,2	2 2 0010 0010	0 0 XXXX XXXX	—			—		Lade Datum (linkes Byte) des aktuellen Datenbausteins in den Akku 1
L DW	1 bis 255	16,5	3 2 0011 0010	0 0 XXXX XXXX	—			—		Lade Datum (Wort) des aktuellen Datenbausteins in den Akku 1
L T	1 bis 127	11	0 2 0000 0010	0 0 XXXX XXXX	—			—		Lade den Zeitwert (dual) der Zeit in den Akku 1
L Z	1 bis 127	10,4	4 2 0100 0010	0 0 XXXX XXXX	—			—		Lade den Zählwert (dual) des Zählers in den Akku 1
L PB	0 bis 127[1)]	50	7 2 0111 0010	0 0 XXXX XXXX	—			—		Lade Peripheriebyte der digitalen Ein/Ausgaben, unter Umgehung des Prozeßabbilds in den Akku 1
L PW	0 bis 126[1)]	52	7 A 0111 1010	0 0 XXXX XXXX	—			—		Lade Peripheriewort der digitalen Ein/Ausgaben, unter Umgehung des Prozeßabbilds in den Akku 1
LC T	1 bis 127	23,1	0 C 0000 1100	0 0 XXXX XXXX	—			—		Lade den Zeitwert (BCD) der Zeit in den Akku 1
LC Z	1 bis 127	22,8	4 C 0100 1100	0 0 XXXX XXXX	—			—		Lade den Zählwert (BCD) des Zählers in den Akku 1
T EB	0 bis 127[1)]	7,1	4 B 0100 1011	0 0 0XXX XXXX	—			—		Transferiere den Akku 1 ins Eingangsbyte des Prozeßabbilds der Eingänge
T EW	0 bis 126[1)]	7,7	5 3 0101 0011	0 0 0XXX XXXX	—			—		Transferiere den Akku 1 ins Eingangswort des Prozeßabbilds der Eingänge
T AB	0 bis 127[1)]	7,1	4 B 0100 1011	8 0 1XXX XXXX	—			—		Transferiere den Akku 1 ins Ausgangsbyte des Prozeßabbilds der Ausgänge
T AW	0 bis 126[1)]	7,7	5 3 0101 0011	8 0 1XXX XXXX	—			—		Transferiere den Akku 1 ins Ausgangswort des Prozeßabbilds der Ausgänge
T MB	0 bis 255	7,1	0 B 0000 1011	0 0 XXXX XXXX	—			—		Transferiere den Akku 1 ins Merkerbyte
T MW	0 bis 254	8,0	1 3 0001 0011	0 0 XXXX XXXX	—			—		Transferiere den Akku 1 ins Merkerwort
T DR	1 bis 255	14,3	2 B 0010 1011	0 0 XXXX XXXX	—			—		Transferiere den Inhalt von Akku 1 in das Wort (rechtes Byte) des aktuellen Datenbausteins
T DL	1 bis 255	13,2	2 3 0010 0011	0 0 XXXX XXXX	—			—		Transferiere den Inhalt von Akku 1 in das Wort (linkes Byte) des aktuellen Datenbausteins
T DW	1 bis 255	13	3 3 0011 0011	0 0 XXXX XXXX	—			—		Transferiere den Akku1 in das Wort des aktuellen Datenbausteins
T PB	0 bis 127[1)]	53	7 3 0111 0011	0 0 XXXX XXXX	—			—		Transferiere den Akku 1 direkt ins Peripheriebyte
T PW	0 bis 126[1)]	55	7 B 0111 1011	0 0 XXXX XXXX	—			—		Transferiere den Akku 1 direkt ins Peripheriewort

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) 1 VKE abhängig von:	2 ERAB	3 ANZ0	4 ANZ1	5 OVR beeinflußt:	Funktion
L KB	0 bis 255	5,0	2 8 0010 1000	0 0 XXXX XXXX	—			—	—	Lade Konstante (1 Byte)-Zahl in den Akku 1
L KC*	2 ASCII-Zeichen	6,6	3 0 0011 0000	1 0 0001 0000	—			—	—	Lade Konstante Charakter in den Akku 1
L KF*	−32768 bis +32767	6,6	3 0 0011 0000	0 4 0000 0100	—			—	—	Lade Konstante-Festpunktzahl in den Akku 1
L KH*	0 bis FFFF	6,6	3 0 0011 0000	4 0 0100 0000	—			—	—	Lade Konstante-Zahl (Hex-Code) in den Akku 1
L KM*	0000 . . . 00 bis 111 . . . 11	6,6	3 0 0011 0000	8 0 1000 0000	—			—	—	Lade Konstante-Bitmuster eines Wortes (2 Byte) in den Akku 1
L KY*	0 bis 255, 0 bis 255	6,6	3 0 0011 0000	2 0 0010 0000	—			—	—	Lade Konstante (2 Byte)-Zahl in den Akku 1
L KT*	0.0 bis 999.3	6,6	3 0 0011 0000	0 2 0000 0010	—			—	—	Lade Konstante (2 Byte)-Zahl als Zeitwert in den Akku 1
L KZ*	0 bis 999	6,6	3 0 0011 0000	0 1 0000 0001	—			—	—	Lade Konstante (2 Byte)-Zahl als Zählwert in den Akku 1

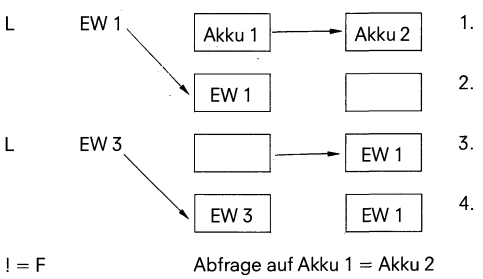
* Dies sind 4 Byte-Befehle, in den die Konstanten in Byte 2 und 3 stehen.

9.1.5 Vergleichsfunktionen

! = F	———	10,7	2 1 0010 0001	8 0 1000 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 a. Gleichheit; Bei Gleichheit VKE=„1";
>< F	———	10,7	2 1 0010 0001	6 0 0110 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 auf Ungleichh.; Bei Ungleichh. VKE=„1";
> F	———	10,7	2 1 0010 0001	2 0 0010 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 a. größer; Akku 2 > Akku 1 VKE=„1";
< F	———	10,7	2 1 0010 0001	4 0 0100 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 a. kleiner; Akku 2 < Akku 1 VKE=„1";
> = F	———	10,7	2 1 0010 0001	A 0 1010 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 a. größer oder gleich; Akku 2 ≥ Akku 1 VKE = „1";
< = F	———	10,7	2 1 0010 0001	C 0 1100 —	—	—		1,2,3,4	Festpunktvergleich von Akku 1 und Akku 2 a. kleiner oder gleich; Akku 2 ≤ Akku 1 VKE = „1";

Hinweis:
Das Automatisierungsgerät hat für Vergleichs- und Rechenoperationen sowie für Digitalverknüpfungen zwei Akkumulatoren.
Laden heißt, daß der Inhalt von Akku 1 nach Akku 2 umgeladen, und daß dann dem Operanden der Ladeoperation entsprechend der Akku 1 neu geladen wird.
Nach zwei Ladeoperationen kann man daher mit Vergleichsoperationen Aussagen über den Inhalt der Akkumulatoren bekommen.

Beispiel:



! = F Abfrage auf Akku 1 = Akku 2

Eine Transferoperation transferiert immer den Inhalt von Akku 1 zu den bei der Transferoperation angegebenen Operanden.

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Wort adr.		Anzeigen 2) abhängig von: VKE 1	ERAB 2	ANZ0 3	beeinflußt: ANZ1 4	OVR 5	Funktion
9.1.6 Bausteinaufrufe												
SPA PB	0 bis 127	24,8	7 0111	5 0101	0 XXXX	0 XXXX	—			2		Springe absolut (unbedingt) zum Programmbaustein
SPA FB	0 bis 47	25,6	3 0011	D 1101	0 XXXX	0 XXXX	—			2		Springe absolut (unbedingt) zum Funktionsbaustein
SPB PB	0 bis 127	25	5 0101	5 0101	0 XXXX	0 XXXX	1			1,2		Springe bedingt zum Programmbaustein
SPB FB	0 bis 47	26	1 0001	D 1101	0 XXXX	0 XXXX	1			1,2		Springe bedingt zum Funktionsbaustein
A DB	1 bis 63	8,8	2 0010	0 0000	0 XXXX	0 XXXX	—			—		Aufruf eines Datenbausteins; Datenbaustein ist so lange gült. b. ein anderer Datenbaustein aufge. wird;
BE	———	18,4	6 0110	5 0101	0 0000	0 0000	—			2		Bausteinende; Abschließen eines Bausteines;
BEA	———	18,4	6 0110	5 0101	0 0000	1 0001	—			2		Bausteinende absolut; darf innerhalb eines Bausteines mehrmals programmiert werden
BEB	———	18,6	0 0000	5 0101	0 0000	0 0000	1			1,2		Bausteinende bedingt, bei VKE = 1
9.1.7 Sonstige Befehle												
NOP 0	———	3,5	0 0000	0 0000	0 0000	0 0000	—			—		Nulloperation (alle Bits gelöscht)
NOP 1	———	3,5	F 1111	F 1111	F 1111	F 1111	—			—		Nulloperation (alle Bits gesetzt)
STP		6,0	7 0111	0 0000	0 0000	3 0011	—			—		Programmierbarer Stopbefehl, (Am Ende des Zyklus geht das AG in den Stop-Zustand)

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0 / Byte 1 / Byte 2 / Byte 3	Anzeigen 2) 1 VKE, 2 ERAB, 3 ANZ0, 4 ANZ1, 5 OVR (abhängig von: / beeinflußt:)	Funktion

9.2.1 Binäre Verknüpfungsfunktion

				Bit Adr.	Wort Adr.				
P	DW[3]	1.0 bis 255.15	22,7	7 0111 C 1100	0 0000 0 XXXX	4 0100 0 XXXX	6 0110 0 XXXX	—	1,2	Prüfe Bit des Datenwortes auf Signalzustand „1".
P	Z[3]	1.0 bis 127.15	21,2	7 0111 C 1100	0 0000 0 XXXX	1 0001 0 XXXX	5 0101 0 XXXX	—	1,2	Prüfe Bit des Zählwortes auf Signalzustand „1".
P	T[3]	1.0 bis 127.15	21,2	7 0111 C 1100	0 0000 0 XXXX	2 0010 0 XXXX	5 0101 0 XXXX	—	1,2	Prüfe Bit des Zeitwortes auf Signalzustand „1".
PN	DW[3]	1.0 bis 255.15	22,5	7 0111 8 1000	0 0000 0 XXXX	4 0100 0 XXXX	6 0110 0 XXXX	—	1,2	Prüfe Bit des Datenwortes auf Signalzustand „0".
PN	Z[3]	1.0 bis 127.15	21,0	7 0111 8 1000	0 0000 0 XXXX	1 0001 0 XXXX	5 0101 0 XXXX	—	1,2	Prüfe Bit des Zählwortes auf Signalzustand „0".
PN	T[3]	1.0 bis 127.15	21,0	7 0111 8 1000	0 0000 0 XXXX	2 0010 0 XXXX	5 0101 0 XXXX	—	1,2	Prüfe Bit des Zeitwortes auf Signalzustand „0".
SU	DW[3]	1.0 bis 255.15	22,7	7 0111 4 0100	0 0000 0 XXXX	4 0100 0 XXXX	6 0110 0 XXXX	—	2	Setze Bit des Datenwortes unbedingt auf „1".
SU	Z[3]	1.0 bis 127.15	21,2	7 0111 4 0100	0 0000 0 XXXX	1 0001 0 XXXX	5 0101 0 XXXX	—	2	Setze Bit des Zählwortes unbedingt auf „1".
SU	T[3]	1.0 bis 127.15	21,2	7 0111 4 0100	0 0000 0 XXXX	2 0010 0 XXXX	5 0101 0 XXXX	—	2	Setze Bit des Zeitwortes unbedingt auf „1".
RU	DW[3]	1.0 bis 255.15	22,5	7 0111 0 0000	0 0000 0 XXXX	4 0100 0 XXXX	6 0110 0 XXXX	—	2	Setze Bit des Datenwortes unbedingt auf „0".
RU	Z[3]	1.0 bis 127.15	21,0	7 0111 0 0000	0 0000 0 XXXX	1 0001 0 XXXX	5 0101 0 XXXX	—	2	Setze Bit des Zählwortes unbedingt auf „0".
RU	T[3]	1.0 bis 127.15	21,0	7 0111 0 0000	0 0000 0 XXXX	2 0010 0 XXXX	5 0101 0 XXXX	—	2	Setze Bit des Zeitwortes unbedingt auf „0".

3) Dies sind 4 Byte-Befehle

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) VKE 1	ERAB 2	ANZ0 3	ANZ1 4	OVR 5	Funktion
					abhängig von:			beeinflußt:		

9.2.2 Digitale Verknüpfungsfunktionen

UW	———	6,0	4 0100	1 1001	0 ———	0 ———	—	3,4	Digitale UND-Verknüpfung von Akku 1 und Akku 2, (Wortw.) Ergeb. w. i. Akku 1 hinterlegt.
OW	———	8,2	4 0100	9 1001	0 ———	0 ———	—	3,4	Digit. ODER-Verknüpfung v. Akku 1 u. Akku 2 (Wortweise); Ergeb. w. i. Akku 1 hinterlegt.
XOW	———	6,5	5 0101	1 0001	0 ———	0 ———	—	3,4	Digit. EXOR-Verknüpfung v. Akku 1 u. Akku 2 (Wortweise); Ergeb. w. i. Akku 1 hinterlegt.

9.2.3 Rechenfunktionen

+ F	———	9,9	7 0111	9 1001	0 ———	0 ———	—	3,4,5	Addiere Akku 1 zu Akku 2; Ergebnis wird in Akku 1 hinterlegt;
– F	———	12,1	5 0101	9 1001	0 ———	0 ———	—	3,4,5	Subtrahiere Akku 1 v. Akku 2; Ergebnis wird i. Akku 1 hinterlegt;

9.2.4 Sprungfunktionen

					Wort-adr. ± 127				
SPA =	"Marke" (4 ASCII-Zeichen)	9,9	2 0010	D 1101	0 XXXX	0 XXXX	—	—	Springe absolut (unbedingt) zur Marke, bestehend a. 4 ASCII-Zeichen. Sprungdistanz ≤ ± 127 Wörter.
SPB =	"Marke" (4 ASCII-Zeichen)	10,4	F 1111	A 1010	0 XXXX	0 XXXX	1	1,2	Springe bedingt (b. VKE=„1") zur Marke, bestehend a. 4 ASCII-Zeichen. Sprungdistanz ≤ ± 127 Wörter.
SPP =	"Marke" (4 ASCII-Zeichen)	10,4	1 0001	5 0101	0 XXXX	0 XXXX	3,4	—	Springe bedingt (b. Ergebnis größer Null) z. Marke besteh. a. 4 ASCII-Zeich. Srungdis. ≤ ±127 Wörter.
SPM =	"Marke" (4 ASCII-Zeichen)	10,2	2 0010	5 0101	0 XXXX	0 XXXX	3,4	—	Springe bedingt (b. Ergebnis kleiner Null) z. Marke, besteh. a. 4 ASCII-Zeich. Sprungd. ≤ ± 127 Wörter.
SPZ =	"Marke" (4 ASCII-Zeichen)	10,4	4 0100	5 0101	0 XXXX	0 XXXX	3,4	—	Springe bedingt (b. Ergebnis gleich Null) z. Marke, besteh. a. 4 ASCII-Zeich. Sprungd. ≤ ± 127 Wörter.
SPN =	"Marke" (4 ASCII-Zeichen)	10,4	3 0011	5 0101	0 XXXX	0 XXXX	3,4	—	Springe bedingt (b. Ergebnis ungl. Null) z. Marke besteh. a. 4 ASCII-Zeich. Sprungd. ≤ ± 127 Wörter.
SPO =	"Marke" (4 ASCII-Zeichen)	10,4	0 0000	D 1101	0 XXXX	0 XXXX	5	—	Springe bedingt (bei Anzeige OVR = 1) zur Marke, besteh. a. 4 ASCII-Zeich. Sprungd. ≤ ± 127 Wörter.

9.2.5 Zeit- und Zählfunktionen

					Wort-adr.				
FRT	0 bis 127	8,8	0 0000	4 0100	0 XXXX	0 XXXX	1	2	Freigabe der Zeit für Neustart (Nur bei positiven Flankenwechsel von VKE)
FRZ	0 bis 127	8,8	4 0100	4 0100	0 XXXX	0 XXXX	1	2	Freigabe des Zählers für Neustart (Nur bei positiven Flankenwechsel von VKE)

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Operations-Code Byte 1	Anzeigen 2) 1 VKE	2 ERAB	3 ANZ0	4 ANZ1	5 OVR	Funktion
					abhängig von:			beeinflußt:		

9.2.6 Schiebefunktionen

						Par.			
SLW	0 bis 15	—	6 0110	1 0001	0 0000	0 XXXX	—	3,4	Schiebe Inhalt von Akku 1 nach links, die rechts frei werdenden Akkuzellen werden mit Nullen aufgefüllt.
SRW	0 bis 15	—	6 0110	9 1001	0 0000	0 XXXX	—	3,4	Schiebe Inhalt von Akku 1 nach rechts, die links frei werdenden Akkuzellen werden mit Nullen aufgefüllt.

9.2.7 Umwandlungsfunktionen

KEW	———	4,7	0 0000	1 0001	0	0	—	—	Einerkomplement von Akku 1
KZW	———	8,2	0 0000	9 1001	0	0	—	3,4	Zweierkomplement von Akku 1; Anzeige von < 0, > 0 oder OV;

9.2.8 Dekrementieren/Inkrementieren

					Dek./Ink. 0 bis 255				
D	1 bis 255	5,7	1 0001	9 1001	0 XXXX	0 XXXX	—	—	Dekrementiere nur das Low-Byte von Akku 1 um einen bestimmten Wert.
I	1 bis 255	4,4	1 0001	1 0001	0 XXXX	0 XXXX	—	—	Inkrementiere nur das Low-Byte von Akku 1 um einen bestimmten Wert.

9.2.9 Bearbeitungsfunktionen

						Wort-adr.				
B	MW[4)]	0 bis 254	9,9	4 0100	E 1110	0 XXXX	0 XXXX	—	—	Bearb. Merkerw.. Die nachfolg. angegeb. Operation wird mit dem im Merkerwort angegebene Parameter kombiniert und ausgeführt.
B	DW[4)]	0 bis 255	12,1	6 0110	E 1110	0 XXXX	0 XXXX	—	—	Bearb. Datenw.. Die nachfolg. angegeb. Operation wird mit dem im Datenwort angegebene Parameter kombiniert und ausgeführt.

9.2.10 Befehlsausgabe sperren/freigeben

BAS	———	5,5	B 1011	E 1110	0	0	1	—	Befehlsausgabe sperren
BAF	———	5,2	F 1111	E 1110	0	0	1	—	Befehlsausgabe freigeben

9.2.11 Alarme sperren/freigeben

AS	———	5,0	0 0000	8 1000	0 0000	0 0000	—	—	Alarmbearbeitung sperren
AF	———	5,0	0 0000	8 1000	8 1000	0 0000	—	—	Alarmbearbeitung freigeben

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

4) Dies sind 4 Byte-Befehle; in Byte 2 steht der Operations-Code und in Byte 3 der Parameter des auszuführenden Befehls.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) 1 VKE	2 ERAB	3 ANZ0	4 ANZ1	5 OVR	Funktion
					abhängig von:			beeinflußt:		
9.2.12 Substitutionsfunktionen				Parameteradr. (Hex.)						UND/ODER Verknüpfungsfunktion
U =	Formaloperand (4ASCII-Zeichen)	X+14,0[5]	0 0000	7 0111	0 00XX	0 XXXX	2		1,2	Und-Funktion; Abfrage eines Formaloperanden auf „1".
UN =	Formaloperand (4ASCII-Zeichen)	X+14,2[5]	2 0010	7 0111	0 00XX	0 XXXX	2		1,2	Und-Funktion; Abfrage eines Formaloperanden auf „0".
O =	Formaloperand (4ASCII-Zeichen)	X+14,2[5]	0 0000	F 1111	0 00XX	0 XXXX	2		1,2	Oder-Funktion; Abfrage eines Formaloperanden auf „1".
ON =	Formaloperand (A4SCII-Zeichen)	X+14,2[5]	2 0010	F 1111	0 00XX	0 XXXX	2		1,2	Oder-Funktion; Abfrage eines Formaloperanden auf „0".
										Speicherfunktionen
S =	Formaloperand (4ASCII-Zeichen)	X+14,2[5]	1 0001	7 0111	0 00XX	0 XXXX	1		2	Setzen (binär) eines Formaloperanden auf „1".
= =	Formaloperand (4ASCII-Zeichen)	X+14,2[5]	1 0001	F 1111	0 00XX	0 XXXX	1		2	Setzen eines Formaloperanden begrenzt auf „1".
RB =	Formaloperanden (4ASCII-Zeichen)	X+14,2[5]	3 0011	7 0111	0 00XX	0 XXXX	1		2	Setzen (binär) eines Formaloperanden auf „0".
RD =	Formaloperand (4ASCII-Zeichen)	X+14,2[5]	3 0011	E 1110	0 00XX	0 XXXX	1		2	Setzen (digital) eines Formaloperanden auf „0" [Operanden T und Z].
										Lade/Transferfunktion
L =	Formaloperand (4ASCII-Zeichen)	X+15,5[5]	4 0100	6 0110	0 00XX	0 XXXX	—		—	Laden eines Formaloperanden. Der Wert des als Formaloperanden vorgegebenen Operanden wird in den Akku 1 geladen [Operanden EB,EW,MB,MW,AB,AW,DR,DL,DW,PB,PW].
LC =	Formaloperand (4ASCII-Zeichen)	X+6,8[5]	0 0000	E 1110	0 00XX	0 XXXX	—		—	Laden codiert eines Formaloperanden. Der Wert der als Formaloperand vorgegebenen Zeit- oder Zählzelle wird BCD-codiert in den Akku 1 geladen [Operanden T,Z].
LW =	Formaloperand (4ASCII-Zeichen)	8,5	3 0011	F 1111	0 00XX	0 XXXX	—		—	Laden des Bitmusters eines Formaloperanden. Das Bitmuster des Formaloperanden wird in den Akku 1 geladen [Operanden KB,KC,KF,KH,KM,KY, KT,KZ].
T =	Formaloperand (4ASCII-Zeichen)	X+14,5[5]	6 0110	6 0110	0 00XX	0 XXXX	—		—	Transferieren zu einem Formaloperanden. Der Inhalt des Akku 1 wird zu dem als Formaloperand vorgegebenen Operanden transferiert [Operanden EB,EW,MB,MW,AB,AW,DR,DL,DW,PB,PW].

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

5) X bedeutet die Zykluszeit des zu substituierten Befehles.

Operation	Parameter	Zyklus-zeit (µs)	Operations-Code Byte 0	Byte 1	Anzeigen 2) VKE 1	ERAB 2	ANZ0 3	ANZ1 4	OVR 5	Funktion
					abhängig von:			beeinflußt:		
				Parameteradr. (Hex.)						Zeit/Zählfunktionen
SI =	Formaloperand (4ASCII-Zeichen)	X+5,5[5]	3 0011	6 0110	0 00XX	0 XXXX	1	2		Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Impuls [Operand T].
SVZ =	Formaloperand (4ASCII-Zeichen)	X+6,7[5]	1 0001	E 1110	0 00XX	0 XXXX	1	2		Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als verlängerter Impuls, bzw. Setzen eines als Formaloperand vorgegebenen Zählers mit dem vorher geladenen Zählwert [Operanden Z,T].
SE =	Formaloperand (4ASCII-Zeichen)	X+5,6[5]	2 0010	6 0110	0 00XX	0 XXXX	1	2		Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Einschaltverzögerung [Operand T].
SAR =	Formaloperand (4ASCII-Zeichen)	X+6,8[5]	1 0001	6 0110	0 00XX	0 XXXX	1	2		Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als Ausschaltverzögerung; bzw. Rückwärtszählen eines als Formaloperand vorgegebenen Zählers [Operanden Z,T].
SSV =	Formaloperand (4ASCII-Zeichen)	X+6,8[5]	2 0010	E 1110	0 00XX	0 XXXX	1	2		Starten einer als Formaloperand vorgegebenen Zeit mit dem vorher geladenen Zeitwert als speichernde Einschaltverzögerung; bzw. Vorwärtszählen eines als Formaloperand vorgegebenen Zählers [Operanden Z,T].
FR =	Formaloperand (4ASCII-Zeichen)	X+6,8[5]	0 0000	6 0110	0 00XX	0 XXXX	1	2		Freigabe eines Formaloperanden für Neustart [Operanden Z,T].
										Bearbeitungsfunktion
B =	Formaloperand (4ASCII-Zeichen)	X+8,5[5]	7 0111	6 0110	0 00XX	0 XXXX	—	—		Bearbeite Formaloperand. Nur die Operationen A DB, SPA PB und SPA FB können substituiert werden.

2) VKE (Verknüpfungsergebnis) ≙ Status; ERAB = 0 bedeutet, daß es sich um eine laufende Verknüpfung handelt; ERAB = 1 bedeutet, daß es sich um eine Erstabfrage handelt; ANZ1 ANZ0 = 00 Ergebnis oder Akku 1 gleich Null; ANZ1 ANZ0 = 01 Ergebnis oder Akku 1 kleiner Null; ANZ1 ANZ0 = 10 Ergebnis oder Akku 1 größer Null; OVR (Überlauf) = 1 bedeutet, daß bei arithmetischen Befehlen der Wert für den Akku zu groß ist.

5) X bedeutet die Zykluszeit des zu substituierten Befehles.

SIEMENS AKTIENGESELLSCHAFT

Änderungen vorbehalten

Bestell-Nr. GWA 4NEB 807 2122-01
Printed in West Germany
132761 683 0.5 Fl. de