

**SIEMENS**

**SIMATIC S5**

**AG S5-135U**  
**CPU 928**

**Programmieranleitung**

Bestell-Nr.: 6ES5 998-1PR11  
Ausgabe 01

Inhalt	
CPU 928 Programmieranleitung C79000-B8500-C633-01	<b>1</b>
Mehrprozessorkommunikation Bedienungsanleitung C79000-B8500-C468-05	<b>2</b>

*Das Tabellenheft CPU 922/CPU 928/CPU 928B/CPU 948*  
*Bestell-Nr.: 6ES5 997-3UA12*  
*ist dem Handbuch beigelegt.*

**Wichtig:**

**Die vorliegende Programmieranleitung  
bezieht sich auf die CPU 928 mit der  
MLFB-Nummer -3UA12 (12 MHz).**

## **Copyright**

Copyright © Siemens AG 1993 All Rights Reserved

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadensersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung.

## **Haftungsausschluß**

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden jedoch regelmäßig überprüft und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen bleiben vorbehalten.

## **Sicherheitstechnische Hinweise**

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise sind durch ein Warndreieck hervorgehoben und je nach Gefährungsgrad folgendermaßen dargestellt:



### **Warnung**

bedeutet, daß Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.



### **Vorsicht**

bedeutet, daß eine leichte Körperverletzung oder ein Sachschaden eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Inbetriebsetzung und Betrieb des Gerätes darf nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieses Handbuchs sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

## SIMATIC S5

### CPU 928

#### Inhalt

<b>1</b>	<b>AG S5-135U - Arbeitsweise und Anwendungsbereich.....</b>	<b>1-1</b>
1.1	Neue Eigenschaften und Funktionen der CPU 928.....	1-7
<b>2</b>	<b>Anwenderprogramm.....</b>	<b>2-1</b>
2.1	Programmiersprache STEP5.....	2-1
2.1.1	Darstellungsarten KOP, FUP und AWL.....	2-2
2.1.2	Strukturierte Programmierung.....	2-3
2.1.3	STEP5-Operationen.....	2-4
2.1.4	Zahlendarstellung .....	2-5
2.1.5	STEP5-Bausteine.....	2-9
2.2	Organisations-, Programm- und Schrittbausteine.....	2-13
2.2.1	Programmierung von OBs, PBs und SBs.....	2-13
2.2.2	Aufruf.....	2-14
2.2.3	Spezielle Organisationsbausteine.....	2-16
2.2.4	Sonderfunktions-Organisationsbausteine.....	2-18
2.3	Funktionsbausteine.....	2-19
2.3.1	Aufbau von Funktionsbausteinen.....	2-20
2.3.2	Programmierung von Funktionsbausteinen.....	2-22
2.3.3	Aufruf und Parametrierung von FBs.....	2-26
2.3.4	Spezielle Funktionsbausteine.....	2-29
2.4	Datenbausteine.....	2-31
2.4.1	Aufbau eines Datenbausteins.....	2-31
2.4.2	Programmierung von Datenbausteinen.....	2-32
2.4.3	Aufschlagen von Datenbausteinen.....	2-33
2.4.4	Spezielle Datenbausteine.....	2-35
<b>3</b>	<b>Programmbearbeitung.....</b>	<b>3-1</b>
3.1	Übersicht.....	3-1
3.1.1	Programmorganisation.....	3-1
3.1.2	Programmspeicherung.....	3-5
3.1.3	Bearbeitung des STEP5-Anwenderprogramms.....	3-6
3.1.4	Festlegungen zur Programmbearbeitung.....	3-9
3.2	STEP5-Operationsvorrat mit Programmierbeispielen.....	3-10
3.2.1	Grundoperationen.....	3-13
3.2.2	Ergänzende Operationen.....	3-37



<b>4</b>	<b>Betriebszustände.....</b>	<b>4-1</b>
4.1	Betriebszustände und Programmbearbeitungsebenen.....	4-1
4.2	Betriebszustand STOP.....	4-6
4.3	Betriebszustand ANLAUF.....	4-9
4.3.1	Neustart und Manueller Wiederanlauf.....	4-11
4.3.2	Automatischer Wiederanlauf.....	4-13
4.3.3	Unterbrechungen im ANLAUF.....	4-14
4.4	Betriebszustand RUN.....	4-16
4.4.1	ZYKLUS: Zyklische Programmbearbeitung.....	4-17
4.4.2	WECKALARME: Zeitgesteuerte Programmbearbeitung.....	4-18
4.4.3	REGLER-ALARM: Bearbeitung von Reglern.....	4-23
4.4.4	PROZESS-ALARM: Alarmgesteuerte Programmbearbeitung.....	4-24
<b>5</b>	<b>Unterbrechungs- und Fehlerbehandlung.....</b>	<b>5-1</b>
5.1	Häufige Fehler im Anwenderprogramm.....	5-1
5.2	Auswertung von Fehlerinformationen.....	5-2
5.3	Steuerbits und Unterbrechungsstack (USTACK).....	5-7
5.4	Fehlerbehandlung über Organisationsbausteine.....	5-18
5.5	Fehler im ANLAUF.....	5-21
5.5.1	DB0-Fehler.....	5-22
5.5.2	DB1-Fehler.....	5-22
5.5.3	DB2-Fehler.....	5-24
5.5.4	DX0-Fehler.....	5-25
5.6	Fehler im ANLAUF und im RUN.....	5-26
5.6.1	BCF (Befehlscodefehler).....	5-27
5.6.2	LZF (Laufzeitfehler).....	5-30
5.6.3	ADF (Adressierfehler).....	5-34
5.6.4	QVZ (Quittungsverzug).....	5-35
5.6.5	ZYK-FE (Zyklusfehler).....	5-36
5.6.6	WECK-FE (Weckfehler).....	5-37
5.6.7	REG-FE (Reglerfehler).....	5-37
5.6.8	ABR (Abbruch).....	5-40
<b>6</b>	<b>Integrierte Sonderfunktionen.....</b>	<b>6-1</b>
6.1	Register-Handling.....	6-5
6.1.1	Zugriff auf das Anzeigenbyte (OB 110).....	6-5
6.1.2	Akku 1, Akku 2, Akku 3 und Akku 4 löschen (OB 111).....	6-7
6.1.3	Akku Roll Up (OB 112) und Akku Roll Down (OB 113).....	6-8
6.2	Strukturbefehle.....	6-10
6.2.1	Zählschleifen (OB 160 bis 163).....	6-10
6.3	Bausteinstack (BSTACK) lesen (OB 170).....	6-12

6.4	Baustein-Handling.....	6-16
6.4.1	Variabler Datenbaustein-Zugriff (OB 180).....	6-16
6.4.2	Datenbausteine (DB/DX) testen (OB 181).....	6-20
6.4.3	Merker in DB übertragen (OB 190, 192).....	6-22
6.4.4	Datenblock in Merkerbereich übertragen (OB 191, 193).....	6-24
6.4.5	Datenbausteine ins DB-RAM übertragen (OB 254, 255).....	6-29
6.5	Mehrprozessor-Kommunikation (OB 200, OB 202 bis 205).....	6-31
6.6	Kachelzugriffe (OB 216 bis 218).....	6-32
6.6.1	Schreiben zu einer Kachel (OB 216).....	6-35
6.6.2	Lesen von einer Kachel (OB 217).....	6-37
6.6.3	Belegen einer Kachel (OB 218).....	6-39
6.7	Vorzeichenerweiterung (OB 220).....	6-43
6.8	Systemfunktionen.....	6-44
6.8.1	"Alarmer gemeinsam sperren" ein-/ausschalten (OB 120) und "Alarmer gemeinsam verzögern" ein-/ausschalten (OB 122).....	6-44
6.8.2	"Weckalarmer einzeln sperren" ein-/ausschalten (OB 121) und "Weckalarmer einzeln verzögern" ein-/ausschalten (OB 123)...	6-46
6.8.3	Zykluszeit einstellen (OB 221).....	6-49
6.8.3	Zykluszeit neu starten (OB 222).....	6-49
6.8.4	Anlaufarten vergleichen (OB 223).....	6-50
6.8.5	Koppelmerker im Block übertragen (OB 224).....	6-50
6.8.6	Wort aus dem Systemprogramm lesen (OB 226).....	6-51
6.8.7	Quersumme des Systemprogramms lesen (OB 227).....	6-52
6.8.8	Statusinformation einer Programmbearbeitungsebene lesen (OB 228).....	6-54
6.9	Funktionen für Standard-Funktionsbausteine (OB 230 - 237)...	6-56
6.10	Schieberegister.....	6-57
6.10.1	Schieberegister initialisieren (OB 240).....	6-60
6.10.2	Schieberegister bearbeiten (OB 241).....	6-63
6.10.3	Schieberegister löschen (OB 242).....	6-64
6.11	Regelung: PID-Algorithmus.....	6-65
6.11.1	PID-Algorithmus initialisieren (OB 250).....	6-72
6.11.2	PID-Algorithmus bearbeiten (OB 251).....	6-73
7	<b>Voreinstellung von Systemverhalten im Datenbaustein DX 0....</b>	<b>7-1</b>
8	<b>Speicherbelegung und Speicherorganisation.....</b>	<b>8-1</b>
8.1	Adreßraumaufteilung in der CPU 928.....	8-2
8.1.1	Adreßraumaufteilung System-RAM.....	8-3
8.1.2	Adreßraumaufteilung Peripherie.....	8-4
8.2	Speicherorganisation in der CPU 928.....	8-7
8.2.1	Bausteinköpfe im Anwenderspeicher und DB-RAM.....	8-7
8.2.2	Bausteinadreßlisten im Datenbaustein DB 0.....	8-8
8.2.3	BA-/BB-Bereich.....	8-12
8.2.4	BS-/BT-Bereich (Systemdatenbelegung).....	8-12

<b>9</b>	<b>Speicherzugriffe über absolute Adressen.....</b>	<b>9-1</b>
9.1	Zugriffe auf Register und den Speicher über eine Adresse im Akku 1.....	9-5
9.2	Speicherblöcke transferieren.....	9-13
9.3	Operationen mit dem BR-Register.....	9-19
9.3.1	Laden des BR-Registers.....	9-19
9.3.2	Verschieben von Registerinhalten.....	9-20
9.3.3	Zugriffe auf den lokalen Speicher.....	9-21
9.3.4	Zugriffe auf den globalen Speicher.....	9-21
9.3.5	Zugriffe auf den Kachelspeicher.....	9-24
<b>10</b>	<b>Mehrprozessorbetrieb.....</b>	<b>10-1</b>
10.1	Hinweise.....	10-1
10.2	Datenaustausch zwischen den Prozessoren.....	10-3
10.2.1	Koppelmerker.....	10-4
10.2.2	Mehrprozessorkommunikation.....	10-8
10.2.3	Zusammenhängende Datenblöcke "geschützt" übertragen.....	10-8
10.3	Peripheriezuteilung.....	10-9
10.3.1	Datenbaustein DB 1.....	10-9
10.4	Anlauf im Mehrprozessorbetrieb.....	10-12
10.5	Testbetrieb.....	10-13
<b>11</b>	<b>Testhilfsmittel: On-line-Funktionen.....</b>	<b>11-1</b>
11.1	On-line-Funktion 'STATUS VARIABLEN'.....	11-3
11.2	On-line-Funktion 'STATUS'.....	11-4
11.3	On-line-Funktion 'BEARBEITUNGSKONTROLLE'.....	11-5
11.4	On-line-Funktion 'STEUERN'.....	11-9
11.5	On-line-Funktion 'STEUERN VARIABLEN'.....	11-9
11.6	On-line-Funktion 'Speicher KOMPRIMIEREN'.....	11-10
11.7	On-line-Funktionen 'START/STOP'.....	11-10
11.8	On-line-Funktion 'AG URLÖSCHEN'.....	11-11
11.9	On-line-Funktion 'AUSGABE ADRESSE'.....	11-11
11.10	On-line-Funktion 'SPEICHERAUSBAU'.....	11-11
11.11	Tabelle: Tätigkeiten an Kontrollpunkten.....	11-12

## **A N H A N G**

<b>A</b>	Technische Daten AG S5-135U.....	A-1
<b>B</b>	Übersicht über die Fehlerkennungen.....	B-1
<b>C</b>	Übersicht über den STEP5-Operationsvorrat.....	C-1
<b>D</b>	STEP5-Operationen (alphabetisch).....	D-1
<b>E</b>	STEP5-Operationen (sortiert nach Befehlscode).....	E-1
<b>F</b>	STEP5-Operationen, die in der CPU 928 nicht enthalten sind.....	F-1
<b>G</b>	Übersicht über die Kennungen der Programmbearbeitungsebenen....	G-1
<b>H</b>	Beispiel zur Auswertung des Unterbrechungsstacks.....	H-1

### **Stichwortverzeichnis**

### **Verzeichnis der Abbildungen, Beispiele und Übersichten**

## Was steht wo?

**Kapitel 1** beschreibt einleitend die Arbeitsweise und den internen Aufbau eines Prozessors. Es skizziert die typische Anlagenstruktur beim Einsatz des Automatisierungsgerätes AG S5-135U und weist auf die neuen Eigenschaften und Funktionen der CPU 928 hin.

**Kapitel 2** erläutert die Struktur des Anwenderprogramms und geht auf die Besonderheiten der Programmiersprache STEP5 ein. Im Anschluß daran werden die verschiedenen STEP5-Programmbausteine charakterisiert und ihre Programmierung beschrieben.

**Kapitel 3** enthält Informationen zur zyklischen Programmbearbeitung in der CPU 928, zur Programmorganisation und zur Programmspeicherung. Es erläutert den gesamten STEP5-Befehlsvorrat und enthält viele Programmierbeispiele. (Zusätzliche Informationen über die STEP5-Befehle finden Sie in der STEP5-Operationsliste. Außerdem verweisen wir Sie auf die Angaben im Literaturverzeichnis.)

**Kapitel 4** beschreibt die verschiedenen Betriebszustände der CPU 928 (Anlauf, Run, Stop) und definiert den Begriff der "Programmbearbeitungsebene". Es erklärt die möglichen Programmbearbeitungsebenen in den einzelnen Betriebszuständen des Prozessors. Außerdem finden Sie wichtige Informationen zur zeit- und alarm-gesteuerten Programmbearbeitung.

**Kapitel 5** behandelt ausführlich das Thema "Fehlersuche und Fehlerbehandlung". Es beschreibt typische Fehler im Anlauf und im Run und gibt Ihnen Tips, wie Sie Fehler ausfindig machen und wie Sie darauf reagieren können. Es erklärt den Aufbau und die Auswertung des Unterbrechungsstacks (USTACK) anhand von Beispielen.

**Kapitel 6** beschreibt die integrierten Sonderfunktionen der CPU 928 und enthält viele Anwendungsbeispiele.

**Kapitel 7** ist dem Aufbau und der Programmierung des Datenbausteins DX 0 gewidmet, mit dem Sie auf einfache Weise bestimmte Leistungen der CPU 928 Ihren Erfordernissen anpassen können (ebenfalls mit Programmierbeispielen).

**Kapitel 8** enthält detaillierte Informationen zu den einzelnen Speicherbereichen der CPU 928. Systemerfahrene Anwender finden dort die Belegung der Systemdaten.

**Kapitel 9** ist interessant für geübtere Anwender, die bereits über sehr gute Systemkenntnisse verfügen. Es enthält alle STEP5-Befehle, mit denen Sie über Absolutadressen auf den gesamten Speicherbereich zugreifen können und beschreibt dabei auch die einzelnen Register der CPU 928.

**Kapitel 10** gibt Ihnen ergänzende Informationen zum Mehrprozessorbetrieb und beschreibt den Aufbau und die Programmierung des Datenbausteins DB 1, der für den Mehrprozessorbetrieb erforderlich ist. Außerdem werden Besonderheiten des Testbetriebs erläutert.

**Kapitel 11** beschreibt einige On-line-Funktionen, die Sie am Programmiergerät zum Testen Ihres Programms aufrufen können, und weist auf Besonderheiten in Zusammenhang mit der CPU 928 hin.

## Abkürzungen

ABBR	Abbruch
ADF	Adressierfehler
AG	Automatisierungsgerät
Akku 1(2,3,4)-L	Low-Wort im Akkumulator 1 (2,3,4), 16 Bit
Akku 1(2,3,4)-H	High-Wort im Akkumulator 1 (2,3,4), 16 Bit
Akku 1(2,3,4)-LL	Low-Byte des Low-Wort im Akku 1 (2,3,4), 8 Bit
Akku 1(2,3,4)-LH	High-Byte des Low-Wort im Akku 1 (2,3,4), 8 Bit
ANZ 0, ANZ 1	Wort-Anzeigen, codiert
AWL	Anweisungsliste
BASP	Befehlsausgabesperre
BSTACK	Bausteinstack
BCD	Binär codierte Dezimalzahl
BCF	Befehlscodefehler
CP	Kommunikationsprozessor
D, DL/DR, DW, DD	Datum (1 Bit), Datum links/rechts (8 Bit), Datenwort (16 Bit), Datendoppelwort (32 Bit)
DB	Datenbaustein
DBA	Datenbausteinanfangsadresse (im Register 6)
DBL	Datenbausteinlänge (im Register 8)
DX	Erweiterter Datenbaustein
EPROM	Erasable Programmable Read Only Memory (löscharer, programmierbarer Nur-Lese-Speicher)
ERAB	Erstabfrage (Bit-Anzeige)
FB	Funktionsbaustein
FUP	Funktionsplan
FX	Erweiterter Funktionsbaustein
IP	Intelligente Peripheriebaugruppe
KOP	Kontaktplan
KOR	Koordinatorbaugruppe
LZF	Laufzeitfehler
M, MB, MW, MD	Merkerbit, Merkerbyte, Merkerwort, Merkerdoppelwort
OB	Organisationsbaustein
OR	Oder (Bit-Anzeige)
OS	Overflow speichernd (Wort-Anzeige)
OV	Overflow (Wort-Anzeige)
PA	Prozeßabbild
PAA	Prozeßabbild der Ausgänge
PAE	Prozeßabbild der Eingänge
PB	Programmbaustein
PB, PW	Peripheriebyte (Programmiergerät PG 675), Peripheriewort
PG	Programmiergerät
Proz.	Prozessor
PY	Peripheriebyte (Programmiergerät PG 685)
QB, QW	Byte, Wort aus Bereich 'Erweiterte Peripherie'
QVZ	Quittungsverzug
RAM	Random Access Memory (Speicher mit wahlfreiem Zugriff)
SAZ	STEP-Adreßzähler (im Register 15)
SB	Schrittbaustein
SF	Sonderfunktion
STA	Status (Bit-Anzeige)
T	Timer (Zeitzellen)
USTACK	Unterbrechungsstack
VKE	Verknüpfungsergebnis (Bit-Anzeige)
Z	Zähler (Zählzellen)
ZYK	Zyklusfehler

## Literaturverzeichnis

Eine Einführung in das Programmieren mit STEP 5 und eine Erläuterung der Programmierung des Automatisierungsgerätes SIMATIC S5-135U ist erläutert in folgenden Handbüchern:

S5-135U programmieren mit STEP 5  
Siemens AG, ISBN 3-8009-1461-1  
(S- und R-Prozessor)

Automatisieren mit SIMATIC S5-135U  
Siemens AG, ISBN 3-8009-1522-7  
(S- und R-Prozessor sowie CPU 928)

Beachten Sie außerdem in Ihrem Gerätehandbuch die übrigen Kapitel zur CPU 928 (Betriebsanleitung, STEP5-Operationsliste).

## 1 Einführung: AG 135U - Arbeitsweise und Anwendungsbereich

Dieses Kapitel ist für Anwender, die zum ersten Mal ein Automatisierungsgerät einsetzen, aber schon andere Mikrocomputersysteme kennen.

### Anlagenstruktur

Ein Automatisierungsgerät (AG) ist ein Computersystem, das speziell für den Einsatz in der Industrie, z.B. zum Steuern eines Fertigungsautomaten, entwickelt wurde. AGs sind modular aufgebaut und bestehen aus einem Baugruppenrahmen mit mindestens einer Prozessorbaugruppe, im folgenden kurz 'Prozessor' genannt, und einer Anzahl von Peripheriebaugruppen. Welche und wieviele Prozessoren und Peripheriebaugruppen eingesetzt werden, hängt von der Automatisierungsaufgabe ab.

Das Automatisierungsgerät S5-135U gehört zur Familie der speicherprogrammierbaren Steuerungen SIMATIC S5. Es ist ein leistungsfähiges Mehrprozessorgerät zur Prozeßautomatisierung (Steuern, Melden, Überwachen, Regeln, Protokollieren) und kann sowohl für den Aufbau einfachster Steuerungen mit binären Signalen als auch zur Lösung umfangreicher Automatisierungsaufgaben eingesetzt werden.

Das Zentralgerät der S5-135U können Sie wahlweise bestücken

- mit einem Prozessor im Einzelprozessorbetrieb oder
- mit einem Koordinator (KOR) und bis zu 4 Prozessoren im Mehrprozessorbetrieb;
- zusätzlich mit Kommunikationsprozessoren (CPs):  
bis zu 7 CPs im Einzelprozessorbetrieb bzw. 4 bis zu 6 (7) CPs im Mehrprozessorbetrieb.

Die weiteren, freien Steckplätze im Zentralgerät der S5-135U stehen für Ein- und Ausgabebaugruppen zur Verfügung. Zur Erweiterung der Peripherie können Erweiterungsgeräte (EGs) an das Zentralgerät angeschlossen werden.

Beachten Sie dazu den Katalog "Automatisierungsgerät S5-135U" ST 54.1, Bestell-Nr. E86010-K4654-A111-A3.



Die folgende Abbildung zeigt den typischen Aufbau einer Anlage mit der S5-135U. Für den Einzelprozessorbetrieb genügen die fett umrandeten Baugruppen.

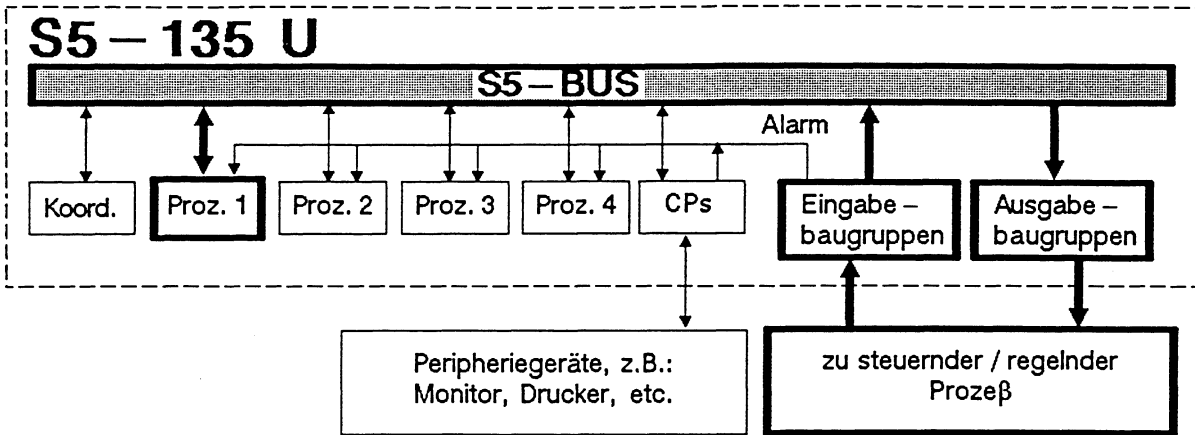


Abb. 1-1: Typische Anlagenstruktur AG S5-135U

### Anwendungsbereich

Im Einzelprozessorbetrieb ist für einfachere Automatisierungsaufgaben je nach Schwerpunkt einer der folgenden Prozessoren einzusetzen:

- S-Prozessor, besonders geeignet für Steuerungsaufgaben (schnelle Bitverarbeitung)
- R-Prozessor, besonders geeignet für Regelungsaufgaben, Rechnen, Kommunikation (schnelle Wortverarbeitung)
- M-Prozessor, zur Meßwertverarbeitung; programmierbar in Assembler und in höheren Programmiersprachen (BASIC, C)
- CPU 928, universell einsetzbar, schnelle Bit- und Wortverarbeitung.

Für komplexere Automatisierungsaufgaben läßt sich das Zentralgerät S5-135U - im Gegensatz zu vielen anderen AGs - durch den gleichzeitigen Einsatz mehrerer Prozessoren zu einem Mehrprozessorgesamtgerät ausbauen:

Der Mehrprozessorbetrieb ist immer dann sinnvoll, wenn der zu beeinflussende Prozeß für einen Prozessor zu umfangreich ist und er sich in mehrere voneinander weitgehend unabhängige Teilaufgaben gliedern läßt. Jeder Teilaufgabe kann ein besonders dafür geeigneter Prozessor zugeordnet werden (siehe oben). Dabei bearbeitet jeder einzelne Prozessor sein individuelles Anwenderprogramm unabhängig von den anderen Prozessoren.

Über einen gemeinsamen Bus (= S5-Bus) greifen die Prozessoren nacheinander auf die Peripheriebaugruppen zu. Eine zusätzliche Baugruppe, der Koordinator, teilt den Prozessoren der Reihe nach in festen Zeitabschnitten den S5-Bus zu. Nur der Prozessor, dem der S5-Bus zugeteilt ist, kann auf die Peripherie zugreifen.

Über den S5-Bus können die Prozessoren miteinander Daten austauschen. Dieser Datenverkehr erfolgt mit Hilfe eines Zwischenspeichers im Koordinator.

### Arbeitsweise

Innerhalb eines Prozessors wiederholt sich ständig folgender Zyklus:

1. Alle dem Prozessor zugeordneten Eingabebaugruppen werden abgefragt und die eingelesenen Werte im Prozeßabbild der Eingänge (PAE) zwischengespeichert.
2. Die im PAE enthaltenen Werte werden durch das Anwenderprogramm verarbeitet und die auszugebenden Werte in das Prozeßabbild der Ausgänge (PAA) eingetragen.
3. Die im Prozeßabbild der Ausgänge enthaltenen Werte werden an die dem Prozessor zugeordneten Ausgabebaugruppen ausgegeben.

Die Zeit, die der Prozessor für diese drei Aufgaben benötigt, wird Zykluszeit genannt.

Der Zyklus muß ausreichend schnell ablaufen. Die Prozeßzustände dürfen sich nicht schneller ändern, als daß der Prozessor darauf reagieren kann. Ansonsten gerät der Prozeß außer Kontrolle. Als maximale Reaktionszeit muß die doppelte Zykluszeit berücksichtigt werden. Die Zykluszeit hängt ab von Art und Umfang des Anwenderprogramms (siehe unten) und ist oftmals nicht konstant.

Für Prozesse, die in konstanten Zeitabschnitten Steuersignale benötigen, kann zusätzlich ein zeitgesteuertes Programm vorgesehen werden. Nach Ablauf des Zeitabschnittes wird das zyklische Programm zur Bearbeitung des zeitgesteuerten Programms unterbrochen. Bei der CPU 928 sind bis zu 9 zeitgesteuerte Programme möglich! Die Zykluszeit erhöht sich um die Bearbeitungszeit des zeitgesteuerten Programms.

Einem Prozeßsignal, auf das besonders schnell reagiert werden muß, kann im Prozessor ein alarmgesteuertes Programm zugeordnet werden. Nach einem Alarm unterbricht der Prozessor das zyklische oder zeitgesteuerte Programm zur Bearbeitung des alarmgesteuerten Programms. Die Zykluszeit erhöht sich um die Bearbeitungszeit des alarmgesteuerten Programms.

Im ungünstigsten Fall setzt sich die Zykluszeit aus der Bearbeitungszeit des zyklischen Programms und der Bearbeitungszeit der u.U. mehrfach aufgerufenen zeit- und alarmgesteuerten Programme zusammen. Jeder Prozessor überwacht die Zykluszeit. Bei Überschreiten eines programmierbaren Grenzwerts unterbricht er die Programmbearbeitung, bringt sich und die anderen in den Stoppzustand und nimmt die Ausgangssignale zurück.

## Programm

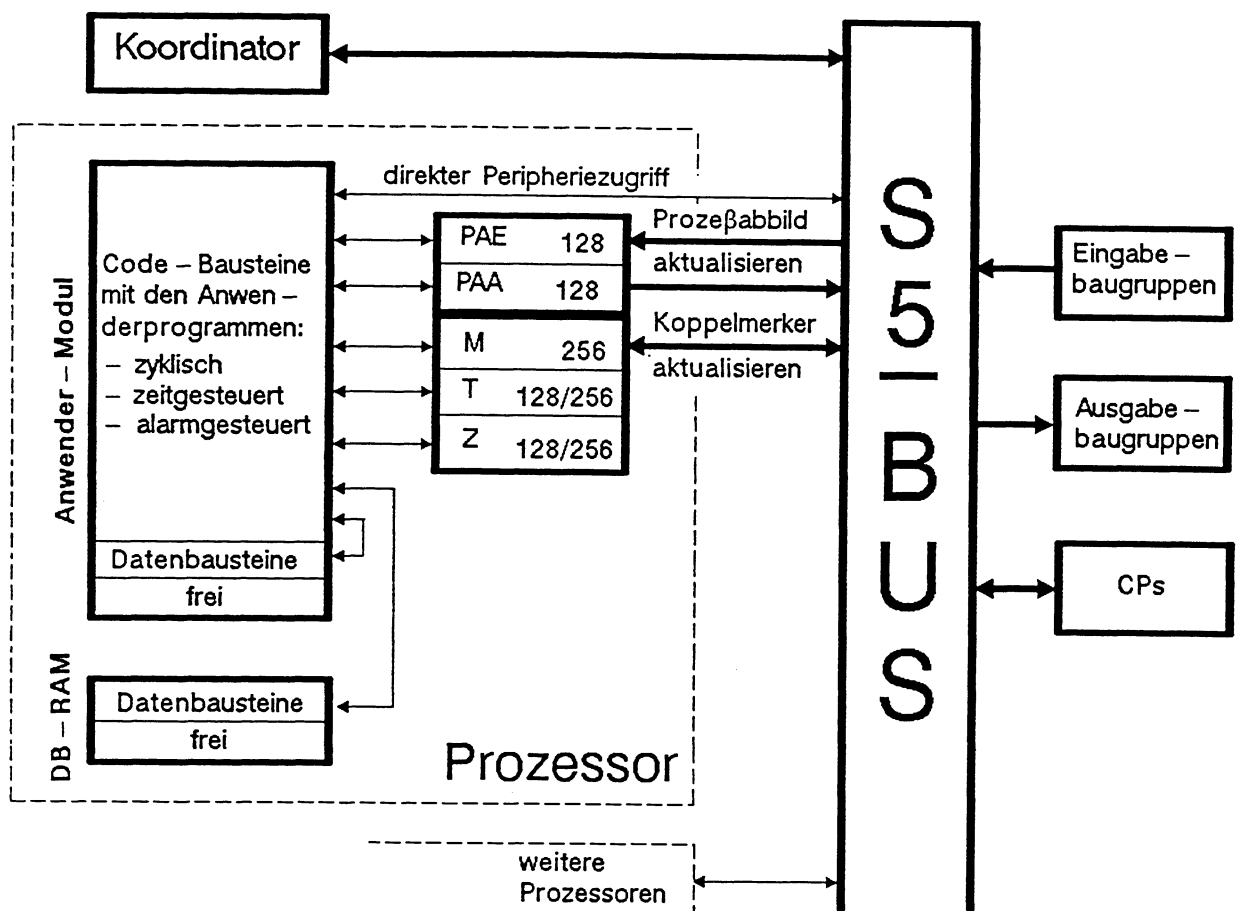
Das in jedem Prozessor vorhandene Programm unterteilt sich in das Anwenderprogramm und das Systemprogramm.

STEP5-Anwenderprogramme für das AG 135U werden mit der speziell für Automatisierungsgeräte entwickelten Programmiersprache STEP5 erstellt (Ausnahme: M-Prozessor). Das Anwenderprogramm ist modular aufgebaut und besteht aus mindestens einem Programmodul (Baustein). Es wird zwischen zwei grundlegenden Bausteintypen unterschieden:

- a) Code-Bausteine: Bausteine, die STEP5-Befehle enthalten.
- b) Datenbausteine: Bausteine mit den Konstanten und Variablen für das STEP5-Programm.

Auf das Systemprogramm hat der Anwender keinen Zugriff. Das Systemprogramm unterstützt alle für ein Automatisierungsgerät typischen Funktionen. Hierzu gehört

- das Aktualisieren der Prozeßabbilder (Eingänge, Ausgänge, Koppelmerker),
- die Aktualisierung der Zeitzellen,
- der Aufruf des zyklischen, zeit- und alarmgesteuerten Programms.



Blockschaltbild eines Prozessors im S5-135U (Mehrprozessorbetrieb)

## Interner Aufbau eines Prozessors

Der Speicher eines Prozessors ist in mehrere Bereiche aufgeteilt. Die wichtigsten Bereiche sind:

- Anwenderspeicher (max. 32K Wörter)

Der Anwenderspeicher befindet sich auf einem steckbaren RAM- oder EPROM-Modul und enthält Code- und Datenbausteine.

- Datenbaustein-RAM (= DB-RAM, max. 23,375K Wörter)

Das DB-RAM ist ein Speicherbereich zur Aufnahme von Datenbausteinen. Datenbausteine, deren Inhalt vom Anwenderprogramm verändert werden soll, müssen vom EPROM-Modul in das DB-RAM kopiert werden.

- Merkerbereich M (256 Bytes)

Der Merkerbereich ist ein Speicherbereich, auf den das Anwenderprogramm sehr schnell zugreifen kann. Der Merkerbereich sollte bevorzugt für oft benötigte Arbeitsdaten verwendet werden.

Auf folgende Datentypen kann zugegriffen werden: Einzelbits, Bytes, Wörter und Doppelwörter.

Einzelne Merkerbytes können als Koppelmerker zum Datenaustausch zwischen den Prozessoren genutzt werden. Koppelmerker werden vom Systemprogramm am Zyklusende über einen Zwischenspeicher im Koordinator aktualisiert.

- Prozeßabbild der Ein- und Ausgänge PAE/PAA (je 128 Bytes)

Auf das Prozeßabbild kann das Anwenderprogramm in gleicher Weise zugreifen wie auf den Merkerbereich. Das Prozeßabbild wird am Zyklusende vom Systemprogramm aktualisiert.

- Peripheriebereich (512 Bytes)

Das Anwenderprogramm kann unter Umgehung des Prozeßabbildes direkt über den S5-Bus auf die Peripheriebaugruppen zugreifen. Folgende Datentypen sind möglich: Bytes und Wörter.

- Zeiten T (128 Zeitzellen bei S- und R-Prozessor, 256 Zeitzellen bei CPU 928)

Zeitzellen werden vom Anwenderprogramm mit einem Zeitwert zwischen 10ms und 9990sec geladen und vom Systemprogramm im Abstand von 10ms heruntergezählt.

- Zähler Z (128 Zähler bei S- und R-Prozessor, 256 Zähler bei CPU 928)

Zählzellen werden vom Anwenderprogramm mit einem Anfangswert (max. 999) geladen und hinauf- bzw. heruntergezählt.

Die STEP5-Befehle können auf folgende Operandenbereiche zugreifen:

- Merkerbereich
- Prozeßabbild der Ein- und Ausgänge
- Peripheriebereich
- Zeiten
- Zähler
- aktueller Datenbaustein

Für den Zugriff auf diese Operandenbereiche verwenden die STEP5-Befehle zwei unterschiedliche Mechanismen:

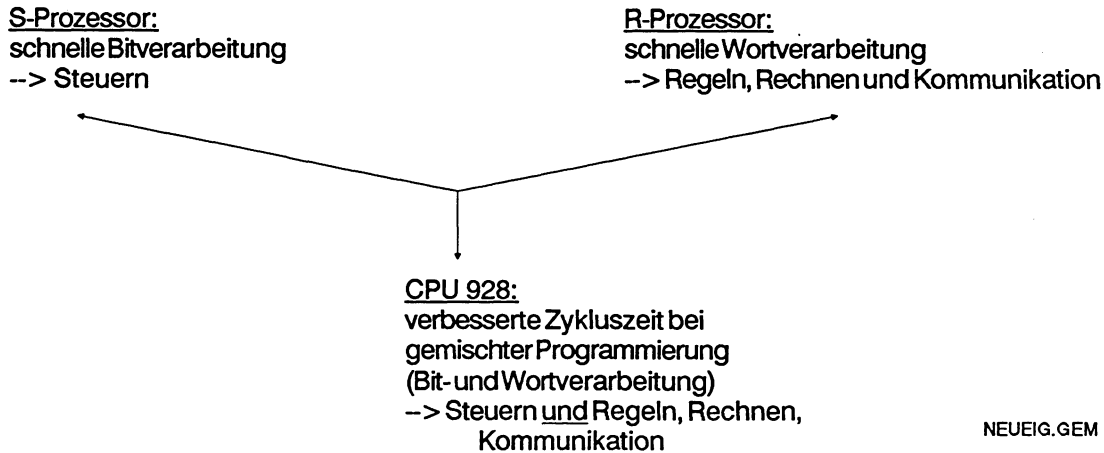
- Der überwiegende Teil der STEP5-Befehle adressiert eine Speicherzelle relativ zum Beginn eines Operandenbereiches. Solange ausschließlich mit diesen Befehlen gearbeitet wird, ist das Programm von den Operandenbereichen getrennt und kann sich im Fehlerfall nicht selbst überschreiben.
- Einige STEP5-Befehle arbeiten mit absoluter Adressierung. Mit diesen Befehlen kann auf den gesamten Speicherbereich zugegriffen werden.

Im Vergleich zu anderen Operandenbereichen hat der aktuelle Datenbaustein keine feste Anfangsadresse und Länge. Der aktuelle Datenbaustein ist der Datenbaustein, dessen Anfangsadresse und Länge in speziellen Registern (siehe unten) eingetragen ist. Das Anwenderprogramm kann - falls keine Befehle mit absoluter Adressierung verwendet werden - ausschließlich auf den aktuellen Datenbaustein zugreifen. Folgende Datentypen sind möglich: Einzelbits, Bytes, Wörter und Doppelwörter. Zugriffe auf den aktuellen Datenbaustein sind langsamer als Zugriffe auf den Merkerbereich.

Außer den oben genannten Speicherbereichen enthält der Prozessor mehrere Register:

- 4 Akkumulatoren (32 Bits), die als Vielzweckregister dienen, z.B. als Hilfsregister beim Speicher-Speicher-Transfer oder als Register für Operanden und Rechenergebnisse.
- 1 Befehlszähler (STEP-Adreßzähler, SAZ), der die Adresse des nächsten auszuführenden Befehls enthält.
- 1 Bausteinstackzeiger (Bausteinstack-Pointer = BSP), der die Einträge in den Bausteinstack verwaltet.
- 1 DBA-Register (DBA = Datenbausteinanfangsadresse), das die Anfangsadresse des aktuellen Datenbausteins enthält.
- 1 DBL-Register (DBL = Datenbausteinlänge), das die Anzahl der Datenwörter des aktuellen Datenbausteins enthält.
- 1 Anzeigenregister.
- 1 BR-Register (BR = Basisadreßregister), das für Adressierungen bei absoluten Speicherzugriffen verwendet wird.

## 1.1 Neue Eigenschaften und Funktionen der CPU 928 (für Anwender von S- bzw. R-Prozessor)



Die CPU 928 ist eine 40 mm breite Baugruppe und belegt damit zwei Steckplätze im Zentralgerät 135U. Sie vereinigt die Vorteile des S-Prozessors (schnelle Bitverarbeitung, optimiert für Steuerungsaufgaben) und des R-Prozessors (schnelle Wortverarbeitung, optimiert für Regelungsaufgaben). Die CPU 928 ist darüber hinaus besonders geeignet zum Überwachen und Melden, zur Kommunikation im Mehrprozessorbetrieb und zum Bedienen und Beobachten. Die CPU 928 ist damit ein Prozessor, den Sie bei der Lösung von Automatisierungsaufgaben universell einsetzen können.

Wenn Sie mit S- bzw. R-Prozessor in der S5-135U bereits vertraut sind, beachten Sie besonders die Beschreibung neuer Eigenschaften und Funktionen der CPU 928 in den folgenden Kapiteln (*kursiv gedruckte Angaben gelten nur für die Version 3UA12!*):

### Kapitel 3.1.1: Programmorganisation

*Die maximale Baustein-Schachtelungstiefe ist auf die Anzahl '62' erhöht worden (CPU 928-3UA11: '30', R-Prozessor: '20').*

*Die maximal zulässige Zykluszeit beträgt nun 6000 ms (CPU 928-3UA11 und R-Prozessor: 4000 ms).*

### Kapitel 3.3: Ergänzende Operationen

Der STEP5-Operationsvorrat wurde erweitert. Folgende Befehle sind neu:

- Befehle zur Addition und Subtraktion von 32-Bit-Festpunktzahlen: +D, -D, ADD DF (Systemoperationen) <sup>1)</sup>
- Befehle zum Laden und Transferieren eines Wortes in den BB- oder BT-Bereich: L BB, T BB, L BT, T BT (Ergänzende Operationen)

<sup>1)</sup> Die Programmierung dieser Befehle ist abhängig vom PG-Typ und vom Ausgabestand der PG-Systemsoftware.

## **Kapitel 4.4.2: WECKALARME**

Sie können jetzt bis zu 9 zeitgesteuerte Programme bearbeiten lassen. Die einzelnen Programme stehen in den Organisationsbausteinen OB 10 bis OB 18. Jeder OB wird in einem anderen Zeitraster aufgerufen: Der OB 10 wird beispielsweise alle 10 ms bearbeitet, der OB 15 alle 500 ms. Weckalarm-OBs mit kürzerem Zeitraster sind höherprior als Weckalarme mit längerem Zeitraster und werden bei Bedarf in diese eingeschachtelt.

## **Kapitel 5.6: Fehler im ANLAUF und im RUN**

Die Fehlerkennungen in Akku 1 und 2 sind erweitert worden.

## **Kapitel 6: Integrierte Sonderfunktionen**

In der CPU 928 stehen neue Sonderfunktionen zur Verfügung. Dieses sind

OB 110	: Zugriff auf das Anzeigenbyte
OB 111	: Akku 1, 2, 3 und 4 löschen
OB 112	: Akku Roll Up
OB 113	: Akku Roll Down
OB 120	: "Alarme gemeinsam sperren" ein-/ausschalten
OB 121	: "Weckalarme einzeln sperren" ein-/ausschalten
OB 122	: "Alarme gemeinsam verzögern" ein-/ausschalten
OB 123	: "Weckalarme einzeln verzögern" ein-/ausschalten
OB 160 bis OB 163:	Zählschleife
OB 170	: Bausteinstack (BSTACK) lesen
OB 180	: Variabler Datenbaustein-Zugriff
OB 181	: Datenbausteine testen
OB 190 und OB 192:	Merker in Datenbausteine übertragen
OB 191 und OB 193:	Datenblöcke in Merkerbereich übertragen
OB 228	: Statusinformation einer Programmbearbeitungsebene lesen

## **Kapitel 7: Datenbaustein DX 0**

Bei der CPU 928 können Sie je 256 Zähler und Zeitzellen einsetzen (R-Prozessor: 128 Zähler, 128 Zeitzellen).

Die Parameter für die alarmgesteuerte Programmbearbeitung sind erweitert worden.

Für die Gleitpunktarithmetik können Sie im DX 0 einstellen, ob der Prozessor mit einer 16-Bit- oder einer 24-Bit-Mantisse rechnen soll.

## **Kapitel 8.1: Adreßraumaufteilung in der CPU 928**

Das Datenbaustein-RAM der CPU 928 ist auf 23,375K Wörter erweitert worden (R-Prozessor: 11,125K Wörter). Dadurch können Sie mehr Datenbausteine bearbeiten lassen als bisher.

Es stehen Ihnen außerdem zwei neue Operandenbereiche mit je 256 Wörtern Länge zur Verfügung, die Sie frei nutzen können: der BB- und der BT-Bereich. Für den Zugriff auf diese Bereiche gibt es neue STEP5-Befehle.

## **Kapitel 9: Speicherzugriffe über absolute Adressen**

Die durch die STEP5-Befehle LIR, TIR, TNB und TNW sinnvoll ansprechbaren Bereiche im Adreßraum sind bei der CPU 928 vergrößert und enthalten weniger Lücken.

### **Kapitel 9.3: Operationen mit dem BR-Register**

Zur einfacheren Adressierung bei den absoluten Speicherzugriffen wurde das BR-Register (BR = Basisadreßregister) eingeführt.

*Es gibt*

- neue Befehle, mit denen das BR-Register geladen oder verändert werden kann (siehe 9.3.1), <sup>1)</sup>
- neue Befehle, um die Inhalte einzelner Register verschieben zu können (siehe 9.3.2), <sup>1)</sup>
- neue Befehle, mit denen auf lokale oder globale Speicherbereiche zugegriffen wird (siehe 9.3.3 und 9.3.4), <sup>1)</sup>
- neue Befehle für Zugriffe auf den Kachelspeicher. <sup>1)</sup>

## **ANHANG A und Operationsliste:**

Mit der Version 3UA12 (12 MHz) sind - im Vergleich zur Version 3UA11 - die Befehls- und Systemlaufzeiten der CPU 928 um ca. <sup>1</sup>/<sub>3</sub> verbessert worden.

## **ANHANG G:**

Die Kennungen der einzelnen Programmbearbeitungsebenen sind geändert bzw. erweitert worden.

<sup>1)</sup> Die Programmierung dieser Befehle ist abhängig vom PG-Typ und dem Ausgabestand der PG-Systemsoftware.



## 2 Anwenderprogramm

### 2.1 Programmiersprache STEP5

Mit der Programmiersprache STEP5 setzen Sie die Aufgaben der Automatisierungstechnik in Programme um, die in den SIMATIC S5-Automatisierungsgeräten ablaufen. In STEP5 können Sie sowohl einfache binäre Funktionen als auch komplexe digitale Funktionen und arithmetische Grundoperationen programmieren.

Der **Befehlsumfang** der Programmiersprache STEP5 gliedert sich in

#### Grundoperationen:

- In allen Bausteinen anwendbar
- Darstellungsarten Kontaktplan (KOP), Funktionsplan (FUP), Anweisungsliste (AWL)

#### Ergänzende Operationen:

- Nur in Funktionsbausteinen anwendbar
- Darstellungsart nur Anweisungsliste (AWL)

#### Systemoperationen:

- Diese gehören zu den Ergänzenden Operationen
- Nur in Funktionsbausteinen anwendbar
- Darstellungsart nur Anweisungsliste (AWL)
- Nur für Anwender mit sehr guten Systemkenntnissen!

### 2.1.1 Darstellungsarten KOP, FUP und AWL

Beim Programmieren in STEP5 können Sie zwischen den drei Darstellungsarten Kontaktplan (KOP), Funktionsplan (FUP) und Anweisungsliste (AWL) wählen, so daß die Programmiermethode dem jeweiligen Anwendungsfall angepaßt werden kann.

Die von den Programmiergeräten (PGs) erzeugte Maschinensprache ist bei den drei Darstellungsarten identisch.

*Wenn Sie beim Programmieren in STEP5 bestimmte Regeln berücksichtigen, kann das PG Ihr Anwenderprogramm von einer Darstellungsart in jede andere übersetzen!*

Während Sie mit den Darstellungsarten Funktions- und Kontaktplan die Möglichkeit haben, Ihr STEP5-Programm graphisch darzustellen, werden in der Anweisungsliste die einzelnen STEP5-Befehle aufgelistet.

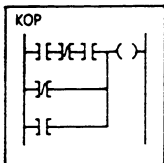
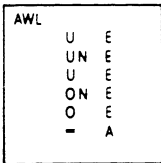
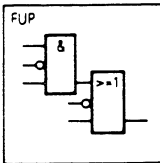
Kontaktplan	Anweisungsliste	Funktionsplan
Programmieren mit grafischen Symbolen wie Stromlaufplan  entspricht DIN 19239	Programmieren mit mnemotechnischen Abkürzungen der Funktionsbezeichnungen  entspricht DIN 19239	Programmieren mit grafischen Symbolen  entspricht IEC 117-15 DIN 40700 DIN 40719 DIN 19239
		

Abb. 2-1: Darstellungsarten der Programmiersprache STEP5

GRAPH 5 ist eine Programmiersprache zur graphischen Darstellung von Ablaufsteuerungen. Sie ist den Darstellungsarten KOP, FUP und AWL übergeordnet. Das mit GRAPH 5 geschriebene Programm in bildhafter Darstellung wird vom Programmiergerät automatisch in ein STEP5-Programm umgesetzt.

### 2.1.2 Strukturierte Programmierung

Das Gesamtprogramm eines Prozessors besteht aus dem

Systemprogramm: Es enthält die Gesamtheit aller Anweisungen und Vereinbarungen zur Realisierung geräteinterner Betriebsfunktionen (z.B. Sicherstellen von Daten bei Ausfall der Versorgungsspannung, Veranlassen von Anwenderreaktionen bei Unterbrechungen usw.).

Es ist in sog. EPROMs (Erasable Programmable Read Only Memory) hinterlegt und damit ein fester Bestandteil des Prozessors. Als Anwender haben Sie *keine Zugriffsmöglichkeit* auf das Systemprogramm.

Anwenderprogramm: Es enthält die Gesamtheit aller vom Anwender programmierten Anweisungen und Vereinbarungen für die Signalverarbeitung, durch die eine zu steuernde Anlage (Prozeß) gemäß der Steuerungsaufgabe beeinflußt wird. Das Anwenderprogramm ist in Bausteine unterteilbar.

Das gesamte Anwenderprogramm kann in einzelne, in sich abgeschlossene Programmabschnitte (Bausteine) aufgeteilt werden. Die Gliederung Ihres Anwenderprogramms verdeutlicht somit auf den ersten Blick die wesentlichen Programmstrukturen oder hebt programmtechnisch zusammenhängende Anlagenteile hervor.

Dieses Verfahren des **"strukturierten Programmierens"** bietet Ihnen folgende Vorteile:

- einfache und übersichtliche Programmierung auch großer Programme,
- Möglichkeit zum Standardisieren von Programmteilen,
- einfache Programmorganisation,
- leichte Änderungsmöglichkeiten,
- einfacher abschnittsweiser Programmtest,
- einfache Inbetriebnahme.

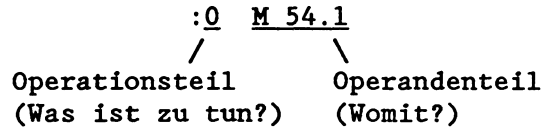
#### Was ist ein Baustein ?

Ein Baustein ist ein durch Funktion, Struktur oder Verwendungszweck abgegrenzter Teil des Anwenderprogramms. Man unterscheidet Bausteine, die Anweisungen zur Signalverarbeitung enthalten (Organisationsbausteine, Programmbausteine, Funktionsbausteine, Schrittbausteine), und Bausteine, die Daten enthalten (Datenbausteine).

### 2.1.3 STEP5-Operationen

Eine STEP5-Operation ist die kleinste selbständige Einheit des Anwenderprogramms. Sie ist die Arbeitsvorschrift für den Prozessor. Eine STEP5-Operation setzt sich zusammen aus einem Operationsteil und einem Operandenteil.

Beispiel:



Den Operandenteil können Sie entweder **absolut** oder **symbolisch** (über die Zuordnungsliste) eingeben.

Beispiel für die absolute Darstellung:                   :U   E 1.4

Beispiel für die symbolische Darstellung:               :U   -Motor1

Beachten Sie zur absoluten und symbolischen Programmierung die Bedienungsanleitung "Programmiergerät PG 685", Bestell-Nr. C79000-B8500-C373-02.

Der Operationsumfang von STEP5 ermöglicht es Ihnen,

- binäre Werte miteinander zu verknüpfen,
- Werte zu laden, zu speichern und zu transferieren,
- Werte miteinander zu vergleichen und arithmetisch zu bearbeiten,
- Zeit- und Zählwerte vorzugeben,
- Zahlendarstellungen umzuwandeln,
- das Anwenderprogramm zu strukturieren,
- die Programmbearbeitung zu beeinflussen, etc.

Die meisten STEP5-Operationen verwenden als Quelle oder Ziel für die Operanden und als Ziel für das Ergebnis einer Operation zwei Register: den Akkumulator 1 (Akku 1) und den Akkumulator 2 (Akku 2). Ein Akkumulator ist jeweils 32 Bit (1 Doppelwort) breit.

Der gesamte STEP5-Operationsvorrat wird ausführlich in Kapitel 3.2 beschrieben. Sie finden dort Programmierbeispiele zu den einzelnen STEP5-Befehlen.

Anhang C enthält zusätzlich eine Übersicht über alle vorhandenen STEP5-Operationen und ihrer zulässigen Parameter in Form einer Liste.

#### 2.1.4 Zahlendarstellung

Damit der Prozessor Zahlenwerte miteinander verknüpfen, verändern oder vergleichen kann, müssen diese in einer binär-codierten Darstellung in die Akkumulatoren geladen werden.

Abhängig von der auszuführenden Operation sind in STEP5 folgende Zahlendarstellungen zulässig:

- Dualzahlen:           a) 16-Bit-Festpunktzahlen  
                          b) 32-Bit-Festpunktzahlen  
                          c) Gleitpunktzahlen
- Dezimalzahlen:       d) BCD-codierte Zahlen

Bei der Ein- und Ausgabe von Zahlenwerten stellen Sie am Programmiergerät das Datenformat (z.B. KF für Festpunkt) ein, in dem Sie den Zahlenwert eingeben bzw. angezeigt haben möchten. Auf diese Weise übernimmt das PG die Umrechnung aus der intern verwendeten Zahlendarstellung in eine direkt lesbare Darstellungsart.

Mit den 16-Bit-Festpunktzahlen und Gleitpunktzahlen können Sie alle arithmetischen Operationen wie Vergleichen, Addieren, Subtrahieren, Multiplizieren und Dividieren ausführen.

BCD-codierte Zahlen werden nur bei der Ein- und Ausgabe verwendet; mit ihnen können direkt *keine* arithmetischen Operationen durchgeführt werden.

Mit 32-Bit-Festpunktzahlen werden Vergleichsoperationen durchgeführt. Außerdem werden sie bei Umwandlungen von BCD-codierten Zahlen in Gleitpunktzahlen als Zwischenstufe benötigt. Mit den neuen Befehlen +D und -D können sie auch für Additionen und Subtraktionen verwendet werden.

Die STEP5-Sprache enthält **Umwandlungsoperationen**, mit denen Sie Zahlen direkt in die wichtigsten Zahlendarstellungen umwandeln lassen können.

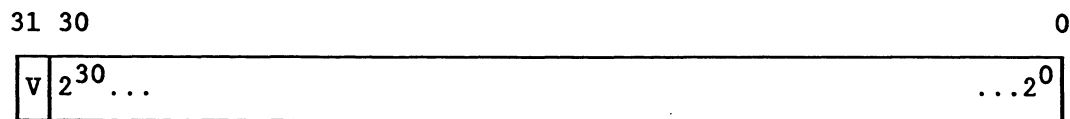
#### 16-Bit- und 32-Bit-Festpunktzahlen

Festpunktzahlen sind ganze, mit einem Vorzeichen versehene Dualzahlen.

Sie sind 16 Bit (= 1 Wort) bzw. 32 Bit (= 2 Wörter) lang, wobei das Bit Nr. 15 bzw. Nr. 31 das Vorzeichen enthält: "0" = positive Zahl; "1" = negative Zahl.

Negative Zahlen werden in ihrem 2er-Komplement dargestellt.

32-Bit-Festpunktzahl:



Angabe des Datenformats für 16-Bit-Festpunktzahlen am PG: KF

Angabe des Datenformats für 32-Bit-Festpunktzahlen: nur KH

zulässiger Zahlenbereich: -32768 bis +32767 (16 Bit)

-2147483648 bis +2147483647 (32 Bit)

(Zur Umwandlung einer 16-Bit-Festpunktzahl in eine 32-Bit-Festpunktzahl siehe Kapitel 3.2.2 "Ergänzende Operationen".)

Festpunktzahlen werden bei einfachen Rechenaufgaben und beim Vergleich von Zahlenwerten verwendet. Da Festpunktzahlen immer ganze Zahlen sind, beachten Sie bitte, daß bei der Ausführung von Divisionen keine Restbildung möglich ist!

### Gleitpunktzahlen

Gleitpunktzahlen sind positive und negative gebrochene Zahlen. Sie belegen immer ein Doppelwort (32 Bit). Eine Gleitpunktzahl wird als Exponentialzahl dargestellt. Die Mantisse ist 24 Bit, der Exponent 8 Bits lang.

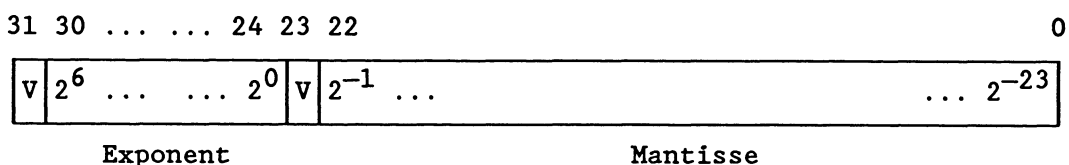
Der Exponent gibt die Größenordnung der Gleitpunktzahl an. Am Vorzeichen des Exponenten erkennen Sie, ob der Betrag der Gleitpunktzahl größer oder kleiner als 0,1 ist.

Die Mantisse gibt die Genauigkeit der Gleitpunktzahl an:

- Genauigkeit bei einer 24-Bit-Mantisse:  $2^{-24} = 0,000000059604$   
(entspricht 7 Nachkommastellen)
- Genauigkeit bei einer 16-Bit-Mantisse:  $2^{-16} = 0,000015258$   
(entspricht 4 Nachkommastellen)

Ist das Vorzeichen der Mantisse "0", ist die Zahl positiv; bei Vorzeichen "1" ist es eine negative Zahl in 2er-Komplement-Darstellung.

Gleitpunktzahl:



In der Voreinstellung rechnet die CPU 928 beim Addieren, Subtrahieren, Multiplizieren und Dividieren mit einer 16 Bit breiten Mantisse (Bit 8 bis 23). Die niederwertigen (rechts stehenden) Bit 0 bis 7 haben dabei immer den Wert '0'!

Wünschen Sie bei Gleitpunktrechnungen eine höhere Genauigkeit (und können Sie dafür kleine Laufzeitverluste in Kauf nehmen), so können Sie im DX 0 "Gleitpunktarithmetik mit 24-Bit-Mantisse" einstellen (siehe Kapitel 7).

Angabe des Datenformats für Gleitpunktzahlen am PG:      KG

Zulässiger Zahlenbereich:  $\pm 0,1469368 \times 10^{-38}$  bis  $\pm 0,1701412 \times 10^{39}$

**Eingabe von Gleitpunktzahlen Z mit dem PG**

$$Z = 12,34567$$

$$L \text{ KG } + \overbrace{1234567} \quad + \overbrace{02}$$

Mantisse      Exponent (Basis 10) mit Vorzeichen

$$Z = +0,1234567 \times 10^{+2} = 12,34567$$

$$Z = -0,005$$

$$L \text{ KG } - \overbrace{50000000} \quad - \overbrace{02}$$

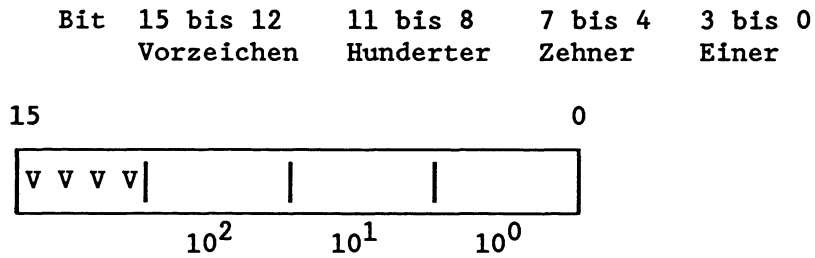
Mantisse      Exponent (Basis 10) mit Vorzeichen

$$Z = -0,5 \times 10^{-2} = -0,005$$

Verwenden Sie Gleitpunktzahlen für die Lösung umfangreicherer Rechenaufgaben, insbesondere bei Multiplikationen und Divisionen, und dann, wenn Sie mit sehr großen oder sehr kleinen Zahlen arbeiten!

## BCD-codierte Zahl

Dezimalzahlen werden als BCD-Zahlen dargestellt. Mit Vorzeichen und 3 Ziffern belegen sie im Akkumulator 16 Bits (1 Wort):



Die einzelnen Ziffern sind positive 4-Bit-Dualzahlen zwischen 0000 und 1001 (0 und 9).

Zulässiger Wertebereich: -999 bis +999

Die linken Bits sind für das Vorzeichen reserviert.

Vorzeichen bei einer positiven Zahl: "0000"

Vorzeichen bei einer negativen Zahl: "1111"



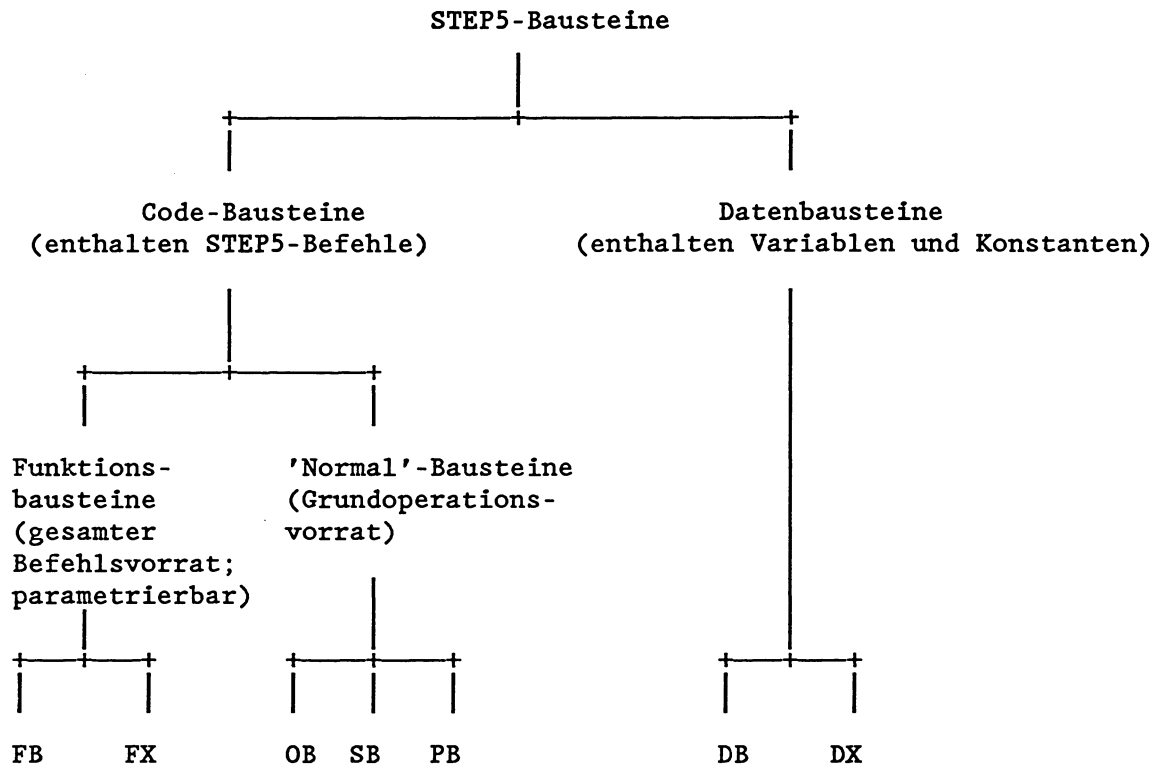
### 2.1.5 STEP5-Bausteine

Ein Baustein ist ein durch Funktion, Struktur oder Verwendungszweck abgegrenzter Teil des Anwenderprogramms.

Ein Baustein ist identifiziert durch

- die Bausteinart (OB, PB, SB, FB, FX, DB, DX)
- die Bausteinnummer (Zahl zwischen 0 und 255).

Die STEP5-Bausteine können folgendermaßen unterteilt werden:



Die Programmiersprache STEP5 unterscheidet folgende Bausteinarten:

- Organisationsbausteine (OB)

Die Organisationsbausteine sind die Schnittstelle zwischen dem Systemprogramm und dem Anwenderprogramm. Sie können in zwei Gruppen unterteilt werden:

OB 1 bis 39 werden vom Systemprogramm aufgerufen und steuern die Programmbearbeitung, das Anlaufverhalten des Prozessors und das Verhalten im Fehlerfall. Diese OBs werden vom Anwender programmiert.

OB 40 bis 255 enthalten Sonderfunktionen des Systemprogramms. Sie werden bei Bedarf vom Anwender aufgerufen.

- Programmbausteine (PB)

Sie werden zur Strukturierung des Anwenderprogramms verwendet und enthalten die nach technologischen oder funktionellen

Gesichtspunkten gegliederten Teilprogramme. Normalerweise enthalten Programmbausteine den größten Teil des Anwenderprogramms.

- **Schrittbausteine (SB)**

Sie sind spezielle Programmbausteine, die zur "schritt"-weisen Bearbeitung von Ablaufketten verwendet werden.

- **Funktionsbausteine (FB/FX)**

Sie dienen zum Programmieren von häufig wiederkehrenden oder auch komplexen Funktionen (z.B. digitale Funktionen, Ablaufsteuerungen, Regelungen, Meldefunktionen). Ein Funktionsbaustein kann von übergeordneten Bausteinen mehrfach aufgerufen werden und bei jedem Aufruf mit neuen Operanden versorgt ("parametriert") werden.

- **Datenbausteine (DB/DX)**

In Datenbausteinen stehen die (festen oder veränderbaren) Daten, mit denen das Anwenderprogramm arbeitet. Diese Bausteinart enthält keine STEP5-Anweisungen und unterscheidet sich in seiner Funktion grundsätzlich von den übrigen Bausteinen.

**So ist ein Baustein aufgebaut:**

Alle Bausteinarten bestehen aus

- einem Bausteinkopf und
- einem Bausteinrumpf.

Der **Bausteinkopf** hat immer eine Länge von 5 Datenwörtern. Vom Programmiergerät werden darin automatisch abgelegt

- die Baustein-Anfangskennung
- die Bausteinart (OB, FB...)
- die Bausteinnummer
- die PG-Kennung
- die Bibliotheksnummer
- die Bausteinlänge (einschließlich Bausteinkopf).

Bausteinkopf im  
Programmspeicher:

Anfangs-	kennung
Baustein-Art	Baustein-Nummer
PG-Kennung	B i b l i o -
t h e k s n u m m e r	
Bausteinlänge inkl. Kopf (Wörter)	

15

0

Die exakten Kennungen von Baustein-Art und Baustein-Nummer entnehmen Sie bitte dem Kapitel 8.2.1.

Im **Bausteinrumpf** sind - abhängig von der Bausteinart - enthalten

- STEP5-Befehle (bei OBs, PBs, SBs, FBs, FXs),
- variable oder konstante Daten (bei DBs, DXs),
- Formaloperandenliste (bei FBs, FXs).

Zu den Bausteinarten DB, DX, FB und FX erzeugt das Programmiergerät zusätzlich einen **Bausteinvorkopf** (DV, DXV, FV, FXV). Diese Bausteinvorköpfe enthalten Informationen über das Datenformat (bei DB und DX) bzw. über die Sprungmarken (bei FB und FX), die nur das Programmiergerät auswerten kann. Die Bausteinvorköpfe werden deshalb nicht in den AG-Speicher übertragen. Als Anwender haben Sie auf den Inhalt eines Bausteinvorkopfes direkt keinen Einfluß.

Ein STEP5-Baustein darf maximal 4096 Wörter im Programmspeicher des Prozessors belegen. Berücksichtigen Sie bei der Eingabe oder Übertragung von Bausteinen mit dem Programmiergerät die Speichergröße des verwendeten PGs!

Von den einzelnen Bausteinarten stehen Ihnen zum Programmieren zur Verfügung:

OB	1	bis	39
FB	0	bis	255
FX	0	bis	255
PB	0	bis	255
SB	0	bis	255
DB	3	bis	255
DX	1	bis	255

Die Datenbausteine DB 1, DB 2 und DX 0 enthalten Parameter. Sie sind für bestimmte Funktionen reserviert und deshalb nicht beliebig verwendbar.

Alle programmierten Bausteine werden vom PG in beliebiger Reihenfolge im Programmspeicher hinterlegt (Bild), der auf dem Prozessor als steckbares RAM- oder EPROM-Modul ausgeführt ist. Die Anfangsadressen der gespeicherten Bausteine werden im Datenbaustein DB 0 hinterlegt.

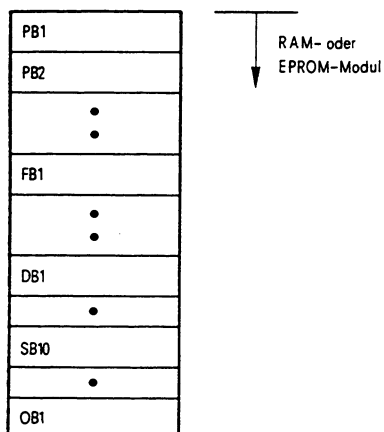


Abb. 2-2: Ablage der Bausteine in den Programmspeicher

Beim Korrigieren von Bausteinen wird der 'alte' Baustein im Speicher für ungültig erklärt und ein neuer Baustein im Speicher eingetragen. Ebenso werden beim Löschen von Bausteinen die Bausteine nicht wirklich gelöscht, sondern nur für ungültig erklärt.

**WICHTIG!**

**Gelöschte und korrigierte Bausteine belegen weiterhin Speicherplatz!**

Mit der On-line-Funktion 'Speicher KOMPRIMIEREN' schaffen Sie Speicherplatz für neue Bausteine: Die Funktion beseitigt alle ungültigen Bausteine im Speicher und schiebt die gültigen zusammen (siehe Kapitel 11.6).

## 2.2 Organisations-, Programm- und Schrittbausteine

Diese drei Bausteinarten unterscheiden sich hinsichtlich Programmierung und Aufruf nicht. Alle drei können Sie wahlweise in den Darstellungsarten KOP, FUP und AWL programmieren.

### 2.2.1 Programmierung

So gehen Sie bei der Programmierung von Organisations-, Programm- und Schrittbausteinen vor:

- Geben Sie zuerst die Bausteinart, dann die Nummer des Bausteins an, den Sie programmieren wollen.

Folgende Nummern stehen Ihnen jeweils zur Verfügung:

Programmbausteine	0 bis 255
Schrittbausteine	0 bis 255
Organisationsbausteine	1 bis 39

- Geben Sie Ihr Anwenderprogramm in STEP5 ein.

#### WICHTIG!

Bei der Programmierung von PBs, SBs und OBs dürfen Sie nur die STEP5-Grundoperationen verwenden!

- Schließen Sie die Programmeingabe mit der Anweisung "BE" (Bausteinende) ab.

#### WICHTIG!

Ein STEP5-Baustein soll immer ein abgeschlossenes Programm enthalten. Logische Verknüpfungen müssen innerhalb eines Bausteins abgeschlossen sein.

In einen Baustein können Sie (je nach Programmiergerät) bis zu 4000 Wörter schreiben.

Der Bausteinkopf, den das PG automatisch erzeugt, belegt 5 Wörter im Programmspeicher.

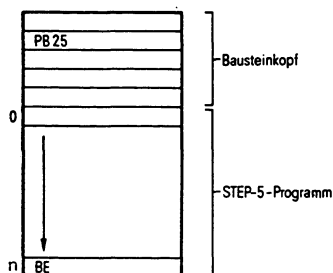


Abb. 2-3: Aufbau eines Organisations-, Programm- und Schrittbausteins

### 2.2.2 Aufruf

Bausteine müssen zum Bearbeiten freigegeben werden. Dies geschieht durch Bausteinaufrufe (Bild).

Diese Bausteinaufrufe können innerhalb eines Organisations-, Programm-, Funktions- oder Schrittbausteins programmiert werden. Sie sind vergleichbar mit Sprüngen in ein Unterprogramm. Jeder Sprung verursacht einen Bausteinwechsel.

Sprünge können sowohl unbedingt als auch bedingt ausgeführt werden:

#### - Unbedingter Aufruf: SPA xx

Der angesprochene Baustein wird *unabhängig* vom vorherigen Verknüpfungsergebnis (= VKE) bearbeitet.

Das VKE ist der Signalzustand im Prozessor, der zur weiteren binären Signalverarbeitung verwendet wird. Das VKE kann beispielsweise mit dem Signalzustand von Operanden verknüpft werden oder es werden Operationen ausgeführt abhängig vom vorherigen VKE: Die sog. "unbedingten Operationen" werden *immer* ausgeführt, die sog. "bedingten Operationen" *ausschließlich* dann, wenn das VKE = 1 ist.

Die Sprunganweisung SPA gehört zu den unbedingten Operationen. Sie hat selbst keinen Einfluß auf das VKE. Dieses wird beim Sprung in den neuen Baustein mitgenommen. Dort kann es zwar ausgewertet, jedoch nicht mehr weiter verknüpft werden.

#### - Bedingter Aufruf: SPB xx

Die Sprunganweisung SPB gehört zu den bedingten Operationen, d.h., der angesprochene Baustein wird *nur* bearbeitet, wenn das vorherige Verknüpfungsergebnis (VKE) = 1 ist. Bei VKE = 0 wird die Sprunganweisung nicht ausgeführt. Das VKE wird dabei jedoch auf "1" gesetzt!

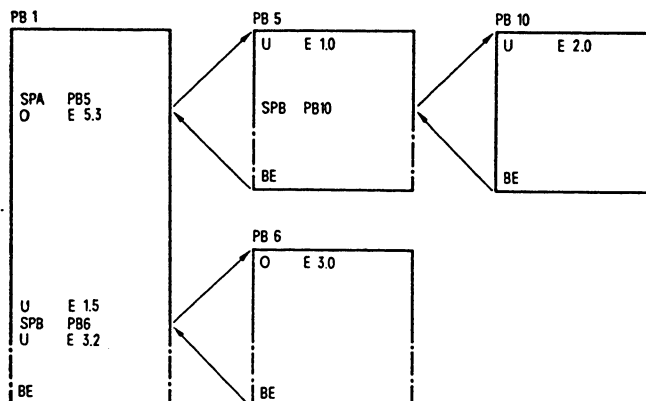


Abb. 2-4: Bausteinaufrufe, die die Bearbeitung eines Programmbausteins freigeben

Nach der Anweisung BE wird in den Baustein zurückgesprungen, in dem der Bausteinaufruf programmiert worden ist. Die Programmbearbeitung wird mit der ersten STEP5-Anweisung nach dem Bausteinaufruf fortgesetzt.

Die Bausteinende-Anweisung BE wird unabhängig vom Verknüpfungsergebnis bearbeitet. Nach BE kann das Verknüpfungsergebnis nicht mehr weiter verknüpft werden. Das unmittelbar vor Ausführung des BE-Befehls vorhandene Verknüpfungs-/Rechenergebnis wird jedoch an den aufrufenden Baustein übergeben und kann dort ausgewertet werden. Bei der Rückkehr aus dem aufgerufenen Baustein werden die Inhalte von Akku 1, Akku 2, Akku 3 und Akku 4, die Anzeigen ANZ0 und ANZ1 und das Verknüpfungsergebnis VKE nicht verändert.

### 2.2.3 Spezielle Organisationsbausteine

Die Schnittstelle zwischen dem Systemprogramm und dem Anwenderprogramm sind die Organisationsbausteine. Die Organisationsbausteine OB 1 bis 39 sind Teile des Anwenderprogramms, die Sie genauso wie Programm-, Funktions- oder Schrittbausteine programmieren. Durch Programmierung dieser OBs können Sie das Verhalten des Prozessors beim Anlauf, während der Programmbearbeitung und im Fehlerfall beeinflussen. Die Organisationsbausteine sind wirksam, sobald sie in den AG-Speicher geladen werden. Dies ist auch im laufenden Betrieb möglich.

Wesentlich ist, daß diese OBs vom Systemprogramm aufgerufen werden, als Reaktion auf bestimmte Ereignisse:

Organisationsbaustein	Funktion und Aufrufkriterium
OB 1	Organisation der zyklischen Programmbearbeitung Aufruf nach Ende einer Anlaufart
OB 2	Organisation der alarmgesteuerten Programmbearbeitung Aufruf durch Signal des S5-Bus (Prozeßalarm)
OB 10 - 18	Organisation der zeitgesteuerten Programmbearbeitung (Weckalarme)
OB 10	Aufruf alle 10 ms
OB 11	Aufruf alle 20 ms
OB 12	Aufruf alle 50 ms
OB 13	Aufruf alle 100 ms
OB 14	Aufruf alle 200 ms
OB 15	Aufruf alle 500 ms
OB 16	Aufruf alle 1 sec
OB 17	Aufruf alle 2 sec
OB 18	Aufruf alle 5 sec
OB 20 - 22	Organisation des Anlaufverhaltens
OB 20	Aufruf bei Anforderung "Neustart"
OB 21	Aufruf bei Anforderung "Manueller Wiederanlauf"
OB 22	Aufruf nach Netzwiederkehr ("Automat. Wiederanlauf")
OB 19, 23-34	Reaktion auf folgende Geräte- oder Programmfehler: <sup>1)</sup>
OB 19	Laufzeitfehler: Aufruf eines nicht geladenen Bausteins
OB 23	Quittungsverzug im Anwenderprogramm (bei Direktzugriff auf Peripheriebaugruppen oder andere S5-Busadressen)
OB 24	Quittungsverzug beim Aktualisieren des Prozeßabbildes und bei Koppelmerkerübertragung
OB 25	Adressierfehler
OB 26	Zykluszeitüberschreitung
OB 27	Befehlscodefehler: Substitutionsfehler

<sup>1)</sup> Ist im Fehlerfall der OB nicht programmiert, geht der Prozessor in den Stoppzustand über. AUSNAHME: Bei nicht vorhandenem OB 23 und 24 (Quittungsverzug) erfolgt keine Reaktion!



Organisations- baustein	Funktion und Aufrufkriterium
OB 28	Stop durch PG-Funktion/Stoppschalter/S5-Bus <sup>1)</sup>
OB 29	Befehlscodefehler: Operations-Code nicht zulässig
OB 30	Befehlscodefehler: Parameter nicht zulässig
OB 31	Sonstige Laufzeitfehler
OB 32	Laufzeitfehler: Transferfehler bei Datenbausteinen
OB 33	Weckfehler
OB 34	Fehler bei Reglerbearbeitung

- <sup>1)</sup> Der OB 28 wird vor Übergang in den Stoppzustand aufgerufen. Der Stoppzustand erfolgt immer, gleichgültig ob und wie der OB 28 programmiert ist.

Nachdem das Systemprogramm den betreffenden Organisationsbaustein aufgerufen hat, wird das darin enthaltene Anwenderprogramm bearbeitet. In der Regel wird danach an das durch den Fehler-Organisationsbaustein unterbrochene Programm zurückgesprungen (Ausnahme: OB 28). Zur Reaktion bei nicht vorhandenem Fehler-OB siehe Kapitel 5.4.

Zu Testzwecken können diese Organisationsbausteine auch vom Anwenderprogramm aufgerufen werden (SPA/SPB OBxxx). Es ist jedoch nicht möglich, z.B. durch Aufruf von OB 28 einen Übergang in den Stoppzustand herbeizuführen oder durch Aufruf von OB 22 einen automatischen Wiederanlauf auszulösen!

#### **WICHTIG!**

**Die speziellen Organisationsbausteine werden vom Anwender programmiert und vom Systemprogramm automatisch aufgerufen!**

## 2.2.4 Sonderfunktions-Organisationsbausteine

Die folgenden Organisationsbausteine enthalten Sonderfunktionen des Systemprogramms. Sie können vom Anwender nicht programmiert (dies gilt für alle OBs mit Nummern zwischen 40 und 255!), sondern lediglich aufgerufen werden. Sie enthalten kein STEP5-Programm. Sonderfunktions-OBs können in allen Code-Bausteinen aufgerufen werden.

### Übersicht 2-5: Sonderfunktions-Organisationsbausteine in der CPU 928

OB 110	Zugriff auf das Anzeigenbyte
OB 111	Akku 1, Akku 2, Akku 3 und Akku 4 löschen
OB 112	Akku Roll Up
OB 113	Akku Roll Down
OB 120	"Alarmer gemeinsam sperren" ein-/ausschalten
OB 121	"Weckalarmer einzeln sperren" ein-/ausschalten
OB 122	"Alarmer gemeinsam verzögern" ein-/ausschalten
OB 123	"Weckalarmer einzeln verzögern" ein-/ausschalten
OB 160-163	Zählschleife
OB 170	Bausteinstack (BSTACK) lesen
OB 180	Variabler Datenbaustein-Zugriff
OB 181	Datenbausteine testen
OB 190, 192	Merker in Datenbausteine übertragen
OB 191, 193	Datenblöcke in Merkerbereich übertragen
OB 200, 202-205	Mehrprozessor-Kommunikation
OB 216-218	Zugriffe auf Kacheln
OB 220	Konvertierung des Akku 1 von 16- auf 32-Bit-Festpunktzahl durch Vorzeichenerweiterung
OB 221	Neue Zykluszeit einstellen und triggern
OB 222	Zykluszeit nachtriggern
OB 223	Stopp bei uneinheitlicher Anlaufart im Mehrprozessorbetrieb
OB 224	Blockübertragung der Koppelmerker im Mehrprozessorbetrieb
OB 226	Inhalt einer Systemprogramm-Speicherzelle byteweise lesen
OB 227	Quersumme des Systemprogrammspeichers lesen
OB 228	Statusinformation einer Programmbearbeitungsebene lesen
OB 230-237	Funktionen für Standard-Funktionsbausteine
OB 240	Schieberegister initialisieren
OB 241	Schieberegister aufrufen
OB 242	Schieberegister löschen
OB 250	PID-Regler initialisieren
OB 251	PID-Regler bearbeiten
OB 254, 255	Datenbausteine ins DB-RAM übertragen

Die ausführliche Beschreibung dieser Sonderfunktionen finden Sie in Kapitel 6.

## 2.3 Funktionsbausteine

Funktionsbausteine (FB/FX) sind ebenso Teile des Anwenderprogramms wie z.B. Programmbausteine. FX-Funktionsbausteine haben den gleichen Aufbau wie FB-Funktionsbausteine und werden ebenso programmiert.

Mit Funktionsbausteinen werden häufig wiederkehrende oder sehr komplexe Funktionen realisiert.

Funktionsbausteine weisen gegenüber den Organisations-, Programm- und Schrittbausteinen vier wesentliche **Unterschiede** auf:

- Funktionsbausteine lassen sich **parametrieren**. D.h.: Die Formaloperanden eines Funktionsbausteins können bei jedem Aufruf durch andere Aktualoperanden ersetzt werden. Damit lassen sich für einen allgemeinen Anwendungsfall erstellte Funktionsbausteine vielseitig verwenden.
- Funktionsbausteine können Sie mit dem gesamten Operationsumfang der Programmiersprache STEP5 programmieren. Dazu gehören neben den Grundoperationen, die in allen Bausteinarten verwendet werden können, auch die **Ergänzenden Operationen** und die **Systemoperationen**.

### **WICHTIG!**

**Ergänzende Operationen sowie Systemoperationen können ausschließlich in Funktionsbausteinen programmiert werden.**

- Funktionsbausteine lassen sich **nur als Anweisungsliste (AWL)** programmieren und dokumentieren.

Der *Aufruf* der Funktionsbausteine ist jedoch auch in den Darstellungsarten FUP und KOP möglich und wird graphisch als Kasten dargestellt.

- Funktionsbausteine können Sie mit einem bis zu 8 Zeichen langen **Namen** versehen.

Jeder Funktionsbaustein stellt innerhalb des Anwenderprogramms eine komplexe, abgeschlossene Funktion dar. Funktionsbausteine können

- als Softwareprodukt von SIEMENS bezogen werden (Standard-Funktionsbausteine auf Mini-Diskette); mit diesen Standard-Funktionsbausteinen können Anwenderprogramme für Steuern, Melden, Regeln, Protokollieren schnell und sicher erstellt werden;

oder

- vom Anwender selbst programmiert werden.

2.3.1 Aufbau von Funktionsbausteinen

Der **Bausteinkopf** (5 Wörter) eines Funktionsbausteins ist gleich aufgebaut wie die Bausteinköpfe der übrigen STEP5-Bausteine.

Der **Bausteinrumpf** hingegen unterscheidet sich in seinem Aufbau grundsätzlich von dem der anderen Bausteinarten. Er enthält das eigentliche Programm des Funktionsbausteins. Die auszuführende Funktion ist darin in Form einer Anweisungsliste in der Programmiersprache STEP5 geschrieben. *Zwischen* dem Bausteinkopf und dem eigentlichen STEP5-Anwenderprogramm benötigt ein Funktionsbaustein weiteren Speicherplatz für die Angabe seines Namens und für die Liste der Formaloperanden. Da diese Liste keine Anweisungen für den Prozessor enthält, wird sie mit einem unbedingten Sprung übersprungen, den das PG automatisch erzeugt. Diese Sprunganweisung wird bei der Ausgabe am PG nicht angezeigt!

Operanden können in einem Funktionsbaustein absolut (z.B. M 2.5) oder symbolisch (z.B. -MOTOR1) eingegeben werden. Die Zuordnung der symbolischen Operanden müssen Sie zuvor in einer *Zuordnungsliste* ablegen.

Bei einem Aufruf des Funktionsbausteins wird nur der Bausteinrumpf bearbeitet.

So sieht ein Funktionsbaustein im AG-Speicher aus:

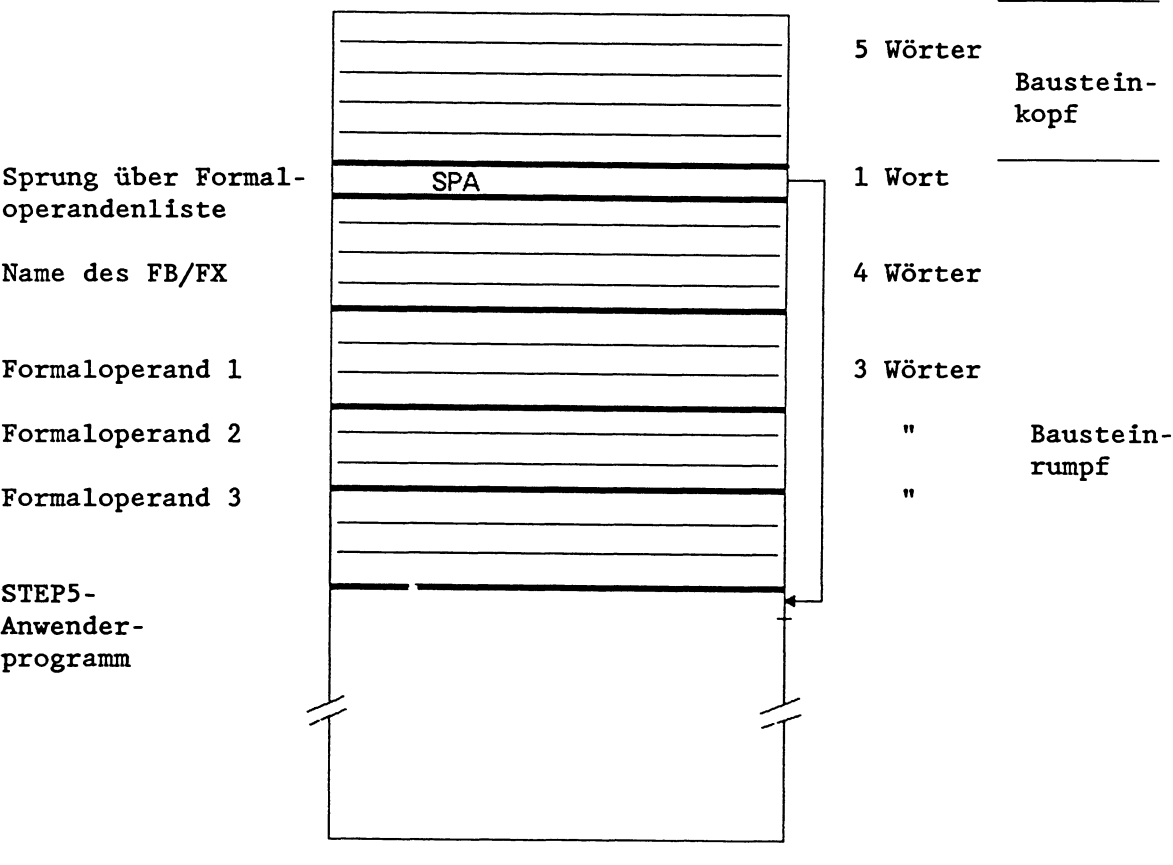


Abb. 2-6: Aufbau eines Funktionsbausteins (FB/FX)

Im Speicher stehen somit alle Angaben, die das Programmiergerät benötigt, um den Funktionsbaustein beim Aufruf graphisch darstellen zu können und um die Operanden bei der Parametrierung und Programmierung des Funktionsbausteins zu überprüfen. Eine fehlerhafte Eingabe wird vom Programmiergerät abgelehnt.

**WICHTIG!**

**Unterscheiden Sie bei der Behandlung von Funktionsbausteinen zwischen**

- a) FB/FX programmieren und**
- b) FB/FX aufrufen und anschließend parametrieren.**

Beim Programmieren legen Sie die Funktion des Bausteins fest. Die eingegebenen Operanden sind dabei Formaloperanden, die eine Platzhalterfunktion ausüben.

Beim Aufrufen eines Bausteins durch einen übergeordneten Baustein (OB, PB, SB, FB, FX) werden die Formaloperanden durch die Aktualoperanden ersetzt: der Funktionsbaustein wird parametriert.

Die folgenden Seiten werden Ihnen diesen Sachverhalt verdeutlichen.

### 2.3.2 Programmierung von Funktionsbausteinen

Bei der Eingabe eines Funktionsbausteins am PG gehen Sie so vor:

- Geben Sie die **Nummer** des Funktionsbausteins ein.

#### **WICHTIG!**

**Anwender-Funktionsbausteine sollten von FB 255 an abfallend nummeriert werden, um nicht mit den Standard-Funktionsbausteinen zu kollidieren, die von FB 1 bis FB 199 nummeriert sind.**

- Sie können als **Bibliotheksnummer** eine Nummer von 0 bis 99 999 eingeben. Dem Funktionsbaustein wird diese Nummer zugeordnet, unabhängig von seiner Bausteinnummer oder seinem Namen.

Sie sollten eine bestimmte Bibliotheksnummer nur einmal vergeben, um einen bestimmten Funktionsbaustein eindeutig identifizieren zu können.

- Geben Sie den **Namen** des Funktionsbausteins ein. Dieser kann bis zu 8 Zeichen lang sein.
- Geben Sie die **Formaloperanden** ein, die Sie im Baustein verwenden (maximal 40 Formaloperanden).

Für jeden Formaloperanden müssen Sie angeben:

1. den Namen des Bausteinparameters,
2. die Art des Bausteinparameters,
3. den Typ des Bausteinparameters.

Der **Name** kann maximal 4 Zeichen lang sein.

Für die **Art des Bausteinparameters** liefert Ihnen das Programmiergerät folgende Auswahl:

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

E, D, B, T oder Z sind Parameter, die bei der graphischen 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.

Für die Bausteinparameterarten E, A und D müssen Sie zusätzlich den **Typ des Parameters** angeben:

BI/BY/W/D                      für Parameterart E, A  
KM/KH/KY/KC/KF/KT/KZ/KG    für Parameterart D

Der Parametertyp gibt an, ob es sich bei E- und A-Parametern um Bit-, Byte-, Wort- oder Doppelwortgrößen handelt und welches Datenformat (z. B. Bitmuster oder Hexadezimalmuster) für D-Parameter gilt.

Art des Parameters	Typ des Paramaters	Zugelassene Aktualoperanden
E, A	BI für einen Operanden mit Bitadresse	E n.m Eingänge A n.m Ausgänge M n.m Merker
	BY für einen Operanden mit Byteadresse	EB n Eingangsbytes AB n Ausgangsbytes MB n Merkerbytes DL n Datenbyte links DR n Datenbyte rechts PY n Peripheriebytes QB n Peripheriebytes aus der erweiterten Peripherie
	W für einen Operanden mit Wortadresse	EW n Eingangswörter AW n Ausgangswörter MW n Merkerwörter DW n Datenwörter PW n Peripheriewörter QW n Peripheriewörter aus der erweiterten Peripherie
	D für einen Operanden mit Doppelwortadresse	ED n Eingangs-Doppelwörter AD n Ausgangs-Doppelwörter MD n Merker-Doppelwörter DD n Datum-Doppelwörter
D	KM für ein Binärmuster (16 Stellen)  KY für zwei byteweise Betragsszahlen im Bereich jeweils von 0 bis 255  KH für ein Hexadezimalmuster bis 4 Stellen  KC für 2 alphanumerische Zeichen  KT für einen Zeitwert (BCD-codiert) mit Zeitraster .0 bis .3 und Zeitwert 0 bis 999	Konstanten

Art des Parameters	Typ des Parameters	Zugelassene Aktualoperanden
D	KZ für einen Zählwert (BCD-codiert) 0 bis 999  KF für eine Festpunktzahl -32768 bis +32767  KG für eine Gleitpunktzahl	
B	Keine Typangabe zulässig	DB n Datenbausteine; ausgeführt wird der Befehl A DB n.  FB n Funktionsbausteine (nur ohne Parameter zulässig) werden unbedingt (SPA ..n) aufgerufen.  PB n Programmbausteine werden unbedingt (SPA ..n) aufgerufen.  SB n Schrittbausteine werden unbedingt (SPA ..n) aufgerufen.
T	Keine Typangabe zulässig	T 0 bis 255 Zeit <sup>1)</sup>
Z	Keine Typangabe zulässig	Z 0 bis 255 Zähler <sup>1)</sup>

<sup>1)</sup> Der Zeitwert bzw. Zählwert ist als Datum zu parametrieren oder als Konstante im Funktionsbaustein zu programmieren.

- Dann geben Sie Ihr STEP5-Programm in Form einer Anweisungsliste ein.

Die Formaloperanden werden dabei durch ein vorangestelltes Gleichheitszeichen gekennzeichnet (z.B. U =X1). Sie können auch mehrmals an verschiedenen Stellen im Funktionsbaustein angesprochen werden.

#### **WICHTIG!**

**Wenn Sie die Reihenfolge oder die Anzahl der Formaloperanden in der Formaloperandenliste ändern, müssen die Substitutionsbefehle im STEP5-Programm des Funktionsbausteins und die Bausteinparameterliste im aufrufenden Baustein entsprechend nachgeführt werden!**



**WICHTIG!**

**Programmieren und ändern Sie Funktionsbausteine grundsätzlich auf Diskette oder Winchester und übertragen Sie sie anschließend in das Automatisierungsgerät!**

- Schließen Sie die Programmeingabe mit 'BE' (Bausteinende) ab.

**Beispiel 2-7: Programmierung eines Funktionsbausteins**

FB 202

NAME: **BEISPIEL**

BEZ : <b>MONI</b>	E/A/D/B/T/Z: <b>E</b>	BI/BY/W/D: <b>BI</b>	Formal- operandenliste
BEZ : <b>BERT</b>	E/A/D/B/T/Z: <b>E</b>	BI/BY/W/D: <b>BI</b>	
BEZ : <b>HANS</b>	E/A/D/B/T/Z: <b>A</b>	BI/BY/W/D: <b>BI</b>	

:U **=MONI**  
:U **=BERT**  
:= **=HANS**

STEP5-Programm

Formal-  
operanden

Parameterart

Parametertyp

### 2.3.3 Aufruf und Parametrierung von Funktionsbausteinen

Jeder Funktionsbaustein kann beliebig oft und an beliebigen Stellen im STEP5-Anwenderprogramm aufgerufen werden. Während das STEP5-Programm immer als Anweisungsliste geschrieben wird, können die Aufrufe von Funktionsbausteinen auch in einer graphischen Darstellung erfolgen (FUP oder KOP).

So gehen Sie bei Aufruf und Parametrierung vor:

- Geben Sie im aufrufenden Baustein die Aufrufanweisung für den Funktionsbaustein ein.

Der Aufruf eines Funktionsbausteins kann innerhalb eines Organisations-, Programm- oder Schrittbusteins oder innerhalb eines anderen Funktionsbausteins programmiert werden.

Der Aufruf kann bedingt oder unbedingt erfolgen:

- Unbedingter Aufruf (SPA FBn für Funktionsbausteine oder BA FXn für erweiterte Funktionsbausteine):

Der angesprochene Funktionsbaustein wird unabhängig vom vorherigen Verknüpfungsergebnis bearbeitet.

- Bedingter Aufruf (SPB FBn für Funktionsbausteine oder BAB FXn für erweiterte Funktionsbausteine):

Der angesprochene Funktionsbaustein wird nur dann bearbeitet, wenn das vorherige Verknüpfungsergebnis  $VKE = 1$  ist. Bei  $VKE = 0$  wird die Sprunganweisung nicht ausgeführt, das VKE dabei jedoch auf 1 gesetzt.

Nach dem unbedingten und bedingten Aufruf kann das Verknüpfungsergebnis nicht weiter verknüpft werden. Es wird jedoch beim Sprung in den angesprochenen Funktionsbaustein mitgenommen und kann dort ausgewertet werden.

Nachdem Sie die Aufrufanweisung (z.B. SPA FB200) eingegeben haben, erscheint automatisch der Name und die Formaloperandenliste des betreffenden Funktionsbausteins:

- Sie ordnen nun jedem einzelnen Formaloperanden den für diesen Aufruf gültigen Aktualoperanden zu, d.h., Sie parametrieren den Funktionsbaustein.

Diese Aktualoperanden können bei den einzelnen Aufrufen unterschiedlich sein: Beim ersten Aufruf des FB200 z.B. Ein- und Ausgänge, beim zweiten Aufruf Merker.

Entsprechend der Formaloperandenliste können Sie bei jedem Aufruf eines Funktionsbausteins maximal 40 Aktualoperanden zuordnen.

### WICHTIG!

Bevor Sie einen Funktionsbaustein aufrufen und parametrieren, müssen Sie diesen Funktionsbaustein programmiert und auf die Programmdiskette überspielt bzw. direkt in den Programmspeicher des Automatisierungsgerätes eingegeben haben!

Nach dem Sprung in den Funktionsbaustein werden bei der Bearbeitung des Funktionsbaustein-Programms anstelle der Formaloperanden die Aktualoperanden aus dem aufrufenden Baustein verwendet.

Aufgrund dieser Eigenschaft der parametrierbaren Funktionsbausteine können Sie diese in Ihrem Anwenderprogramm vielseitig verwenden.

**Beispiel: Aufruf und Parametrierung eines Funktionsbausteins mit den Darstellungsarten AWL und KOP/FUP in einem Programmbaustein**

#### - Darstellungsart AWL

```
PB25
      :SPA FB201
NAME :E-ANTR
ZU-E :   DW1
RME  :   E 3.5
ESB  :   M 2.5
UEZ  :   T 2
ZEIT :   KT10.1
ZU-A :   DW1
BEA  :   A 2.3
LSL  :   A 6.0
```

Formal- operanden	Aktualoperanden

#### - Darstellungsart KOP/FUP

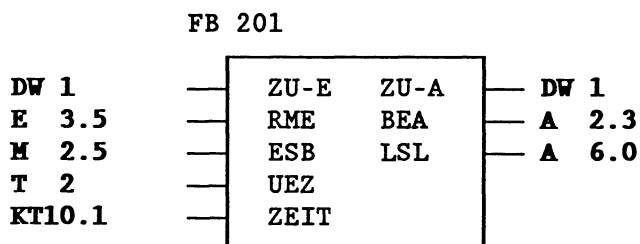


Abb. 2-8: Aufruf und Parametrierung eines Funktionsbausteins

Folgendes (vollständiges) **Beispiel** soll Ihnen noch einmal das Programmieren, Aufrufen und Parametrieren eines Funktionsbausteins verdeutlichen. Sie können es selbst leicht nachvollziehen.

Der Funktionsbaustein FB 202 wird programmiert:

FB 202

NAME : BEISPIEL

BEZ : MONI	E/A/D/B/T/Z : E	BI/BY/W/D : BI	Formal- operandenliste
BEZ : BERT	E/A/D/B/T/Z : E	BI/BY/W/D : BI	
BEZ : HANS	E/A/D/B/T/Z : A	BI/BY/W/D : BI	

: U = MONI  
: U = BERT  
: = HANS

STEP5-Programm

Formal-  
operanden

Parameterart

Parametertyp

Der Funktionsbaustein FB 202 wird im Programmbaustein PB 25 aufgerufen und parametrier:

- Darstellungsart AWL

PB25

: SPA FB 202  
NAME : BEISPIEL  
MONI : E 13.5  
BERT : M 17.7  
HANS : A 23.0

Formal-  
operanden

Aktualoperanden

- Darstellungsart KOP/FUP

FB 202

E 13.5	—	MONI	HANS	—	A 23.0
M 17.7	—	BERT			

Folgendes Programm wird nach Sprung in den FB 202 ausgeführt:

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

## 2.3.4 Spezielle Funktionsbausteine

### - Standardfunktionsbausteine

Neben den Funktionsbausteinen, die der Anwender selbst programmiert, gibt es die Standardfunktionsbausteine, die als fertiges Softwareprodukt zu beziehen sind. Sie enthalten allgemein verwendbare Standardfunktionen (z.B. Meldefunktionen, Ablaufsteuerungen etc.).

Standardfunktionsbausteine belegen die Nummern FB 1 bis FB 199.

Wenn Sie Standardfunktionsbausteine beziehen, beachten Sie die speziellen Hinweise in der dazugehörigen Beschreibung (belegte Bereiche, Konventionen etc.).

Die Standardfunktionsbausteine für die S5-135U, ihre Laufzeit, ihr Speicherbedarf und die von ihnen belegten Variablen sind im Katalog ST 57 "Software für Automatisierungsgeräte der U-Reihe" aufgeführt.

### Beispiel für einen Standardfunktionsbaustein

Gleitpunkttradizierer	RAD:GP	FB 6 für S5-115U
		FB 6 für S5-135U
		FB 19 für S5-150U

Der Funktionsbaustein RAD:GP radiziert eine Gleitpunktzahl (8-Bit-Exponent und 24-Bit-Mantisse), d.h., er bildet die Quadratwurzel. Das Ergebnis ist ebenfalls eine Gleitpunktzahl (8-Bit-Exponent und 24-Bit-Mantisse), wobei das niederwertigste Bit der Mantisse nicht gerundet wird.

Der Funktionsbaustein setzt für die weitere Verarbeitung gegebenenfalls die Kennung "Radikand negativ".

Zahlenbereich:

Radikand	-0,1469368 Exp. -38 bis +0,1701412 Exp. +39
Wurzel	+0,3833434 Exp. -19 bis +0,1304384 Exp. +20

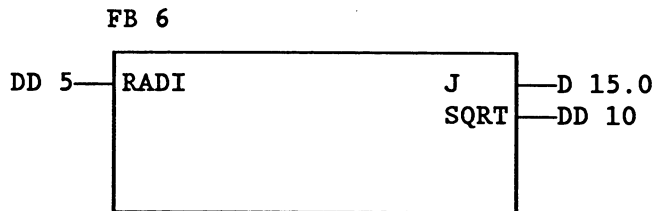
Funktion:  $Y = \sqrt{A}$   
Y = SQRT; A = RAD

### Aufruf des Funktionsbausteins FB 6:

- Darstellungsart AWL

```
      : A DB 17
      :
      :
      : SPA FB 6
NAME  : RAD : GP
RADI  : DD 5
J     : D 15.0
SQRT  : DD 10
```

- Darstellungsart KOP



DD = Datendoppelwort

Im obigen Beispiel wird eine Gleitpunktzahl, die im DD 5 mit 8-Bit-Exponent und 24-Bit-Mantisse bereitgestellt ist, radiziert. Das Ergebnis, wieder eine 32-Bit-Gleitpunktzahl, wird im DD 10 abgelegt. Vorher muß der entsprechende Datenbaustein aufgeschlagen werden. Der Parameter J (Parameterart: A, Parameter-typ: BI) gibt das Vorzeichen des Radikanden an: J = 1 bei negativem Radikanden. Belegte Merkerwörter: MW 238 bis 254.

### - Funktionsbaustein FB 0

Wenn der Organisationsbaustein OB 1 nicht programmiert ist, ruft das Systemprogramm anstelle des OB 1 zyklisch den FB 0 auf.

#### **WICHTIG!**

**Der FB 0 sollte deshalb nur zur Programmierung der zyklischen Programmbearbeitung verwendet werden! (Er darf keine Parameter enthalten.)**

Da Sie in einem Funktionsbaustein den gesamten Operationsvorrat der Programmiersprache STEP5 zur Verfügung haben, eignet sich die Programmierung des FB 0 - anstelle des OB 1 - besonders dann, wenn Sie ein kurzes und zeitkritisches Programm bearbeiten lassen wollen.

Sind sowohl OB 1 als auch FB 0 programmiert, wird nur der Organisationsbaustein OB 1 zyklisch bearbeitet.

## 2.4 Datenbausteine

In Datenbausteinen (DB/DX) sind die festen oder variablen Daten abgelegt, mit denen das Anwenderprogramm arbeitet. In Datenbausteinen werden keine STEP5-Operationen bearbeitet.

Die Daten eines Datenbausteins können sein:

- beliebige Bitmuster, z.B. für Anlagenzustände,
- Zahlen (hexadezimal, dual, dezimal) für Zeitwerte, Rechenergebnisse,
- alphanumerische Zeichen, z.B. für Meldetexte.

### 2.4.1 Aufbau eines Datenbausteins

Ein Datenbaustein besteht aus den Teilen

- Bausteinvorkopf (DV, DXV)
- Bausteinkopf
- Bausteinrumpf.

Der **Bausteinvorkopf** wird automatisch angelegt. Er enthält die Datenformate der im Bausteinrumpf eingegebenen Datenwörter. Als Anwender haben Sie keinen Einfluß auf das Anlegen des Bausteinvorkopfes.

#### WICHTIG!

Wenn Sie einen Datenbaustein vom AG oder vom EPROM-Modul auf Diskette übertragen, wird der dazugehörige Bausteinvorkopf gelöscht. Aus diesem Grund dürfen Sie einen Datenbaustein mit unterschiedlichen Datenformaten nie im AG ändern und ihn anschließend auf die Diskette zurückübertragen, sonst wird allen Datenwörtern dieses DBs automatisch das Datenformat zugeordnet, das Sie in der *Voreinstellungsmaske* gewählt haben.

Der **Bausteinkopf** belegt 5 Wörter im Speicher und enthält

- die Bausteinkennung
- die Kennung des Programmiergerätes
- die Bausteinnummer
- die Bibliotheksnummer
- die Bausteinlänge (inkl. Länge des Bausteinkopfes).

Der **Bausteinrumpf** enthält in aufsteigender Reihenfolge, beginnend mit Datenwort DW 0, die Datenwörter, mit denen das Anwenderprogramm arbeitet. Jedes Datenwort belegt im Speicher 1 Wort (16 Bits).

Ein Datenbaustein darf insgesamt bis zu 2000 Wörter im Speicher des Prozessors belegen. Berücksichtigen Sie beim Eingeben und Übertragen von Datenbausteinen mit dem PG die Speichergröße Ihres Programmiergerätes!

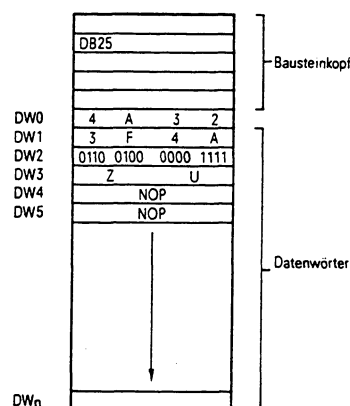


Abb. 2-9: Aufbau eines Datenbausteins

#### 2.4.2 Programmierung von Datenbausteinen

So erstellen Sie einen Datenbaustein:

- Geben Sie eine Datenbaustein-**Nummer** zwischen 3 und 255 (bei DB-Datenbausteinen) oder 1 und 255 (bei DX-Datenbausteinen) ein.

##### **WICHTIG!**

Die Datenbausteine DB 0, DB 1, DB 2 und DX 0 sind für bestimmte Funktionen reserviert und damit nicht frei verwendbar (siehe Kapitel 2.4.4)!

- Geben Sie die einzelnen **Datenwörter** im gewünschten Datenformat ein.

Zulässige Datenformate:

Beispiele:

KM = Bitmuster  
 KH = Hexadezimalzahl  
 KY = Byte  
 KF = Festpunktzahl  
 KG = Gleitpunktzahl  
 KC = Zeichen  
 KT = Zeitwert eines Zeitglieds  
 KZ = Zählerwert  
 ZL = Zuordnung in Zuordnungsliste  
 (nicht mit PG-Software S5DOS)

00100110 00111111  
 263F  
 38,63  
 +9791  
 +1356123+12  
 ?!ABCD123-+.,%  
 055.2  
 234  
 MOTOR1 = A 12.5

##### **WICHTIG!**

Die Eingabe von Datenwörtern wird nicht mit der Bausteinende-Anweisung 'BE' abgeschlossen!



### 2.4.3 Aufschlagen von Datenbausteinen

Ein Datenbaustein (DB/DX) kann nur unbedingt aufgeschlagen werden. Dies ist möglich innerhalb eines Organisations-, Programm-, Schritt- oder Funktionsbausteins. Ein bestimmter Datenbaustein kann mehrfach im Programm aufgeschlagen werden.

So schlagen Sie Datenbausteine auf:

- DB-Datenbausteine mit der Anweisung A DB..
- DX-Datenbausteine mit der Anweisung AX DX..

Der **Zugriff** auf die in dem aufgeschlagenen Datenbaustein gespeicherten Daten erfolgt bei der Programmbearbeitung durch die Lade- und Transferbefehle:

Mit einem **Ladebefehl** wird der Inhalt des adressierten Datenwortes in den Akku 1 übertragen und vom Prozessor verarbeitet.

Ladebefehle:            L DW.. (Wort)  
                         L DR.. (rechtes Byte)  
                         L DL.. (linkes Byte)  
                         L DD.. (Doppelwort)

Mit einem **Transferbefehl** werden Daten aus dem Akku 1 in das adressierte Datenwort übertragen.

Transferbefehle:        T DW..  
                         T DR..  
                         T DL..  
                         T DD..

Beim Laden wird der Inhalt eines Datenwortes nicht verändert.

Beim Transferieren wird der alte Inhalt eines Datenwortes überschrieben.

#### WICHTIG!

- Vor dem Zugriff auf ein Datenwort müssen Sie im Anwenderprogramm den gewünschten Datenbaustein aufschlagen, da nur so der Prozessor das richtige Datenwort findet! Das adressierte Datenwort muß im aufgeschlagenen Baustein enthalten sein, sonst erkennt das Systemprogramm bei dem Befehl T Dx einen Transferfehler oder lädt beim Befehl L Dx zufällige Werte.
- Mit Lade- und Transferbefehlen können Sie nur bis zu Datenwortnummer 255 zugreifen!

### Beispiel: Transferieren von Datenwörtern

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

Dazu geben Sie folgende Anweisungen ein:

A DB10        (DB 10 aufrufen)  
L DW1        (DW 1 in den Akku übertragen)  
A DB20        (DB 20 aufrufen)  
T DW1        (DW 1 aus Akku in DW 1 übertragen)

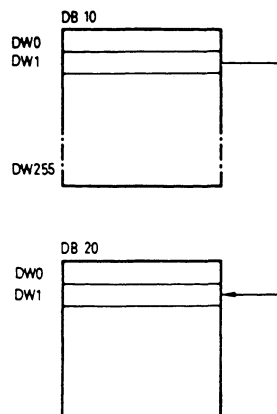


Abb. 2-10: Aufrufen von Datenbausteinen und Zugriff auf Datenwörter

Nach Aufrufen eines Datenbausteins beziehen sich alle folgenden Anweisungen mit dem Operandenbereich D auf den aufgerufenen Baustein.

Der aufgerufene Datenbaustein bleibt auch dann gültig, wenn durch eine Sprunganweisung (z.B. SPA/SPB PB 20) die Programmbearbeitung in einem anderen Baustein fortgesetzt wird.

Wenn in diesem Baustein nun ein anderer Datenbaustein aufgerufen wird, ist dieser nur im aufgerufenen Baustein (PB 20) gültig. Nach Rücksprung in den aufrufenden Baustein gilt wieder der alte Datenbaustein.

### WICHTIG!

Ein aufgerufener Datenbaustein bleibt also gültig bis

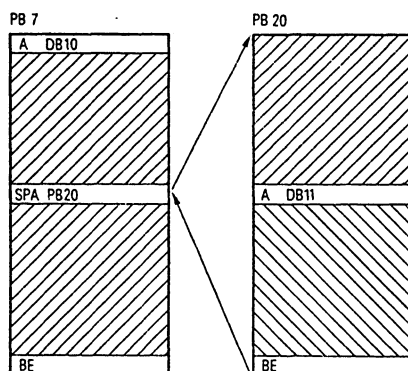
- oder        a) ein anderer Datenbaustein aufgerufen wird
- oder        b) ein Rücksprung in einen übergeordneten Baustein erfolgt
- oder        c) ein aufrufender Baustein mit 'BE' abgeschlossen wird.

### Beispiel: Gültigkeitsbereich bei Datenbausteinen

Im Programmbaustein PB 7 wird der Datenbaustein DB 10 aufgeschlagen (A DB10). In der folgenden Programmbearbeitung werden die Daten dieses Datenbausteins bearbeitet.

Nach dem Aufruf (SPA PB20) wird der Programmbaustein PB 20 bearbeitet. Der Datenbaustein DB 10 ist jedoch nach wie vor gültig. Erst mit dem Aufschlagen des Datenbausteins DB 11 (A DB11) wird der Datenbereich gewechselt. Bis zum Ende des Programmbausteins PB 20 (BE) ist nun der Datenbaustein DB 11 gültig.

Nach Sprung zurück in den Programmbaustein PB 7 ist wieder der Datenbaustein DB 10 gültig.



/// Gültigkeitsbereich des DB 10  
\\ Gültigkeitsbereich des DB 11

Abb. 2-11: Gültigkeitsbereich eines aufgerufenen Datenbausteins

#### 2.4.4 Spezielle Datenbausteine

Die Datenbausteine DB 0, DB 1, DB 2 und DX 0 sind für bestimmte Funktionen reserviert. Sie werden vom Systemprogramm verwaltet und sind für den Anwender nicht beliebig verwendbar.

##### - Datenbaustein DB 0 (siehe Kapitel 8.2.2)

Der Datenbaustein DB 0 enthält die Adreßliste mit den Anfangsadressen aller Bausteine, die sich im Anwenderspeicher oder im Datenbaustein-RAM des Prozessors befinden. Diese Adreßliste wird vom Systemprogramm bei der Initialisierung (nach jedem Netz-ein und nach Umräumen) erzeugt und bei der Eingabe oder Änderung von Bausteinen mit dem PG automatisch aktualisiert.

##### - Datenbaustein DB 1 (siehe Kapitel 10.3)

Der Datenbaustein DB 1 enthält die Liste der digitalen Ein- und Ausgänge (P-Peripherie mit relativen Byteadressen von 0 bis 127) sowie der Koppelmerkereingänge und -ausgänge, die dem Prozessor zugeordnet sind, und gegebenenfalls eine Zeitenblocklänge.

Bei Mehrprozessorbetrieb muß der Anwender den DB 1 für jeden beteiligten Prozessor erstellen. Im Einprozessorbetrieb wird der DB 1 eingesetzt, um die Zykluszeit zu verringern, da nur die im DB 1 angegebenen Ein- und Ausgänge, Koppelmerkerein- und -ausgänge oder Zeiten aktualisiert werden.

- **Datenbaustein DB 2** (siehe Kapitel 4.4.3)

Der Datenbaustein DB 2 dient zur Parametrierung der Reglerstruktur R64 durch den Anwender. Die Regelungsfunktion kann als Softwareprodukt bezogen werden und arbeitet systemprogrammunterstützt.

Beachten Sie zur Regelung die Beschreibung "Kompaktregelung im R-Prozessor des AG 135U", Bestell-Nr. C79000-B8500-C365-03.

- **Datenbaustein DX 0** (siehe Kapitel 7)

Durch die Programmierung des Datenbausteins DX 0 können Sie die Voreinstellungen bestimmter Systemprogrammfunktionen (z.B. bei der Bearbeitung des Anlaufs) ändern und damit die Leistungen des Systemprogramms Ihren Erfordernissen anpassen.

### **3 Programmbearbeitung**

#### **3.1 Übersicht**

Das STEP5-Anwenderprogramm kann auf verschiedene Art und Weise bearbeitet werden.

Normalerweise herrscht die zyklische Programmbearbeitung vor: Dabei wird der Organisationsbaustein OB 1 zyklisch durchlaufen und das dort verwaltete Anwenderprogramm von Anfang an über verschiedene Bausteinaufrufe hinweg durchgehend bearbeitet.

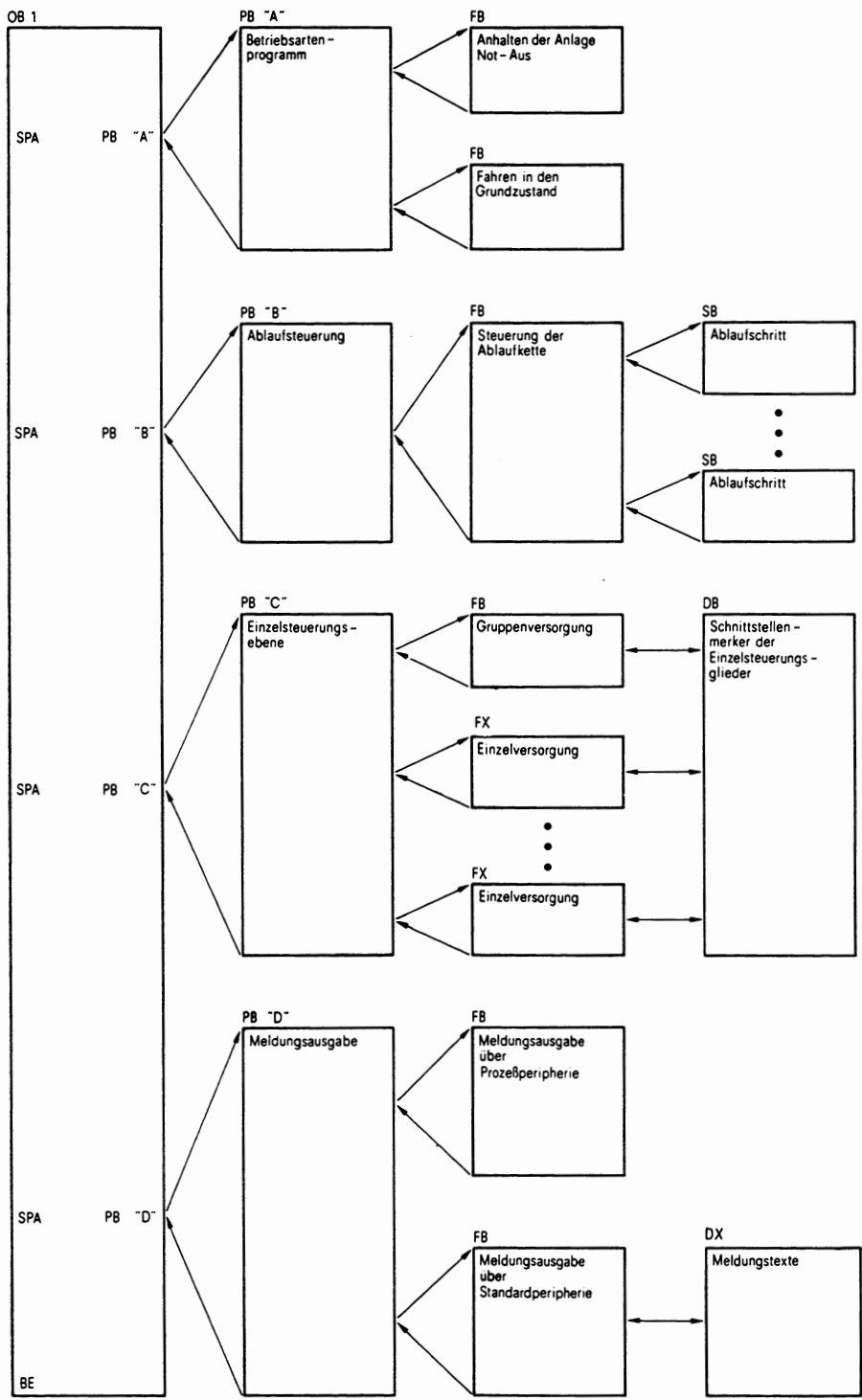
##### **3.1.1 Programmorganisation**

Mit der Programmorganisation legen Sie fest, ob und in welcher Reihenfolge die von Ihnen erstellten Bausteine bearbeitet werden sollen. Dazu programmieren Sie in den Organisationsbausteinen bedingte oder unbedingte Aufrufe der gewünschten Bausteine.

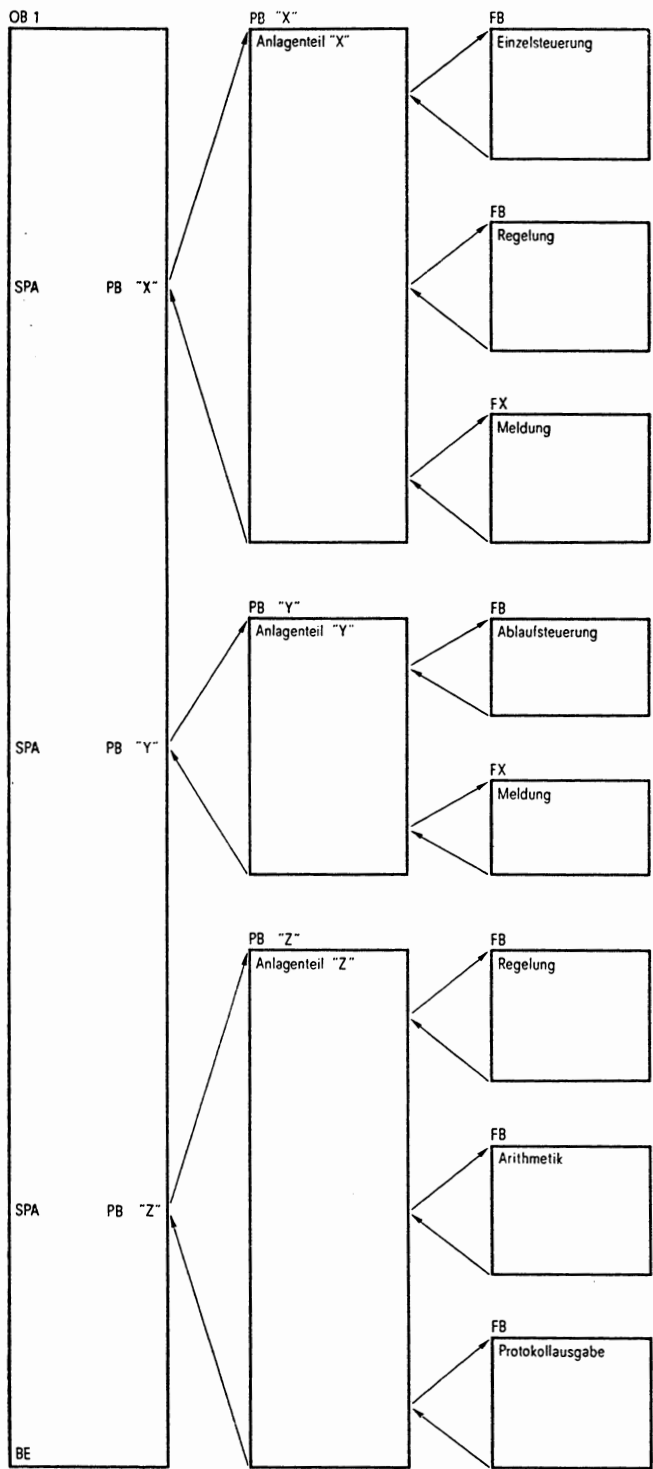
Im Programm der einzelnen Organisations-, Programm-, Funktions- und Schrittbausteine können weitere Programm-, Funktions- und Schrittbausteine in beliebiger Kombination (nacheinander oder ineinander verschachtelt) aufgerufen werden.

Das Anwenderprogramm sollte zweckmäßigerweise so organisiert sein, daß es die wesentlichen Programmstrukturen oder programmtechnisch zusammenhängende Anlagenteile hervorhebt.

Beispiel 3-1: Organisation des Anwenderprogramms nach Programmstruktur



**Beispiel 3-2: Organisation des Anwenderprogramms nach Anlagenstruktur**



### WICHTIG!

Sie können maximal 62 Bausteine ineinander schachteln. Werden mehr als 62 Bausteine aufgerufen, meldet der Prozessor einen Fehler.

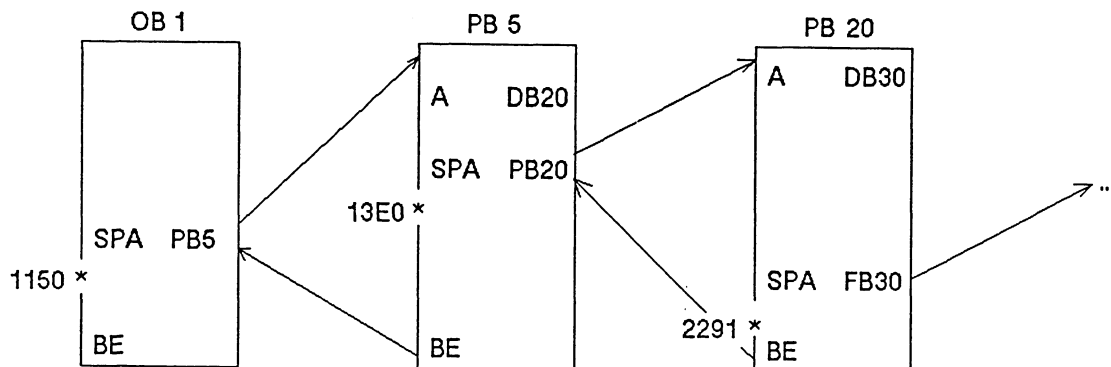
So ermitteln Sie die Schachtelungstiefe Ihres Programms (bezogen auf Beispiel, folgende Seite):

- Addieren Sie alle von Ihnen programmierten Organisationsbausteine (im Beispiel auf der folgenden Seite: 4 OBs).
- Addieren Sie die Schachtelungstiefen der einzelnen Organisationsbausteine (im Beispiel:  $2 + 2 + 1 + 0 = 5$ ).
- Beide Beträge zusammen ergeben die Programm-Schachtelungstiefe (im folgenden Beispiel:  $4 + 5 =$  Schachtelungstiefe 9).  
Sie darf den Wert 62 nicht überschreiten!

Die Lage eines Bausteins im Anwenderspeicher (oder DB-RAM) ist festgelegt durch seine **Baustein-Anfangsadresse**: Dies ist bei Code-Bausteinen die Adresse derjenigen Zelle im Speicher, in der sich der erste STEP5-Befehl des Bausteins befindet (bei FB und FX der SPA-Befehl über die Formaloperandenliste); bei Datenbausteinen die Adresse der Speicherzelle, in der sich das Datenwort DW 0 des Bausteins befindet.

Damit der Prozessor bei einem Bausteinaufruf (SPA/SPB xx, A DB) den aufgerufenen Baustein im Speicher findet, sind die Anfangsadressen aller programmierten Bausteine in der Bausteinadreßliste im Datenbaustein DB 0 eingetragen. Der DB 0 wird vom Systemprogramm verwaltet, als Anwender können Sie ihn nicht aufschlagen!

Um nach Abarbeitung des aufgerufenen Bausteins den Rückweg in den aufrufenden Baustein zu finden, speichert der Prozessor bei jedem Aufruf eines neuen Bausteins die **Rücksprungadresse**: Die Rücksprungadresse ist die Adresse derjenigen Zelle im Speicher, in der die dem Bausteinaufruf folgende STEP5-Anweisung steht. Außerdem speichert er die **Anfangsadresse und Länge des Datenbausteins**, der an dieser Stelle gültig ist.



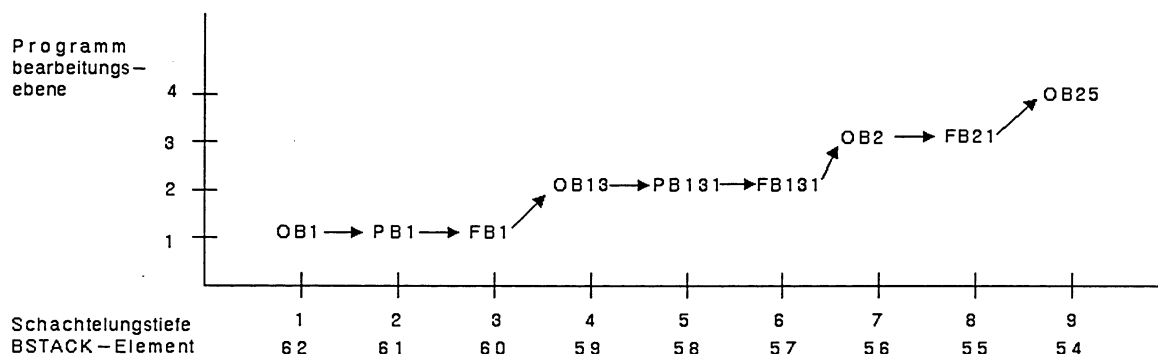
\* Rücksprungadressen



Diese Angaben werden in den **Bausteinstack (BSTACK)** eingetragen. Der BSTACK wird von unten gefüllt: Der erste Eintrag entspricht dem BSTACK-Element 62, der zweite Eintrag dem BSTACK-Element 61, usw.. Ist der aufgerufene Baustein vollständig bearbeitet und der Rücksprung in den aufrufenden Baustein erfolgt, werden die betreffenden Einträge wieder gelöscht.

Nach 62 Einträgen (BSTACK-Element 1) ist der Bausteinstack gefüllt. Bei Überschreiten der zulässigen Schachtelungstiefe geht der Prozessor in den Stoppzustand über.

**Beispiel: Bausteinschachtelungstiefe und Bausteinstack (BSTACK)**



### 3.1.2 Programmspeicherung

Damit der Prozessor das Anwenderprogramm bearbeiten kann, muß es in den Programmspeicher geladen werden. Es gibt zwei unterschiedliche Möglichkeiten:

- Wenn Sie ein steckbares **RAM-Modul** verwenden, können Sie Ihr Anwenderprogramm direkt vom Programmiergerät in den Prozessor übertragen.

Bei einem RAM-Modul läßt sich der Speicherinhalt schnell und häufig ändern. Eine Pufferbatterie verhindert, daß bei Netz-Aus das Anwenderprogramm im Speicher gelöscht wird (zur Pufferbatterie siehe Betriebsanleitung des Zentralgerätes S5-135U).

Alle programmierten Bausteine werden in beliebiger Reihenfolge im RAM-Modul gespeichert. Sobald sie einen Baustein ändern, ändert sich auch die Reihenfolge der Bausteine im Speicher.

Datenbausteine DB und DX werden im RAM-Modul abgelegt, bis dieses gefüllt ist, danach im Datenbaustein-RAM des Prozessors.

- b) Das gesamte Anwenderprogramm wird in einem steckbaren **EPROM-Modul** fest hinterlegt. In einem EPROM-Modul ist das Anwenderprogramm auch bei Netz-aus und fehlender Pufferbatterie voll geschützt.

Der Inhalt eines EPROM-Moduls kann nicht ohne weiteres verändert werden. Aus diesem Grund müssen Datenbausteine, die *variable* Daten enthalten und im Ablauf des Anwenderprogramms geändert werden sollen, im Neustart aus dem EPROM-Modul in das Datenbaustein-RAM des Prozessors kopiert werden (siehe Sonderfunktions-OBs 254 und 255, Kapitel 6.4.5).

Stellt der Prozessor beim Durchsuchen des Anwenderspeichers einen Fehler fest, so fordert er Umlöschen an und geht in Stop. Nach dem Umlöschen laden Sie Ihr Anwenderprogramm erneut in den Speicher.

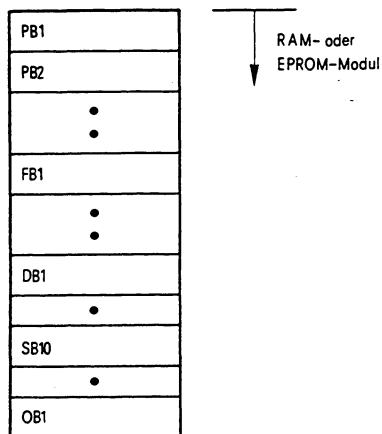


Abb. 3-4: Ablage der Bausteine in den Programmspeicher

### 3.1.3 Bearbeitung des STEP5-Anwenderprogramms

Das Anwenderprogramm kann auf verschiedene Art und Weise bearbeitet werden. Bei den speicherprogrammierbaren Steuerungen herrscht im allgemeinen die **zyklische Programmbearbeitung** vor.

Am Ende eines Anlaufs ruft das Systemprogramm automatisch den Organisationsbaustein OB 1 (bzw. den FB 0) auf. Der Prozessor beginnt dort mit der ersten STEP5-Anweisung des Anwenderprogramms und bearbeitet der Reihe nach alle Anweisungen des Anwenderprogramms. Ist er am Ende des Programms angelangt, beginnt er mit dem nächsten Zyklus wieder am Programmanfang.

In jedem Zyklus führt das Systemprogramm bestimmte Tätigkeiten aus:

- Es startet die Zykluszeit.
- Es aktualisiert das Prozeßabbild der Eingänge.
- Es aktualisiert die Eingangskoppelmerker.

- Es ruft die Anwenderschnittstelle auf.
- Es aktualisiert das Prozeßabbild der Ausgänge.
- Es aktualisiert die Ausgangskoppelmerker.

### **Zykluszeit**

Das Systemprogramm überwacht die Zeit, die der Prozessor zur Bearbeitung des Anwenderprogramms benötigt. Zu Beginn der Programmbearbeitung wird die zu überwachende Zykluszeit vom Systemprogramm gestartet.

Standardmäßig ist sie auf den maximal zulässigen Wert von 150ms eingestellt.

Als Anwender haben Sie die Möglichkeit, die Zykluszeit selbst einzustellen bzw. sie während der zyklischen Programmbearbeitung neu zu starten (siehe DX 0, Sonderfunktions-OBs 221 und 222).

Die *Gesamtzykluszeit* setzt sich zusammen aus der Laufzeit des Anwenderprogramms und der Laufzeit für den zyklischen Teil des Systemprogramms (siehe Bild, folgende Seite).

Die Laufzeit des Anwenderprogramms wiederum ergibt sich aus der Summe der Laufzeiten aller aufgerufenen Bausteine bei einem Programmdurchlauf (vom Aufruf des OB 1 bzw. FB 0 bis zu seinem Bearbeitungsende). Wenn Sie also einen bestimmten Baustein n-mal aufrufen, müssen Sie seine Laufzeit n-mal bei der Summenbildung berücksichtigen.

### **Prozeßabbild der Ein- und Ausgänge (PAE und PAA)**

Vor Beginn der STEP5-Programmbearbeitung werden die Signalzustände der Eingabe-Peripheriebaugruppen gelesen und ins Prozeßabbild der Eingänge (im Systemdatenspeicher des Prozessors) übertragen. Aus dem Prozeßabbild der Eingänge errechnet das STEP5-Anwenderprogramm das Prozeßabbild der Ausgänge. Nach Bearbeitung des STEP5-Programms werden die Signalzustände des Prozeßabbilds der Ausgänge zu den Ausgabe-Peripheriebaugruppen übertragen.

Das Prozeßabbild ist somit ein Speicherbereich, dessen Inhalt nur einmal pro Zyklus an die Peripherie ausgegeben bzw. von der Peripherie eingelesen wird.

### **WICHTIG!**

**Ein Prozeßabbild existiert nur für Ein- und Ausgabebytes der P-Peripherie mit Byteadressen von 0 bis 127!**

### **Koppelmerker**

Die Koppelmerker dienen zum Datenaustausch zwischen den einzelnen Prozessoren (im Mehrprozessorbetrieb) bzw. zwischen dem Prozessor und den Kommunikationsprozessoren.

Vor Beginn der STEP5-Programmbearbeitung werden die Eingangskoppelmerker des Prozessors eingelesen. Nach Bearbeitung des STEP5-Programms werden die Ausgangskoppelmerker zum Koordinator und zu den Kommunikationsprozessoren übertragen.

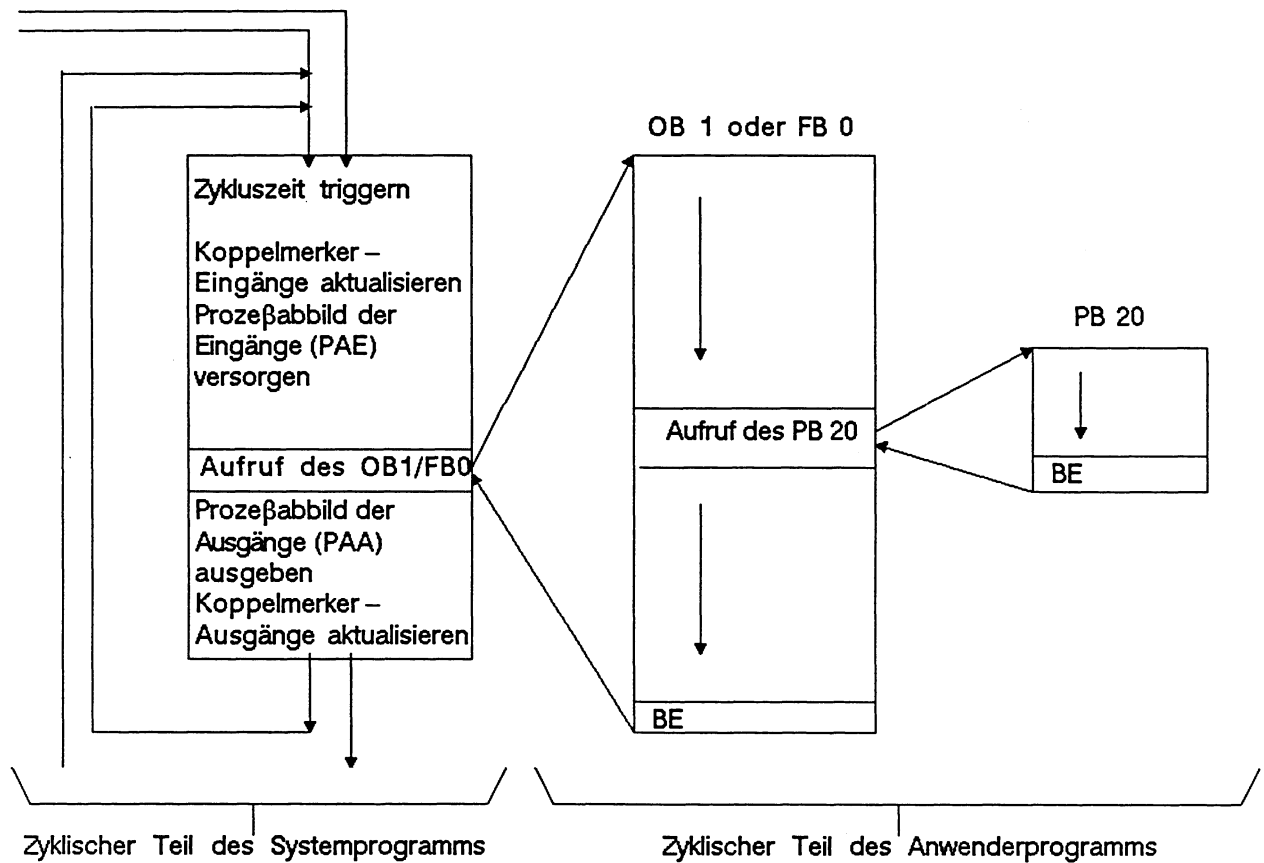


Abb. 3-5: Zyklische Programmbearbeitung

### Unterbrechungsstellen

Die zyklische Programmbearbeitung kann unterbrochen werden durch

- eine prozeßalarmgesteuerte Programmbearbeitung
- eine zeitgesteuerte Programmbearbeitung.

Sie kann unterbrochen bzw. ganz abgebrochen werden

- beim Auftreten eines Geräte- oder Programmfehlers
- durch Bedienung (PG-Funktion, Stoppschalter).

### 3.1.4 Festlegungen zur Programmbearbeitung

Als Anwender haben Sie zwei Möglichkeiten, das Verhalten des Prozessors beim Anlauf, während der zyklischen Programmbearbeitung und im Fehlerfall zu beeinflussen:

- a) durch die Programmierung der Organisationsbausteine OB 1 bis OB 34 (Schnittstellen zwischen System- und Anwenderprogramm, siehe Kapitel 2.2.3) und
- b) durch die Programmierung des Datenbausteins DX 0 (siehe Kapitel 7).

Die **Organisationsbausteine OB 1 bis OB 34** sind die Schnittstellen zwischen dem System- und dem Anwenderprogramm, da sie einerseits vom Systemprogramm aufgerufen werden und andererseits wie 'normale' Bausteine mit STEP5-Anwenderprogramm gefüllt werden können. In diesen Organisationsbausteinen können weitere Bausteine aufgerufen werden. Mit dem darin enthaltenen STEP5-Programm legt der Anwender die Reaktion des Prozessors auf bestimmte Ereignisse fest.

Die Organisationsbausteine OB 1 bis OB 34 sind wirksam, sobald sie in den Programmspeicher geladen werden (**auch im laufenden Betrieb**).

Werden sie vom Anwender nicht programmiert, erfolgt entweder keine Reaktion des Prozessors, oder - in den meisten Fehlerfällen - er geht in den Stoppzustand über (siehe hierzu Kapitel 5.4).

Eine andere Möglichkeit, das Verhalten des Prozessors zu beeinflussen, besteht in der Programmierung des **Datenbausteins DX 0**. Die Funktionen, die das Systemprogramm ausführt, sind standardmäßig vorgegeben. Durch die Angabe bestimmter Parameter im DX 0 können Sie die Standardvoreinstellungen einiger Systemprogramm-funktionen ändern.

Wie die Organisationsbausteine läßt sich der DX 0 im laufenden Betrieb in den Programmspeicher laden. **Er wird jedoch erst mit dem nächsten Neustart wirksam.**

Wird der DX 0 vom Anwender nicht programmiert, gelten die Voreinstellungen.

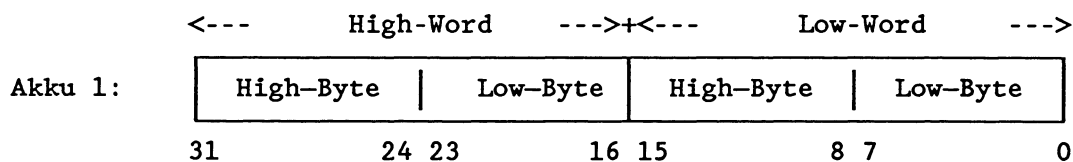
### 3.2 STEP5-Operationsvorrat mit Programmierbeispielen

Die STEP5-Operationen lassen sich in verschiedene Gruppen einteilen:

- Zu den binären Funktionen zählen die binären Verknüpfungsoperationen, die Speicheroperationen und die Zeit- und Zähloperationen.
- Zu den digitalen Funktionen zählen die Lade- und Transferoperationen, die Vergleichs- und die arithmetischen Operationen.
- Zu den organisatorischen Funktionen gehören die Sprungoperationen, Stopp- und Bausteinende-Operationen, Anweisungen zum Erzeugen oder Aufrufen eines Datenbausteins etc.

#### Die Akkumulatoren als Hilfsregister

Ein überwiegender Teil der STEP5-Operationen verwendet als Quelle für die Operanden und als Ziel für die Ergebnisse zwei Register (32 Bit): Akkumulator 1 (Akku 1) und Akkumulator 2 (Akku 2).



Die Akkumulatoren werden abhängig vom auszuführenden STEP5-Befehl beeinflusst.

#### Beispiele:

- Bei den Ladeoperationen wird als Ziel immer der Akku 1 verwendet. Der alte Inhalt von Akku 1 wird in den Akku 2 (Stack Lift) geschoben. Die Akkumulatoren 3 und 4 werden bei allen Ladeoperationen nicht verändert.
- Arithmetische Befehle verknüpfen den Inhalt von Akku 1 und Akku 2, schreiben das Ergebnis in den Akku 1 und übertragen den Inhalt des Akku 3 nach Akku 2 und den Inhalt des Akku 4 nach Akku 3 (Stack Drop).
- Beim Addieren einer Konstanten (ADD BF/KF) zum Inhalt des Akku 1 werden die Akkus 2, 3 und 4 hingegen nicht verändert.

## Ergebnisanzeigen

Es gibt Befehle für die Verarbeitung einzelner Bit-Informationen und Befehle für die Verarbeitung von Wort-Informationen (8, 16 oder 32 Bit).

In den beiden Gruppen gibt es anzeigensetzende Befehle und anzeigenauswertende Befehle (siehe Operationsliste, Anzeigenbeeinflussung). Entsprechend den beiden Befehlsgruppen gibt es Bit-Anzeigen (Bit 0 bis 3) und Wort-Anzeigen (Bit 4 bis 7). Das Anzeigenbyte kann am Programmiergerät ausgelesen werden und sieht folgendermaßen aus:

Wort-Anzeigen				Bit-Anzeigen			
ANZ1	ANZ0	OV	OS	OR	STA	VKE	ERAB
Bit 7	6	5	4	3	2	1	0

### Bit-Anzeigen:

#### ERAB Erstabfrage

Mit ihr beginnt eine logische Verknüpfung. Am Ende einer logischen Verknüpfungskette (Speicheroperationen) wird ERAB = 0 gesetzt. Befehle, die ERAB = 0 setzen (z.B. Ergebniszuzuweisung = A2.4), wirken VKE-begrenzend (siehe Operationsliste), d.h., das Verknüpfungsergebnis bleibt konstant. Es kann zwar ausgewertet (z.B. durch VKE-abhängige Befehle), jedoch nicht weiter verknüpft werden. Erst ab der nächsten Verknüpfungsanweisung (= Erstabfrage) wird das Verknüpfungsergebnis neu aufgebaut und ERAB = 1 gesetzt.

#### VKE Verknüpfungsergebnis

Ergebnis bit-breiter Verknüpfungen. Wahrheitsaussage bei den Vergleichsbefehlen (siehe Operationsliste, binäre Verknüpfungsoperationen bzw. Vergleichsoperationen).

#### STA Status

Gibt bei Bit-Befehlen den logischen Zustand des gerade abgefragten oder gesetzten Bits an. Der Status wird bei binären Verknüpfungsoperationen - ausgenommen U(, O(, ), 0 - und bei Speicheroperationen aktualisiert.

#### OR Oder

Sagt dem Prozessor, daß folgende UND-Verknüpfungen vor einer ODER-Verknüpfung (UND vor ODER) behandelt werden müssen.

### Wort-Anzeigen:

#### OV Overflow

Gibt an, ob bei der eben abgeschlossenen arithmetischen Operation der zulässige Zahlenbereich überschritten worden ist.

#### OS Overflow speichernd

Das Over-Bit ist gespeichert. Dient dazu, im Verlaufe mehrerer arithmetischer Operationen zu erkennen, ob irgendwann ein Fehler durch Überlauf (Over) aufgetreten ist.

#### ANZ1 und ANZ0

Codierte Ergebnisanzeigen, deren Interpretation aus der folgenden Tabelle ersichtlich wird:

Wortergebnisanzeigen		Festpunkt-rechnung, Ergebnis	Verknüpfungen, digital	Vergleich Inhalte von Akku 1 mit Akku 2	Schieben: letztes geschobenes Bit
ANZ1	ANZ0				
0	0	Erg. = 0	= 0	Akku 2 = Akku 1	0
0	1	Erg. < 0	-	Akku 2 < Akku 1	-
1	0	Erg. > 0	≠ 0	Akku 2 > Akku 1	1

Zur unmittelbaren Auswertung der Anzeigen stehen Sprungoperationen zur Verfügung (siehe Kapitel 3.2.2).



### 3.2.1 Grundoperationen

#### ● Binäre Verknüpfungsoperationen

Operation	Parameter	Funktion
)		Klammer zu
U (		UND-Verknüpfung von Klammersausdrücken
O (		ODER-Verknüpfung von Klammersausdrücken
O		ODER-Verknüpfung von UND-Funktionen
U		UND-Verknüpfung
O		ODER-Verknüpfung
E	0.0 bis 127.7	mit Abfrage eines Eingangs auf Signalzustand "1"
A	0.0 bis 127.7	mit Abfrage eines Ausgangs auf Signalzustand "1"
M	0.0 bis 255.7	mit Abfrage eines Merkers auf Signalzustand "1"
D	0.0 bis 255.15	mit Abfrage eines Datenwortes auf Signalzustand "1"
N E	0.0 bis 127.7	mit Abfrage eines Eingangs auf Signalzustand "0"
N A	0.0 bis 127.7	mit Abfrage eines Ausgangs auf Signalzustand "0"
N M	0.0 bis 255.7	mit Abfrage eines Merkers auf Signalzustand "0"
N D	0.0 bis 255.15	mit Abfrage eines Datenwortes auf Signalzustand "0"
T	0 bis 255	mit Abfrage einer Zeit auf Signalzustand "1"
N T	0 bis 255	mit Abfrage einer Zeit auf Signalzustand "0"
Z	0 bis 255	mit Abfrage eines Zählers auf Signalzustand "1"
N Z	0 bis 255	mit Abfrage eines Zählers auf Signalzustand "0"

Die binären Verknüpfungsoperationen erzeugen als Ergebnis das Verknüpfungsergebnis (VKE).

Am Anfang einer Verknüpfungskette hängt die Bildung des VKEs bei der ersten Verknüpfungsoperation (Erstabfrage) nur vom abgefragten Signalzustand (Status) ab und davon, ob er negiert wird (N = Negation), jedoch nicht von der Verknüpfungsart (O = ODER, U = UND).

Innerhalb einer Verknüpfungskette wird das VKE aus Verknüpfungsart, bisherigem VKE und dem abgefragten Signalzustand gebildet. Eine Verknüpfungskette wird durch einen VKE-begrenzenden (ERAB = 0) Befehl (z.B. Speicheroperationen) abgeschlossen.

Bis zur nächsten Erstabfrage bleibt das VKE unverändert. Es kann zwar ausgewertet, jedoch nicht weiter verknüpft werden.

### Beispiel:

Programm	Status	VKE	ERAB
:			
=A 0.0	0	0	0 ← VKE-begrenzt
U E1.0	1	1	1 ← Erstabfrage
U E1.1	1	1	1
U E1.2	0	0	1
=A 0.1	0	0	0 ← VKE-begrenzt, Ende der Verknüpfungs- kette

### ● Speicheroperationen

Operation	Parameter	Funktion
S		Setzen
R		Rücksetzen
=		Zuweisen
E	0.0 bis 127.7	eines Eingangs im PAE
A	0.0 bis 127.7	eines Ausgangs im PAA
M	0.0 bis 255.7	eines Merkers
D	0.0 bis 255.15	eines Datenwortbits

### ● Lade-, Transfer- und Vergleichsoperationen

Operation	Parameter	Funktion
L		Laden
T		Transferieren
E B	0 bis 127	eines Eingabebytes vom/zum PAE
E W	0 bis 126	eines Eingabewortes vom/zum PAE
E D	0 bis 124	eines Eingabe-Doppelwortes vom/zum PAE
A B	0 bis 127	eines Ausgabebytes vom/zum PAA
A W	0 bis 126	eines Ausgabewortes vom/zum PAA
A D	0 bis 124	eines Ausgabe-Doppelwortes vom/zum PAA
M B	0 bis 255	eines Merkerbytes
M W	0 bis 254	eines Merkerwortes
M D	0 bis 252	eines Merker-Doppelwortes
D R	0 bis 255	eines Datums (rechtes Byte) aus DB, DX
D L	0 bis 255	eines Datums (linkes Byte) aus DB, DX
D W	0 bis 255	eines Datenwortes
D D	0 bis 254	eines Daten-Doppelwortes
P B/	0 bis 127	eines Peripheriebytes der Digitaleingaben
P Y		bzw. -ausgaben (P-Bereich)
P B/	128 bis 255	eines Peripheriebytes der Analog- oder
P Y		Digitaleingaben bzw. -ausgaben (P-Bereich)
Q B	0 bis 255	eines Bytes der erweiterten Peripherie (Q-Bereich)

● Lade-, Transfer- und Vergleichsoperationen (Fortsetzung)

Operation	Parameter	Funktion
L T		Laden Transferieren
P W	0 bis 126	eines Peripheriewortes der Digital- eingaben bzw. -ausgaben (P-Bereich)
P W	128 bis 254	eines Peripheriewortes der Analog- oder Digitaleingaben bzw. -ausgaben (P-Bereich)
Q W	0 bis 254	eines Wortes der erweiterten Peripherie (Q-Bereich)
L		Laden
K M	16-Bit-Muster	einer Konstanten als Bitmuster
K H	0 bis FFFF	einer Konstanten im Hexadezimalcode
K F	-32 768 bis +32 767	einer Konstanten als Festpunktzahl
K Y	0 bis 255 für jedes Byte	einer Konstanten, 2 Bytes
K B	0 bis 255	einer Konstanten, 1 Byte
K C	2 alphanumer. Zeichen	einer Konstanten, 2 ASCII-Zeichen
K T	0.0 bis 999.3	eines Zeitwertes (Konstante)
K Z	0 bis 999	eines Zählwertes (Konstante)
K G	1)	einer Konstanten als Gleitpunktzahl (32 Bit)
T	0 bis 255	eines Zeitwertes
Z	0 bis 255	eines Zählwertes
LC T	0 bis 255	BCD-codiertes Laden eines Zeitwertes
LC Z	0 bis 255	BCD-codiertes Laden eines Zählwertes
! =		Vergleich auf gleich
> <		Vergleich auf ungleich
>		Vergleich auf größer
> =		Vergleich auf größer/gleich
<		Vergleich auf kleiner
< =		Vergleich auf kleiner/gleich
F		zweier Festpunktzahlen (16 Bit)
D		zweier Festpunktzahlen (32 Bit)
G		zweier Gleitpunktzahlen (32 Bit)

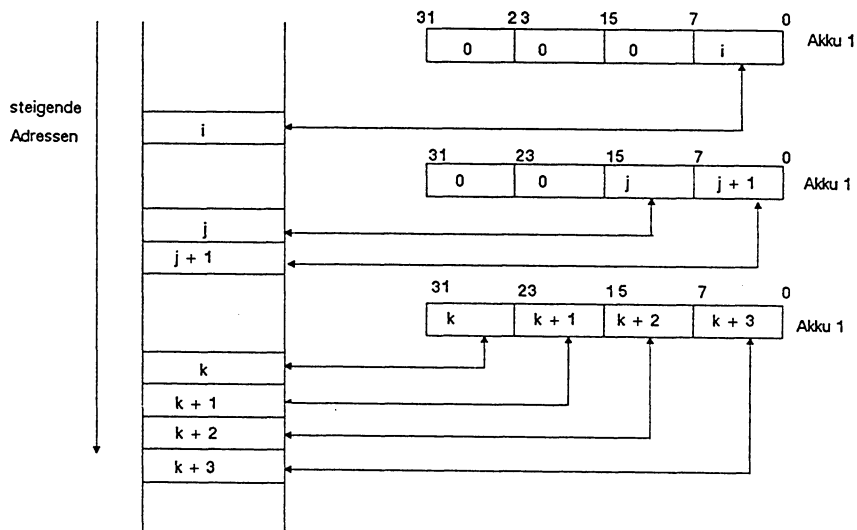
Ladebefehle schreiben den adressierten Wert in den Akku 1, dessen vorheriger Inhalt in den Akku 2 gerettet wird (Stack Lift).

Transferbefehle schreiben den Inhalt des Akku 1 in die adressierte Speicherzelle.

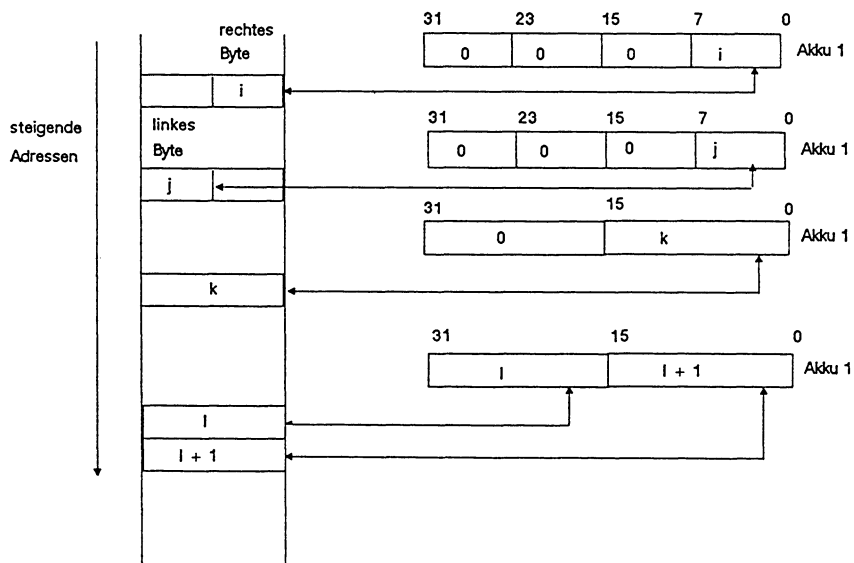
1)  $\pm 0.1469368 \times 10^{-38}$  bis  $\pm 0.1701412 \times 10^{39}$

Beispiel: Laden / Transferieren eines Bytes, Wortes oder Doppelwortes aus einem / in einen **byteweise** organisierten Speicherbereich (PAE, PAA, Merker, Peripherie).

:L EW 5      Geladen werden Byte 5 und 6 des PAE in den Akku 1.  
:L MD 10     Geladen werden die Merkerbytes 10 bis 13.



Beispiel: Laden / Transferieren eines Bytes, Wortes oder Doppelwortes aus einem / in einen **wortweise** organisierten Speicherbereich.



Worte oder Doppelworte werden innerhalb des Speichers, beginnend mit dem höchstwertigsten Byte oder Wort, in Richtung aufsteigender Adressen abgelegt.

Beim Laden eines Bytes oder Wortes werden die überzähligen Bits im Akku 1 gelöscht.

Die Ladeoperationen beeinflussen die Anzeigen nicht. Transferoperationen löschen das OS-Bit. Die Vergleichsbefehle erzeugen als Ergebnis das VKE und die Wortanzeigen ANZ1 und ANZ0. Verglichen werden immer die Inhalte der Akkus 1 und 2 (siehe Programmbeispiele und Operationsliste).

Die Peripherie kann durch Lade- und Transferoperationen angesprochen werden:

1. direkt:  
mit L/T PY, PW, QB, QW oder
2. über das Prozeßabbild:  
mit L/T EB, EW, ED, AB, AW, AD und mit Verknüpfungsoperationen.

Bei den Transferoperationen T PY 0 bis 127 und T PW 0 bis 126 wird parallel das Prozeßabbild der Ausgänge nachgeführt.

Das Prozeßabbild stellt einen Speicherbereich dar, dessen Inhalt nur *einmal pro Zyklus* an die Peripherie ausgegeben (Prozeßabbild der Ausgänge, PAA) bzw. von der Peripherie eingelesen (Prozeßabbild der Eingänge, PAE) wird. Damit wird vermieden, daß häufiges Ändern des logischen Zustandes eines Bits innerhalb eines Programmzyklus zum "Flattern" des zugehörigen Peripherieausgangs führt.

Beachten Sie zur Peripherie folgende Punkte:

- Ein Prozeßabbild der Ein- und Ausgänge existiert für je 128 Ein- und Ausgabebytes der P-Peripherie mit Byteadressen von 0 bis 127.
- Für den gesamten Bereich der Q-Peripherie und den Bereich der P-Peripherie mit relativen Byteadressen von 128 bis 255 existiert kein Prozeßabbild! (Zur Adreßraumaufteilung der Peripherie siehe Kapitel 8.1.2.)
- Ein-/Ausgabebaugruppen mit Adressen der Q-Peripherie dürfen nur in Erweiterungsgeräten stecken (nicht im Zentralgerät).
- In einem Erweiterungsgerät kann man entweder nur P-Peripherie oder nur Q-Peripherie verwenden.
- Falls in einem Erweiterungsgerät Relativadressen der P-Peripherie verwendet werden, sind diese Adressen für Peripheriebaugruppen im Zentralgerät nicht mehr zulässig (Doppeladressierung!).

## ● Zeit- und Zähloperationen

Um eine Zeit durch einen Startbefehl oder einen Zähler durch einen Setzbefehl zu laden, muß der Wert vorher in den Akku 1 geladen werden.

Zu bevorzugen sind folgende Ladeoperationen:

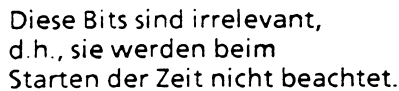
Für Zeiten: L KT, L EW, L AW, L MW, L DW.

Für Zähler: L KZ, L EW, L AW, L MW, L DW.

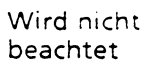
Operation		Parameter	Funktion
S I	T	0 bis 255	Starten einer Zeit als Impuls
S V	T	0 bis 255	Starten einer Zeit als verlängerter Impuls
S E	T	0 bis 255	Starten einer Zeit als Einschalt- verzögerung
S S	T	0 bis 255	Starten einer Zeit als speichernde Einschaltverzögerung
S A	T	0 bis 255	Starten einer Zeit als Ausschalt- verzögerung
R	T	0 bis 255	Rücksetzen einer Zeit
S	Z	0 bis 255	Setzen eines Zählers
R	Z	0 bis 255	Rücksetzen eines Zählers
Z V	Z	0 bis 255	Vorwärtszählen eines Zählers
Z R	Z	0 bis 255	Rückwärtszählen eines Zählers

Bei der Ausführung der Zeit- bzw. Zähloperationen SI, SE, SV, SS, SA und S wird der im Akku 1 stehende Wert in die Zeit- bzw. Zählzelle gebracht (entspricht dem Transferbefehl) und die entsprechende Operation veranlaßt.

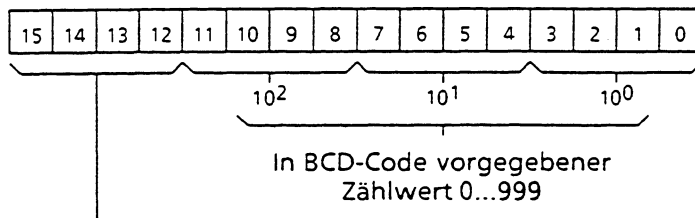
Für den Zeitwert



Belegung der Bits:

[illegible]

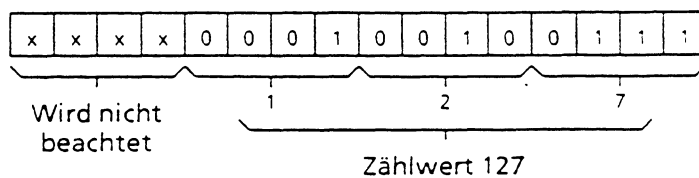
## Für den Zählwert



Diese Bits sind irrelevant,  
d.h., sie werden beim  
Setzen des Zählers nicht beachtet.

Beispiel: Es soll ein Zählwert von 127 vorgegeben werden.

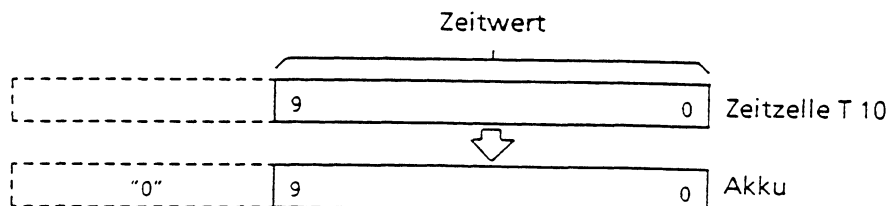
Belegung der Bits:



In der Zeit- bzw. Zählzelle selbst liegt der Zeit- bzw. Zählwert dualcodiert vor. Zur Abfrage der Zeit bzw. des Zählers kann der Wert der Zeit- bzw. Zählzelle direkt oder BCD-codiert in den Akku 1 geladen werden.

## Beispiele

Direktes Laden von **Zeitwerten**:

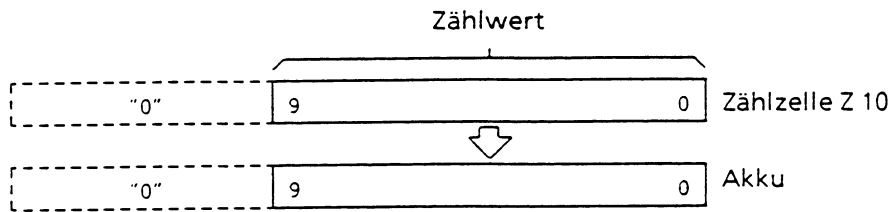


L T 10 Direktes Laden des dualen Zeitwertes der Zeit T 10  
in den Akku

Das Zeitraster wird nicht mitgeladen.

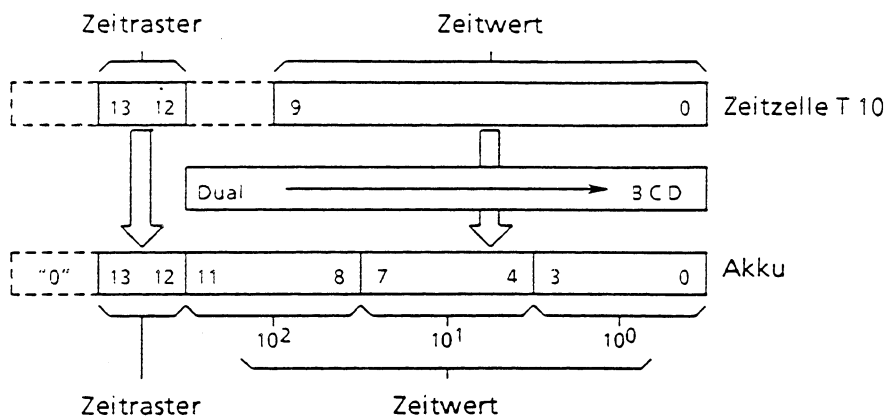


### Direktes Laden von Zählwerten:



**L Z 10** Direktes Laden des Zählwertes des Zählers Z 10 in den Akku

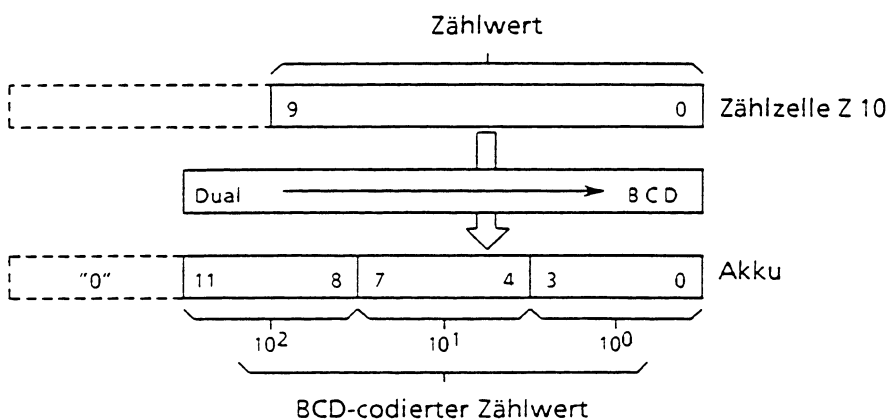
### Codiertes Laden von Zeitwerten:



**LC T 10** Codiertes Laden des Zeitwertes und des Zeitrasters der Zeit T 10 in den Akku

Das Zeitraster wird mitgeladen.

### Codiertes Laden von Zählwerten:



**LC Z 10** Codiertes Laden des Zählwerts des Zählers Z 10 in den Akku

Beim codierten Laden werden die Zustandsbits 14 und 15 der Zeitzellen bzw. 12 bis 15 der Zählzellen nicht geladen. An ihrer Stelle steht 0 im Akku 1. Der nun im Akku stehende Wert kann weiterverarbeitet werden.

### ● Arithmetische Operationen

Operation	Parameter	Funktion
+	F	Addition zweier Festpunktzahlen (16 Bit)
-	F	Subtraktion zweier Festpunktzahlen (16 Bit)
x	F	Multiplikation zweier Festpunktzahlen (16 Bit)
:	F	Division zweier Festpunktzahlen (16 Bit)
+	G	Addition zweier Gleitpunktzahlen (32 Bit)
-	G	Subtraktion zweier Gleitpunktzahlen (32 Bit)
x	G	Multiplikation zweier Gleitpunktzahlen (32 Bit)
:	G	Division zweier Gleitpunktzahlen (32 Bit)

Die arithmetischen Operationen verknüpfen den Inhalt von Akku 1 und Akku 2 (siehe Operationsliste). Das Ergebnis steht anschließend im Akku 1. Die Rechenregister werden durch eine arithmetische Operation wie folgt verändert:

vorher:    <Akku 1>   <Akku 2>   <Akku 3>   <Akku 4>  
               ↓            ↙            ↘  
 nachher:   <Ergebn.> <Akku 3>   <Akku 4>   <Akku 4>

Der alte Inhalt von Akku 2 geht verloren!

Beachten Sie, daß innerhalb der Ergänzenden Operationen Befehle zur Subtraktion und Addition von Doppelwort-Festpunktzahlen zur Verfügung stehen.

### ● Bausteinaufrufe

Operation	Parameter	Funktion
S P A		Sprung unbedingt
S P B		Sprung bedingt (nur wenn VKE = 1)
O B	1 bis 39	zu einem Organisationsbaustein
O B	40 bis 255	zu einer Systemprogramm-Sonderfunktion
P B	0 bis 255	zu einem Programmbaustein
F B	0 bis 255	zu einem FB-Funktionsbaustein
S B	0 bis 255	zu einem Schrittbaustein
B A F X	0 bis 255	Sprung unbedingt zu einem FX-Funktionsbaustein
B A B F X	0 bis 255	Sprung bedingt zu einem FX-Funktionsbaustein (nur wenn VKE = 1)
A D B	3 bis 255	Aufruf eines DB-Datenbausteins
A X D X	1 bis 255	Aufruf eines DX-Datenbausteins
B E		Bausteinende
B E B		Bausteinende bedingt (nur wenn VKE = 1)
B E A		Bausteinende absolut

### ● Nulloperationen

Operation	Parameter	Funktion
N O P	0	Nulloperation
N O P	1	Nulloperation
B L D	0 bis 255	Bildaufbauanweisung für das PG (wird vom Prozessor wie eine Nulloperation behandelt)

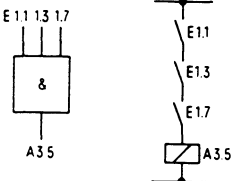
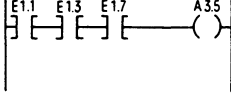
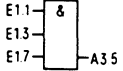
### ● Stop-Anweisung

Operation	Parameter	Funktion
S T P		Prozessor geht in Stop

**Programmierbeispiele für Verknüpfungs-, Speicher-, Zeit-, Zähl- und Vergleichsoperationen**

**• Verknüpfungsoperationen**

UND-Verknüpfung

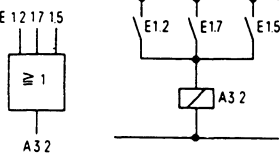
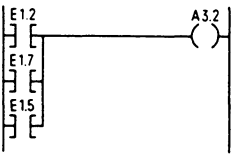
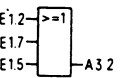
Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 1.1 U E 1.3 U E 1.7 = A 3.5		

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		
	Anweisungs- liste	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.

## ● Verknüpfungsoperationen (Fortsetzung)

### UND-vor-ODER-Verknüpfung

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 1.5 U E 1.6 O 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		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	O E 6.0 O E 6.1 U( 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.

## ● Verknüpfungsoperationen (Fortsetzung)

### ODER-vor-UND-Verknüpfung

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> U( O E 1.4 O E 1.5 ) U( O E 2.0 O E 2.1 ) = A 3.0           </pre>		

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.

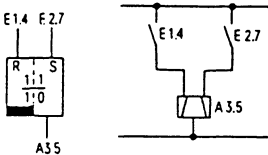
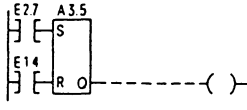
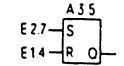
### Abfrage auf Signalzustand „0“

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> U E 1.5 UN E 1.6 = A 3.0           </pre>		

Am Ausgang A 3.0 erscheint Signalzustand „1“ nur dann, wenn der Eingang E 1.5 den Signalzustand „1“ (Schließer betätigt) und der Eingang E 1.6 den Signalzustand „0“ (Öffner nicht betätigt) führt.

## ● Speicheroperationen

### RS-Speicherglied für speichernde Signalausgabe

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 2.7 S A 3.5 U E 1.4 R A 3.5		

Signalzustand "1" am Eingang E 2.7 bewirkt das Setzen des Speicherglieds (Signalzustand "1" am Ausgang A 3.5). Wechselt der Signalzustand am Eingang E 2.7 nach "0", so bleibt dieser Zustand erhalten, d.h., das Signal wird gespeichert.

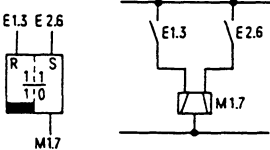
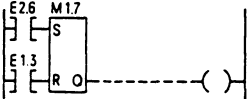
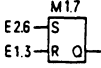
Signalzustand "1" am Eingang E 1.4 bewirkt das Rücksetzen des Speicherglieds (Signalzustand "0" am Ausgang A 3.5).

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 (Rücksetzen vorrangig).

## ● Speicheroperationen (Fortsetzung)

### RS-Speicherglied mit Merkern

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 2.6 S M 1.7 U E 1.3 R M 1.7		

Signalzustand „1“ am Eingang E 2.6 bewirkt das Setzen des Speichergliedes.

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 Speichergliedes.

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 1.3) während der Bearbeitung des übrigen Programms wirksam (Rücksetzen vorrangig).



## ● Speicheroperationen (Fortsetzung)

### Nachbildung eines Wischrelais

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> 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           </pre>		

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 ("Flankenmerker") und M 2.0 ("Impulsmerker") 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 rückgesetzt.

Der Merker M 2.0 führt also während eines einzigen Programmdurchlaufs Signalzustand „1“.

### Binäruntersetzer (T-Kippglied)

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> 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 U M 1.1 U A 3.0 = M 2.0 U M 1.1 UN A 3.0 UN M 2.0 S A 3.0 U M 2.0 R A 3.0           </pre>		

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.

## • Zeitoperationen

### Impuls

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> U  E 3.0 L  KT 10.2 SI T 1 U  T 1 =  A 4.0           </pre>		

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).

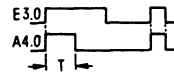
Die Abfragen U T bzw. O T liefern Signalzustand „1“, solange die Zeit läuft.

#### KT 10.2:

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



DU und DE sind digitale Ausgänge der Zeitzelle. Am Ausgang DU steht der Zeitwert dualcodiert, am Ausgang DE BCD-codiert mit Zeitraster an.

## ● Zeitoperationen (Fortsetzung)

### Verlängerter Impuls

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> U  E 3.1 L  EW15 SV T 2 U  T 2 =  A 4.1 </pre>		

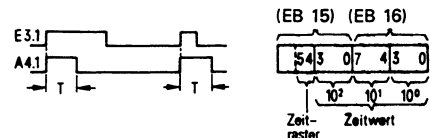
Bei Verknüpfungsergebnis „1“ und erstmaliger Bearbeitung wird das Zeitglied gestartet.

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)



### Einschaltverzögerung

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	<pre> U  E 3.5 L  KT9.2 SE T 3 U  T 3 =  A 4.2 </pre>		

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“, 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



## ● Zeitoperationen (Fortsetzung)

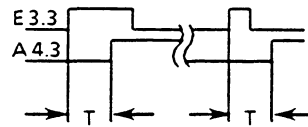
### Speichernde Einschaltverzögerung

Vorlage	STEP-5-Darstellung		
	Anwei- sungsliste	Kontaktplan	Funktionsplan
	U E 3.3 L KT 10.2 SS T 4 U E 3.2 R T 4 U T 4 = A 4.3		

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

Bei Verknüpfungsergebnis "0" bleibt das Zeitglied unbeeinflusst.

Die Abfragen UT bzw. OT liefern Signalzustand "1", wenn die Zeit abgelaufen ist. Der Signalzustand wird erst dann "0", wenn das Zeitglied mit der Funktion RT zurückgesetzt worden ist.



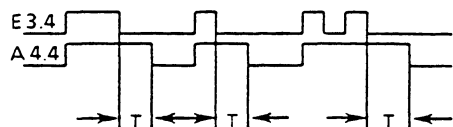
### Ausschaltverzögerung

Vorlage	STEP-5-Darstellung		
	Anwei- sungsliste	Kontaktplan	Funktionsplan
	U E 3.4 L KT 10.2 SA T 5 U T 5 = A 4.4		

Wenn das Verknüpfungsergebnis am Starteingang von "1" nach "0" wechselt, wird die Zeit gestartet. Sie läuft mit der programmierten Zeitdauer ab.

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

Die Abfragen UT bzw. OT liefern Signalzustand "1", wenn die Zeit läuft oder das Verknüpfungsergebnis am Eingang "1" ist.



## ● Zähloperationen

### Zähler setzen

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 4.1 L EW20 S Z 1		

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 der Zähler wieder gesetzt (Flankenauswertung).

Der für die Flankenauswertung des Setzeingangs erforderliche Merker ist im Zählwort mitgeführt.

DU und DE sind digitale Ausgänge der Zählerzelle. Am Ausgang DU steht der Zählwert dualcodiert, am Ausgang DE BCD-codiert an.

### Zähler zurücksetzen

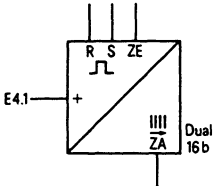
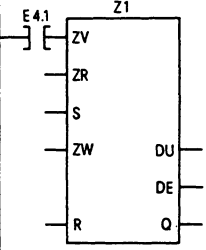
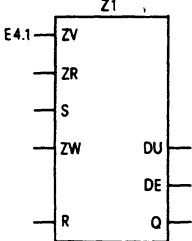
Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 4.2 R Z 1 U Z 1 = A 2.4		

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

Bei Verknüpfungsergebnis „0“ bleibt der Zähler unbeeinflusst.

## ● Zähloperationen (Fortsetzung)

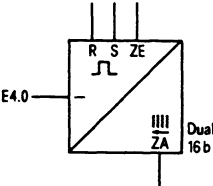
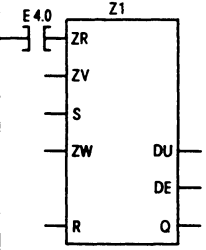
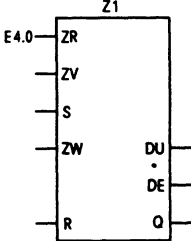
### Vorwärts zählen

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 4.1 ZV Z 1		

Der Wert des adressierten Zählers wird um 1 erhöht, maximal bis zum Zählwert 999. Die Funktion ZV wird nur bei einer positiven Flanke (von "0" nach "1") der vor ZV programmierten Verknüpfung ausgeführt. Die für die Flankenauswertung der Zählergänge 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.

### Rückwärts zählen

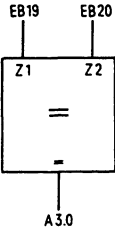
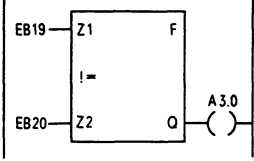
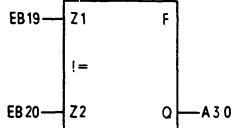
Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	U E 4.0 ZR Z 1		

Der Wert des adressierten Zählers wird um 1 erniedrigt, maximal bis zum Zählwert 0. Die Funktion wird nur bei einer positiven Flanke (von "0" nach "1") der vor ZR programmierten Verknüpfung wirksam. Die für die Flankenauswertung der Zählergänge 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.

## ● Vergleichsoperationen

### Vergleich auf gleich

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	L EB19 L EB20 ! = F = A 3.0		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleichsfunktion verglichen. Der Vergleich ergibt ein binäres Verknüpfungsergebnis.

VKE = „1“: Vergleich ist erfüllt, wenn  
Akku 1-L = Akku 2-L

VKE = „0“: Vergleich ist nicht erfüllt, wenn  
Akku 1-L ≠ Akku 2-L

Die Anzeigen ANZ 1 und ANZ 0 werden gemäß Tabelle auf Seite 3-12 gesetzt.

Akku 2-H und Akku 1-H bleiben beim 16-Bit-Festpunktvergleich an der Operation unbeteiligt.

Beim Festpunktvergleich (! = D) und Gleitpunktvergleich (! = G) werden die gesamten Inhalte von Akku 1 und Akku 2 (32 Bit) miteinander verglichen.

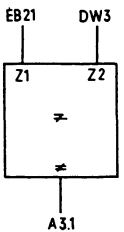
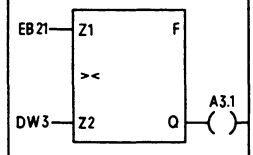
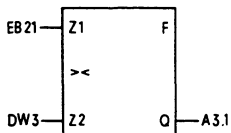
Beim Vergleich wird die Zahlendarstellung der Operanden berücksichtigt, d.h., der Inhalt von Akku 1-L und Akku 2-L wird hier als Festpunktzahl interpretiert.

O	EB 19	Akku 2-L
---	-------	----------

O	EB 20	Akku 1-L
---	-------	----------

● Vergleichsoperationen (Fortsetzung)

Vergleich auf ungleich

Vorlage	STEP-5-Darstellung		
	Anweisungs- liste	Kontaktplan	Funktionsplan
	L EB21 L DW3 >< F = A 3.1		

Der zuerst angegebene Operand wird mit dem nachfolgenden Operanden entsprechend der Vergleichsfunktion verglichen. Der Vergleich ergibt ein binäres Verknüpfungsergebnis.

VKE = „1“: Vergleich ist erfüllt, wenn  
Akku 1-L ≠ Akku 2-L  
VKE = „0“: Vergleich ist nicht erfüllt, wenn  
Akku 1-L = Akku 2-L

Die Anzeigen ANZ 1 und ANZ 0 werden gemäß Tabelle auf Seite 3-12 gesetzt.

Akku 2-H und Akku 1-H bleiben beim 16-Bit-Festpunktvergleich an der Operation unbeteiligt.

Beim Festpunktvergleich (32 Bit) und Gleitpunktvergleich sind auch Akku 2-H und Akku 1-H beim Vergleich beteiligt.

Entsprechendes gilt für die Vergleiche auf größer, größer gleich, kleiner und kleiner gleich (siehe Operationsliste).

Beim Vergleich wird die Zahlendarstellung der Operanden berücksichtigt, d.h., der Inhalt von Akku 1-L und Akku 2-L wird hier als Festpunktzahl interpretiert.

O	EB 21	Akku 2-L
---	-------	----------

DW 3	Akku 1-L
------	----------



### 3.2.2 Ergänzende Operationen

Der ergänzende Operationsvorrat kann nur in den Funktionsbausteinen (FB und FX) verwendet werden. Der Gesamtoperationsvorrat für Funktionsbausteine besteht daher aus den Grundoperationen und den ergänzenden Operationen.

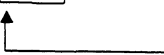
Zu den ergänzenden Funktionen gehören die Systemoperationen: Mit den Systemoperationen können Sie z.B. den Speicher an beliebiger Stelle überschreiben oder den Inhalt der Arbeitsregister des Prozessors verändern. *Deshalb sollten Sie (wenn überhaupt) die Systemoperationen nur mit äußerster Vorsicht anwenden.* Systemoperationen können nur programmiert werden, wenn sie in der Voreinstellungsmaske des Programmiergerätes freigegeben werden.

Beachten Sie zum Thema "Systemoperationen" das Kapitel 9 "Speicherzugriffe".

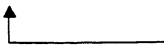
Bei den Funktionsbausteinen werden die Operationen nur in AWL dargestellt. Die Programme der Funktionsbausteine können also nicht in graphischer Form (KOP oder FUP) programmiert werden.

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

#### ● Binäre Verknüpfungen

Operation	Beschreibung
U = <input type="text"/>	UND Funktion, Abfrage eines Formaloperanden auf Signalzustand „1“.
UN = <input type="text"/>	UND-Funktion, Abfrage eines Formaloperanden auf Signalzustand „0“.
O = <input type="text"/>	ODER-Funktion, Abfrage eines Formaloperanden auf Signalzustand „1“.
ON = <input type="text"/>	ODER-Funktion, Abfrage eines Formaloperanden auf Signalzustand „0“.
	Formaloperand einsetzen.
	Als Aktualoperand sind binär adressierte Eingänge, Ausgänge, Daten und Merker (Parameterart: E, A; Parametertyp: BI) sowie Zeiten und Zähler (Parameterart: T, Z) zugelassen.

#### ● Speicheroperationen

Operation	Beschreibung
S = <input type="text"/>	Setzen (binär) eines Formaloperanden.
RB = <input type="text"/>	Rücksetzen (binär) eines Formaloperanden.
= = <input type="text"/>	Zuweisen des Verknüpfungsergebnisses an einen Formaloperanden.
	Formaloperand einsetzen.
	Als Aktualoperand sind binär adressierte Eingänge, Ausgänge, Daten und Merker zugelassen (Parameterart: E, A; Parametertyp: BI).

## ● Zeit- und Zähloperationen

Operation	Beschreibung
FR      T 0 bis 255	<p>Freigabe einer Zeit für Neustart Die Operation wird nur bei steigender Flanke des Verknüpfungsergebnisses ausgeführt. Sie bewirkt einen Neustart der Zeit, wenn bei der Startoperation Verknüpfungsergebnis „1“ anliegt.</p>
FR      Z 0 bis 255	<p>Freigabe eines Zählers Die Operation wird nur bei steigender Flanke des Verknüpfungsergebnisses ausgeführt. Sie bewirkt ein Setzen, Vorwärts- oder Rückwärtszählen des Zählers, wenn an der entsprechenden Operation Verknüpfungsergebnis „1“ anliegt.</p>
FR      = <input type="text"/>	<p>Freigabe eines Formaloperanden für Neustart (Beschreibung siehe F T bzw. F Z, je nach Formaloperand; Parameterart: T, Z).</p>
RD      = <input type="text"/>	<p>Rücksetzen (digital) eines Formaloperanden (Parameterart: T, Z).</p>
SI      = <input type="text"/>	<p>Starten einer als Formaloperand vorgegebenen Zeit mit dem im Akku hinterlegten Wert als Impuls (Parameterart: T)</p>
SE      = <input type="text"/>	<p>Starten einer als Formaloperand vorgegebenen Zeit mit dem im Akku hinterlegten Wert als Einschaltverzögerung (Parameterart: T).</p>
SVZ    = <input type="text"/>	<p>Starten einer als Formaloperand vorgegebenen Zeit mit dem im Akku hinterlegten Wert als verlängerter Impuls bzw. Setzen eines als Formaloperand vorgegebenen Zählers auf den im Akku 1 hinterlegten Zählwert (Parameterart: T, Z).</p>
SSV    = <input type="text"/>	<p>Starten einer als Formaloperand vorgegebenen Zeit mit dem im Akku hinterlegten Wert als speichernde Einschaltverzögerung bzw. Vorwärtszählen eines als Formaloperand vorgegebenen Zählers (Parameterart: T, Z).</p>
SAR    = <input type="text"/>	<p>Starten einer als Formaloperand vorgegebenen Zeit mit dem im Akku hinterlegten Wert als Ausschaltverzögerung bzw. Rückwärtszählen eines als Formaloperand vorgegebenen Zählers (Parameterart: T, Z).</p>
	<p>Formaloperand einsetzen.</p> <p>Als Aktualoperand sind Zeiten und Zähler zugelassen; Ausnahme: Bei SI und SE nur Zeiten.</p> <p>Der Zeit- bzw. Zählerwert kann wie bei den Grundoperationen oder als Formaloperand wie folgt vorgegeben werden:</p> <p>Setzen des Zeit- bzw. Zählwerts mit dem im BCD-Code vorliegenden Wert des als Formaloperanden vorgegebenen Operanden EW, AW, MW, DW (Parameterart: E; Parametertyp: W) bzw. als Konstante (Parameterart: D; Parametertyp: KT, KZ).</p>

## Beispiele

Funktionsbausteinanruf	Programm im Funktionsbaustein	ausgeführtes Programm
:SPA FB203 NAME :BEISPIEL ANNA : E 10.3 BERT : T 17 HANS : A 18.4	:U =ANNA :L KT 010.2 :SSV =BERT :U =BERT := =HANS	:U E 10.3 :L KT 010.2 :SS T 17 :U T 17 := A 18.4
:SPA FB204 NAME :BEISPIEL MAXI : E 10.5 IRMA : E 10.6 EVA : E 10.7 DORA : Z 15 EMMA : M 58.3	:U =MAXI :SSV =DORA :U =IRMA :SAR =DORA :U =EVA :L KZ100 :SVZ =DORA :UN =DORA := =EMMA	:U E 10.5 :ZV Z 15 :U E 10.6 :ZR Z 15 :U E 10.7 :L KZ 100 :S Z 15 :UN Z 15 := M 58.3
:SPA FB205 NAME :BEISPIEL KURT : E 10.4 CARL : T 18 EGON : EW20 MAUS : M 100.7	:U =KURT :L =EGON :SVZ =CARL :U =CARL := =MAUS	:U E 10.4 :L EW20 :SV T 18 :U T 18 := M 100.7

## • Lade- und Transferoperationen

Operation	Beschreibung
L = _____	Laden eines Formaloperanden Der Wert des als Formaloperanden vorgegebenen Operanden wird in den Akku geladen (Parameterart: E, T, Z, A; Parametertyp: BY, W, D).
LC = _____	Codiertes Laden eines Formaloperanden Der Wert der als Formaloperand vorgegebenen Zeit- oder Zählzelle wird BCD-codiert in den Akku geladen (Parameter: T, Z).
LW = _____	Laden des Bitmusters eines Formaloperanden Das Bitmuster des Formaloperanden wird in den Akku geladen (Parameterart: D; Parametertyp: KF, KH, KM, KY, KC, KT, KZ).
LD = _____	Laden des Bitmusters eines Formaloperanden Das Bitmuster des Formaloperanden wird in den Akku geladen (Parameterart: D; Parametertyp: KG).
T = _____	Transferieren zu einem Formaloperanden Der Akkumulatorinhalt wird zu dem als Formaloperand vorgegebenen Operanden transferiert (Parameterart: E, A; Parametertyp: BY, W, D).
Formaloperanden einsetzen	

Als Aktualoperand sind die den Grundoperationen entsprechenden Operanden zugelassen. Bei LW sind ein Datum in Form eines Binär- (KM) oder eines Hexadezimalmusters (KH), 2 byteweise Betragsszahlen (KY), Zeichen (KC), Festpunktzahl (KF), Zeitwerte (KT) und Zählwerte (KZ) zugelassen. Bei LD ist eine Gleitpunktzahl als Datum zugelassen.

Operation	Parameter	Beschreibung
L BA	0 bis 255	Laden eines Wortes in den Akku 1 aus dem Bereich "Anschaltungsdaten" (BA-Bereich)
L BB	0 bis 255	Laden eines Wortes in den Akku 1 aus dem Bereich "Anschaltungsdaten" (BB-Bereich)
L BS	0 bis 255	Laden eines Wortes in den Akku 1 aus dem Bereich "Systemdaten" (BS-Bereich) (frei: BS 60 bis 63)
L BT	0 bis 255	Laden eines Wortes in den Akku 1 aus dem Bereich "Systemdaten" (BT-Bereich)
T BA	0 bis 255	Transferieren des Akku 1 zu einem Wort des Bereichs "Anschaltung" (BA-Bereich)
T BB	0 bis 255	Transferieren des Akku 1 zu einem Wort des Bereichs "Anschaltung" (BB-Bereich)
T BS <sup>1)</sup>	0 bis 255	Transferieren des Akkus 1 zu einem Wort des Bereichs "System" (BS-Bereich) (frei: BS 60 bis 63)
T BT <sup>1)</sup>	0 bis 255	Transferieren des Akkus 1 zu einem Wort des Bereichs "System" (BT-Bereich)

<sup>1)</sup> Systemoperation

Im Gegensatz zu den Bereichen BA, BB und BT dürfen vom BS-Bereich nur die Wörter BS 60 bis BS 63 für Anwenderzwecke frei genutzt werden.

Beachten Sie dazu das Kapitel 8.2.4 "BS-/BT-Bereich".

● Rechenoperationen

Operation	Beschreibung
ENT	Eintrag von Daten in die Akkus 3 und 4, die bei arithmetischen Operationen mitverwendet werden: Die Inhalte von Akku 2 und 3 werden in die Akkus 3 und 4 geladen.

Es findet ein Stack Lift in die Akkus 3 und 4 statt:

<Akku 4> := <Akku 3>  
<Akku 3> := <Akku 2>  
<Akku 2> := <Akku 2>  
<Akku 1> := <Akku 1>

Die Akkus 1 und 2 werden nicht verändert. Der alte Inhalt des Akku 4 geht verloren.

Beispiel

Folgender Bruch soll ausgerechnet werden:  $(30 + 3 \times 4)/6 = 7$

	Akku 1	Akku 2	Akku 3	Akku 4
Vorbelegung der Akkus vor der arithmetischen Operationskette	a	b	c	d
L KF 30	30	a	c	d
L KF 3	3	30	c	d
ENT	3	30	30	c
L KF 4	4	3	30	c
* F	12	30	c	c
+ F	42	c	c	c
L KF 6	6	42	c	c
/ F	7	c	c	c

Operation	Parameter	Beschreibung
ADD BF <sup>1)</sup>	-128 bis +127	Addiere Byte-Konstante (Festpunkt) zum Akku 1
ADD KF <sup>1)</sup>	-32 768 bis +32 767	Addiere Festpunktkonstante (Wort) zum Akku 1

<sup>1)</sup> Systemoperation

Operation	Parameter	Beschreibung
ADD DF <sup>1)2)</sup>	-2147483648 bis +2147483647	Addiere Doppelwort-Festpunkt-konstante zum Akku 1
+D <sup>1)2)</sup>		Addiere zwei Doppelwort-Festpunkt-konstanten (Akku 1 + Akku 2)
-D <sup>1)2)</sup>		Subtrahiere zwei Doppelwort-Festpunkt-konstanten (Akku 1 - Akku 2)
TAK <sup>1)</sup>		Tausche den Inhalt von Akku 1 und Akku 2

<sup>1)</sup> Systemoperation

<sup>2)</sup> Die Programmierung ist abhängig vom PG-Typ und vom Ausgabestand der PG-Systemsoftware möglich.

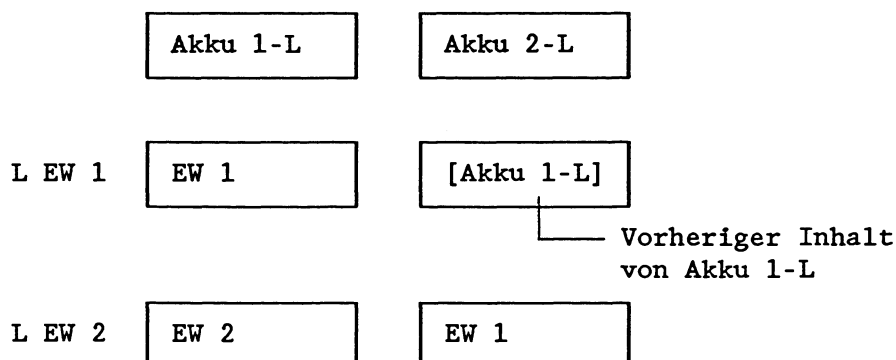
### ● Digitalverknüpfungen

Operation	Beschreibung
UW	UND-Verknüpfung von Akku 1-L und Akku 2-L
OW	ODER-Verknüpfung von Akku 1-L und Akku 2-L
XOW	Exklusiv-ODER-Verknüpfung von Akku 1-L und Akku 2-L

Die Akkus 3 und 4 werden nicht beeinflusst, jedoch die Anzeigen ANZ1 und ANZ 0 (siehe Wortergebnisanzeigen).

Durch zwei Ladeoperationen können Akku 1 und Akku 2 entsprechend den Operanden der Ladeoperation geladen werden. Anschließend lassen sich die Inhalte beider Akkus digital verknüpfen.

#### Beispiel:



UND-Verknüpfung von EW 2 und EW 1:

UW	Ergebnis	EW 1
----	----------	------

## Organisatorische Funktionen

### ● Sprungoperationen

Das Sprungziel für unbedingte und bedingte Sprünge wird symbolisch angegeben (maximal 4 Zeichen, beginnend mit einem Buchstaben). 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  $\pm 127$  Wörter umfaßt und eine STEP5-Anweisung aus mehr als einem Wort bestehen kann. Sprünge dürfen nur innerhalb eines Bausteins durchgeführt werden; Sprünge über Netzwerke hinweg sind unzulässig.

### **WICHTIG!**

**Sprunganweisung und Sprungziel müssen in einem Netzwerk liegen. Pro Netzwerk ist nur eine Symboladresse für Sprungziele zugelassen.**

**Ausnahme:** Dies gilt nicht für den Sprung SPR, bei dem als Parameter eine absolute Sprungdistanz angegeben wird.

Operation	Beschreibung
SPA = adr	Sprung unbedingt Der unbedingte Sprung wird unabhängig von Bedingungen ausgeführt.
SPB = adr	Sprung bedingt Der bedingte Sprung wird ausgeführt, wenn VKE = 1 ist. Bei VKE = 0 wird die Anweisung nicht ausgeführt und das Verknüpfungsergebnis auf VKE = 1 gesetzt.
SPZ = adr	Sprungbedingung: ANZ1, ANZ0 Der Sprung wird nur dann ausgeführt, wenn ANZ1 = 0 und ANZ0 = 0 ist. Das Verknüpfungsergebnis wird nicht verändert.
SPN = adr	Sprungbedingung: ANZ1, ANZ0 Der Sprung wird nur dann ausgeführt, wenn ANZ1 $\neq$ ANZ0 ist. Das Verknüpfungsergebnis wird nicht verändert.
SPP = adr	Sprungbedingung: ANZ1, ANZ0 Der Sprung wird nur dann ausgeführt, wenn ANZ1 = 1 und ANZ0 = 0 ist. Das Verknüpfungsergebnis wird nicht verändert.
SPM = adr	Der Sprung wird nur dann ausgeführt, wenn ANZ1 = 0 und ANZ0 = 1 ist. Das Verknüpfungsergebnis wird nicht verändert.

adr = Symboladresse (maximal 4 Zeichen)

Operation	Beschreibung
SPO = adr	Sprung bei Überlauf (Overflow) Der Sprung wird ausgeführt, wenn die Anzeige OV = 1 ist. Wenn kein Überlauf vorliegt (OV = 0), wird der Sprung nicht ausgeführt. Das Verknüpfungsergebnis wird nicht verändert. Ein Überlauf entsteht, wenn bei gegebener Zahlendarstellung der zulässige Bereich durch eine arithmetische Operation überschritten wird.
SPS = adr	Sprung, wenn die Anzeige OS (Overflow speichernd) gesetzt ist (OS = 1).
SPR <sup>1)</sup> -32 768 bis +32 767	Beliebiger Sprung innerhalb eines Funktionsbausteins; wird immer ausgeführt, unabhängig von Bedingungen

adr = Symboladresse (maximal 4 Zeichen)

<sup>1)</sup> Systemoperation

#### ● Schiebeoperationen

Operation	Beschreibung
SLW 0 bis 15	Schieben nach links (von rechts werden Nullen nachgezogen)
SRW 0 bis 15	Schieben nach rechts (von links werden Nullen nachgezogen)
SLD 0 bis 32	Schieben eines Doppelwortes nach links (von rechts werden Nullen nachgezogen)
SVW 0 bis 15	Schieben mit Vorzeichen nach rechts (von links wird Bit 15 nachgezogen)
SVD 0 bis 32	Schieben eines Doppelwortes mit Vorzeichen nach rechts (von links wird Bit 31 nachgezogen)
RLD 0 bis 32	Rotieren nach links
RRD 0 bis 32	Rotieren nach rechts

Bei den Schiebefunktionen ist nur der Akku 1 an der Ausführung beteiligt. Der Parameterteil dieser Befehle gibt an, um wie viele Stellen der Akku-Inhalt geschoben bzw. rotiert wird. Bei SLW, SRW und SVW ist nur das niederwertige Wort bei den Schiebefunktionen beteiligt, bei SLD, SVD, RLD und RRD der gesamte Inhalt des Akku 1 (32 Bit).

Die Schiebefunktionen werden unabhängig von Bedingungen ausgeführt.



Das zuletzt hinausgeschobene Bit kann mit Sprungfunktionen abgefragt werden: Mit SPZ kann gesprungen werden, wenn das zuletzt geschobene Bit = 0 ist, mit SPN, wenn das Bit = 1 ist.

Die Anzeigen ANZ0 und ANZ1 werden beeinflusst:

ANZ1	ANZ0	Schieben: letztes geschobenes Bit
0	0	0
0	1	-
1	0	1

### Beispiele

STEP5-Programm:            Inhalt der Datenwörter:

```
:L   DW52           KH = 14AF
:SLW 4
:T   DW53           KH = 4AF0
```

STEP5-Programm:            Inhalt von Akku 1 (hexadezimal):

```
:L   ED0            2348 ABCD
:SLW 4              2348 BCD0
:SRW 4              2348 0BCD
:SLD 4              3480 BCD0
:SVW 4              3480 FBCE
:SVD 4              0348 0FBC
:RLD 4              3480 FBCE
:RRD 4              0348 0FBC
:BE
```

Anwendungen:            Multiplikation mit 2er-Potenz,  
z.B. neuer Wert = alter Wert x 8

```
:L   MW10
:SLW 3
:T   MW10           Achtung: Positive Bereichsgrenze
                        nicht überschreiten!
```

Division durch 2er-Potenz,  
z.B. neuer Wert = alter Wert : 4

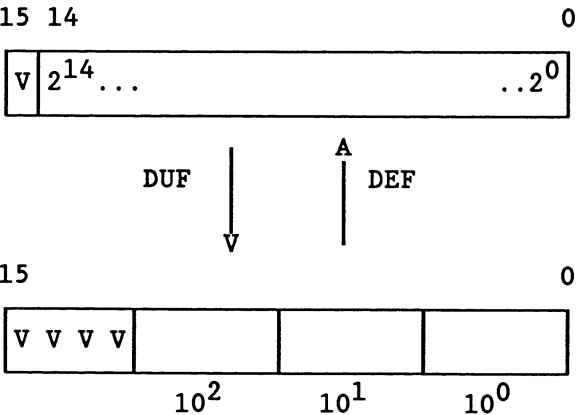
```
:A   DB5
:L   DW0
:SRW 2
:T   DW0
```

● Umwandlungsoperationen

Operation	Bedeutung
KEW	Bildung des 1er-Komplements von Akku 1 (16 Bit)
KZW	Bildung des 2er-Komplements von Akku 1 (16 Bit)
KZD	Bildung des 2er-Komplements von Akku 1 (32 Bit)
DEF	Festpunkt wandlung (16 Bit) von BCD in dual
DUF	Festpunkt wandlung (16 Bit) von dual in BCD
DED	Doppelwort wandlung (32 Bit) von BCD in dual
DUD	Doppelwort wandlung (32 Bit) von dual in BCD
FDG	Wandlung einer Festpunktzahl (32 Bit) in eine Gleitpunktzahl (32 Bit); siehe OB 220 : Vorzeichenerweiterung
GFD	Wandlung einer Gleitpunktzahl in eine Festpunktzahl (32 Bit)

**DEF:**  
Der im Akku 1-L (Bit 0 bis Bit 15) stehende Wert wird als BCD-codierte Zahl interpretiert. Nach der Umwandlung steht im Akku 1-L eine 16-Bit-Festpunktzahl.

**DUF:**  
Der im Akku 1-L (Bit 0 bis Bit 15) stehende Wert wird als 16-Bit-Festpunktzahl interpretiert. Nach der Umwandlung steht im Akku 1-L eine BCD-codierte Zahl.



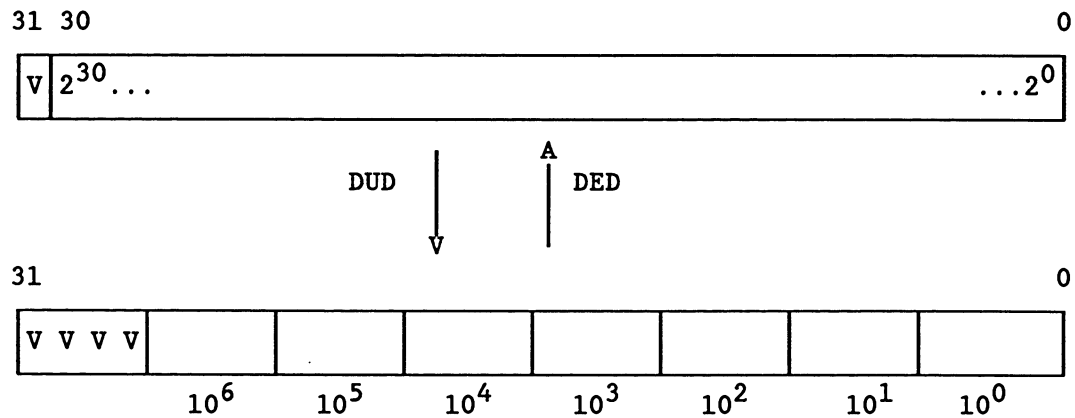
V (Vorzeichen): 0 = positiv  
1 = negativ

**DED:**

Der im Akku 1 (Bit 0 bis Bit 31) stehende Wert wird als BCD-codierte Zahl interpretiert. Nach der Umwandlung steht im Akku 1 eine 32-Bit-Festpunktzahl.

**DUD:**

Der im Akku 1 (Bit 0 bis Bit 31) stehende Wert wird als 32-Bit-Festpunktzahl interpretiert. Nach der Umwandlung steht im Akku 1 eine BCD-codierte Zahl.



V (Vorzeichen): 0 = positiv  
1 = negativ

**FDG:**

Der im Akku 1 (Bit 0 bis Bit 31) stehende Wert wird als 32-Bit-Festpunktzahl interpretiert. Nach der Umwandlung steht im Akku 1 eine Gleitpunktzahl (Exponent und Mantisse).

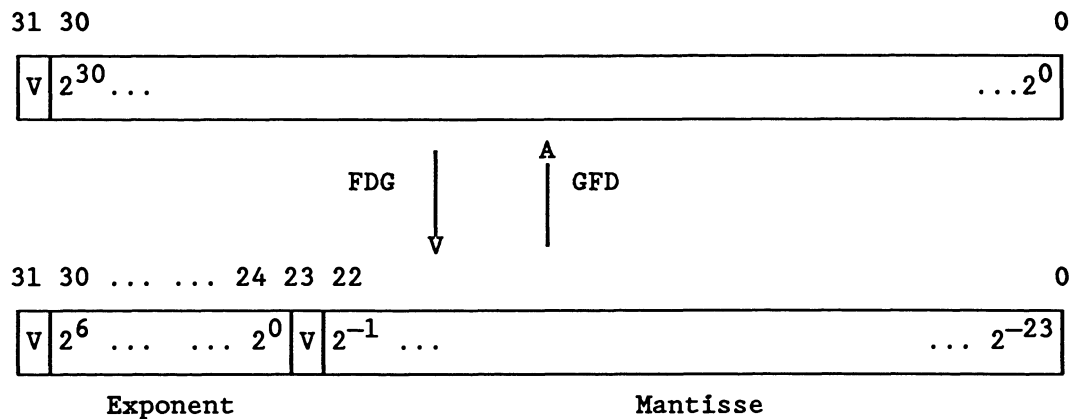
**GFD:**

Der im Akku 1 (Bit 0 bis Bit 31) stehende Wert wird als Gleitpunktzahl interpretiert. Nach der Umwandlung steht im Akku 1 eine 32-Bit-Festpunktzahl.

Gleitpunktzahlen  $\geq 0$  oder  $\leq -1$  werden dabei, falls nötig, auf die nächst kleinere ganze Zahl abgerundet.

Gleitpunktzahlen  $< 0$  und  $> -1$  werden dabei auf 0 aufgerundet.

Beispiele: +5,7 --> 5  
-2,3 --> -3  
-0,6 --> 0  
+0,9 --> 0



**KEW, KZW:**

### Beispiele

Der Inhalt des Datenwortes 64 soll Bit für Bit invertiert ('umgekehrt') und in Datenwort 78 abgelegt werden.

STEP5-Programm: Belegung der Datenwörter:

```
:L DW64 KM = 0011111001011011
:KEW
:T DW78 KM = 1100000110100100
```

Der Inhalt des Datenwortes 207 ist als Festpunktzahl zu interpretieren und mit umgekehrtem Vorzeichen im Datenwort 51 abzulegen.

STEP5-Programm: Belegung der Datenwörter:

```
:L DW207 KF = + 51
:KZW
:T DW51 KF = - 51
```

### ● Dekrementieren/Inkrementieren

Operation	Beschreibung
D 1 bis 255	Dekrementieren
I 1 bis 255	Inkrementieren
Parameter	

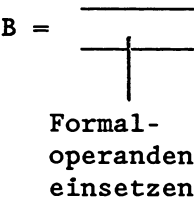
Der Akkumulatorinhalt 1 wird um die als Parameter angegebene Zahl dekrementiert (erniedrigt) bzw. inkrementiert (erhöht). Die Operationsausführung ist unabhängig von Bedingungen. Sie beschränkt sich auf das rechte Byte (ohne Übertrag).

### Beispiel

STEP5-Programm:      Belegung der Datenwörter:

:L	DW7	KH = 1010
:I	16	
:T	DW8	KH = 1020
:D	33	
:T	DW9	KH = 10FF

● **Bearbeitungsoperationen**

Operation	Beschreibung
B DW 0 bis 255 (Operation)	Bearbeite Datenwort Die nachfolgend angegebene Operation wird mit dem im Datenwort angegebenen Parameter kombiniert und ausgeführt.
B MW 0 bis 254 (Operation)	Bearbeite Merkerwort Die nachfolgend angegebene Operation wird mit dem im Merker angegebenen Parameter kombiniert und ausgeführt.
B =  Formal- operanden einsetzen	Bearbeite Fomaloperanden (Parameterart: B): Nur A DB, SPA PB, SPA FB, SPA SB können substituiert werden.
BI <sup>1)</sup> <sup>2)</sup>	Bearbeite über einen Formaloperanden (indirekt). Die Nummer des auszuführenden Formaloperanden steht im Akku 1.
B BS <sup>1)</sup> <sup>2)</sup> 60 bis 63	Befehl, der im Bereich Systemdaten (BS) steht, soll ausgeführt werden (freie Systemdaten: BS 60 bis 63).

- <sup>1)</sup> Systemfunktion  
<sup>2)</sup> Der Wert, der im Systemdatum oder im Formaloperanden steht, wird als Operationscode einer STEP5-Operation interpretiert, die dann ausgeführt wird. Zulässige Operationen wie bei B MW und B DW.

Mit B DW oder B MW dürfen alle Operationen kombiniert werden, außer den folgenden:

- alle Zweiwort- und Dreiwortbefehle, siehe dazu Anhang D, (erlaubt sind E DB, EX DX, SES, SEF, AX DX, BA FX und BAB FX,)
- Operationen mit Formaloperanden in Funktionsbausteinen,
- SPA/SPB OB, SPA/SPB PB, FB, .....

Das PG prüft die Zulässigkeit der Kombinationen nicht.

B    MW 14  
L    EB 0

120
-----

 MW 14

—————>    L    EB 120    (= ausgeführter Befehl)

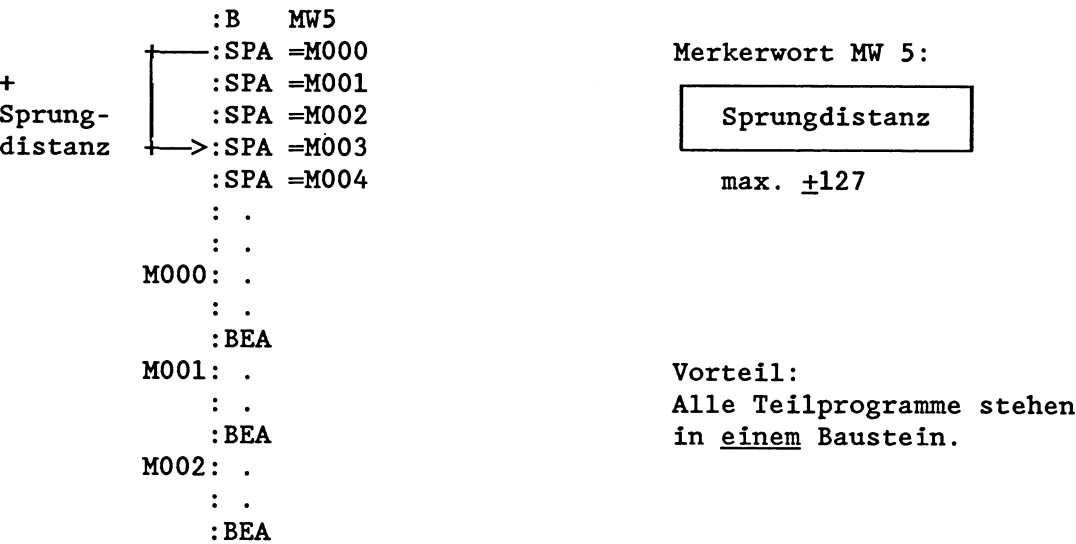
Beispiel (Bearbeite Datenwort)

Es sollen die Inhalte der Datenwörter DW 20 bis DW 100 auf Signalzustand "0" gesetzt werden. Das Indexregister für den Parameter der Datenwörter ist DW 1.

```

      :L   KF 20   Versorgung des Indexregisters
      :T   DW1
M001  :L   KB 0    Rücksetzen
      :B   DW1
      :T   DW0
      :L   DW1     Erhöhen des Indexregisters
      :L   KF 1
      :+F
      :T   DW1
      :L   KF 100
      :<=F
      :SPB =M001   Sprung, wenn Index im Bereich liegt
      ...         Weiteres STEP5-Programm
```

Anwendung: Sprungverteiler für Unterprogrammtechnik



### ● Prozeßalarme sperren/freigeben

AS	Prozeßalarmbearbeitung sperren
AF	Prozeßalarmbearbeitung freigeben

"Alarme sperren/freigeben" kann z. B. angewendet werden, wenn bei einer zeitgesteuerten Bearbeitung die prozeßalarmgesteuerte Bearbeitung unterdrückt werden soll. In dem Programmteil, der zwischen den Anweisungen AS und AF steht, ist dann die prozeßalarmgesteuerte Bearbeitung nicht mehr möglich.

Beachten Sie hierzu die Sonderfunktion OB 120 "Alarme sperren", Kapitel 6.8.1.

### ● Sonstige Operationen

Operation	Parameter	Beschreibung
E DB	3 bis 255	Erzeugen eines DB-Datenbausteins im DB-RAM
EX DX	1 bis 255	Erzeugen eines DX-Datenbausteins im DB-RAM

#### E DB: Erzeuge Datenbaustein

Der Befehl E DBxxx erzeugt einen DB-Datenbaustein mit der Nummer xxx (zwischen 3 und 255) im internen Datenbaustein-RAM des Prozessors.

Vor dem Programmieren der Anweisung müssen Sie die Anzahl der Datenwörter, die der neue DB haben soll, im Akku 1-L hinterlegen. Der dazugehörige Bausteinkopf wird vom E DB/EX DX-Befehl erzeugt. Ein Datenbaustein darf (inkl. Bausteinkopf) maximal 4091 Wörter im Speicher belegen.

Falls der entsprechende Datenbaustein schon existiert, die Länge des DBs unzulässig ist oder der Platz im DB-Ram nicht ausreicht, ruft das Systemprogramm den OB 31 auf. Wenn dieser nicht programmiert ist, geht der Prozessor aufgrund eines Laufzeitfehlers in den Stoppzustand. Im Akku 1 sind dann Fehlerkennungen hinterlegt.

Der Befehl EX DXxxx erzeugt im DB-RAM einen DX-Datenbaustein und arbeitet wie E DBxxx (zulässige Parameter: 1 bis 255).



Operation	Parameter	Beschreibung
SES	0 bis 31	Setzen einer Semaphore
SEF	0 bis 31	Freigeben einer Semaphore

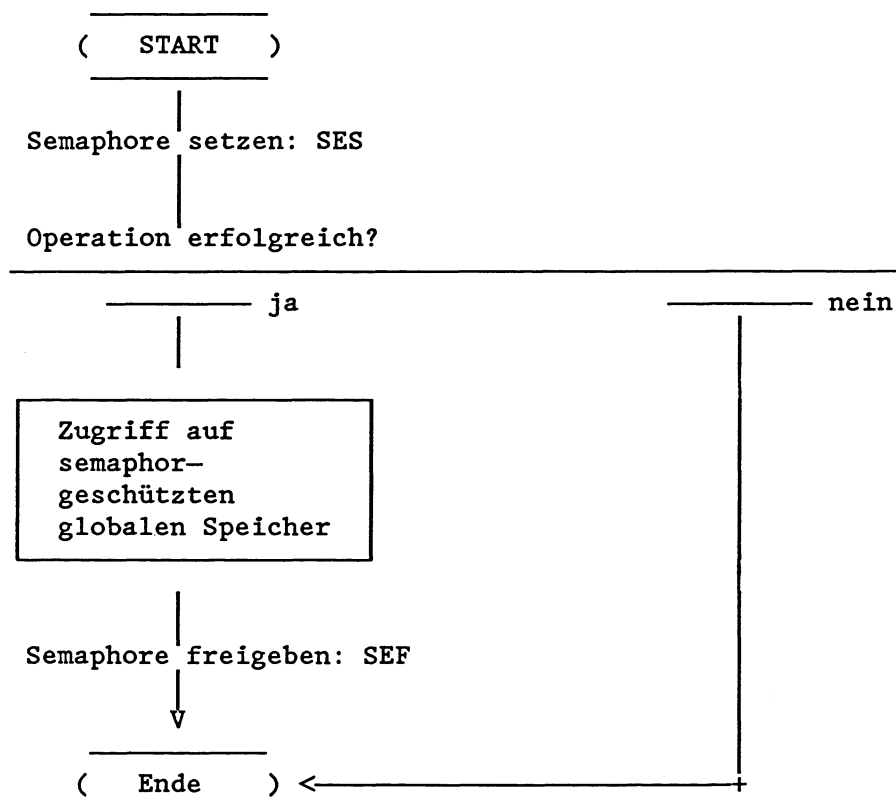
### SES/SEF: Semaphore setzen/freigeben

Benutzen zwei oder mehr Prozessoren eines Automatisierungsgerätes bestimmte globale Speicherbereiche (Peripherie, CPs, IPs) gemeinsam, besteht die Gefahr, daß die Prozessoren einander Daten überschreiben oder daß ungültige Zwischenstände der Daten ausgelesen werden. Deshalb ist es erforderlich, den Zugriff der Prozessoren auf die gemeinsamen Speicherbereiche zu koordinieren.

Die Koordinierung der einzelnen Prozessoren ist mit den Semaphoren und den Befehlen SES und SEF möglich: Nur nach erfolgreichem Setzen der vereinbarten Semaphore (SES) greift jeder der am Mehrprozessorbetrieb beteiligten Prozessoren auf den gemeinsamen Speicherbereich zu. Eine Semaphore xx kann dabei immer nur durch einen einzigen Prozessor gesetzt werden. Gelingt einem Prozessor das Setzen der Semaphore nicht, muß er auf den Zugriff verzichten.

Ebenso muß ein Prozessor auf einen weiteren Zugriff verzichten, nachdem er die Semaphore wieder freigegeben hat (SEF).

Alle beteiligten Prozessoren müssen einen Funktionsbaustein mit folgender Programmstruktur enthalten:



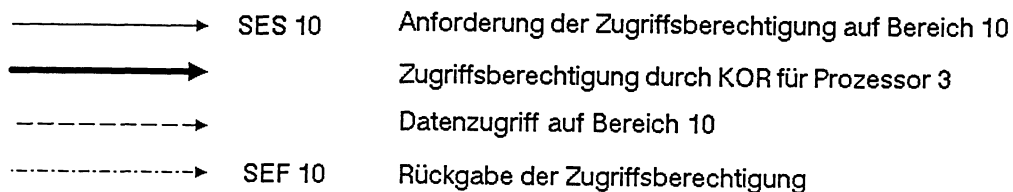
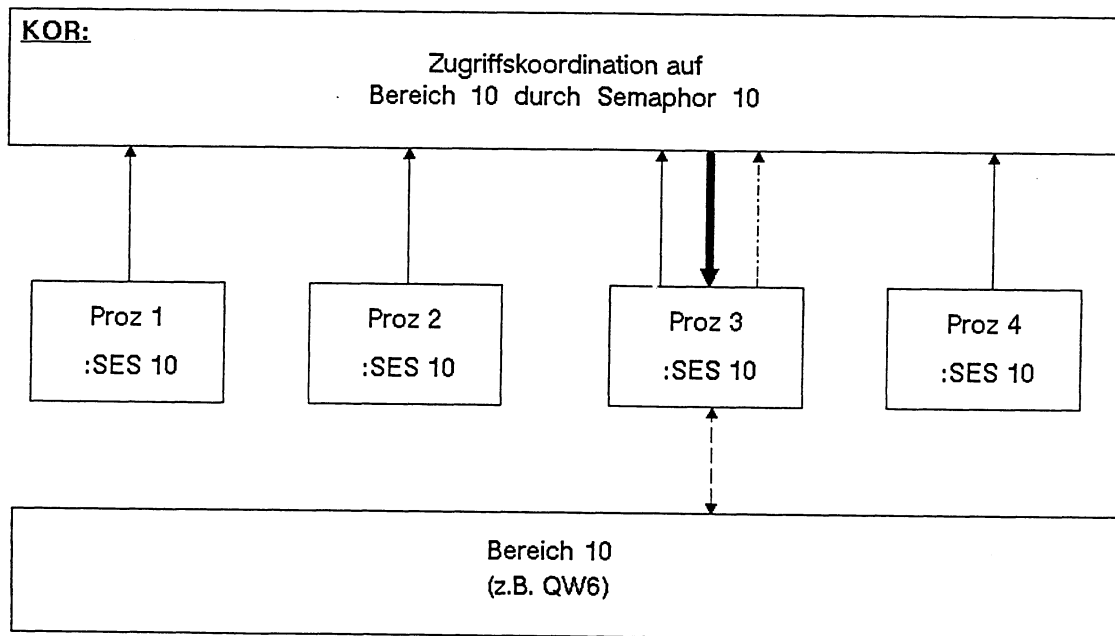
Durch Anwendung der Befehle SES und SEF ist gewährleistet, daß ein Prozessor zusammengehörige Informationen "geschützt" in einen/aus einem bestimmten Speicherbereich übertragen kann, ohne dabei durch einen anderen Prozessor unterbrochen zu werden.

#### WICHTIG!

Die Befehle **SES xx** und **SEF xx** müssen von allen Prozessoren verwendet werden, die synchronisiert auf einen gemeinsamen globalen Speicherbereich (Adressen > F000H) zugreifen sollen.

Der Befehl **SES xx** (Semaphore setzen) belegt für den befehlsausführenden Prozessor ein bestimmtes Byte im Koordinator (vorausgesetzt, dieses ist nicht bereits durch einen anderen Prozessor belegt). Solange sich der Prozessor dort eingetragen hat, dürfen die übrigen Prozessoren auf den mit der Semaphore (Nummer 0 bis 31) geschützten Speicherbereich nicht mehr zugreifen. Der Bereich ist damit für alle anderen Prozessoren gesperrt.

Der Befehl **SEF xx** (Semaphore freigeben) setzt das Byte im Koordinator wieder zurück. Dadurch wird der geschützte Speicherbereich für die anderen Prozessoren wieder les- bzw. beschreibbar. Eine Semaphore kann nur von demjenigen Prozessor freigegeben werden, von der sie gesetzt wurde.



Vor jedem Setzen bzw. Freigeben einer bestimmten Semaphore fragen die Befehle SES und SEF den Zustand (= Status) dieser Semaphore ab. Die Anzeigen ANZ0 und ANZ1 werden dabei wie folgt beeinflusst:

ANZ1	ANZ 0	Bedeutung	Auswertung
0	0	Semaphore ist von anderem Prozessor gesetzt worden und kann nicht gesetzt/freigegeben werden	SPZ
1	0	Semaphore wird gesetzt/freigegeben	SPN, SPP

#### WICHTIG!

Der Vorgang des Abfragens einer bestimmten Semaphore (= Lesevorgang) und der Vorgang des Setzens bzw. des Freigebens der Semaphore (= Schreibvorgang) bilden eine Einheit. Kein anderer Prozessor kann während dieser Vorgänge auf diese Semaphore zugreifen!

Zur Anwendung der Semaphoren beachten Sie bitte die folgenden Punkte:

- Eine Semaphore ist eine globale Größe, d.h., die Semaphore mit der Nummer 16 ist auch bei Einsatz von z.B. drei Prozessoren im gesamten System nur **e i n m a l** vorhanden.
- Die Befehle SES und SEF müssen von **a l l e n** Prozessoren verwendet werden, deren Zugriff auf einen gemeinsamen Speicherbereich koordiniert erfolgen soll.
- Alle beteiligten Prozessoren müssen die **g l e i c h e** Anlaufart durchführen. Bei Neustart werden alle Semaphoren gelöscht, bei einem manuellen oder automatischen Wiederanlauf bleiben sie erhalten.
- Der Anlauf im Mehrprozessorbetrieb muß synchronisiert erfolgen. Aus diesem Grund ist **k e i n** Testbetrieb erlaubt.

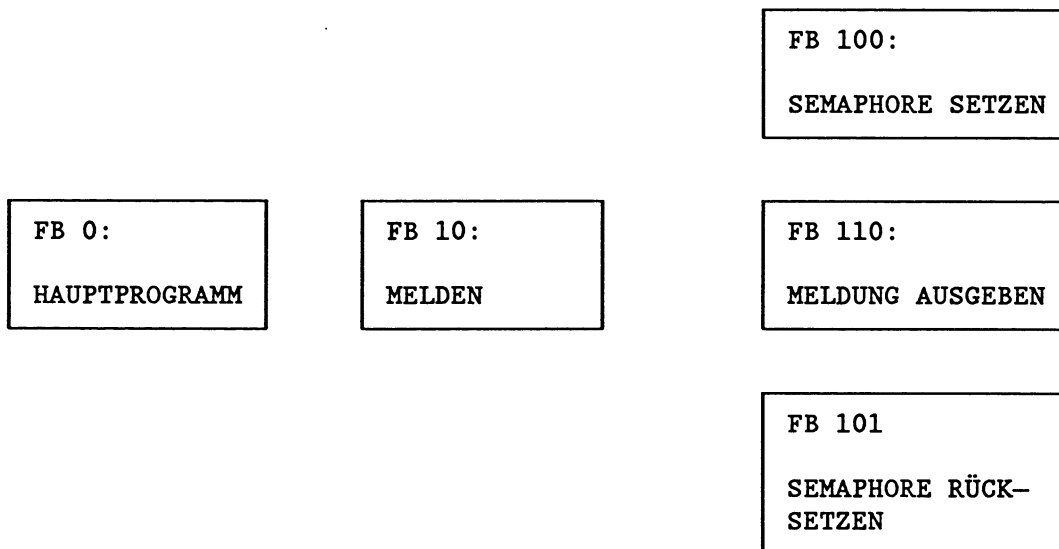
## Anwendungsbeispiel für Semaphoren

In einem AG S5-135U stecken vier Prozessoren, die über einen gemeinsamen Speicherbereich der Q-Peripherie (QW 6) Statusmeldungen an ein Statusmeldegerät abgeben. Jede Statusmeldung muß 10 Sekunden lang ausgegeben werden und darf erst dann von einer neuen Meldung des gleichen oder eines anderen Prozessors überschrieben werden.

Die Benutzung des Peripheriewortes QW 6 (erweiterte Peripherie, kein Prozeßabbild) wird über eine Semaphore gesteuert. Es darf nur derjenige Prozessor seine Meldung auf das QW 6 schreiben, der durch erfolgreiches Setzen der zugeordneten Semaphore diesen Bereich für sich belegen konnte. Für die Dauer von jeweils 10 Sekunden bleibt die Semaphore gesetzt (TIMER T10). Erst nach Ablauf des Timers gibt der Prozessor die Semaphore und damit den belegten Bereich für die anderen Prozessoren wieder frei. Das QW 6 kann mit einer neuen Meldung beschrieben werden.

Ist beim Versuch eines Prozessors, die Semaphore zu setzen, diese bereits durch einen anderen Prozessor gesetzt worden, versucht der Prozessor im nächsten Zyklus erneut, die Semaphore zu setzen und seine Meldung auszugeben.

Das folgende Programm kann in allen vier Prozessoren - mit einer jeweils anderen Meldung - ablaufen. Folgende Bausteine werden geladen:



Es werden 5 Merker verwendet:

- M 10.0 = 1: Eine Meldung ist angefordert oder in Bearbeitung.
- M 10.1 = 1: Die Semaphore ist erfolgreich gesetzt worden.
- M 10.2 = 1: Der Timer ist gestartet.
- M 10.3 = 1: Die Meldung ist übertragen.
- M 10.4 = 1: Die Semaphore ist rückgesetzt.

## FB0

NAME:MAIN

```
      :U    M 10.0      Falls keine Meldung aktiv,
      :SPB  =M001
      :
      :UN    E 0.0
      :BEB
      :
      :L    KH2222      Meldung erzeugen und
      :T    MW12
      :UN    M 10.0
      :S    M 10.0      Merker 'MELDUNG' setzen
      :
M001 :SPA FB10          FB 'MELDE' aufrufen
NAME :MELDE
      :
      :BE
```

## FB 10

NAME:MELDE

```
      :UN    M 10.1      Wenn keine Semaphore gesetzt,
      :SPB  FB100        FB 'Semaphore setzen' aufrufen
NAME :SEMASET
      :
      :U    M 10.1      Wenn Semaphore gesetzt
      :UN    M 10.2      und Timer nicht gestartet,
      :S    M 10.2
      :L    KT010.2      Timer starten
      :SV    T 10
      :
      :U    M 10.2      Wenn Timer gestartet
      :UN    M 10.3      und keine Meldung übertragen wird,
      :SPB  FB110        FB 'Meldung ausgeben' aufrufen
NAME :MELDAUSG
      :
      :U    M 10.2      Wenn Timer gestartet
      :UN    M 10.4      und Semaphore nicht zurückgesetzt
      :UN    T 10        und Timer abgelaufen,
      :SPB  FB101        FB 'Semaphore rücksetzen' aufrufen
NAME :SEMARESE
      :
      :UN    M 10.4      Wenn Semaphore rückgesetzt,
      :BEB
      :
      :L    KH0000
      :T    MB10        alle Merker rücksetzen
      :BE
```

**FB100**

NAME: SEMASET

:SES 10	Semaphore Nr. 10 setzen
:SPZ =M001	
:UN M 10.1	Falls Semaphore erfolgreich gesetzt,
:S M 10.1	Merker 'SEMA-GESETZT' setzen
M001 :BE	

**FB110**

NAME: MELDAUSG

:L MW12	Meldung an die
:T QW6	Peripherie übertragen
:UN M 10.3	
:S M 10.3	Merker 'MELDUNG-ÜBERTRAGEN' setzen
:BE	

**FB101**

NAME: SEMARESE

:SEF 10	Semaphore Nr. 10 freigeben
:SPZ =M001	
:UN M 10.4	
:S M 10.4	Merker 'SEMAPHORE-RÜCKGESETZT' setzen
M001 :BE	

## 4 Betriebszustände

### 4.1 Betriebszustände und Programmbearbeitungsebenen

Der Prozessor kennt drei Betriebszustände:

Betriebszustand STOP

Betriebszustand ANLAUF

Betriebszustand RUN

Wie in Kapitel 2 beschrieben, ruft das Systemprogramm beim Auftreten bestimmter Ereignisse die dafür vorgesehenen Organisationsbausteine (OB 1 bis OB 34) auf, in denen der Anwender die weitere Reaktion des Prozessors festlegen kann. Tritt beispielsweise beim Aktualisieren des Prozeßabbilds im RUN ein Quittungsverzug auf, ruft das Systemprogramm den OB 24 auf.

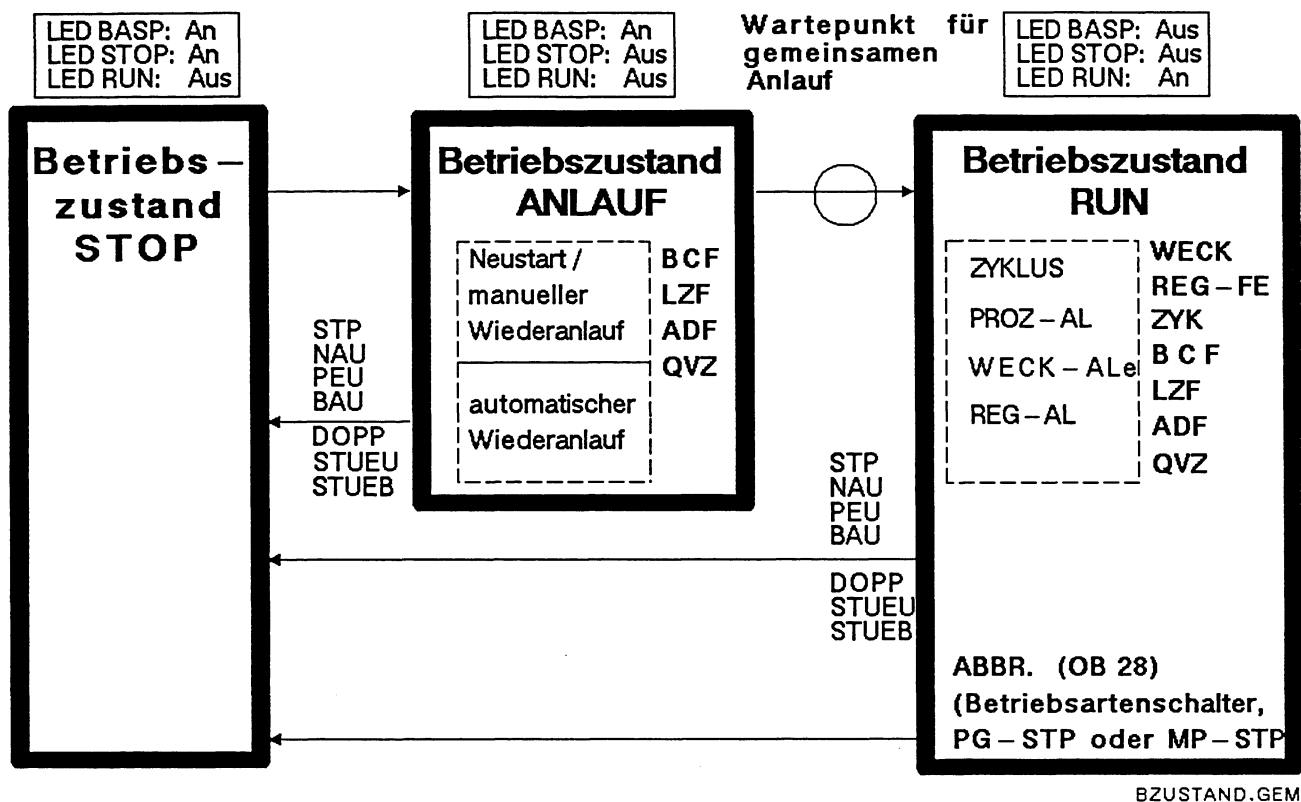
Einige Ereignisse können nur im Betriebszustand ANLAUF, einige nur im Betriebszustand RUN und manche sowohl im ANLAUF als auch im RUN auftreten (siehe folgende Seite).

Nach Aufruf eines Organisationsbausteins führt der Prozessor das darin enthaltene STEP5-Anwenderprogramm aus. Dabei wird ein neuer Registersatz angelegt (Register: Akku 1 bis 4, Bausteinstack-Pointer, Datenbaustein-Anfangsadresse, STEP-Adreßzähler). Ist durch das Auftreten des Ereignisses die 'normale' Programmbearbeitung unterbrochen worden, setzt der Prozessor nach der Bearbeitung des OBs - inklusive aller dort eingeschachtelten Bausteine - die Programmbearbeitung an der Unterbrechungsstelle fort. Dort gelten wieder die alten Register-Inhalte.

Einem oder mehrerer dieser Organisationsbausteine ist nun jeweils eine sog. Programmbearbeitungsebene zugeordnet: Wird z.B. der OB 2 aufgerufen, ist damit die Programmbearbeitungsebene 'PROZESSALARM' aktiviert. Die Programmbearbeitungsebene 'BEFEHLS-CODEFEHLER' wird aktiviert durch den Aufruf von OB 27 oder OB 29 oder OB 30.

Auf der folgenden Seite sehen Sie eine Übersicht über die Betriebszustände und Programmbearbeitungsebenen im AG 135U, CPU 928.

## Übersicht 4-1: Betriebszustände und Programmbearbeitungsebenen



### Programmbearbeitungsebenen im ANLAUF:

NEUSTART/MANUELLER WIEDERANLAUF  
AUTOMATISCHER WIEDERANLAUF  
**BCF** (Befehlscodefehler)  
**LZF** (Laufzeitfehler)  
**ADF** (Adressierfehler)  
**QVZ** (Quittungsverzug)

### Programmbearbeitungsebenen im RUN:

<b>ZYKLUS</b>	(zyklische Programmbearbeitung)
<b>WECKALARM 5sec</b>	(zeitgesteuerte Programmbearbeitung)
<b>WECKALARM 2sec</b>	( " " )
<b>WECKALARM 1sec</b>	( " " )
<b>WECKALARM 500ms</b>	( " " )
<b>WECKALARM 200ms</b>	( " " )
<b>WECKALARM 100ms</b>	( " " )
<b>WECKALARM 50ms</b>	( " " )
<b>WECKALARM 20ms</b>	( " " )
<b>WECKALARM 10ms</b>	( " " )
<b>REGLERALARM</b>	(zeitgesteuerte Bearbeitung von Reglern)
<b>PROZESSALARM</b>	(alarmgesteuerte Programmbearbeitung)
<b>WECK</b>	(Weckfehler)
<b>REG</b>	(Reglerfehler)
<b>ZYK</b>	(Zyklusfehler)
<b>BCF</b>	(Befehlscodefehler)
<b>LZF</b>	(Laufzeitfehler)
<b>ADF</b>	(Adressierfehler)
<b>QVZ</b>	(Quittungsverzug)
<b>ABBR</b>	(Abbruch)



Eine Programmbearbeitungsebene ist ferner durch folgende **Merkmale** charakterisiert:

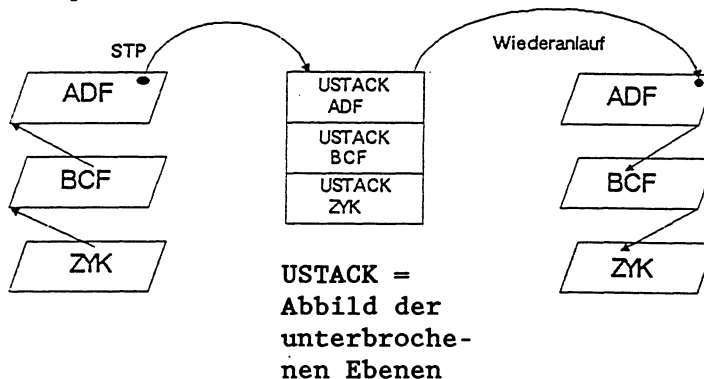
- Jede Programmbearbeitungsebene hat ihr spezifisches Systemprogramm.

*Beispiel:*

In der Bearbeitungsebene ZYKLUS aktualisiert das Systemprogramm das Prozeßabbild der Ein- und Ausgänge, triggert die Zykluszeit und ruft die Verwaltung der PG-Schnittstelle auf (Systemkontrollpunkt).

- Für jede Programmbearbeitungsebene legt das Systemprogramm im Unterbrechungsfall einen eigenen Unterbrechungsstack (USTACK) an und vermerkt darin u.a. die Ebene, die unterbrochen worden ist (siehe EBENE).

*Beispiel:*



- Die Programmbearbeitungsebenen haben eine feste Priorität. Davon abhängig können sie einander unterbrechen bzw. ineinander verschachtelt werden.

Die "Wiederanlauf- und die Fehlerebenen" unterscheiden sich von den "Grundebenen" dadurch, daß sie grundsätzlich *sofort* und an *Befehlsgrenzen* eingeschachtelt werden, sobald das entsprechende Ereignis auftritt. Sie können sowohl in die Grundebenen als auch gegenseitig eingeschachtelt werden. Bei Fehlern hat der zuletzt aufgetretene immer die höchste Priorität.

Eine "Grundebene" hingegen kann in die jeweils niedripore nur an *Bausteingrenzen* eingeschachtelt werden, es sei denn, diese Voreinstellung ist durch eine entsprechende Programmierung des DX 0 geändert worden (siehe Kapitel 7).

Priorität der "Grundebenen":

ZYKLUS  
WECKALARME <sup>1)</sup>  
REGLERALARM  
PROZESSALARM



aufsteigende Priorität

<sup>1)</sup> Die Weckalarme sind untereinander ebenfalls priorisiert, wobei kürzere Weckalarme höherprior sind als längere.

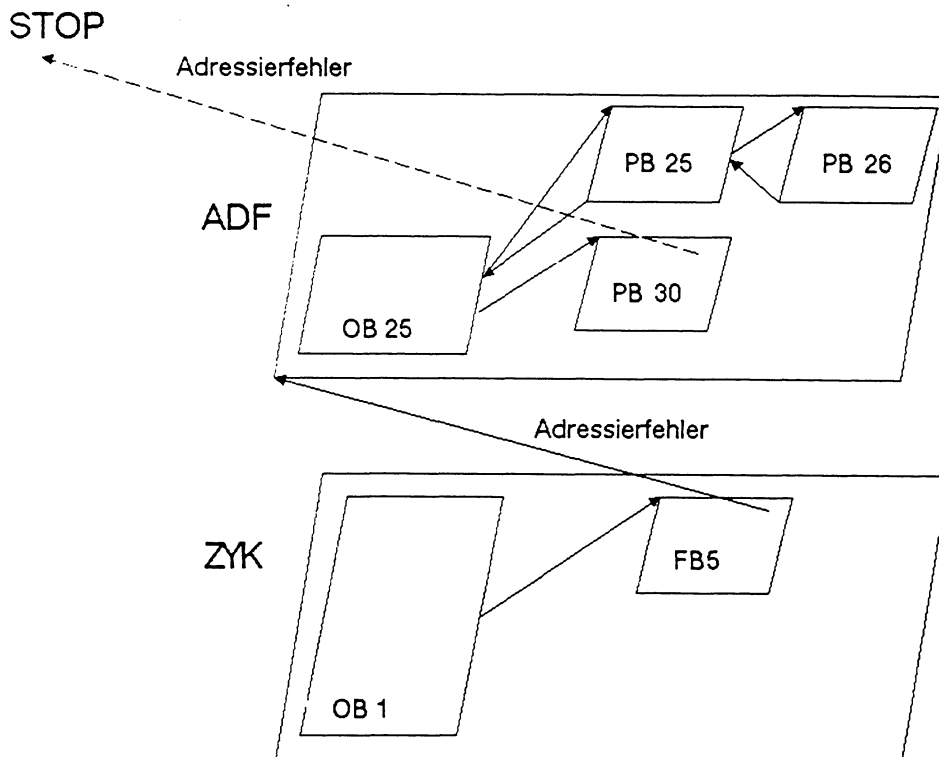
*Beispiel:*

Bei der Bearbeitung eines Weckalarms tritt ein Prozeßalarm auf. Da der Prozeßalarm eine höhere Priorität besitzt, wird die Bearbeitung der Weckalarm-Ebene an der nächsten Bausteingrenze unterbrochen und die PROZESSALARM-Programmbearbeitungsebene eingeschachtelt. Tritt bei der Bearbeitung des Prozeßalarms nun z.B. eine falsche Adressierung auf, so wird der Prozeßalarm sofort an der nächsten Befehlsgrenze unterbrochen, um die ADF-Ebene einzuschachteln.

- Eine einmal aktivierte Fehlerebene (ADF, BCF, LZF, QVZ, REG, ZYK), die noch nicht vollständig abgearbeitet ist, kann nicht noch einmal aktiviert werden, auch nicht, wenn eine andere Programmbearbeitungsebene dazwischengeschachtelt ist. In diesem Fall geht das AG wegen Doppelaufwurf einer Programmbearbeitungsebene (im USTACK: 'DOPP') unmittelbar in Stop (Ausnahme: Weckfehler, siehe dort). Im USTACK mit der Tiefe '01' ist die Kennung 'DOPP' sowie die doppelt aufgerufene Fehlerebene angekreuzt.

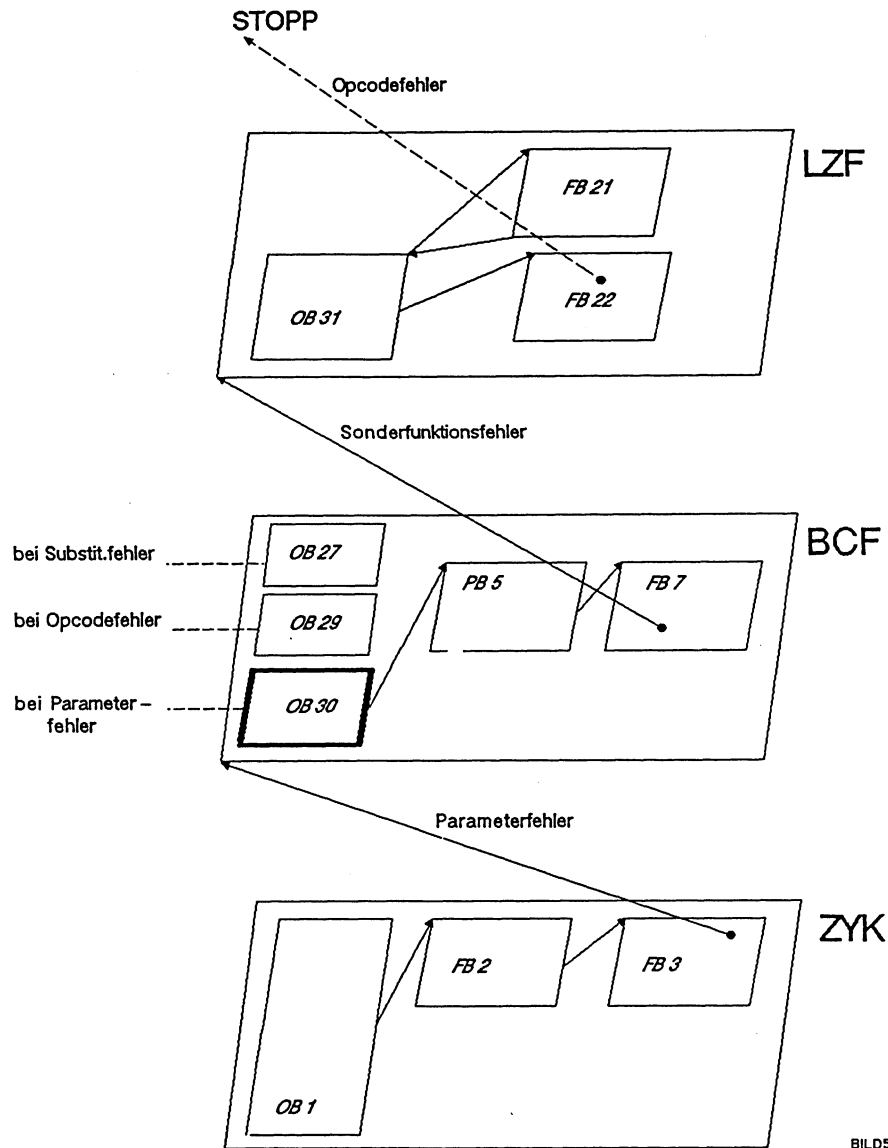
*Beispiel 1:*

Bei der Bearbeitung der ADF-Ebene (AW-Schnittstelle OB 25) tritt ein weiterer Adressierfehler auf. Da die ADF-Ebene noch aktiviert ist, kann sie kein zweites Mal aufgerufen werden: Der Prozessor geht in Stop.



## Beispiel 2:

Bei Auftreten eines Operationscodefehlers in der LZF-Programmbearbeitungsebene versucht das Systemprogramm, die BCF-Ebene (AW-Schnittstelle OB 29) aufzurufen. Diese ist jedoch schon durch das Auftreten eines Parameterfehlers (AW-Schnittstelle OB 30) aktiviert worden und nicht vollständig abgearbeitet. Ein erneuter Aufruf der BCF-Ebene an dieser Stelle ist deshalb nicht zulässig: Der Prozessor geht in Stop.



Die einzelnen Programmbearbeitungsebenen mit den dazugehörigen Anwenderschnittstellen werden in den folgenden Kapiteln genauer beschrieben:

Kapitel 4.3 beschreibt die "Grundebenen" im ANLAUF.

Kapitel 4.4 beschreibt die "Grundebenen" im RUN.

Kapitel 5.5 und 5.6 beschreiben die "Fehlerebenen" im ANLAUF und im RUN.

## 4.2 Betriebszustand STOP

Der Betriebszustand STOP ist durch folgende Merkmale charakterisiert:

- Es findet keine Anwenderprogrammbearbeitung statt.
- Hat eine Programmbearbeitung stattgefunden, so bleiben die Werte von Zählern, Zeiten, Merkern und Prozeßabbildern beim Übergang in den Stoppzustand erhalten.
- Das Signal BASP (Befehlsausgabe sperren) ist ausgegeben. Damit sind alle digitalen Ausgaben gesperrt.

*Ausnahme:* Im Testbetrieb wird BASP nicht ausgegeben!  
(Zum Testbetrieb siehe Kapitel 10.5.)

- Hat eine Programmbearbeitung stattgefunden, so ist im Stoppzustand ein Unterbrechungsstack (USTACK) vorhanden.

### LEDs auf der Frontplatte der Zentralbaugruppe:

RUN-LED:	aus	
STOP-LED:	an	(dauernd oder blinkend)
BASP-LED:	an	(außer im Testbetrieb)

Die STOP-LED kann Ihnen dabei signalisieren, welche Ursachen für den momentanen Stoppzustand in Frage kommen. Dauerlicht oder Blinken der STOP-LED haben eine bestimmte Bedeutung, die im folgenden kurz beschrieben wird.

### Die STOP-LED zeigt Dauerlicht:

Der Betriebszustand STOP wurde ausgelöst

im *Einzelprozessorbetrieb*:

- a) bei Betätigen des Betriebsartenschalters von 'RUN' auf 'STOP'
- b) bei PG-Funktion 'AG-STOP'
- c) bei Gerätefehlern (BAU, PEU, NAU)
- d) nach Urlöschen

im *Mehrprozessorbetrieb*:

- a) bei Betätigen des Betriebsartenschalters am KOR auf 'STOP'
- b) ein anderer Prozessor ist aufgrund einer Störung in Stop gegangen (der *nicht* fehlerverursachende Prozessor zeigt Dauerlicht)

- c) bei PG-Funktion 'AG-STOP' an einem Prozessor
- d) bei PG-Funktion 'BEARBEITUNGSKONTROLLE ENDE' an einem anderen Prozessor

**Die STOP-LED blinkt langsam:** (ca. 0,5 mal pro Sekunde)

Ein langsames Blinken der STOP-LED signalisiert Ihnen in den meisten Fällen einen Fehler.

Im Mehrprozessorbetrieb wird durch langsames Blinken derjenige Prozessor gekennzeichnet, der den Stoppzustand (durch einen Fehler) verursacht hat.

Die STOP-LED blinkt langsam

- a) bei Programmierung eines Stoppbefehls im Anwenderprogramm
- b) bei Fehlbedienungen (DB1-Fehler, Wahl einer unzulässigen Anlaufart, etc.)
- c) bei Programmier- und Gerätefehlern (Aufruf eines nicht geladenen Bausteins, Adressierfehler, Quittungsverzug, Befehlscodefehler etc);  
als zusätzlicher Hinweis auf die Fehlerursache leuchten:  
ADF-LED  
QVZ-LED  
ZYK-LED
- d) bei PG-Funktion 'BEARBEITUNGSKONTROLLE ENDE' an diesem Prozessor

**Die STOP-LED blinkt schnell:** (ca. 2 mal pro Sekunde)

Ein schnelles Blinken der STOP-LED signalisiert Ihnen die Warnung: "Urlöschen ist angefordert"!

### **Urlöschen anfordern**

- a) Das System fordert Urlöschen an:

Nach jedem Einschalten der Netzspannung und nach Durchführen des Urlöschens durchläuft der Prozessor eine Initialisierungsroutine. Wenn nun bei der Initialisierung Fehler festgestellt werden, geht der Prozessor in den Stoppzustand mit schnellem Blinken.

- |                  |  |
|------------------|--|
| Mögliche Fehler: | 1. Inhalte der RAMs sind leer<br>Abhilfe: Prozessor urlöschen  |
|                  | 2. Anwender-EPROM ist leer oder steckt nicht<br>Abhilfe: Programmiertes EPROM stecken und<br>Prozessor urlöschen |

Die Störungsursache muß beseitigt und danach der Prozessor (erneut) urgelöscht werden.

b) Der Anwender fordert Urlöschen an:

Durch folgende Bedienschritte fordern Sie Urlöschen an:

- Betätigen Sie den Betriebsartenschalter von 'RUN' nach 'STOP'.

Ergebnis: Der Prozessor befindet sich im Stoppzustand.  
Die STOP-LED zeigt Dauerlicht.

- Halten Sie den Wahlschalter in Stellung 'OVERALL RESET' fest; gleichzeitig betätigen Sie den Betriebsartenschalter von 'STOP' nach 'RUN' und wieder nach 'STOP'.

Ergebnis: Urlöschen ist angefordert.  
Die STOP-LED blinkt schnell.

#### **WICHTIG!**

Soll das von Ihnen angeforderte Urlöschen nicht ausgeführt werden, so wählen Sie jetzt eine Anlaufart.

#### **Urlöschen durchführen**

Unabhängig davon, ob die Urlöschanforderung vom System oder vom Anwender kommt, führen Sie das Urlöschen folgendermaßen durch:

- Halten Sie den Wahlschalter in Stellung 'OVERALL RESET' fest; gleichzeitig betätigen Sie den Betriebsartenschalter von 'STOP' nach 'RUN' und wieder nach 'STOP'.

Ergebnis: Urlöschen wird durchgeführt.  
Die STOP-LED zeigt Dauerlicht.

- Oder: Durch die PG-Funktion 'LÖSCHEN'.  
(Beim Urlöschen mit dem PG kann die manuelle Urlöschanforderung durch Schalterbetätigung entfallen! Die Stellung von Wahl- und Betriebsartenschalter ist nicht relevant.)

Ergebnis: Urlöschen wird durchgeführt.  
Die STOP-LED zeigt Dauerlicht.

#### **WICHTIG!**

Haben Sie Urlöschen durchgeführt, so ist als Anlaufart nur ein Neustart zulässig!

Ausgänge aus dem Stoppzustand:

- Wahl einer Anlaufart  
(siehe Kapitel 4.3)
- Urlöschen, anschließend Neustart
- Testbetrieb (bei Mehrprozessorbetrieb, siehe Kapitel 10.5)

### 4.3 Betriebszustand ANLAUF

Der Betriebszustand ANLAUF ist durch folgende Merkmale charakterisiert:

- Der ANLAUF ist der Übergang vom Betriebszustand STOP in den Betriebszustand RUN.
- Es gibt drei verschiedene, über Bedienung wählbare Anlaufarten: den Neustart, den Manuellen Wiederanlauf und den Automatischen Wiederanlauf.  
Nach einem Neustart wird das zyklische Anwenderprogramm von Anfang an bearbeitet. Nach einem Wiederanlauf wird die zyklische Anwenderprogrammbearbeitung an der Unterbrechungsstelle fortgesetzt.
- Für alle drei Anlaufarten ruft das Systemprogramm jeweils einen anderen Organisationsbaustein auf, in dem Sie ein bestimmtes Anlaufprogramm hinterlegen können. Die Länge des STEP5-Anlaufprogramms in den OBs ist nicht beschränkt. Es wird zeitlich nicht überwacht. In den Anlauf-OBs können weitere Bausteine aufgerufen werden.
- Die Werte von Zählern, Zeiten, Merkern und Prozeßabbildern werden in den einzelnen Anlaufarten unterschiedlich behandelt.
- Das Signal BASP (Befehlsausgabe sperren) ist ausgegeben. Damit sind alle digitalen Ausgaben gesperrt.  
*Ausnahme:* Im Testbetrieb wird BASP nicht ausgegeben!  
(Zum Testbetrieb siehe Kapitel 10.5.)

#### LEDs auf der Frontplatte der Zentralbaugruppe:

RUN-LED:	aus
STOP-LED:	aus
BASP-LED:	an (außer im Testbetrieb)

#### Anmerkung:

Hinweise zum "Anlaufverhalten im Mehrprozessorbetrieb" finden Sie in Kapitel 10.4.

#### So lösen Sie einen Neustart aus:

- Halten Sie den Wahlschalter in Stellung 'RESET'; gleichzeitig betätigen Sie den Betriebsartenschalter von 'STOP' nach 'RUN'.
- Oder: Durch die PG-Funktion 'AG-START' (--> Neustart).

Ein Neustart ist **erforderlich** nach

- Stacküberlauf (USTACK: 'STUEU, STUEB')
- Doppelaufruf einer Programmbearbeitungsebene (USTACK: 'DOPP')
- Urlöschen (Steuerbits: 'URGELOE')
- Anlaufabbruch (Steuerbits: 'ANL-ABB')
- Stop nach PG-Funktion "Bearbeitungskontrolle Ende"

Ein Neustart ist **immer zulässig**, sofern vom System nicht "Urlöschen" angefordert ist!

So lösen Sie einen Manuellen Wiederanlauf aus:

- Der Wahlschalter befindet sich in Mittelstellung.
- Betätigen Sie den Betriebsartenschalter von 'STOP' nach 'RUN'.
- Oder: Durch die PG-Funktion 'AG-START' (--> Wiederanlauf).

Ein Manueller Wiederanlauf ist immer zulässig mit Ausnahme der Fälle, in denen ausschließlich ein Neustart erlaubt ist (siehe oben) oder vom System "Urlöschen" angefordert ist.

Auslösen des Automatischen Wiederanlaufs:

Bei Netzspannungsausfall/Netz-aus im ANLAUF oder RUN und anschließender Netzspannungswiederkehr/Netz-ein durchläuft der Prozessor eine Initialisierungsroutine und versucht dann automatisch, einen Wiederanlauf durchzuführen.

Voraussetzung:

- Die Schalter an allen Prozessoren und am Koordinator stehen unverändert auf 'RUN'.
- Bei der Initialisierung sind keine Fehler aufgetreten.

Soll Ihr Prozessor bei Netzspannungsausfall und anschließender Spannungswiederkehr hingegen einen *Automatischen Neustart* durchführen, so ändern Sie die Voreinstellung durch Programmierung des Datenbausteins DX 0.

#### **WICHTIG!**

Ein Manueller bzw. ein Automatischer Wiederanlauf ist nur zulässig, wenn das Anwenderprogramm im Stoppzustand nicht verändert wurde.



#### 4.3.1 Neustart und Manueller Wiederanlauf

Folgende Tabelle enthält eine Gegenüberstellung der Anlaufarten Neustart und Manueller Wiederanlauf.

<b>zeitliche Abfolge</b>	<b>Neustart</b>	<b>Manueller Wiederanlauf</b>
Auslösung:  ↓  V	Stoppschalter von Stellung STOP in Stellung RUN und Wahlschalter in Stellung RESET  PG-Funktion AG-START (Neustart)  Einschalten der Versorgungsspannung, wenn im DX 0 "automat. Neustart" eingetragen ist.	Stoppschalter von Stellung STOP in Stellung RUN und Wahlschalter in Mittelstellung  PG-Funktion AG-START (manueller Wiederanlauf)
Systemprogrammleistungen:  ↓  V  ↓  V  ↓  V  ↓  V	<ul style="list-style-type: none"> <li>- Bausteinadreßliste im DB0 erhalten</li> <li>- Prozeßabbild der Eingänge löschen</li> <li>- Prozeßabbild der Ausgänge löschen</li> <li>- Merker, Zeiten, Zähler löschen</li> <li>- digitale/analoge Peripherie löschen (je 2 x 128 Bytes)</li> <li>- Globale Koppelmerker löschen (256 Bytes)</li> <li>- Semaphoren löschen (alle 32)</li> <li>- DB1 vorhanden: dort eingetragene digit. E/A und Koppelmerker-E/A in die PA-Listen übernehmen</li> <li>- DB1 nicht vorhanden: tatsächlich existierende Baugruppen (nur digit. E/A) in die PA-Listen übernehmen (Koppelmerker werden ignoriert)</li> <li>- Regler-Parametrierungs-Baustein DB2 auswerten</li> </ul>	<ul style="list-style-type: none"> <li>- Bausteinadreßliste im DB0 erhalten</li> <li>- Merker, Zeiten, Zähler erhalten</li> <li>- Globale Koppelmerker erhalten</li> <li>- Semaphoren erhalten</li> </ul>
V	- Anwenderschnittstelle OB 20 aufrufen (falls vorhanden)	- Anwenderschnittstelle OB 21 aufrufen (falls vorhanden)
V	- Anlauf im Mehrprozessorbetrieb synchronisieren	- Anlauf im Mehrprozessor-synchronisieren

## **Neustart: Programmieren des Organisationsbausteins OB 20**

Wenn der Prozessor einen Neustart durchführt, wird automatisch der OB 20 aufgerufen. Sie können dort ein STEP5-Programm hinterlegen, das einmalig vor Beginn der zyklischen Programmbearbeitung bestimmte Tätigkeiten ausführt:

Sie können z.B.

- Merker setzen
- Zeiten starten
- mit direkten Peripheriezugriffen Ausgänge setzen
- den Datenverkehr des Prozessors mit Peripheriebaugruppen vorbereiten.
- die Synchronisierung mit CPs durchführen (Hantierungsbaustein SYNCHRON, siehe Kapitel 6.9).

Schließen Sie den OB 20 mit 'BE' (Bausteinende) ab!

Nach der Bearbeitung des OB 20 beginnt die zyklische Programmbearbeitung durch Aufruf des OB 1 bzw. FB 0.

Ist der OB 20 nicht programmiert, beginnt der Prozessor am Ende eines Neustarts (nach den Systemleistungen) sofort mit der zyklischen Programmbearbeitung.

Wenn Sie im DX 0 einen "Automatischen Neustart bei Netzwiederkehr" eingetragen haben, wird bei Netzwiederkehr ebenfalls der OB 20 aufgerufen.

## **Manueller Wiederanlauf: Programmieren des Organisationsbausteins OB 21**

Wenn der Prozessor einen Manuellen Wiederanlauf durchführt, wird der OB 21 aufgerufen. Sie können dort ein STEP5-Programm hinterlegen, das einmalig vor Wiederaufnahme der zyklischen Programmbearbeitung bestimmte Tätigkeiten ausführt.

Schließen Sie den OB 21 mit 'BE' (Bausteinende) ab!

Nach der Bearbeitung des OB 21 wird die zyklische Programmbearbeitung an der Abbruchstelle mit der nächsten Anweisung fortgesetzt. Es gilt:

- Das BASP-Signal (Befehlsausgabe sperren) bleibt während der Bearbeitung des Restzyklus erhalten und wird erst mit Beginn des nächsten (vollständigen) Zyklus zurückgenommen.
- Das Prozeßabbild der Ausgänge bleibt am Ende des Restzyklus zurückgesetzt und wird erst am Ende des nächsten (vollständigen) Zyklus aktualisiert.

Ist der OB 21 nicht programmiert, beginnt der Prozessor am Ende eines Manuellen Wiederanlaufs sofort an der Abbruchstelle.

#### 4.3.2 Automatischer Wiederanlauf

Die *Funktion* des Automatischen Wiederanlaufs ist mit der des Manuellen Wiederanlaufs weitgehend identisch. Er unterscheidet sich vom Manuellen Wiederanlauf hauptsächlich in der Art seiner Auslösung.

zeitliche Abfolge	Automatischer Wiederanlauf
Auslösung: ↓ V	Netzspannungswiederkehr nach Netzspannungsausfall
System- programm- leistungen: ↓ V ↓ V ↓ V	<ul style="list-style-type: none"><li>- Bausteinadreßlisten im DB 0 erhalten</li><li>- Merker, Zeiten, Zähler erhalten</li><li>- Koppelmerker erhalten</li><li>- Semaphore erhalten</li></ul>
↓ V	<ul style="list-style-type: none"><li>- Anwenderschnittstelle OB 22 aufrufen (falls vorhanden)</li></ul>
↓ V	<ul style="list-style-type: none"><li>- Anlauf im Mehrprozessorbetrieb synchronisieren</li></ul>

#### Automatischer Wiederanlauf: Programmieren des Organisationsbausteins OB 22

Bei Rückkehr der Versorgungsspannung versucht der Prozessor, das unterbrochene Programm wieder aufzunehmen. Dabei wird zunächst der OB 22 aufgerufen, in dem Sie ebenfalls ein STEP5-Programm hinterlegen können, das einmalig vor Wiederaufnahme der zyklischen Programmbearbeitung von Ihnen gewünschte Tätigkeiten ausführt.

Wollen Sie grundsätzlich verhindern, daß Ihr Prozessor einen automatischen Wiederanlauf durchführt, so programmieren Sie im OB 22 eine Stopp-Anweisung und schließen ihn mit 'BE' (Bausteinende) ab!

```
OB 22 : STP          (Stop)
       : BE           (Bausteinende)
```

Ergebnis: Der Prozessor geht bei Rückkehr der Versorgungsspannung in Stop.

Nach der Bearbeitung des OB 22 wird die zyklische Programmbearbeitung an der Abbruchstelle mit der nächsten Anweisung fortgesetzt. Nach einem Netzausfall mit anschließender Netzwiederkehr gilt:

- Das BASP-Signal (Befehlsausgabe sperren) bleibt während der Bearbeitung des Restzyklus erhalten und wird erst mit Beginn des nächsten (vollständigen) Zyklus zurückgenommen.
- Das Prozeßabbild der Ausgänge bleibt am Ende des Restzyklus zurückgesetzt und wird erst am Ende des nächsten (vollständigen) Zyklus aktualisiert.

#### **4.3.3 Unterbrechungen im ANLAUF**

Ein Anlaufprogramm kann unterbrochen werden durch

- Netzspannungsausfall
- Betätigen des Stoppschalters
- Programm- und Gerätefehler (siehe Kapitel 5.6).

Soll der unterbrochene Anlauf anschließend mit einer der drei möglichen Anlaufarten fortgesetzt werden, so beachten Sie bitte die folgenden Punkte.

Bei **Netzspannungsausfall im Anlauf** und anschließender Netzwiederkehr müssen drei Fälle unterschieden werden:

1. Der Prozessor führt gerade einen Neustart (OB 20) durch. Nach Netzausfall und anschließender Netzwiederkehr wird der Organisationsbaustein OB 22 (Automatischer Wiederanlauf) an der unterbrochenen Stelle im OB 20 eingeschachtelt.
2. Der Prozessor führt gerade einen Manuellen Wiederanlauf (OB 21) durch. Nach Netzausfall und anschließender Netzwiederkehr wird der Organisationsbaustein OB 22 (Automatischer Wiederanlauf) an der unterbrochenen Stelle im OB 21 eingeschachtelt.
3. Der Prozessor führt bereits einen Automatischen Wiederanlauf (OB 22) durch. Nach Netzausfall und anschließender Netzwiederkehr wird kein zweiter OB 22 eingeschachtelt: Der unterbrochene OB 22 wird nach Rückkehr der Netzspannung nicht fortgesetzt, sondern abgebrochen, und es wird stattdessen der neu aufgerufene OB 22 bearbeitet.

#### **Stop durch Schalter im Anlauf und anschließender Manueller Wiederanlauf**

Brechen Sie einen beliebigen Anlauf durch Betätigen des Stoppschalters (oder PG-Funktion 'AG-STOP') ab und lösen anschließend

einen Manuellen Wiederanlauf aus, so wird der unterbrochene Anlauf an der Abbruchstelle fortgesetzt. Es wird kein OB 21 eingeschachtelt!

#### **Stop durch Schalter im Anlauf und anschließender Neustart**

Brechen Sie einen beliebigen Anlauf durch Betätigen des Stoppschalters (oder PG-Funktion 'AG-STOP') ab und lösen anschließend einen Neustart aus, so wird der unterbrochene Anlauf abgebrochen und ein Neustart (falls vorhanden, mit OB 20) durchgeführt.

#### 4.4 Betriebszustand RUN

Der Betriebszustand RUN ist durch folgende Merkmale charakterisiert:

- Das Anwenderprogramm wird zyklisch bearbeitet.
- Alle im Programm gestarteten Zähler und Zeiten 'laufen', die Prozeßabbilder werden zyklisch aktualisiert.
- Das Signal BASP (Befehlsausgabe sperren) ist aufgehoben. Damit sind alle digitalen Ausgaben freigegeben.
- Die Koppelmerker werden zyklisch aktualisiert.

LEDs auf der Frontplatte der Zentralbaugruppe:

RUN-LED:	an
STOP-LED:	aus
BASP-LED:	aus

#### WICHTIG!

Ist vor dem Übergang in den Betriebszustand RUN ein Automatischer oder Manueller Wiederanlauf durchgeführt worden, erlischt die BASP-LED erst dann, wenn der Restzyklus abgearbeitet und das Prozeßabbild aktualisiert worden ist.

#### WICHTIG!

Der Betriebszustand "RUN" wird nur über den Betriebszustand "ANLAUF" erreicht!

Im Betriebszustand RUN gibt es 12 Grundprogrammbearbeitungsebenen:

- |                     |   |
|---------------------|---|
| 1. den Zyklus:      | Das Anwenderprogramm wird zyklisch bearbeitet.  |
| 2. 9 Weckalarme:    | Das Anwenderprogramm wird zeitgesteuert bearbeitet.   |
| 3. den Regleralarm: | Zusätzlich zum Anwenderprogramm wird eine vorgegebene Anzahl an Reglern zeitgesteuert bearbeitet. |
| 4. den Prozeßalarm: | Das Anwenderprogramm wird alarmgesteuert bearbeitet.  |

Sie unterscheiden sich in folgenden Punkten:

- a) Sie werden durch unterschiedliche Ereignisse ausgelöst.
- b) In jeder Programmbearbeitungsebene führt das Systemprogramm unterschiedliche Funktionen aus.
- c) Für jede Programmbearbeitungsebene existiert als Anwenderschnittstelle ein anderer Organisationsbaustein bzw. Funktionsbaustein.

#### **4.4.1 ZYKLUS: Zyklische Programmbearbeitung**

Bei den speicherprogrammierbaren Steuerungen herrscht im allgemeinen die **zyklische Programmbearbeitung** vor.

##### **Auslösung:**

Hat der Prozessor das Anlaufprogramm fehlerfrei beendet, so beginnt er mit der zyklischen Programmbearbeitung.

##### **Leistungen des Systemprogramms:**

- Zu Beginn des Zyklus stellt es die zu überwachende Zykluszeit ein.
- Es aktualisiert das Prozeßabbild der Eingänge (PAE).
- Es aktualisiert die Eingangskoppelmerker.
- Es ruft die Anwenderschnittstelle auf: Der OB 1 wird bearbeitet.
- Am Ende des Zyklus aktualisiert es das Prozeßabbild der Ausgänge (PAA).
- Es aktualisiert die Ausgangskoppelmerker.

##### **Anwenderschnittstelle: OB 1 bzw. FB 0**

Bei der zyklischen Programmbearbeitung wird als Anwenderschnittstelle regelmäßig der Organisationsbaustein OB 1 oder der Funktionsbaustein FB 0 aufgerufen. Das STEP5-Anwenderprogramm im OB 1 oder FB 0 wird von Anfang an über verschiedene Bausteinaufrufe hinweg durchgehend bearbeitet. Nach den Systemleistungen beginnt der Prozessor wieder von vorne mit der ersten STEP5-Anweisung im OB 1 (bzw. FB 0).

Im OB 1 programmieren Sie die Aufrufe der Programm-, Funktions- und Schrittbausteine, die im zyklischen Programm bearbeitet werden sollen.

Wenn Sie ein kurzes, zeitkritisches Anwenderprogramm haben, bei dem Sie auf eine strukturierte Programmierung verzichten können, programmieren Sie den FB 0: Da er über den gesamten Operationsvorrat von STEP5 verfügt, können Sie Bausteinaufrufe einsparen und dadurch die Laufzeit des Programms verkürzen.

#### **WICHTIG!**

Programmieren Sie entweder den OB 1 oder den FB 0.

Wenn sowohl OB 1 als auch FB 0 programmiert sind, wird nur der OB 1 vom Systemprogramm aufgerufen. Wenn Sie den FB 0 als Anwenderschnittstelle verwenden, darf dieser keine Parameter enthalten!

#### **Unterbrechungsstellen**

Die zyklische Programmbearbeitung kann an *Bausteingrenzen* unterbrochen werden durch

- eine prozeßalarmgesteuerte Programmbearbeitung
- eine Reglerbearbeitung
- eine zeitgesteuerte Programmbearbeitung

(Durch Parametrierung des DX 0 können diese Unterbrechungen auch an *Befehlsgrenzen* erfolgen!)

Sie kann an *Befehlsgrenzen* unterbrochen bzw. ganz abgebrochen werden

- beim Auftreten eines Geräte- oder Programmfehlers
- durch Bedienung (PG-Funktion, Stoppschalter).

#### *Hinweis:*

Die Rechenregister Akku 1, 2, 3 und 4 können über Zyklusgrenzen hinweg - vom Ende eines Programmzyklus bis zum Beginn des nächsten - als Datenspeicher verwendet werden.

#### **4.4.2 WECKALARME: Zeitgesteuerte Programmbearbeitung**

Eine zeitgesteuerte Bearbeitung liegt vor, wenn ein von einer "inneren Uhr" kommendes Zeitsignal (Weckalarm) den Prozessor veranlaßt, die zyklische Programmbearbeitung 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 die Bearbeitung fort.

Dadurch werden in einem festgelegten Zeitraster bestimmte Programmabschnitte automatisch in die zyklische Programmbearbeitung eingeschoben.

In der CPU 928 können Sie bis zu 9 verschiedene Programme zeitgesteuert bearbeiten lassen, von denen jedes in einem anderen Zeitraster aufgerufen wird.



### Auslösung:

Ein Weckalarm wird automatisch in dem ihm zugehörigen Zeitraster ausgelöst, vorausgesetzt, der entsprechende OB ist programmiert.

### Anwenderschnittstellen: OB 10 bis OB 18

Als Anwenderschnittstelle wird bei Auftreten eines bestimmten Weckalarms an der nächsten Bausteingrenze (bzw. Befehlsgrenze) der zugehörige Organisationsbaustein aufgerufen.

Zuordnung:	Organisationsbaustein	Zeitraster
	OB 10	Aufruf alle 10 ms
	OB 11	Aufruf alle 20 ms
	OB 12	Aufruf alle 50 ms
	OB 13	Aufruf alle 100 ms
	OB 14	Aufruf alle 200 ms
	OB 15	Aufruf alle 500 ms
	OB 16	Aufruf alle 1 sec
	OB 17	Aufruf alle 2 sec
	OB 18	Aufruf alle 5 sec

Programmieren Sie beispielsweise im OB 13 denjenigen Programmteil, der alle 100 ms in die zyklische Programmbearbeitung eingeschoben werden soll.

Sie können alle, beliebig viele oder keinen dieser insgesamt 9 OBs programmieren. Ist keiner der 9 Organisationsbausteine programmiert, so findet grundsätzlich keine zeitgesteuerte Programmbearbeitung statt.

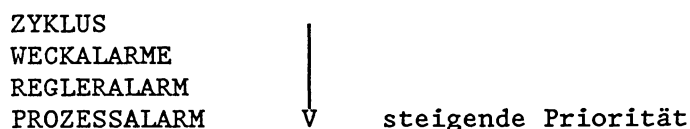
Bei jedem Aufruf eines Weckalarm-OBs (OB 10 bis OB 18) wird im Akku 1 mitgeteilt, wieviele Zeitraster seit dem letzten Aufruf des Weckalarm-OBs aufgetreten sind. Dabei gilt:

Akku 1 := Anzahl der Zeitraster - 1

Steht bei Aufruf des OB 11 beispielsweise die Zahl "5" im Akku 1, bedeutet dies, daß seit dem letzten Aufruf des OB 11 120ms (6 Zeitraster) vergangen sind. Solange kein Weckfehler vorliegt, wird im Akku 1 eine "0" übergeben.

### Priorisierung der Weckalarme

Die Programmbearbeitungsebenen WECKALARME sind innerhalb der Grundebenen folgendermaßen eingeordnet:



Die Priorität der einzelnen Weckalarme ist ebenfalls festgelegt:

OB 18	(längstes Zeitraster)		V	steigende Priorität
OB 17				
OB 16				
.				
.				
OB 11				
OB 10	(kürzestes Zeitraster)			

#### **WICHTIG!**

**OBs mit kürzeren Zeitrastern sind grundsätzlich höherprior und können OBs mit längeren Zeitrastern unterbrechen!**

#### **Unterbrechungsstellen**

Eine zeitgesteuerte Programmbearbeitung kann entweder an *Bausteingrenzen* (Voreinstellung) oder an *Befehlsgrenzen* (Programmierung DX 0) unterbrochen werden durch

- eine erneute zeitgesteuerte Bearbeitung
- eine Reglerbearbeitung
- eine prozeßalarmgesteuerte Bearbeitung

und an *Befehlsgrenzen* durch

- einen Programm- oder Gerätefehler.

#### Beispiel:

Während der Bearbeitung des OB 14 (Aufruf alle 200 ms) wird der OB 12 aufgerufen (Aufruf alle 50 ms). Der OB 14 wird an der nächsten Baustein- oder Befehlsgrenze unterbrochen und es wird der OB 12 bearbeitet. Erst wenn dieser vollständig bearbeitet ist (möglicherweise unterbrochen durch einen Regleralarm, einen Prozeßalarm, eine Fehlerbehandlung, einen OB 10 oder 11), wird die Programmbearbeitung im OB 14 fortgesetzt und zuendegeführt.

#### **Weckfehler**

##### **WICHTIG!**

**Eine zeitgesteuerte Programmbearbeitung kann nicht durch dieselbe zeitgesteuerte Programmbearbeitung unterbrochen werden!**

Wenn ein bestimmter Weckalarm-OB noch nicht vollständig bearbeitet ist und zum zweiten Mal aufgerufen wird, weil die Zeit abgelaufen ist, liegt ein Weckfehler vor. Ebenso kommt es zu einem Weckfehler, wenn ein bestimmter OB zum zweiten Mal aufgerufen wird, ohne daß der erste Aufruf bearbeitet worden wäre. Dies ist denkbar bei der Einstellung "Weckalarme unterbrechen an Bausteingrenzen", besonders dann, wenn Ihr STEP5-Programm langlaufende Bausteine beinhaltet.

Liegt ein Weckfehler vor, so wird die Fehlerprogrammbearbeitungsebene WECKFE aktiviert und das Systemprogramm ruft als Anwenderschnittstelle den **OB 33** auf. Im OB 33 können Sie die gewünschte Reaktion auf diesen Zustand programmieren.

Beim Aufruf des OB 33 hinterlegt das Systemprogramm in den Akkus 1 und 2 zusätzliche Informationen, die den ersten aufgetretenen Fehler näher erläutern:

Fehlerkennung Akku1      Akku2		Erläuterung
1001H	0016H	Weckfehler bei OB 10 (10 ms)
1001H	0014H	Weckfehler bei OB 11 (20 ms)
1001H	0012H	Weckfehler bei OB 12 (50 ms)
1001H	0010H	Weckfehler bei OB 13 (100 ms)
1001H	000EH	Weckfehler bei OB 14 (200 ms)
1001H	000CH	Weckfehler bei OB 15 (500 ms)
1001H	000AH	Weckfehler bei OB 16 (1 sec)
1001H	0008H	Weckfehler bei OB 17 (2 sec)
1001H	0006H	Weckfehler bei OB 18 (5 sec)

Hinweis: Die Kennung im Akku 2 ist die Ebenenkennung des fehlererzeugenden Weckalarms.

Ist der OB 33 nicht programmiert, geht der Prozessor in den Stoppzustand. Dann ist am Programmiergerät bei "Ausgabe USTACK" in den Steuerbits "WECKFE" angekreuzt, im USTACK ist die Ebenenkennung des entsprechenden Weckalarms (EBENE) angegeben.

Soll die Programmbearbeitung bei einem aufgetretenen Weckfehler fortgesetzt werden, programmieren Sie entweder im OB 33 die Bausteinende-Anweisung "BE", oder Sie ändern die Voreinstellung im DX 0 dahingehend, daß bei einem aufgetretenen Weckfehler und nicht programmiertem OB 33 die Programmbearbeitung trotzdem *fortgesetzt* wird. Nach der Bearbeitung des OB 33 wird das Programm an der Unterbrechungsstelle fortgesetzt.

Hinweise:

- Beachten Sie im Hinblick auf die zeitgesteuerte Programmbearbeitung die neuen Sonderfunktionen **OB 120**, **OB 121**, **OB 122** und **OB 123**, mit denen Sie die Bearbeitung von Weckalarmen für einen bestimmten Programmteil sperren bzw. verzögern können. (Dies ist möglich entweder für *alle* programmierten Weckalarme oder für *einzelne* von ihnen.)

- Je 'schneller' eine zeitgesteuerte Programmbearbeitungsebene ist, umso größer wird die Gefahr von Weckfehlern: Weckalarme mit kurzen Zeitrastern (z.B. der 10ms- und der 20ms-Weckalarm) werden im allgemeinen erfordern, daß sie auf *Unterbrechung an Befehlsgrenzen* eingestellt werden. Dies bedingt, daß auch der Regleralarm und der Prozeßalarm auf Unterbrechung an Befehlsgrenzen eingestellt sind (siehe Kapitel 7, DX 0-Parametrierung).

#### 4.4.3 REGLER-ALARM: Bearbeitung von Reglern

In der CPU 928 ist neben der zyklischen, der zeit- und alarmgesteuerten Programmbearbeitung auch die zusätzliche Bearbeitung von Reglern möglich. In vom Anwender festgelegten Zeitabschnitten (= Abtastzeit) wird die zyklische oder zeitgesteuerte Programmbearbeitung unterbrochen und der jeweilige Regler bearbeitet. Danach kehrt der Prozessor zur Unterbrechungsstelle im zyklischen oder zeitgesteuerten Programm zurück und setzt dort die Bearbeitung fort.

##### **Auslösung:**

Ein Regleralarm wird nach Ablauf der vom Anwender gewählten Abtastzeit ausgelöst.

##### **Leistungen des Systemprogramms:**

- Es verwaltet die Anwenderschnittstelle für die Reglerbearbeitung.
- Es aktualisiert das Reglerprozeßabbild.

##### **Anwenderschnittstelle: Standard-Funktionsbaustein "Reglerstruktur R64"**

Bei der Reglerbearbeitung wird als Anwenderschnittstelle der R64-Standard-Funktionsbaustein aufgerufen. Dieser ermöglicht in Zusammenhang mit dem Regler-Parametrierungs-Baustein DB 2 die Bearbeitung von bis zu 64 Reglern.

Für jeden Regler parametrieren Sie einen bestimmten Datenbaustein. Im Datenbaustein DB 2, der sogenannten "Reglerliste", legen Sie fest, welche Regler zu welchem Zeitpunkt vom Systemprogramm zu bearbeiten sind. Der DB 2 ist für diese Aufgabe reserviert!

(Bei der Parametrierung, der Inbetriebnahme und beim Test des R64-Standard-FBs werden Sie unterstützt durch ein spezielles Programmpaket: "COMREG". Bestell-Nr. für PG 685: 6ES5895-3SA11.)

##### **Unterbrechungsstellen**

Eine Reglerbearbeitung kann entweder an *Bausteingrenzen* (Voreinstellung) oder an *Befehlsgrenzen* (Programmierung DX 0) unterbrochen werden durch

- eine prozeßalarmgesteuerte Bearbeitung.

Sie kann an *Befehlsgrenzen* unterbrochen werden durch

- einen Programm- oder Gerätefehler.

#### 4.4.4 PROZESSALARM: Alarmgesteuerte Programmbearbeitung

Eine alarmgesteuerte Programmbearbeitung liegt vor, wenn ein S5-Bus-Signal einer interruptfähigen Digitaleingabebaugruppe (z.B. 6ES5 432-4UA11) oder einer entsprechend arbeitenden IP-Baugruppe den Prozessor veranlaßt, die Programmbearbeitung zu unterbrechen und einen spezifischen Programmteil zu bearbeiten. Nach der Bearbeitung dieses Programms kehrt der Prozessor zur Unterbrechungsstelle zurück und setzt dort die Bearbeitung fort.

##### Auslösung:

Der aktive Zustand einer Interruptleitung auf dem S5-Bus löst den Prozeßalarm aus. Abhängig vom Steckplatz ist jedem Prozessor jeweils eine der 7 Interruptleitungen zugeordnet (siehe Betriebsanleitung CPU 928).

##### Anwenderschnittstelle: OB 2

Als Anwenderschnittstelle wird bei Auftreten eines Prozeßalarms der OB 2 aufgerufen. Im OB 2 programmieren Sie ein spezifisches Programm, das im Falle eines Prozeßalarms bearbeitet werden soll.

Ist der OB 2 nicht programmiert, wird die Programmbearbeitung nicht unterbrochen. Es findet keine alarmgesteuerte Programmbearbeitung statt.

##### Unterbrechungsstellen

Eine prozeßalarmgesteuerte Programmbearbeitung kann nur unterbrochen werden durch

- einen Programm- oder Gerätefehler (*an Befehlsgrenzen*).

##### WICHTIG!

**Eine alarmgesteuerte Programmbearbeitung kann nicht durch eine zeitgesteuerte Programmbearbeitung oder eine erneute alarmgesteuerte Programmbearbeitung unterbrochen werden!**

Treten während der alarmgesteuerten Programmbearbeitung erneut Prozeßalarme auf, so werden diese solange ignoriert, bis der OB 2 vollständig bearbeitet ist (inkl. aller im OB 2 aufgerufenen Bausteine).

Dann kehrt der Prozessor an die Unterbrechungsstelle zurück und bearbeitet das Programm bis zur nächsten Bausteingrenze. Erst dann wird ein neuer Prozeßalarm akzeptiert und erneut der OB 2 aufgerufen. Dadurch wird auch bei einem Daueralarm das zyklische Programm bearbeitet. (Dies gilt nicht für den Fall, daß im DX 0 "Prozeßalarm unterbricht an Befehlsgrenzen" eingestellt ist).

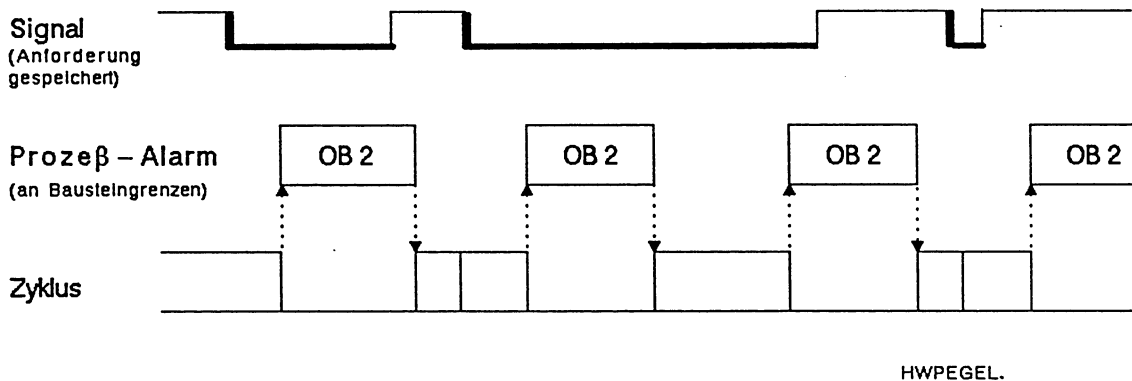
### Prozeßalarmsignal: Pegelgetriggert

Standardmäßig ist das Prozeßalarmsignal bei der CPU 928 pegelgetriggert. D.h.: Der aktive Zustand der Interruptleitung setzt eine Anforderung, die an der nächsten Baustein- oder Befehlsgrenze zur Bearbeitung des OB 2 führt. Diese Anforderung ist gespeichert und wird erst durch den Bausteinende-Befehl (BE) des OB 2 zurückgesetzt.

Daraus resultiert:

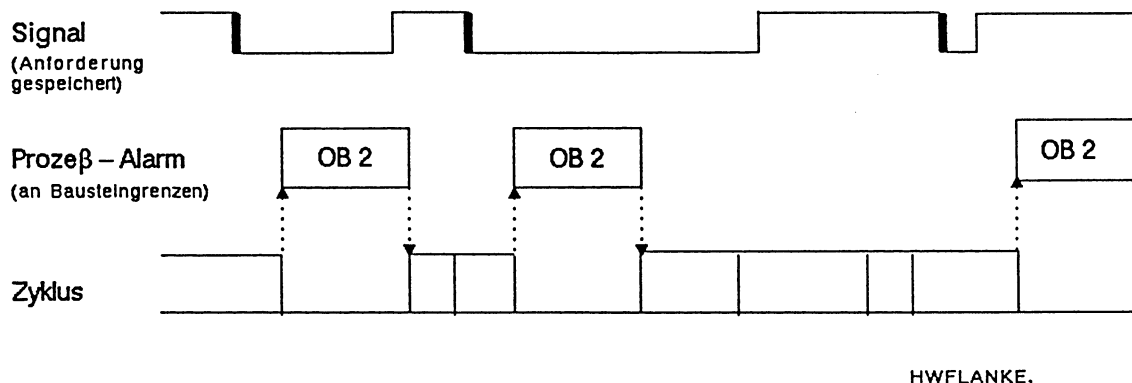
- Mehrfache Alarmer werden nicht erkannt.
- Alarmer, die während der Bearbeitung des OB 2 auftreten und kürzer sind als der OB 2, werden nicht erkannt.
- Der OB 2 wird auch dann aufgerufen, wenn bei Erreichen der Bausteingrenze der Signalzustand der Interruptleitung bereits wieder passiv ist (siehe Abbildung).

Der aufgerufene OB 2 wird vollständig bearbeitet. Steht am Ende des OB 2 der Pegel immer noch bzw. erneut an, wird im zyklischen Programm ein Baustein bearbeitet und anschließend wieder der OB 2 aufgerufen. Steht der Pegel nicht mehr an, wird erst beim nächsten Signalzustandswechsel (von inaktiv nach aktiv) erneut der OB 2 aufgerufen.



### Prozeßalarmsignal: Flankengetriggert

Diese Einstellung erreichen Sie über die Parametrierung des DX 0. Nach Abarbeitung des OB 2 kann ein neuer Prozeßalarm nur durch einen Signalzustandswechsel (von inaktiv nach aktiv) ausgelöst werden. Auch beim flankengetriggerten Prozeßalarm bleibt die Anforderung zur Bearbeitung eines OB 2 solange gespeichert, bis dieser vollständig bearbeitet ist. Signalzustandswechsel, die während einer OB 2-Bearbeitung auftreten, werden ignoriert.



## **Sperren der prozeßalarmgesteuerten Bearbeitung**

Ein alarmgesteuertes Programm wird an einer Bausteingrenze oder einer STEP5-Befehlsgrenze in das zyklische Programm eingeschoben.

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) oder eine Befehlsfolge nicht unterbrochen werden darf (z.B. beim Lesen oder Schreiben von zusammengehörenden Werten).

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

- Programmieren Sie diesen Programmteil so, daß er keinen Bausteinwechsel enthält und behalten Sie die Voreinstellung im DX 0 ("Prozeßalarme an Bausteingrenzen") bei. Programmteile, die keinen Bausteinwechsel enthalten, können dann auch nicht unterbrochen werden.
- Schreiben Sie das Programm selbst in ein alarmgesteuertes Programm. Hier kann es von keinem weiteren Alarm unterbrochen werden.
- Programmieren Sie den STEP5-Befehl 'AS' (Prozeßalarme sperren). Mit dem Befehl 'AF' (Prozeßalarme freigeben) geben Sie die Alarmbearbeitung wieder frei. Zwischen diesen beiden Befehlen wird keine alarmgesteuerte Programmbearbeitung durchgeführt, der dazwischen stehende Programmteil kann durch auftretende Prozeßalarme nicht unterbrochen werden.

'AS' und 'AF' sind nur in Funktionsbausteinen möglich (Ergänzender Operationsvorrat)!

- Verwenden Sie die neuen Sonderfunktionen OB 120 und OB 122, mit denen Sie die Bearbeitung von auftretenden Prozeßalarmen für einen bestimmten Programmteil sperren oder verzögern lassen können.

## **Priorisierung von alarm- und zeitgesteuerter Programmbearbeitung gegeneinander**

Wenn während einer zeitgesteuerten Programmbearbeitung ein Prozeßalarm auftritt, wird das Programm an der nächsten Unterbrechungsstelle (Baustein- oder Befehlsgrenze) unterbrochen und der Prozeßalarm bearbeitet. Danach wird die zeitgesteuerte Programmbearbeitung zu Ende geführt.

Wenn während der alarmgesteuerten Programmbearbeitung ein Weckalarm auftritt, wird zuerst die alarmgesteuerte Programmbearbeitung abgeschlossen. Erst dann wird die zeitgesteuerte Programmbearbeitung aufgenommen.

Wenn gleichzeitig ein Prozeßalarm und ein Weckalarm auftreten, dann wird an der nächsten Unterbrechungsstelle zuerst der Prozeßalarm bearbeitet. Erst wenn dieser abgearbeitet ist, wird der noch anstehende Weckalarm bearbeitet.



## Reaktionszeit

Die Reaktionszeit auf eine Weckalarmanforderung entspricht der Bearbeitungszeit eines Bausteins bzw. eines STEP5-Befehls (je nach gewählter Voreinstellung). Wenn jedoch zum Zeitpunkt der Unterbrechung der zyklischen Programmbearbeitung noch Prozeßalarme anstehen, wird das zeitgesteuerte Programm erst dann bearbeitet, nachdem alle anstehenden Prozeßalarme vollständig abgearbeitet sind.

Die maximale Reaktionszeit zwischen dem Auftreten und der Bearbeitung eines Weckalarms wächst in diesem Fall um die Bearbeitungszeit der Prozeßalarme. Wollen Sie für einen bestimmten Weckalarm-OB xy einen Weckfehler weitgehend ausschließen, beachten Sie folgende Regel:

$A + B < C$  wobei  $A$  = Summe der Bearbeitungszeiten aller höher-priorigen Programmbearbeitungsebenen (Prozeß-, Regler-, Weckalarm-OBs)  
 $B$  = Bearbeitungszeit des Weckalarm-OB xy  
 $C$  = Zeitraster des Weckalarm-OB xy

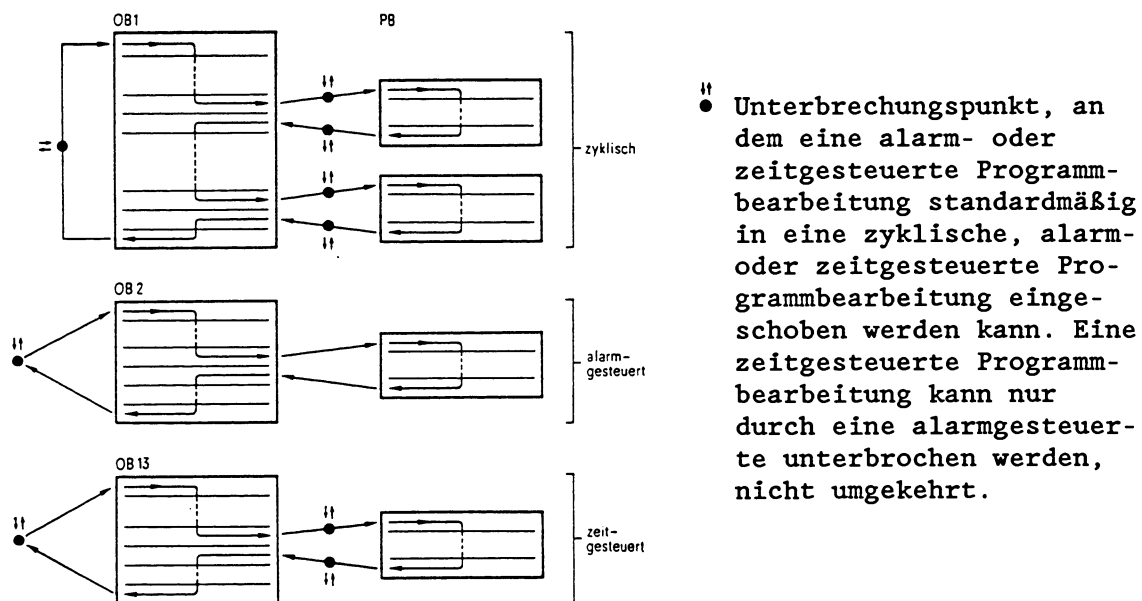


Abb. 4-2: Unterbrechungsgesteuerte Programmbearbeitung an Bausteingrenzen

### WICHTIG!

Wenn Sie Ihr Anwenderprogramm nicht nur zyklisch, sondern auch zeit- und/oder alarmgesteuert bearbeiten lassen, besteht die Gefahr, daß z.B. Merker, die im zyklischen Programm als Zwischenmerker verwendet werden, bei der Unterbrechung der zyklischen Programmbearbeitung durch eine eingeschobene zeit- oder alarmgesteuerte Bearbeitung überschrieben werden. Aus diesem Grund müssen Sie zu Beginn einer zeit- oder alarmgesteuerten Programmbearbeitung die Signalzustände der Merker in einen Datenbaustein "retten" und am Ende der unterbrechenden Bearbeitung wieder zurück in die Merker laden.

Zu diesem Zweck stehen Ihnen vier Sonderfunktions-Organisationsbausteine zur Verfügung: OB 190 und OB 192 "Merker in Datenbaustein übertragen" und OB 191 und 193 "Datenblöcke in Merkerbereich übertragen" (siehe dort).

## 5 Unterbrechungs- und Fehlerbehandlung

Das Systemprogramm kann fehlerhaftes Arbeiten des Prozessors, Fehler in der Systemprogrammbearbeitung oder Auswirkungen einer fehlerhaften Programmierung durch den Anwender feststellen.

### 5.1 Häufige Fehler im Anwenderprogramm

Die folgende Liste enthält eine Aufzählung von Fehlern, die bei der Inbetriebnahme des Anwenderprogramms am häufigsten auftreten, die jedoch schon bei der Erstellung des Programms leicht zu vermeiden sind.

Beachten Sie aus diesem Grund bei der Erstellung Ihres STEP5-Programms bitte folgende Punkte:

- Bei der Angabe von Byte-Adressen für Ein- und Ausgänge müssen für diese Adressen die entsprechenden Baugruppen im Zentralgerät oder Erweiterungsgerät stecken.
- Achten Sie darauf, daß alle Operanden mit den korrekten Parametern versorgt werden.
- Ausgänge, Merker, Zeiten und Zähler sollten nicht an mehreren Stellen im Programm mit entgegengesetzt wirkenden Operationen bearbeitet werden.
- Sorgen Sie dafür, daß alle im Programm aufgerufenen Datenbausteine vorhanden sind und ausreichend lang sind.
- Überprüfen Sie, ob alle aufgerufenen Bausteine auch tatsächlich im Speicher vorhanden sind.
- Vorsicht beim nachträglichen Ändern von Funktionsbausteinen. Kontrollieren Sie, ob die FBs mit den richtigen Operanden parametrisiert sind und ob alle Aktualoperanden angegeben sind.
- Zeiten sollten nur einmal im Zyklus abgefragt werden (z.B. U T1).

## 5.2 Auswertung von Fehlerinformationen

Wenn im Anlauf oder bei der zyklischen Bearbeitung des Anwenderprogramms ein Fehler auftritt, haben Sie verschiedene "Informationsquellen" zur Verfügung, um diesem Fehler auf die Spur zu kommen.

### a) LEDs auf der Frontplatte des Prozessors

Orientieren Sie sich im Falle eines unerwünschten Stoppzustandes an den Leuchtdioden. Sie können Ihnen Hinweise auf die Fehlerursache geben:

- STOP-LED zeigt Dauerlicht
- STOP-LED blinkt langsam
- STOP-LED blinkt schnell

Die unterschiedlichen Erscheinungsbilder der STOP-LED deuten auf bestimmte Unterbrechungs- und Fehlerursachen hin.

Beachten Sie hierzu die Ausführungen im Kapitel 4.2 "Betriebszustand STOP".

Die **Fehler-LEDs** auf der Frontplatte zeigen Dauerlicht bei

- ADF (Adressierfehler)
- QVZ (Quittungsverzug)
- ZYK (Zykluszeitfehler).

### b) On-line-Funktion "Ausgabe USTACK" (siehe Kapitel 5.3)

Über die On-line-Funktionen "AG INFO" und anschließend "Ausgabe USTACK" erhalten Sie Auskunft über die Zustände der Steuerbits und den Inhalt des Unterbrechungsstacks (= USTACK).

In den **USTACK** trägt das Systemprogramm beim Übergang in den Stoppzustand alle Informationen ein, die es für einen Wiederanlauf benötigt. Diese Einträge sind bei der Fehlerdiagnose eine wertvolle Hilfe.

Der vollständige USTACK läßt sich nur im Stoppzustand ausgeben.

Vor der Ausgabe des eigentlichen USTACK werden zuerst die Zustände der **Steuerbits** angezeigt. Diese markieren den aktuellen Betriebszustand und bestimmte Eigenschaften des Prozessors und des Anwenderprogramms und geben zusätzliche Hinweise auf die Fehlerursache.

Die On-line-Funktion "Ausgabe USTACK" lässt sich nicht nur im STOP, sondern auch in den Betriebszuständen ANLAUF und RUN auslösen. Sie erhalten in diesem Fall jedoch nur eine Ausgabe der Steuerbits.

**c) Systemdatum BS 3 und 4 (siehe Kapitel 5.5)**

Wenn Ihr Prozessor schon beim Anlauf infolge eines Fehlers zurück in den Stoppzustand geht, so wird in den Systemdatenwörtern BS 3 und BS 4 die Fehlerursache genauer definiert. In diesem Fall handelt es sich um Fehler, auf die das Systemprogramm beim Aufbau der Adreßlisten im DB 0 oder bei der Auswertung des DB 1, DB 2 oder DX 0 stößt.

Systemdatenwort BS 3: KH = EA03 (absolute Speicheradresse)

Systemdatenwort BS 4: KH = EA04 (absolute Speicheradresse)

Über die Fehlerkennung im Systemdatenwort BS 3 finden Sie heraus, was für ein Fehler aufgetreten ist.

Über die Fehlerkennung im Systemdatenwort BS 4 finden Sie heraus, wo der Fehler aufgetreten ist.

Die Fehlerkennungen sind im Datenformat KH eingetragen.

**Auswertung von Systemdatenwort BS 3 und BS 4 mit dem Programmiergerät:**

- Mit der On-line-Funktion "Auskunft ADRESSE" (KH = EA03 bzw. EA04) können Sie den Inhalt der beiden Systemdatenwörter direkt auslesen und so die Fehlerursache ermitteln.

**d) Akku 1 und Akku 2 (siehe Kapitel 5.6)**

Treten bei der STEP5-Programmbearbeitung im Anlauf oder im Zyklus Fehler auf, für die als Anwenderschnittstelle ein bestimmter Organisationsbaustein vorhanden ist, so hinterlegt das Systemprogramm automatisch beim Aufruf des jeweiligen Organisationsbausteins in den Akkumulatoren Akku 1 und Akku 2 zusätzliche Fehlerinformationen, die die Fehlerursache näher erläutern.

Über die Fehlerkennung im Akku 1 finden Sie heraus, was für ein Fehler aufgetreten ist.

Über die Fehlerkennung im Akku 2 (falls vorhanden) finden Sie heraus, wo der Fehler aufgetreten ist.

Die Fehlerkennungen sind im Datenformat KH eingetragen.

#### **Auswertung von Akku 1 und 2 mit dem Programmiergerät:**

- Mit der On-line-Funktion "Ausgabe USTACK" können Sie den Inhalt der beiden Akkumulatoren direkt aus dem USTACK lesen und so die genaue Fehlerursache ermitteln.

#### **Auswertung von Akku 1 und 2 mit STEP5:**

- Da die Fehlerkennungen automatisch beim Aufruf eines Fehler-Organisationsbausteins im Akku 1 und 2 abgelegt werden, können Sie diese Kennungen bei der Programmierung Ihres Fehler-OBs berücksichtigen.

Es ist somit möglich, in einem Organisationsbaustein unterschiedliche Reaktionen auf verschiedene Fehler vorzusehen, in Abhängigkeit von der dort übergebenen Fehlerkennung.

#### **e) On-line-Funktion "Ausgabe BSTACK"**

Über die On-line-Funktionen "AG INFO" und anschließend "Ausgabe BSTACK" können Sie sich nach einem aufgetretenen Fehler im Stoppzustand den Inhalt des Bausteinstacks (= BSTACK) ausgeben lassen (siehe Kapitel 3.1.1).

Im BSTACK sind, ausgehend vom OB 1 bzw. FB 0, alle Bausteine aufgeführt, die nacheinander bis zum Übergang in den Stoppzustand aufgerufen und noch nicht zuendebearbeitet worden sind. Da der BSTACK von unten gefüllt wird, steht in der obersten Zeile der BSTACK-Ausgabe derjenige Baustein, der als letzter bearbeitet wurde und in dem der Fehler aufgetreten ist.

Bei der Auswertung der obersten Zeile erhalten Sie folgende Informationen:

BAUST.-NR.	Bausteinart und -nummer des vor Erreichen des Stoppzustandes bearbeiteten Bausteins
BAUST.-ADR.	absolute Anfangsadresse dieses Bausteins im Programmspeicher
RÜCKSPR.-ADR.	Absolutadresse des nächsten zu bearbeitenden Befehls in diesem Baustein. Mit diesem Befehl setzt der Prozessor nach der Bedienung "Manueller Wiederanlauf" das Programm fort.
REL.-ADR.	Relativadresse (= RÜCKSPR.-ADR. - BAUST.-ADR.) des nächsten zu bearbeitenden Befehls in diesem Baustein  (Relativadressen können vom PG in Betriebsart "Eingabesperre" (Schlüsselschalter) angezeigt werden.)

DB-NR.	Nummer des zuletzt aufgeschlagenen Datenbausteins
DB-ADR.	absolute Anfangsadresse dieses Datenbausteins (Adresse des Datenwortes DW 0) im Programmspeicher

**Beispiel: "AUSGABE BSTACK" auswerten**

BAUST.-NR.	BAUST.-ADR.	RUECKSPR.-ADR.	REL.-ADR	DB-NR.	DB-ADR.
OB 23	0063	0064	0001	13	0078
FB 5	006A	0072	0008	13	0078
FB 6	008A	0091	0007	100	0098
OB 1	009D	009E	0001		

Im obigen Beispiel ist der Stoppzustand im OB 23 bei der Bearbeitung derjenigen STEP5-Anweisung aufgetreten, die im Speicher unter der Absolutadresse '0064 - 1 = 0063' angeordnet ist.

Der OB 23 (QVZ-Fehler-OB) ist im FB 5 an der relativen Adresse '0008 - 1 = 0007' aufgerufen worden.

Im FB 6 ist der Datenbaustein DB 100 aufgeschlagen worden. Beim Übergang des Prozessors in den Stoppzustand war Datenbaustein DB 13 gültig.

Der Datenbaustein DB 13 wurde im FB 5 aufgeschlagen.

## **Zusammenfassung**

Machen Sie sich auf der Suche nach der Fehlerursache alle Informationen zunutze, die Ihnen zur Verfügung stehen.

Dies können sein:

### **1. LEDs auf der Frontplatte des Prozessors**

Bestimmte Erscheinungsbilder weisen auf bestimmte Fehler- oder Unterbrechungsursachen hin.

### **2. On-line-Funktion "Ausgabe USTACK":**

Sie erhalten in jedem Fall die Ausgabe der Steuerbits, im Stoppzustand die Ausgabe des USTACKs.

### **3. Systemdatenwörter BS 3 und 4:**

Bei Fehlern im Anlauf finden Sie in Systemdatenwort BS 3 und 4 genauere Hinweise auf die Fehlerursache.

### **4. Akku 1 und Akku 2:**

Beim Aufruf der Fehler-Organisationsbausteine hinterlegt das Systemprogramm in Akku 1 und Akku 2 zusätzliche Fehlerinformationen.

### **5. On-line-Funktion "Ausgabe BSTACK":**

Im Stoppzustand können Sie in der obersten Zeile des BSTACKs den Baustein und darin die Adresse des Befehls ermitteln, bei dessen Bearbeitung der Fehler aufgetreten ist.

### 5.3 Steuerbits und Unterbrechungsstack

Über die On-line-Funktionen "AG-INFO" und dann "Ausgabe USTACK" können Sie sowohl Betriebszustand, Eigenschaften des Prozessors und des Anwenderprogramms als auch eventuelle Fehler- und Unterbrechungsursachen analysieren.

#### WICHTIG!

Die Ausgabe der Steuerbits erhalten Sie in jedem Betriebszustand, die Ausgabe des USTACKs nur im Stop.

- Die **Steuerbits** geben den aktuellen bzw. vorausgegangenen Betriebszustand und die Störungsursache an.  
Sind mehrere Fehler aufgetreten, werden in den Steuerbits alle aufgetretenen Fehler dargestellt.
- Im **USTACK** wird die jeweilige Unterbrechungsstelle (Adressen) mit den dort aktuellen Anzeigen und Akkuinhalten sowie die Störungsursache angegeben.  
Sind mehrere Unterbrechungen aufgetreten, so wird ein mehrstufiger Unterbrechungsstack aufgebaut:  
Tiefe 01 = letzte Unterbrechungsursache,  
Tiefe 02 = vorletzte Unterbrechungsursache etc..

Bei USTACK-Überlauf erfolgt ein sofortiger Übergang in den Stoppzustand. Anschließend ist ein Neustart erforderlich.

Die Bedeutung der einzelnen Abkürzungen in den Steuerbits und im Unterbrechungsstack wird nachfolgend erläutert. Sie erhalten diese Abkürzungen am PG 685.

#### WICHTIG!

Der Text am Bildschirm Ihres Programmiergerätes kann von dem hier abgedruckten abweichen. Trotzdem ist die Beschreibung der einzelnen *Positionen* auf dem Bildschirm in dieser Programmieranleitung **gültig!**

---

#### STEUERBITS

>>STP<<	STP-6	FE-STP	BARBEND	PG-STP	STP-SCH	STP-BEF	MP-STP
>>ANL<<	ANL-6	NEUST X	M W A	A W A	ANL-2	NEU-ZUL X	MWA-ZUL X
>>RUN<< X	RUN-6	EINPROZ X	BARB	OB1GEL	FBOGEL X	OBPROZA	OBWECKA
32KWRAM	16KWRAM	8KWRAM X	EPROM	KM-AUS	KM-EIN	DIG-EIN X	DIG-AUS X
URGLOE	URL-IA	STP-VER	ANL-ABB	UA-PG	UA-SYS	UA-PRFE	UA-SCH
DX0-FE	FE-22	MOD-FE	RAM-FE	DB0-FE	DB1-FE	DB2-FE	KOR-FE
N A U	P E U	B A U	STUE-FE	Z Y K	Q V Z	A D F	WECK-FE
B C F	FE-6	FE-5	FE-4	FE-3	L Z F	REG-FE	DOPP-FE

---



Bei der PG-Ausgabe des USTACKs werden auf der 1. Bildschirmseite die Zustände der Steuerbits angezeigt.

Die folgenden Steuerbits markieren den aktuellen bzw. vorausgegangenen Betriebszustand des Prozessors und geben Auskunft über bestimmte Eigenschaften des Prozessors und des STEP5-Anwenderprogramms.

*Die Steuerbits lassen sich in allen Betriebszuständen ausgeben!*  
So können Sie sich z.B. jederzeit vergewissern, ob der Organisationsbaustein OB 2 geladen und somit eine alarmgesteuerte Programmbearbeitung möglich ist.

STP    Prozessor ist im Betriebszustand STOP; die folgenden Steuerbits geben die Ursache für den Stoppzustand an:

- STP-6    nicht belegt
- FE-STP   Fehler-Stop: Stoppzustand nach NAU (Netzausfall), PEU (Peripherie unklar), BAU (Batterie unklar), STUEB (BSTACK-Überlauf), STUEU (USTACK-Überlauf), DOPP (Doppelfehler) oder Prozessorfehler
- BARBEND   Bearbeitungskontrolle-Ende: Stoppzustand nach On-line-Funktion "Bearbeitungskontrolle-Ende" (Neustart erforderlich)
- PG-STP   PG-Stop: Stoppzustand durch Befehl vom PG
- STP-SCH   Stop-Schalter: Stoppzustand durch Stoppschalter in Stellung STOP
- STP-BEF   Stop-Befehl:
  - a) Stoppzustand nach Bearbeitung der STEP5-Operation 'STP'
  - b) Stoppzustand nach Stoppbefehl vom Systemprogramm, wenn Fehler-Organisationsbaustein nicht programmiert ist.
- MP-STP   Mehrprozessor-Stop:
  - a) Wahlschalter am KOR in Stellung STOP oder
  - b) Stop eines anderen Prozessors im Mehrprozessorbetrieb

ANL    Prozessor ist im Betriebszustand ANLAUF:

- ANL-6    nicht belegt
- NEUST    Neustart ist angefordert oder aktiv oder wurde als letzter Anlauf durchgeführt.
- M W A    Manueller Wiederanlauf ist angefordert oder aktiv oder wurde als letzter Anlauf durchgeführt.
- A W A    Automatischer Wiederanlauf nach Netzspannungsausfall ist angefordert oder aktiv oder wurde als letzter Anlauf durchgeführt.

ANL-2      nicht belegt

NEU-ZUL    Neustart als nächster Anlauf zulässig

MWA-ZUL    Manueller Wiederanlauf als nächster Anlauf zulässig

RUN      Prozessor ist im Betriebszustand RUN (zyklische Programm-  
bearbeitung ist aktiv):

RUN-6      nicht belegt

EINPROZ    Einzelprozessorbetrieb

BARB      On-line-Funktion "Bearbeitungskontrolle" ist aktiv.

OB1GEL    Organisationsbaustein OB 1 ist in den Anwenderspeicher  
geladen worden.  
Die zyklische Programmbearbeitung wird durch den OB 1  
bestimmt.

FBOGEL    Funktionsbaustein FB 0 ist in den Anwenderspeicher  
geladen worden.  
Die zyklische Programmbearbeitung wird durch den FB 0  
bestimmt, wenn kein OB 1 geladen ist. Wenn FB 0 und  
OB 1 geladen sind, gilt der OB 1 für die zyklische  
Programmbearbeitung.

OBPROZA    Prozeßalarm-Organisationsbaustein OB 2 geladen, d.h.  
prozeßalarmgesteuerte Programmbearbeitung möglich

OBWECKA    Weckalarm-Organisationsbaustein geladen, d.h. zeit-  
gesteuerte Programmbearbeitung möglich

32KWRAM    Anwenderspeichermodule ist ein RAM mit  $32 \times 2^{10}$  Wörtern.

16KWRAM    Anwenderspeichermodule ist ein RAM mit  $16 \times 2^{10}$  Wörtern.

8KWRAM    Anwenderspeichermodule ist ein RAM mit  $8 \times 2^{10}$  Wörtern.

EPROM      Anwenderspeichermodule ist ein EPROM.

KM-AUS    Adreßliste für KoppelmerkerAusgänge aus DB 1 vorhanden

KM-EIN    Adreßliste für Koppelmerkereingänge aus DB 1 vorhanden

DIG-EIN    Adreßliste für digitale Eingänge vorhanden

DIG-AUS    Adreßliste für digitale Ausgänge vorhanden

URGELOE    Prozessor wurde urgelöscht (Neustart erforderlich).

URL-IA    Prozessor wird momentan urgelöscht.

STP-VER    Prozessor hat Stoppzustand der Zentralbaugruppe verursacht.

ANL-ABB	Abbruch während des Anlaufs (Neustart erforderlich)
UA-PG	PG hat Urlöschen angefordert.
UA-SYS	Systemprogramm hat Urlöschen angefordert (kein Anlauf möglich); Urlöschen muß durchgeführt werden.
UA-PRFE	Urlöschanforderung wegen Prozessorfehler
UA-SCH	Urlöschvoranforderung durch Schalterbedienung: Urlöschen durchführen bzw. Wahl einer Anlaufart, wenn ange- fordertes Urlöschen nicht durchgeführt werden soll.

Die folgenden Steuerbits markieren Fehler, die in den Betriebszu-  
ständen ANLAUF (z.B. beim ersten Neustart) und RUN (z.B. bei der  
zeitgesteuerten Programmbearbeitung) auftreten können.

Sind mehrere Fehler aufgetreten, werden in den letzten drei  
Zeilen der Steuerbits alle bisher aufgetretenen (und noch nicht  
bearbeiteten!) Unterbrechungsursachen angezeigt. Beachten Sie  
hierzu das Systemdatum BS 2: Es beinhaltet das UAMK (Unter-  
brechungsanzeigen-Sammelwort, 16 Bit), in dem ebenfalls alle  
aufgetretenen und noch nicht bearbeiteten Fehler eingetragen  
sind (Kapitel 8.2.4).

#### Fehler im ANLAUF:

DX0-FE	Parametrierfehler im DX 0
FE-22	nicht belegt
MOD-FE	Inhalt des Anwendermoduls ist fehlerhaft (bei RAM ist Ur- löschen erforderlich).
RAM-FE	Inhalt des Betriebssystem-RAMs oder des DB-RAMs ist fehler- haft (Urlöschen erforderlich).
DB0-FE	Aufbau der Baustein-Adreßlisten im DB0 ist fehlerhaft
DB1-FE	Aufbau der Adreßlisten im DB 1 für Prozeßabbild-Aktuali- sierung ist fehlerhaft: <ul style="list-style-type: none"> <li>a) DB 1 bei gestecktem Koordinator oder bei Mehrprozessorbe- trieb nicht programmiert;</li> <li>b) im DB 1 angegebene Byteadressen für Ein- und Ausgänge bzw. Koppelmerker quittieren nicht bei Neustart auf den entsprechenden Baugruppen.</li> </ul>
DB2-FE	Fehler bei der Auswertung des Parametrierungs-Datenbausteins DB 2 der Reglerstruktur R64

#### Fehler im ANLAUF oder im RUN:

KOR-FE	Fehler beim Datenaustausch mit dem Koordinator
NAU	Netzspannungsausfall im Zentralgerät

PEU        Peripherie unklar = Spannungsausfall bei Erweiterungsgerät

BAU        Batterie ist fehlerhaft = Ausfall der Pufferbatterie (Zentralgerät)

STUE-FE   Unterbrechungs- oder Bausteinstack übergelaufen (Schachtelungstiefe zu groß; Neustart erforderlich)

ZYK        Zykluszeit überschritten

QVZ        Quittungsverzug beim Datenaustausch mit Peripherie

ADF        Adressierfehler bei Eingängen oder Ausgängen:

Fehler hervorgerufen durch Zugriff auf das Prozeßabbild, wobei Peripheriebaugruppen angesprochen wurden, die beim letzten Neustart nicht gesteckt, defekt oder nicht im DB 1 angegeben waren.

WECK-FE   Weckfehler:

Vor oder während der Bearbeitung eines bestimmten Weckalarm-OBs ist dieser ein zweites Mal aufgerufen worden.

BCF        Befehlscodefehler:

a) Substitutionsfehler: Bearbeiteter STEP5-Befehl ist nicht substituierbar

b) Operationscodefehler: Bearbeiteter STEP5-Befehl ist falsch

c) Parameterfehler: Parameter des bearbeiteten STEP5-Befehls ist falsch

FE-6       nicht belegt

FE-5       nicht belegt

FE-4       Power-down-Fehler:

Bearbeitung eines vorausgegangenen Netzausfalls (NAU) durch das Systemprogramm ist fehlerhaft abgelaufen; der Wiederanlauf ist deswegen gesperrt.

FE-3       nicht belegt

LZF        Laufzeitfehler:

a) Aufgerufener Baustein nicht geladen

b) Transferfehler bei Datenbausteinen

c) Sonstige Laufzeitfehler

REG-FE    Fehler bei der Bearbeitung der Reglerstruktur R64 im Zyklus

DOPP-FE   Doppelfehler:

Eine noch aktive Fehlerprogrammverarbeitungsebene (ADF, BCF, LZF, QVZ, REG, ZYK) wird ein zweites Mal aktiviert (Neustart erforderlich).

Nach der Ausgabe der Steuerbits auf dem PG-Bildschirm und Betätigen der Übernahmetaste erscheint auf der 2. Bildschirmseite der USTACK. Das Systemprogramm trägt hier beim Übergang in den Stoppzustand alle Informationen ein, die es für einen Neustart oder Wiederanlauf benötigt.

# **WICHTIG!**

**Der Text am Bildschirm Ihres Programmiergerätes kann von dem hier abgedruckten abweichen. Trotzdem ist die Beschreibung der einzelnen Positionen des Bildschirms in dieser Programmieranleitung gültig!**

---

## **USTACK**

TIEFE: 02

BEF-REG: C70A	SAZ: 00F3	DB-ADR: 0000	BA-ADR: 0000
BST-STP: 0000	FB-NR.: 226	DB-NR.: -NR.:	
	REL-SAZ: 0006	DBL-REG: 0000	
EBENE: 0004	UAMK: 0100	UALW: 0000	
AKKU1: 0000 C464	AKKU2: 0000 00FF	AKKU3: 0000 0000	AKKU4: 0000 0000
KLAMMERN: KE1 111	KE2 100	KE3 111	
ERGEBNISANZEIGE:	ANZ1 ANZ0 OVFL OVFLS ODER STATUS VKE ERAB		
	X	X	X
STOERUNGSURSACHE:	NAU PEU BAU MPSTP ZYK QVZ ADF STP		
		X	X
	BCF S-6 LZF REG STUEB STUEU WECK DOPP		

---

Die folgenden USTACK-Kennungen enthalten Angaben über die Fehlerstelle, mit denen im Anwenderprogramm die Anweisung gefunden werden kann, bei deren Bearbeitung der Prozessor in Stop gegangen ist.

**TIEFE** Stufe des USTACKS bei Fehlerschachtelung

TIEFE 01 = zuletzt aufgetretene Unterbrechungsursache  
TIEFE 02 = vorletzte aufgetretene Unterbrechungsursache  
.....

**BEF-REG** Befehlsregister:  
Enthält Maschinencode (erstes Wort) des zuletzt bearbeiteten Befehls einer unterbrochenen Programmbearbeitungsebene

**BST-STP** Bausteinstack-Pointer:  
Enthält die Anzahl der im Bausteinstack (BSTACK) eingetragenen Elemente

EBENE Z      Gibt Ebene der Programmbearbeitung an, die unterbrochen worden ist:

Z :    0002 = Neustart  
      0004 = Zyklus  
      0006 = Weckalarm 5 sec (OB 18)  
      0008 = Weckalarm 2 sec (OB 17)  
      000A = Weckalarm 1 sec (OB 16)  
      000C = Weckalarm 500 ms (OB 15)  
      000E = Weckalarm 200 ms (OB 14)  
      0010 = Weckalarm 100 ms (OB 13)  
      0012 = Weckalarm 50 ms (OB 12)  
      0014 = Weckalarm 20 ms (OB 11)  
      0016 = Weckalarm 10 ms (OB 10)  
      0018 = nicht belegt  
      001A = nicht belegt  
      001C = Reglerbearbeitung  
      001E = nicht belegt  
      0020 = nicht belegt  
      0022 = nicht belegt  
      0024 = Prozeßalarm  
      0026 = nicht belegt  
      0028 = nicht belegt  
      002A = nicht belegt  
      002C = Übergang in den Stoppzustand bei Stop im  
              Mehrprozessorbetrieb, Stoppschalter oder  
              PG-Stop  
      002E = nicht belegt  
      0030 = Weckfehler  
      0032 = Reglerfehler  
      0034 = Zyklusfehler  
      0036 = nicht belegt  
      0038 = Befehlscodefehler  
      003A = Laufzeitfehler  
      003C = Adressierfehler  
      003E = Quittungsverzug  
      0040 = nicht belegt  
      0042 = nicht belegt  
      0044 = Manueller Wiederanlauf  
      0046 = Automatischer Wiederanlauf

SAZ            STEP-Adreßzähler:

Enthält die Absolutadresse des zuletzt bearbeiteten Befehls einer unterbrochenen Programmbearbeitungsebene im Programmspeicher.  
Bei Fehler zeigt der SAZ genau auf den fehlerverursachenden Befehl!

Liegt der Fehler nicht im STEP5-Anwenderprogramm, so steht der SAZ auf '0', der Inhalt von BEF-REG ist irrelevant.

...NR.        Bausteinart und -nummer des zuletzt bearbeiteten Bausteins

REL-SAZ       Relativer Step-Adreßzähler:  
Enthält die Relativadresse (bezogen auf die Bausteinanfangsadresse) des zuletzt bearbeiteten Befehls im zuletzt bearbeiteten Baustein.

(Relativadressen können vom PG in Betriebsart "Eingabesperre" (Schlüsselschalter) angezeigt werden oder bei der Ausgabe des Bausteins auf den Drucker.)

UAMK	Unterbrechungsanzeigen-Sammelwort: Im UAMK sind alle bisher aufgetretenen und noch nicht zuendebearbeiteten Unterbrechungsursachen angezeigt (siehe "Systemdatenbelegung", Kapitel 8.2.4)
UALW	Unterbrechungsanzeigen-Löschwort (siehe "Systemdatenbelegung", Kapitel 8.2.4)
DB-ADR	Absolute Anfangsadresse des zuletzt aufgeschlagenen Datenbausteins im Programmspeicher (DW 0)  (DB-ADR = 0000, wenn kein DB aufgeschlagen wurde)
DB-NR.	Nummer des zuletzt aufgeschlagenen Datenbausteins
DBL-REG	Länge des zuletzt aufgeschlagenen Datenbausteins
BA-ADR	Absolutadresse im Programmspeicher für den nächsten zu bearbeitenden Befehl im zuletzt aufrufenden Baustein
...Nr.	Bausteinart und -nummer des zuletzt aufrufenden Bausteins
AKKU1...4	Inhalt der Rechenregister zum Unterbrechungszeitpunkt  In bestimmten Fehlerfällen werden vom Systemprogramm bei Unterbrechungen in den Akkus 1 und 2 Fehlerkennungen hinterlegt, die die Unterbrechungsursachen näher erläutern.
KLAMMERN	Anzahl der Klammerebenen: 'KEx abc' x = 1 bis 7 Ebenen  a = OR (Oder, siehe Bitanzeigen) b = VKE (Verknüpfungsergebnis, siehe Bitanzeigen) c = 1: U( c = 0: O(
ERGEBNIS- ANZEIGE:	siehe Kapitel 3.2

Die folgenden Abkürzungen stellen die wichtigsten Störungsursachen dar. Es sind nur diejenigen Unterbrechungsursachen angekreuzt, die in der gerade angezeigten Programmbearbeitungsebene (siehe EBENE!) aufgetreten sind.

Bei den Angaben der Störungsursachen handelt es sich um die Wiedergabe des Unterbrechungsanzeigensammelwortes (UAMK, 16 Bits; siehe Kapitel 8.2.4). Teilweise sind die Angaben hier mit denen der Steuerbits identisch.

NAU	Netzspannungsausfall im Zentralgerät
PEU	Peripherie unklar = Spannungsausfall bei Erweiterungsgerät

BAU	Batterie unklar = Ausfall der Pufferbatterie (Zentralgerät)
MPSTP	Mehrprozessor-Stopp: a) Wahlschalter am KOR in Stellung STOP oder b) Stop eines anderen Prozessors im Mehrprozessorbetrieb
ZYK	Zykluszeit überschritten
QVZ	Quittungsverzug beim Datenaustausch mit Peripherie
ADF	Adressierfehler bei Eingängen und Ausgängen
STP	Stoppzustand durch Stoppschalter in Stellung STOP Stoppzustand durch Befehl vom PG Stoppzustand nach Bearbeitung der STEP5-Operation 'STP' Stoppzustand nach Stoppbefehl vom Systemprogramm, wenn Fehler-Organisationsbaustein nicht programmiert ist.
BCF	Befehlscodefehler: Fehler, die während der Befehlsdekodierung erkannt werden  a) Substitutionsfehler: Bearbeiteter STEP5-Befehl ist nicht substituierbar.  b) Operationscodefehler: Bearbeiteter STEP5-Befehl ist falsch.  c) Parameterfehler: Parameter des bearbeiteten STEP5-Befehls ist falsch.
S-6	nicht belegt
LZF	Laufzeitfehler: Fehler, die während der Befehlsausführung erkannt werden  a) Aufgerufener Baustein nicht geladen  b) Transferfehler bei Datenbausteinen  c) Sonstige Laufzeitfehler
REG	Fehler bei der Bearbeitung der Reglerstruktur R64 im Zyklus
STUEB	Bausteinstack übergelaufen (Schachtelungstiefe zu groß; Neustart erforderlich)
STUEU	Unterbrechungsstack übergelaufen (Schachtelungstiefe zu groß; Neustart erforderlich)
WECK	Weckfehler: Vor oder während der Bearbeitung eines Weckalarm-OBs wird derselbe OB zum zweiten Mal aufgerufen
DOPP	Doppelfehler: Eine noch aktive Fehlerprogrammbearbeitungsebene (ADF, BCF, LZF, QVZ, REG, ZYK) wird ein zweites Mal aktiviert (Neustart erforderlich).



## Beispiele für die Auswertung des USTACK

Die folgende Abbildung zeigt Ihnen den Aufbau des USTACK in Zusammenhang mit den aufgetretenen Unterbrechungen.

1. Die Programmbearbeitungsebene "ZYKLUS" (OB 1) wird unterbrochen durch das Auftreten eines Weckalarms (100ms-Zeitraster).
2. Daraufhin wird die Programmbearbeitungsebene "WECKALARM" aktiviert und der OB 13 bearbeitet.
3. Durch das Auftreten eines Prozessalarms wird die Ebene "WECKALARM" verlassen, die Ebene "PROZESSALARM" aktiviert und der OB 2 bearbeitet.
4. Ein falscher Adressierbefehl führt dazu, daß die Ebene "ADF" aktiviert und dort der OB 25 bearbeitet wird. In seinem Fehlerbehandlungsprogramm hat der Anwender einen Stoppbefehl (STP) programmiert: Der Prozessor bricht die Programmbearbeitung ab.

Vor dem endgültigen Übergang in den Stoppzustand sind insgesamt 4 verschiedene Programmbearbeitungsebenen unterbrochen worden. Wenn Sie sich nun am PG den USTACK ausgeben lassen, bekommen Sie entsprechend einen vierstufigen USTACK, zuoberst der USTACK mit der Tiefe 01, in dem die Kennung der zuletzt unterbrochenen Programmbearbeitungsebene (= ADF) vermerkt ist. Sie können den USTACK nun 'hinunterschalten' bis zum USTACK mit der Tiefe 04, der die Programmbearbeitungsebene ZYKLUS repräsentiert, die als erste unterbrochen wurde.

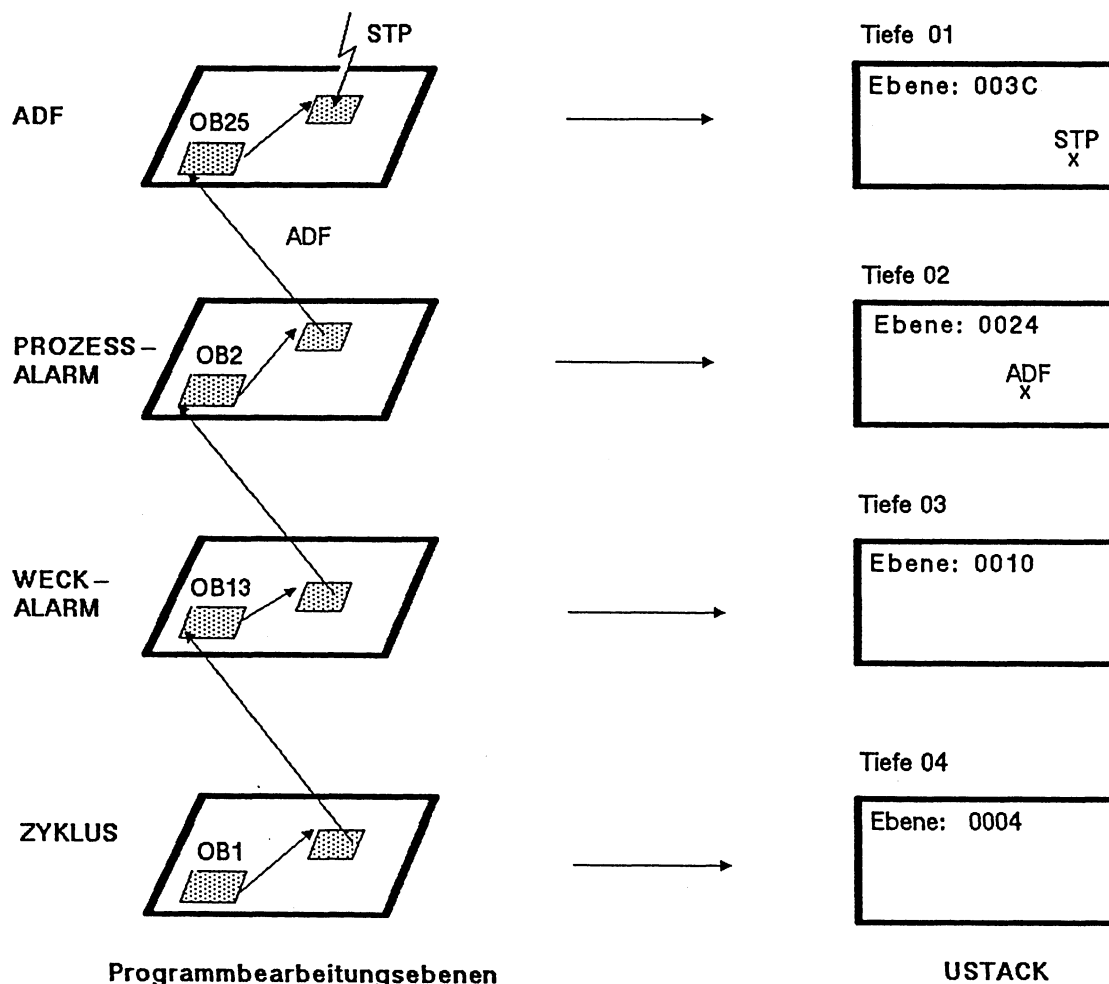
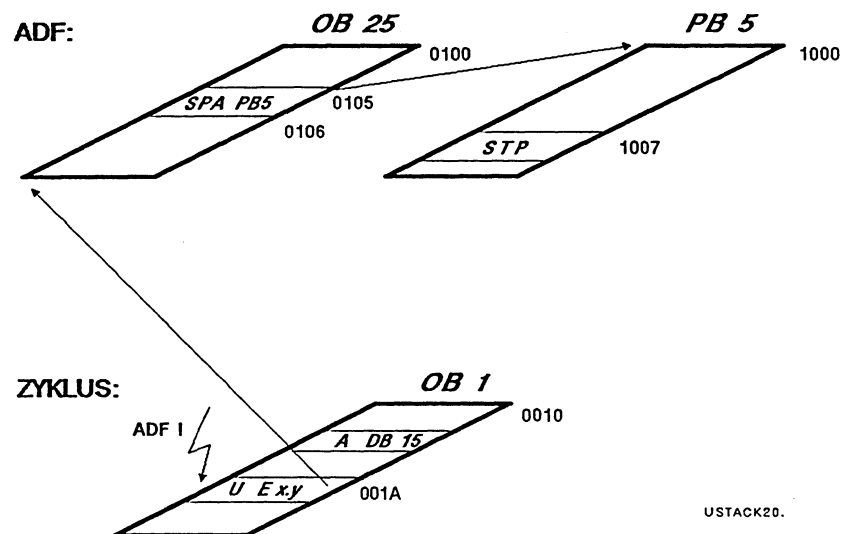


Abb. 5-1: Beispiel für den USTACK-Aufbau

Im folgenden Beispiel erkennt der Prozessor bei der Ausführung des Befehls 'U Ex.y' im OB 1 einen Adressierfehler. Dies führt zur Bearbeitung des OB 25. Aufgrund eines STP-Befehls im PB 5 geht der Prozessor in den Stoppzustand über.

Zwei unterbrochene Programmbearbeitungsebenen führen zum Aufbau eines zweistufigen USTACK:



#### USTACK

TIEFE: 02

BEF-REG: **U Ex.y** SAZ: 001A DB-ADR: BA-ADR: 0000

BST-STP: 1 OB-NR.: 1 DB-NR.: 16 -NR.:

REL-SAZ: 000A DBL-REG:

EBENE: 0004 UAMK: 0200 UALW: 0000

AKKU1:

ERGEBNISANZEIGE: .....

STOERUNGSURSACHE: ADF  
x

#### USTACK

TIEFE: 01

BEF-REG: **STP** SAZ: 1007 DB-ADR: BA-ADR: 0106

BST-STP: 3 PB-NR.: 5 DB-NR.: 16 OB-NR.: 25

REL-SAZ: 0007 DBL-REG:

EBENE: 003C UAMK: 0300 UALW: 0000

AKKU1:

ERGEBNISANZEIGE: .....

STOERUNGSURSACHE: STP  
x

## 5.4 Fehlerbehandlung über Organisationsbausteine

Wenn das Systemprogramm einen bestimmten Fehler erkannt hat, ruft es den für diesen Fall vorgesehenen Organisationsbaustein auf. Durch entsprechende Programmierung dieses Organisationsbausteins können Sie nun das weitere Verhalten des Prozessors festlegen.

Abhängig davon, wie Sie den Organisationsbaustein programmieren, können Sie

- die normale Programmbearbeitung fortsetzen lassen,
- den Prozessor in Stop bringen
- und/oder
- ein spezielles 'Fehlerprogramm' bearbeiten lassen.

Für die folgenden Fehlerursachen sind Organisationsbausteine vorhanden:

Fehlerursache	Aufruf von	Reaktion bei nicht progr. OB
Aufruf eines nicht geladenen Bausteins (LZF)	OB 19	Stop
Quittungsverzug im Anwenderprogramm bei Zugriff auf Peripheriebaugruppen (QVZ)	OB 23	Keine
Quittungsverzug beim Aktualisieren des Prozeßabbildes und bei Koppelmerkerübertragung (QVZ)	OB 24	Keine
Adressierfehler (ADF)	OB 25	Stop
Zykluszeitüberschreitung (ZYK)	OB 26	Stop
Substitutionsfehler (BCF)	OB 27	Stop
Betriebsartenschalter auf STOP, PG-Funktion 'AG-STOP', Stop vom S5-Bus (Mehrprozessorbetrieb)	OB 28	Stop
Operationscodefehler (BCF)	OB 29	Stop
Parameterfehler (BCF)	OB 30	Stop
Sonstige Laufzeitfehler (LZF)	OB 31	Stop
Transferfehler bei Datenbausteinen (LZF)	OB 32	Stop
Weckfehler (WECK-FE)	OB 33	Stop
Fehler bei der Bearbeitung der Reglerstruktur R64 (REG-FE)	OB 34	Stop

Die Reaktion bei nicht programmiertem Organisationsbaustein ist fehlerabhängig:

**a) keine Unterbrechung der zyklischen Programmbearbeitung**

Tritt ein Quittungsverzug auf und OB 23 oder OB 24 sind nicht geladen, so wird die zyklische Programmbearbeitung nicht unterbrochen. Es erfolgt keine Reaktion des Prozessors.

Soll der Prozessor bei QVZ in den Stoppzustand übergehen, so muß der Organisationsbaustein eine Stopp-Anweisung enthalten und mit BE abgeschlossen werden.

Programm für Stop:

```
:  
:  
:STP  
:BE
```

**b) Stoppzustand**

In allen übrigen Fehlerfällen geht der Prozessor sofort in den Stoppzustand, wenn die zugehörigen Organisationsbausteine vom Anwender nicht programmiert worden sind.

Soll in Ausnahmefällen (z. B. während der Inbetriebsetzung) der eine oder andere Fehler die zyklische Programmbearbeitung nicht unterbrechen, so genügt eine Bausteinende-Anweisung im jeweiligen Organisationsbaustein.

Programm für Betrieb ohne Unterbrechung:

```
:  
:  
:BE
```

**WICHTIG!**

Eine Ausnahme bildet der Organisationsbaustein OB 28: Hier erfolgt immer ein Übergang in den Stoppzustand, unabhängig davon, ob und wie der OB 28 programmiert ist.

Wenn Sie den betreffenden Organisationsbaustein nicht programmieren wollen, haben Sie die Möglichkeit, durch entsprechende Programmierung des Datenbausteins DX 0 den Übergang des Prozessors in den Stoppzustand zu verhindern.

## **Unterbrechungen bei der Bearbeitung der Fehler-Organisationsbausteine**

Nachdem das Systemprogramm den betreffenden Organisationsbaustein aufgerufen hat, wird das darin enthaltene Anwenderprogramm bearbeitet.

Tritt während der Bearbeitung eines Organisationsbausteins erneut ein Fehler auf, wird wie in der zyklischen Programmbearbeitung das Programm an der nächsten Befehlsgrenze unterbrochen und der entsprechende Organisationsbaustein aufgerufen.

Die Organisationsbausteine werden in der Reihenfolge bearbeitet, in der sie aufgerufen werden. Wieviele Fehler-Organisationsbausteine ineinandergeschachtelt werden können, ist abhängig von

### **a) der Art der aufgetretenen Fehler:**

Es können keine Organisationsbausteine ineinandergeschachtelt werden, die derselben Programmbearbeitungsebene angehören! (Zur Zuordnung der Fehler-OBs zu den Programmbearbeitungsebenen siehe folgendes Kapitel.)

Bei der Bearbeitung des OB 27 (Programmbearbeitungsebene BCF) kann beispielsweise ein OB 32 (Programmbearbeitungsebene LZF), nicht jedoch ein OB 29 oder OB 30 (ebenfalls BCF) eingeschachtelt werden.

Bei Doppelaufruf einer Programmbearbeitungsebene geht der Prozessor unmittelbar in Stop.

### **b) der Anzahl der zu diesem Zeitpunkt aktivierten Programmbearbeitungsebenen:**

Für jede aktivierte Programmbearbeitungsebene benötigt das Systemprogramm bei Unterbrechungen besonderen Speicherplatz zum Anlegen des USTACKs. Reicht dieser Speicherplatz nicht mehr aus, so kommt es zu einem USTACK-Überlauf.

Bei USTACK-Überlauf geht der Prozessor unmittelbar in Stop.

### **c) der Anzahl der zu diesem Zeitpunkt aufgerufenen Bausteine:**

Bei BSTACK-Überlauf geht der Prozessor unmittelbar in Stop.

## **5.5 Fehler im ANLAUF**

Bei der Initialisierung und im Anlauf auftretende Störungs- und Fehlerursachen können dazu führen, daß das Anlaufprogramm abgebrochen wird und der Prozessor in den Stoppzustand übergeht.

Im Anlaufprogramm (Organisationsbausteine OB 20, 21 und 22) auftretende Unterbrechungsursachen werden wie im Zyklus behandelt.

Ausnahme: Bei einem Stop im Anlauf wird kein Organisationsbaustein OB 28 aufgerufen.

### **Mögliche Unterbrechungs- und Fehlersursachen ohne dazugehörigen Fehler-Organisationsbaustein)**

#### **STP:**

Stoppbefehl vom Systemprogramm (bei FE-STP) oder im Anwenderprogramm

#### **BAU:**

Ausfall der Pufferbatterie am Zentralgerät

#### **NAU:**

Ausfall der Versorgungsspannung am Zentralgerät

#### **PEU:**

Ausfall der Versorgungsspannung an einem Erweiterungsgerät

#### **STUEU:**

Stacküberlauf beim Unterbrechungsstack (USTACK)

#### **STUEB:**

Stacküberlauf beim Bausteinstack (BSTACK) bei zu großer Schachteltiefe

#### **DOPP:**

Doppelaufruf einer Fehlerprogrammbearbeitungsebene (zum Doppelfehler siehe Beispiele auf Seite 4-4 ff)

#### **RAM-FE:**

Fehler bei der Initialisierung: Inhalt des Betriebssystem-RAMs oder des DB-RAMs defekt

#### **MOD-FE:**

Fehler bei der Initialisierung: Inhalt des Anwendermoduls (RAM- oder EPROM-Modul) nicht korrekt

#### **DB0-FE:**

Fehler beim Aufbau der Bausteinadreßliste (DB 0)

#### **DB1-FE:**

Fehler bei der Auswertung des DB 1 zum Aufbau der Adreßliste für die Prozeßabbildaktualisierung

#### **DB2-FE:**

Fehler bei der Auswertung des DB 2 der Reglerstruktur R64

#### **DX0-FE:**

Fehler bei der Auswertung des Datenbausteins DX 0  
(DB0-FE, DB1-FE, DB2-FE und DX0-FE: siehe folgende Seiten!)

### 5.5.1 DBO-FE (DB0-Fehler):

Fehler beim Aufbau der Bausteinadreßliste (Datenbaustein DB 0)

Der DB 0 wird vom Systemprogramm nach Netz-ein aufgebaut. Bei einem DBO-Fehler finden Sie in den Systemdatenwörtern BS 3 und BS 4 Fehlerkennungen, die den aufgetretenen Fehler näher definieren.

absolute Speicheradresse:    BS 3    KH = EA03  
                                  BS 4    KH = EA04

(Hinweise zur Auswertung der Fehlerinformationen in den Systemdatenwörtern BS 3 und BS 4 finden Sie im Kapitel 5.2)

Fehlerkennung BS3        BS4		Erläuterung
8001H	yyyyH	Falsche Bausteinlänge yyyy = Adresse des Bausteins mit falscher Länge
8002H	yyyyH	Berechnete Endadresse des Bausteins im Speicher falsch yyyy = Bausteinadresse
8003H	yyyyH	Ungültige Bausteinkennung yyyy = Adresse des Bausteins mit falscher Kennung
8004H	yyyyH	Zu große Organisationsbausteinnummer (erlaubt: OB 1 bis OB 39) yyyy = Adresse des Bausteins mit falscher Nummer
8005H	yyyyH	Datenbausteinnummer 0 (erlaubt: DB 1 bis DB 255) yyyy = Adresse des Bausteins mit falscher Nummer

### 5.5.2 DB1-FE (DB1-Fehler):

Fehler bei der Auswertung des DB1 zum Aufbau der Adreßliste für die Prozeßabbildaktualisierung

- fehlender DB 1 im Mehrprozessorbetrieb oder
- fehlerhafte DB1-Adreßliste bei Neustart

Auch bei einem DB1-Fehler finden Sie in den Systemdatenwörtern BS 3 und BS 4 Fehlerkennungen, die den aufgetretenen Fehler näher definieren.

absolute Speicheradresse:    BS 3    KH = EA03  
                                  BS 4    KH = EA04

(Hinweise zur Auswertung der Fehlerinformationen in den Systemdatenwörtern BS 3 und BS 4 finden Sie im Kapitel 5.2)

Fehlerkennung BS3	BS4	Erläuterung
0410H	yyyyH	Unzulässige Kennung 1. Kopfkennung fehlt oder fehlerhaft (korrekt KC MASK01) 2. Kennung unzulässig (zulässig KH DE00, DA00, CE00, CA00, BB00) 3. Endekennung fehlt oder fehlerhaft (korrekt KH EEEE) yyyyH = Unzulässige Kennung
0411H	yyyyH	"Digitale Eingänge", Anzahl Adressen unzulässig (zulässig 0...128) yyyy = Unzulässige Anzahl Adressen
0412H	yyyyH	"Digitale Ausgänge", Anzahl Adressen unzulässig (zulässig 0...128) yyyy = Unzulässige Anzahl Adressen
0413H	yyyyH	"Koppelmerker-Eingänge", Anzahl Adressen unzulässig (zulässig 0...256) yyyy = Unzulässige Anzahl Adressen
0414H	yyyyH	"Koppelmerker-Ausgänge", Anzahl Adressen unzulässig (zulässig 0...256) yyyy = Unzulässige Anzahl Adressen
0415H	yyyyH	Ungültige Anzahl Zeitzellen (erlaubt: 256) yyyy = Unzulässige Anzahl Zeitzellen
0419H	yyyyH	Quittungsverzug bei digitalen Eingängen yyyy = Adresse des nicht quittierten Eingangsbytes
041AH	yyyyH	Quittungsverzug bei digitalen Ausgängen yyyy = Adresse des nicht quittierten Ausgangsbytes
041BH	yyyyH	Quittungsverzug bei Koppelmerker-Eingang yyyy = Adresse des nicht quittierten Merkerbytes
041CH	yyyyH	Quittungsverzug bei Koppelmerker-Ausgang yyyy = Adresse des nicht quittierten Merkerbytes



### 5.5.3 DB2-FE (DB2-Fehler):

Fehler bei der Auswertung des Parametrierungs-Datenbaustein DB 2 der Reglerstruktur R64 (Reglerinitialisierung)

Bei einem DB2-Fehler finden Sie in den Systemdatenwörtern BS 3 und BS 4 Fehlerkennungen, die den aufgetretenen Fehler näher definieren.

absolute Speicheradresse:    BS 3    KH = EA03  
                                 BS 4    KH = EA04

(Hinweise zur Auswertung der Fehlerinformationen in den Systemdatenwörtern BS 3 und BS 4 finden Sie im Kapitel 5.2)

Fehlerkennung BS3            BS4		Erläuterung
0421H	DByyH	Datenbaustein nicht geladen yy = Nummer des nicht geladenen Datenbausteins
0422H	FByyH	Funktionsbaustein nicht geladen yy = Nummer des nicht geladenen Funktionsbausteins
0423H	FByyH	Funktionsbaustein nicht erkannt yy = Nummer des nicht erkannten Funktionsbausteins
0424H	FByyH	Funktionsbaustein mit falscher PG-Software geladen yy = Nummer des Funktionsbausteins
0425H	DByyH	Falsche Regler-Datenbaustein-Länge yy = Nummer des Datenbausteins
0426H	-	Für das Verschieben der Regler-DBs vom Anwender- EPROM in das DB-RAM ist der Speicherplatz im DB-RAM nicht ausreichend

#### 5.5.4 DX0-FE (DX0-Fehler):

Fehler bei der Auswertung des Datenbausteins DX 0

Bei einem DX0-Fehler finden Sie in den Systemdatenwörtern BS 3 und BS 4 Fehlerkennungen, die den aufgetretenen Fehler näher definieren.

absolute Speicheradresse:    BS 3    KH = EA03  
                                      BS 4    KH = EA04

(Hinweise zur Auswertung der Fehlerinformationen in den Systemdatenwörtern BS 3 und BS 4 finden Sie im Kapitel 5.2)

Fehlerkennung BS3            BS4		Erläuterung
0431H	yyyyH	Unzulässige Kennung 1. Kopfkennung fehlt oder fehlerhaft (korrekt KC MASKX0) 2. Blockkennung unzulässig 3. Endekennung fehlt oder fehlerhaft (korrekt KH EEEE) yyyy = Unzulässige Kennung
0432H	yyyyH	Unzulässiger Parameter yyyy = Unzulässiger Parameter
0434H	yyyyH	Nicht erlaubte Anzahl Zeitzellen (erlaubt: 0...256) yyyy = Falsche Anzahl Zeitzellen
0435H	yyyyH	Unerlaubte Zykluszeit (erlaubt: 1ms bis 6000ms) yyyy = Falsche Zeitgröße

## 5.6 Fehler im RUN und im ANLAUF

Im Betriebszustand RUN kann eine zyklische, eine zeit- oder alarmgesteuerte Programmbearbeitung oder eine Reglerbearbeitung an Befehlsgrenzen unterbrochen werden durch das Auftreten bestimmter Störungen.

Bei der Initialisierung und im Betriebszustand ANLAUF auftretende Unterbrechungsursachen führen ebenfalls dazu, daß das Anlaufprogramm abgebrochen wird und der Prozessor in den Stoppzustand übergeht bzw. den für diesen Fehlerfall vorgesehenen Organisationsbaustein aufruft. Im Anlaufprogramm auftretende Unterbrechungsursachen werden wie im Zyklus behandelt.

Man unterscheidet zwischen Störungen, die den Prozessor direkt in den Betriebszustand STOP überführen (z.B. STUEU) und Störungen, bei deren Auftreten das Systemprogramm vor Übergang in den Stoppzustand bestimmte Organisationsbausteine aufruft (z.B. ADF), die der Anwender programmieren kann.

### **Mögliche Unterbrechungs- und Fehlerursachen *ohne* dazugehörigem Fehler-OB**

#### **STP:**

Stoppbefehl vom Systemprogramm (bei Maschinenfehler) oder im Anwenderprogramm

#### **BAU:**

Ausfall der Pufferbatterie am Zentralgerät

#### **NAU:**

Ausfall der Versorgungsspannung am Zentralgerät

#### **PEU:**

Ausfall der Versorgungsspannung an einem Erweiterungsgerät

#### **STUEU:**

Stacküberlauf beim Unterbrechungsstack (USTACK)

#### **STUEB:**

Stacküberlauf beim Bausteinstack bei zu großer Schachtelungstiefe

#### **DOPP:**

Doppelaufruf einer Fehlerprogrammbearbeitungsebene (zum Doppelfehler siehe Beispiele auf Seite 4-4 ff)

Bei allen genannten Störungen erfolgt ein sofortiger Übergang in den Stoppzustand. Dabei wird ein USTACK aufgebaut, in dem die aufgetretene Störung angezeigt wird.

(Hinweise zur Auswertung des USTACKs finden Sie in Kapitel 5.3)

**Mögliche Unterbrechungs- und Fehlerursachen mit dazugehörigem Fehler-OB im RUN und im ANLAUF**

**BCF:**

Befehlscodefehler	1. Substitutionsfehler	OB 27
	2. Operationscodefehler	OB 29
	3. Parameterfehler	OB 30

**LZF:**

Laufzeitfehler	1. Aufruf eines nicht geladenen Bausteins	OB 19
	2. Transferfehler bei DBs	OB 32
	3. Sonstige Laufzeitfehler	OB 31

**ADF:**

Adressierfehler		OB 25
-----------------	--	-------

**QVZ:**

Quittungsverzug	1. im Anwenderprogramm bei Zugriff auf Peripheriebaugruppen	OB 23
	2. bei der Prozeßabbildaktualisierung	OB 24

**Mögliche Unterbrechungs- und Fehlerursachen mit dazugehörigem Fehler-OB nur im RUN**

**ZYK-FE:**

Zyklusfehler		OB 26
--------------	--	-------

**WECK-FE:**

Weckfehler		OB 33
------------	--	-------

**REG-FE:**

Reglerfehler		OB 34
--------------	--	-------

**ABBR:**

Abbruch		OB 28
---------	--	-------

Die folgenden Kapitel beschreiben jede dieser Fehlerursachen genauer.

**5.6.1 BCF (Befehlscodefehler)**

Ein Befehlscodefehler tritt dadurch auf, daß der Prozessor einen STEP5-Befehl des Anwenderprogramms nicht interpretieren oder ausführen kann. Alle zulässigen Befehlscodes sind im Anhang aufgelistet.

Der Befehl, der den entsprechenden Befehlscodefehler verursacht, wird nicht ausgeführt. Falls ein BCF-Organisationsbaustein programmiert ist, wird dieser aufgerufen, bearbeitet und anschließend mit dem nächsten Befehl im unterbrochenen Anwenderprogramm fortgefahren. Bei nicht programmiertem BCF-OB geht der Prozessor in den Stop.

Es werden folgende Befehlscodefehler unterschieden:

**a) BCF = Substitutionsfehler**

Wenn in einem Funktionsbaustein eine Operation mit einem Formaloperanden ausgeführt werden soll, so ersetzt der Prozessor bei der Bearbeitung des Anwenderprogramms diesen Formaloperanden durch den im Aufruf des Funktionsbausteins stehenden Aktualoperanden.

Der Prozessor erkennt eine unzulässige Substitution. Das Systemprogramm unterbricht daraufhin die Bearbeitung des Anwenderprogramms und ruft den Organisationsbaustein **OB 27** auf.

Im Akku 1 stehen dabei zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Fehlerkennung Akkul      Akku2		Erläuterung
1801H	-	Substitutionsfehler beim Befehl BBS
1802H	-	Substitutionsfehler beim Befehl BDW, BMW
1803H	-	Substitutionsfehler beim Befehl B=, BI=
1804H	-	Substitutionsfehler beim Befehl L=, T=
1805H	-	Substitutionsfehler beim Befehl U=, UN=, O=, ON=, ==, S= und RB=

**b) BCF = Operationscodefehler**

Ein unzulässiger Operationscode tritt auf, wenn ein Befehl programmiert worden ist, der nicht im STEP5-Befehlsumfang des Prozessors liegt (z.B. können RU- und SU-Befehle mit dem PG programmiert, jedoch von den R- und S-Prozessoren und der CPU 928 im AG 135U nicht interpretiert werden).

Beim Erkennen eines unzulässigen Operationscodes wird an dieser Stelle die Bearbeitung des Anwenderprogramms unterbrochen und der Organisationsbaustein **OB 29** aufgerufen.

Bei Aufruf des OB 29 stehen im Akku 1 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Ein Operationscodefehler sollte nicht quittiert werden: Der Prozessor erkennt nicht, ob es sich bei dem fehlerhaften Befehl um einen Einwort- oder Mehrwortbefehl handelt. Hat der Prozessor den OB 29 bearbeitet, versucht er, das Programm mit dem nächsten Befehlswort fortzusetzen. Falls es sich dabei um das zweite Wort eines Mehrwortbefehls handelt, erkennt er entweder einen weiteren Befehlscodefehler oder führt dieses Wort als gültigen Befehl aus.

Fehlerkennung Akku1      Akku2		Erläuterung
1811H	-	Befehl mit unzulässigem Opcode
1812H	-	Unzulässiger Opcode bei einem Befehl, bei dem das High-Byte des 1. Befehlswortes den Wert 68H enthält
1813H	-	Unzulässiger Opcode bei einem Befehl, bei dem das High-Byte des 1. Befehlswortes den Wert 78H enthält
1814H	-	Unzulässiger Opcode bei einem Befehl, bei dem das High-Byte des 1. Befehlswortes den Wert 70H enthält
1815H	-	Unzulässiger Opcode bei einem Befehl, bei dem das High-Byte des 1. Befehlswortes den Wert 60H enthält

### c) BCF = Parameterfehler

Ein unzulässiger Parameter tritt auf, wenn ein Befehl mit einem Parameter, der für den entsprechenden Prozessor unzulässig ist, programmiert worden ist (z.B. Aufruf eines reservierten Datenbausteins), oder wenn eine nicht vorhandene Sonderfunktion aufgerufen wird.

Wenn ein unzulässiger Parameter vom Prozessor erkannt wird, unterbricht das Systemprogramm die Bearbeitung des Anwenderprogramms und ruft den Organisationsbaustein **OB 30** auf.

Ist der OB 30 nicht programmiert, geht der Prozessor in den Stoppzustand über.

Beim Aufruf des OB 30 stehen im Akku 1 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Fehlerkennung Akku1      Akku2		Erläuterung
		Unzulässiger Parameter bei:
1821H	-	A DB 0, 1, 2
182BH	-	SPA(B) OB 0
182CH	-	SPA(B) OB > 39: Sonderfunktion nicht vorhanden
182DH	-	AX DX0
182EH	-	L MW/T MW/L PW/T PW/L QW/T QW/L DD/T DD/B MW 255
182FH	-	L EW/T EW/L AW/T AW 127
1830H	-	L MD/T MD 253, 254, 255

Fehlerkennung Akku1      Akku2		Erläuterung
1831H	-	L ED/T ED/L AD/T AD 125, 126, 127
1832H	-	RLD/RRD/SVD/SLD 33-255
1833H	-	SLW/SRW/LIR/TIR 16-255
1834H	-	SES/SEF 32-255
1835H	-	U=/UN=/O=/ON=/S=/RB=/==/RD=/FR=/SI=/SE=/SVZ=/SSV=/ SAR=/L=/LC=/LW=/T= 0, 127-255
1836H	-	B=/LD= 0, 126-255
1837H	-	MBR mit Konstante > 0FFFFH ( $2^{16} - 2^{19} \# 0$ )

### 5.6.2 LZF (Laufzeitfehler)

Ein Laufzeitfehler tritt dadurch auf, daß der Prozessor während der Bearbeitung eines STEP5-Befehls einen Fehler erkennt.

Der Befehl, der den entsprechenden Laufzeitfehler verursacht, wird nicht ausgeführt. Falls ein LZF-Organisationsbaustein programmiert ist, wird dieser aufgerufen und anschließend mit dem nächsten Befehl im unterbrochenen Anwenderprogramm fortgefahren. Bei nicht programmiertem LZF-OB geht der Prozessor in den Stop.

Es werden folgende Laufzeitfehler unterschieden:

#### a) LZF = Aufruf eines nicht geladenen Bausteins

Wenn im Anwenderprogramm ein Baustein aufgerufen oder aufgeschlagen wird, der nicht vorhanden ist, erkennt das Systemprogramm einen Fehler. Dies gilt für alle Bausteinarten und sowohl für die bedingte als auch die unbedingte Aufruf-Anweisung.

Wenn der Aufruf oder das Aufschlagen eines nicht geladenen Bausteins erkannt wird, ruft das Systemprogramm den Organisationsbaustein OB 19 auf. Im OB 19 können Sie das weitere Verhalten des Prozessors festlegen. Falls ein OB 19 programmiert ist, wird dieser aufgerufen und anschließend die Bearbeitung des unterbrochenen STEP5-Programms mit dem nächsten Befehl fortgesetzt. Wenn der OB 19 hingegen nicht programmiert ist, geht der Prozessor beim Aufruf oder Aufschlagen eines nicht geladenen Bausteins in den Stoppzustand.

Bei Aufruf des OB 19 stehen im Akku 1 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Fehlerkennung Akkul      Akku2		Erläuterung
1A01H	-	Nicht geladener Datenbaustein bei ADB
1A02H	-	Nicht geladener Datenbaustein bei AXDX
1A03H	-	Nicht geladener Baustein bei SPA(B) FB, OB, PB, SB
1A04H	-	Nicht geladener Baustein bei BA(B) FX
1A05H	-	Nicht geladener Datenbaustein bei OB 254 bzw. 255

#### b) LZF = Transferfehler

Beim Transferieren von Daten in Datenbausteine (DB, DX) vergleicht der Prozessor die Länge des aufgerufenen DBs mit dem im Transfer-Befehl stehenden Parameter. Wird durch den angegebenen Parameter die Datenbausteinlänge überschritten, so wird die Transfer-Anweisung nicht ausgeführt, um ein irrtümliches Überschreiben von Daten im Speicher zu verhindern.

Ein Transferfehler wird auch festgestellt, wenn ein einzelnes Bit innerhalb eines nicht vorhandenen Datenwortes abgefragt oder verändert werden soll.

Ein Transferfehler wird ebenfalls erkannt, wenn ein Zugriff auf ein Datenwort stattfinden soll, bevor ein Datenbaustein aufgeschlagen ist (mit A DBn bzw. AX DXn).

Beim Erkennen eines Transferfehlers ruft das Systemprogramm den Organisationsbaustein OB 32 auf. Der Befehl, der den Transferfehler verursacht hat, wird nicht mehr bearbeitet. Wenn der OB 32 nicht programmiert ist, geht der Prozessor in den Stoppzustand.

Bei Aufruf von OB 32 stehen im Akku 1 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Fehlerkennung Akkul      Akku2		Erläuterung
1A11H	-	Zugriff mit U/UN D, O/ON D, S/R D, =D auf ein nicht definiertes Datenwort
1A12H	-	Transferfehler bei TDR auf ein nicht definiertes Datenwort
1A13H	-	Transferfehler bei TDL auf ein nicht definiertes Datenwort
1A14H	-	Transferfehler bei TDW auf ein nicht definiertes Datenwort
1A15H	-	Transferfehler bei TDD auf ein nicht definiertes Datenwort



### c) Sonstige Laufzeitfehler

Hierzu gehören alle Laufzeitfehler, die nicht einer der vorherigen Laufzeitfehlerarten (Transferfehler oder Aufruf eines nicht geladenen Bausteins) zugeordnet werden können.

Beim Erkennen eines dieser Laufzeitfehler ruft das Systemprogramm den Organisationsbaustein **OB 31** auf. Der den Fehler verursachende Befehl (bzw. die Sonderfunktion) wird nicht weiterbearbeitet. Wenn der OB 31 nicht programmiert ist, geht der Prozessor in den Stoppzustand.

Soll die Programmbearbeitung bei Auftreten eines der unten aufgeführten Fehler weiterlaufen, genügt die Bausteinende-Anweisung **BE** im **OB31**.

Bei Aufruf des **OB 31** stehen im Akku 1 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern:

Fehlerkennung Akku1      Akku2		Erläuterung
1A21H	-	Fehler bei EDB, EXDX: Datenbaustein existiert bereits
1A22H	-	Fehler bei EDB, EXDX: Unzulässige Datenbaustein-Länge ( $> 4 \times 2^{10}$ Wörter)
1A23H	-	Fehler bei EDB, EXDX: Speicherplatz im RAM reicht nicht aus
1A25H	-	Fehler bei BI=: Unzulässiger Parameter im Akku 1 ( $< 1$ oder $> 125$ )
1A29H	-	Klammerstackunter- oder -überlauf nach U(, O(, ).
1A2AH	-	Fehler bei A DB, AX DX: Bausteinlänge im Datenbausteinkopf zu klein (Länge $< 5$ Wörter)
1A2BH	-	Funktionsbaustein mit falscher PG-Software geladen
1A2CH	-	Fehler bei ACR: Unzulässige Kachelnummer
1A31H	-	Sonderfunktionsfehler beim OB 254 bzw. OB 255 (Kopieren) oder OB 250: Ziel-Datenbaustein bereits im DB-RAM vorhanden
1A32H	-	Sonderfunktionsfehler beim OB 254 bzw. OB 255 (Duplizieren): Ziel-Datenbaustein bereits im DB-RAM vorhanden
1A33H	-	Sonderfunktionsfehler beim OB 254 bzw. OB 255: Speicherplatz im DB-RAM reicht nicht aus
1A3AH	-	Sonderfunktionsfehler beim OB 221: Unzulässiger Wert für die neue Zykluszeit (Zykluszeit $< 1\text{ms}$ oder $> 6000\text{ms}$ )

Fehlerkennung Akkul      Akku2		Erläuterung
1A3BH	-	Sonderfunktionsfehler beim OB 223: Anlaufarten der am Mehrprozessorbetrieb beteiligten Prozessoren sind unterschiedlich
1A41H	-	Sonderfunktionsfehler beim OB 240, OB 241 oder OB 242: unzulässige Schieberegister- oder Datenbaustein-Nummer (Nr. < 192)
1A42H	-	Sonderfunktionsfehler beim OB 241: Schieberegister nicht initialisiert
1A43H	-	Sonderfunktionsfehler beim OB 240: Speicherplatz im DB-RAM reicht nicht aus.
1A44H	-	Sonderfunktionsfehler beim OB 240: Datenwort DW 0 des Datenbausteins hat nicht den Inhalt '0'.
1A45H	-	Sonderfunktionsfehler beim OB 240: unzulässige Schieberegisterlänge im DW 1 (nicht zwischen 2 und 256)
1A46H	-	Sonderfunktionsfehler beim OB 240: unzulässige Zeigerposition oder Zeigeranzahl
1A47H	-	Sonderfunktionsfehler beim OB 120
1A48H	-	Sonderfunktionsfehler beim OB 122
1A49H	-	Sonderfunktionsfehler beim OB 110
1A50H	-	Fehler bei LRW, TRW: Die errechnete Speicheradresse <BS + Konstante> liegt außerhalb 0 - EDFFH
1A51H	-	Fehler bei LRD, TRD: Die errechnete Speicheradresse <BS + Konstante> liegt außerhalb 0 - EDFEH
1A52H	-	Fehler bei TSG, LB GB, LW GW, TB GB, TW GW: Die errechnete Linearadresse <BS + Konstante> liegt außerhalb 0 - EFFFH
1A53H	-	Fehler bei LB GW, LW GD, TB GW, TW GD: Die errechnete Linearadresse <BS + Konstante> liegt außerhalb 0 - EFFEh
1A54H	-	Fehler bei LB GD, TB GD: Die errechnete Linearadresse <BS + Konstante> liegt außerhalb 0 - EFFCH
1A55H	-	Fehler bei TSC, LB CB, LW CW, TB CB, TW CW: Die errechnete Kacheladresse <BS + Konstante> liegt außerhalb F400H - FBFFH
1A56H	-	Fehler bei LB CW, LW CD, TB CW, TW CD: Die errechnete Kacheladresse <BS + Konstante> liegt außerhalb F400H - FBFEH

Fehlerkennung Akku1      Akku2		Erläuterung
1A57H	-	Fehler bei LB CD, TB CD: Die errechnete Kachel- adresse <BS + Konstante> liegt außerhalb F400H - FBFCH
1A58H	-	Fehler bei TNW, TNB: Der Quellblock liegt nicht vollständig in einem dieser Bereiche: 0000-7FFF Anwenderspeicher 8000-DD7F Datenbaustein-RAM DD80-EDFF System-RAM (DB0,BA,BB,BS,BT,T,Z) EE00-EFFF Merker, Prozeßabbild F000-FFFF Peripherie
1A59H	-	Fehler bei TNW, TNB: Der Zielblock liegt nicht vollständig in einem dieser Bereiche: 0000-7FFF Anwenderspeicher 8000-DD7F Datenbaustein-RAM DD80-EDFF System-RAM (DB0,BA,BB,BS,BT,T,Z) EE00-EFFF Merker, Prozeßabbild F000-FFFF Peripherie

### 5.6.3 ADF (Adressierfehler)

Ein Adressierfehler tritt auf, wenn mit einer STEP5-Operation ein Ein- oder Ausgang im Prozeßabbild angesprochen wird, dem zum Zeitpunkt des letzten Neustarts keine Peripheriebaugruppe zugeordnet war (Baugruppe war nicht gesteckt, defekt oder nicht im Datenbaustein DB 1 des Prozessors angegeben).

Das Systemprogramm unterbricht nun die Bearbeitung des Anwenderprogramms und ruft den Organisationsbaustein OB 25 auf. Nach der Bearbeitung des im OB 25 programmierten Programms wird mit dem nächsten Befehl des unterbrochenen Programms fortgefahren, d.h., die STEP5-Anweisung, die den ADF verursacht hat, wird nicht ausgeführt.

Wenn der OB 25 nicht programmiert ist, geht der Prozessor beim Auftreten eines Adressierfehlers in den Stoppzustand, es sei denn, Sie haben für diesen Fall eine Fortsetzung der Programmbe-  
arbeitung im Datenbaustein DX0 festgelegt.

Die Adressierfehler-Überwachung kann durch entsprechende Programmierung des DX0 auch ganz unterdrückt werden.

Bei Auftreten eines Adressierfehlers werden keine Fehlerkennungen in Akku 1 oder Akku 2 übergeben!

#### 5.6.4 QVZ (Quittungsverzug)

Ein Quittungsverzug tritt auf, wenn sich eine Ein- oder Ausgabebaugruppe nach einer Adressierung innerhalb einer bestimmten Zeit nicht mit dem RDY-Signal (Ready) zurückmeldet. Die Ursache des Quittungsverzugs kann ein Defekt auf der Baugruppe sein oder das Entfernen der Baugruppe während des Betriebs.

Folgende Quittungsverzugsfehler unterbrechen die Anwenderprogramm-bearbeitung und rufen einen entsprechenden Organisationsbaustein auf:

1. Quittungsverzug im Anwenderprogramm bei Direktzugriff über den S5-Bus auf CP, IP, KOR oder auf eine Peripheriebaugruppe (z.B. mit Lade- und Transferbefehlen L/T P...bzw. Q...):  
Das Systemprogramm ruft den Organisationsbaustein **OB 23** auf.

In den Akkus 1 und 2 stehen dabei zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern:

Fehlerkennung Akkul      Akku2		Erläuterung
1E23H	yyyyH	Quittungsverzug (QVZ) im Anwenderprogramm bei Zugriff auf Peripherie yyyy = QVZ-Adresse

2. Quittungsverzug beim Aktualisieren des Prozeßabbildes für Ein- und Ausgänge und Transfer der Koppelmerker:  
Das Systemprogramm ruft den Organisationbaustein **OB 24** auf.

In den Akkus 1 und 2 stehen dabei zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern:

Fehlerkennung Akkul      Akku2		Erläuterung
1E25H	yyyyH	Quittungsverzug beim Aktualisieren der digitalen Ausgänge yyyy = Adresse des nicht quittierten Ausgangs-bytes
1E26H	yyyyH	Quittungsverzug beim Aktualisieren der digitalen Eingänge yyyy = Adresse des nicht quittierten Eingangs-bytes
1E27H	yyyyH	Quittungsverzug beim Aktualisieren der Koppelmerker-Ausgänge yyyy = Adresse des nicht quittierten Merkerbytes
1E28H	yyyyH	Quittungsverzug beim Aktualisieren der Koppelmerker-Eingänge yyyy = Adresse des nicht quittierten Merkerbytes

Wenn die aufgerufenen Organisationsbausteine nicht programmiert sind, wird die Bearbeitung des Anwenderprogramms fortgesetzt. Ein Quittungsverzug verlängert jedoch die Laufzeit des ihn verursachenden STEP5-Befehls.

Bei einem aufgetretenen Quittungsverzug liest der Prozessor 'ersatzweise' den Wert "00H" ein und arbeitet, falls der QVZ vom Anwender quittiert wird, mit diesem Wert weiter.

Wenn der Quittungsverzug zum Stop des Prozessors führen soll, muß im OB 23 bzw. 24 der Stoppbefehl STP programmiert sein.

Durch entsprechende Programmierung des DX0 können Sie im Falle eines QVZ auch bei nicht programmierten OBs 23/24 einen System-stop veranlassen.

#### 5.6.5 ZYK-FE (Zykluszeitfehler)

Die Zykluszeit umfaßt die gesamte Zeitdauer einer Bearbeitung des zyklischen Programms. Eine Überschreitung der im Prozessor eingestellten Zykluszeit kann ausgelöst werden z.B. durch fehlerhafte Programmierung, durch eine Programmschleife in einem Funktionsbaustein, durch Ausfall des Taktgenerators oder durch Systemleistungen wie z.B. Prozeßabbildaktualisierung in Verbindung mit langen Programmen.

Wenn eine Zykluszeitüberschreitung auftritt, unterbricht das Systemprogramm die Bearbeitung des Anwenderprogramms und ruft den Organisationsbaustein OB 26 auf. Die Zykluszeit wird dabei neu gestartet (getriggert). Falls die Zykluszeit erneut abläuft, bevor der OB 26 zuende bearbeitet ist, geht der Prozessor mit Doppelfehler in den Stoppzustand.

Die Zykluszeit ist variabel (1 bis 6000 msec) und nachtriggerbar (siehe oben). Unabhängig von der Zykluszeit wird 150 msec nach Ablauf der Zykluszeit BASP ausgegeben, wenn der OB 26 zu diesem Zeitpunkt noch nicht zuende bearbeitet ist.

Wenn der OB 26 nicht programmiert ist, geht der Prozessor in den Stoppzustand über, es sei denn, die Voreinstellung im DX0 ist vom Anwender geändert worden.

Der Anwender kann die Zykluszeit individuell vorgeben, entweder durch einen Eintrag im DX0 oder durch einen Aufruf des Sonderfunktions-Organisationsbausteins OB 221.

Im zyklischen Programm kann die Zykluszeitüberwachung durch einen Aufruf des SF-OBs 222 "nachgetriggert" werden.

Bei Auftreten eines Zykluszeitfehlers werden keine Fehlerkennungen in Akku 1 oder Akku 2 übergeben!

### 5.6.6 WECK-FE (Weckfehler)

Wenn für einen bestimmten Weckalarm-OB eine erneute Anforderung auftritt, bevor seine letzte Anforderung vollständig bearbeitet ist, erkennt das Systemprogramm einen Weckfehler und ruft den Organisationsbaustein **OB 33** auf. Beachten Sie hierzu auch Kapitel 4.4.2 "WECKALARME"!

In den Akkus 1 und 2 hinterlegt das Systemprogramm zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern:

Fehlerkennung Akkul      Akku2		Erläuterung
1001H	0016H	Weckfehler bei OB 10 (10 ms)
1001H	0014H	Weckfehler bei OB 11 (20 ms)
1001H	0012H	Weckfehler bei OB 12 (50 ms)
1001H	0010H	Weckfehler bei OB 13 (100 ms)
1001H	000EH	Weckfehler bei OB 14 (200 ms)
1001H	000CH	Weckfehler bei OB 15 (500 ms)
1001H	000AH	Weckfehler bei OB 16 (1 sec)
1001H	0008H	Weckfehler bei OB 17 (2 sec)
1001H	0006H	Weckfehler bei OB 18 (5 sec)

**Hinweis:** Die Kennung im Akku 2 ist die Ebenenkennung des fehlererzeugenden Weckalarms.

Ist der OB 33 nicht programmiert, geht der Prozessor in den Stoppzustand. Durch entsprechende Programmierung im DX0 können Sie bei aufgetretenem Weckfehler und nicht programmiertem OB 33 die Programmbearbeitung jedoch weiterlaufen lassen.

Beachten Sie, daß ein erneuter Aufruf der bereits aktivierten Fehlerprogrammbearbeitungsebene "Weckfehler" nicht zu einem Doppelfehler (DOPP) führt!

### 5.6.7 REG-FE (Reglerfehler)

Ein Fehler beim Bearbeiten der vom Systemprogramm unterstützten Standard-Funktionsbausteine der Reglerstruktur R 64 wird als Reglerfehler erkannt.

#### WICHTIG!

Während z.B. ein Weckfehler vom Systemprogramm immer dann erkannt wird, wenn ein bestimmter Weckalarm-OB nicht innerhalb seines Zeitrasters (z.B. der OB 13 innerhalb von 100 ms) begonnen und zu Ende bearbeitet wurde (s.o.), wird eine fehlerhafte Bearbeitung des Regelungsprogramms erst bei Aufruf der Programmbearbeitungsebene **REGELUNG** erkannt und im **USTACK** angezeigt.

Bei Auftreten eines Reglerfehlers wird die Programmbearbeitungsebene **REGELUNG** verlassen und die Ebene **REGLERFEHLER (EBENE: 001CH)** mit dem Organisationsbaustein **OB 34** aufgerufen.

Die weitere Reaktion des Prozessors hängt ab von der Programmierung des **OB 34**:

- a) Wenn der **OB 34** nicht programmiert ist, geht der Prozessor in Stop.  
Durch Ausgabe des **USTACKs** läßt sich die Fehlerursache ermitteln.
- b) Wenn der **OB 34** programmiert ist, wird das darin enthaltene **STEP5-Programm** (z.B. Auswertung von Akku 1 und 2, davon abhängig die Fehlerbehandlung) bearbeitet. Im Anschluß daran wird die Reglerbearbeitung an der unterbrochenen Stelle fortgesetzt.

Sollen Reglerfehler grundsätzlich ignoriert werden, so genügt ein Bausteinende-Befehl **BE** im **OB 34**.

Soll die Reglerbearbeitung bei aufgetretenem Reglerfehler auch bei nicht programmiertem **OB 34** fortgesetzt werden, müssen Sie die Voreinstellung im **DX0** entsprechend ändern.

Bei Aufruf des **OB 34** stehen in den Akkus 1 und 2 zusätzliche Informationen, die den aufgetretenen Fehler näher erläutern.

Fehlerkennung Akku1      Akku2		Erläuterung
0801H	DByyH	Abtastzeitfehler yy = Nummer des betreffenden Regler-Datenbausteins
0802H	DByyH	Regler-Datenbaustein nicht geladen yy = Nummer des nicht geladenen Datenbausteins
0803H	FByyH	Regler-Funktionsbaustein nicht geladen yy = Nummer des nicht geladenen Funktionsbausteins
0804H	FByyH	Regler-Funktionsbaustein nicht erkannt yy = Nummer des nicht erkannten Funktionsbausteins
0805H	FByyH	Regler-Funktionsbaustein mit falscher PG-Software geladen yy = Funktionsbaustein-Nr.
0806H	DByyH	Falsche Regler-Datenbaustein-Länge yy = Datenbaustein-Nr.
0880H	yyyyH	Quittungsverzug (QVZ) während der Reglerbearbeitung

In allen 7 Fehlerfällen wird am Programmiergerät in den Steuerbits die Fehlerkennung **REG-FE** angekreuzt. Wenn Sie ein PG ohne **S5-DOS-Betriebssystem** verwenden, ist die vorletzte Stelle in der unteren Zeile der Steuerbits-Maske zwar nicht bezeichnet, wird jedoch ebenso angekreuzt. In der **USTACK-Maske** der Ebene **"REGELUNG"** ist als Störungsursache **REG** angekreuzt.

## **Abtastzeitfehler**

Nach Ablauf der vorgegebenen Abtastzeit wird das zyklische Programm an der nächsten **Bausteingrenze** abgebrochen und die Reglerbearbeitung eingeschoben. Nun ist es möglich, daß die Bearbeitung "langer" Bausteine zu viel Zeit in Anspruch nimmt und in der Folge die Reglerbearbeitung "außer Tritt" gerät: Es liegt ein Abtastzeitfehler vor.

Ein Abtastzeitfehler kann wie die übrigen Reglerfehler behandelt werden (wie unter a) und b) beschrieben) oder über eine Maske unterdrückt werden. In diesem Fall wird bei Auftreten eines Abtastzeitfehlers die Programmbearbeitung nicht unterbrochen.

Beachten Sie dazu die Beschreibung "Kompaktregelung im R-Prozessor des AG 135U", C79000-B8500-C365-03!

Ein Abtastzeitfehler kann eventuell verhindert werden, indem Sie die Voreinstellung im DX0 "Bearbeitung des Regler- und Prozeßalarms an Bausteingrenzen" in "Bearbeitung des Regler- und Prozeßalarms an Befehlsgrenzen" abändern.



### 5.6.8 ABBR (Abbruch)

Wenn im Betriebszustand RUN der Stoppzustand angefordert wird durch

- a) Betriebsartenschalter am Prozessor von RUN auf STOP,
- b) On-line-Funktion 'AG-STOP',
- c) Schalter am Koordinator auf STOP (im Mehrprozessorbetrieb),

so ruft das Systemprogramm den OB 28 auf. Nach der Bearbeitung des OB 28 geht der Prozessor in den Stoppzustand über.

#### **WICHTIG!**

**Der Übergang in den Stoppzustand erfolgt unabhängig davon, ob und wie der OB 28 programmiert ist.**

Es werden keine Fehlerkennungen in Akku 1 oder Akku 2 übergeben!

## 6 Integrierte Sonderfunktionen

Das Betriebssystem der CPU 928 bietet Ihnen Sonderfunktionen an, die Sie bei Bedarf mit einem bedingten (SPB OB x) oder einem unbedingten (SPA OB x) Bausteinaufruf anwenden können. Für diese Sonderfunktionen sind die Organisationsbausteine OB 40 bis 255 reserviert.

Diese Funktionen werden als *Integrierte* Sonderfunktionen bezeichnet, da sie ein fester Bestandteil des Systemprogramms sind. Als Anwender können Sie diese Sonderfunktionen zwar aufrufen, nicht jedoch lesen oder ändern.

### **WICHTIG!**

**Der Befehl SPA OB > 39 wirkt nicht wie ein 'echter' Bausteinwechsel. Es werden keine Alarme eingeschachtelt!**

### **Sonderfunktionen mit Pseudobefehlsgrenzen**

Einige der Sonderfunktionen sind langlaufende Sonderfunktionen, die sog. Pseudobefehlsgrenzen enthalten. D.h.: Die Ausführung der Sonderfunktion erfolgt in mehreren Teilschritten. Wenn nun während der Ausführung eines Teilschritts ein Fehler (z.B. ZYK) oder eine Unterbrechung (z.B. Weck- oder Prozeßalarm an Befehlsgrenzen) auftritt, so wird am Ende dieses Teilschritts an der Pseudobefehlsgrenze der entsprechende Organisationsbaustein eingeschachtelt.

Diejenigen Sonderfunktionen, die Pseudobefehlsgrenzen enthalten, sind in der folgenden Tabelle gekennzeichnet.

### **Fehler bei der Bearbeitung von Sonderfunktionen**

Hinsichtlich ihrer Fehlerreaktion können zwei Gruppen von Sonderfunktionen unterschieden werden.

#### Gruppe 1:

Zur Gruppe 1 zählen alle Sonderfunktionen, bei denen im Fehlerfall ein bestimmter Fehler-Organisationsbaustein aufgerufen wird, in dem Sie das weitere Verhalten des Prozessors festlegen können.

Stößt der Prozessor bei der Bearbeitung der Sonderfunktion z.B. auf eine falsche Parametrierung, erkennt er einen Laufzeitfehler und ruft den OB 31 auf. Oder ist z.B. die aufgerufene Sonderfunktion nicht vorhanden, erkennt er einen Befehlscodefehler und versucht, den OB 30 aufzurufen. Falls die Fehler-OBs 30 bzw. 31 nicht programmiert sind oder einen STP-Befehl enthalten, geht der Prozessor in den Stoppzustand. In den Steuerbits und im USTACK ist 'LZF' bzw. 'BCF' angekreuzt. In den Akkus der Fehlerbearbeitungsebene sind Fehlerkennungen hinterlegt, die den Fehler näher beschreiben. Falls der OB 30 bzw. OB 31 programmiert ist (und keinen STP-Befehl enthält), wird das Anwenderprogramm nach der Bearbeitung des OB 30 bzw. 31 mit dem nächsten Befehl fortgesetzt. In diesem Fall sind die Akkus unverändert.

## Gruppe 2:

Manche Sonderfunktionen verwenden zur Behandlung von Sonderfunktions-spezifischen Fehlern einen anderen Mechanismus: Sie beeinflussen VKE oder ANZ0/ANZ1.

Wenn bei der Bearbeitung dieser Sonderfunktionen ein Fehler auftritt, wird in den meisten Fällen das VKE gesetzt ( $VKE = 1$ ) und kann daher mit einem SPB-Befehl (Springe bedingt) ausgewertet werden.

Sie können somit in diesen Fällen das Verknüpfungsergebnis (VKE) als Signal für "Fehler" oder "Kein Fehler" auswerten und davon abhängig für den Fehlerfall ein spezielles Fehlerprogramm vorsehen.

Bei einigen Sonderfunktionen werden die Ergebnisanzeigen ANZ0 und ANZ1 durch die Bearbeitung der Sonderfunktion beeinflusst und können ebenfalls abgefragt werden.

## Übersicht über die Integrierten Sonderfunktionen in der CPU 928

OB 110	Zugriff auf das Anzeigenbyte
OB 111	Akku 1, 2, 3 und 4 löschen
OB 112	Akku Roll Up
OB 113	Akku Roll Down
OB 120	"Alarme gemeinsam sperren" ein-/ausschalten
OB 121	"Weckalarme einzeln sperren" ein-/ausschalten
OB 122	"Alarme gemeinsam verzögern" ein-/ausschalten
OB 123	"Weckalarme einzeln verzögern" ein-/ausschalten
OB 160 - 163	Zählschleife
OB 170	Bausteinstack (BSTACK) lesen
OB 180	Variabler Datenbaustein-Zugriff
OB 181	Datenbausteine (DB/DX) testen
OB 190, 192	Merker in Datenbausteine übertragen
OB 191, 193	Datenblöcke in Merkerbereich übertragen
OB 200 <sup>1)</sup> , 202 <sup>1)</sup> , 203, 204 <sup>1)</sup> und 205	Funktionen zur Mehrprozessor-Kommunikation
OB 216 - 218	Kachelzugriffe
OB 220	Vorzeichenerweiterung
OB 221 <sup>2)</sup>	Zykluszeit einstellen
OB 222	Zykluszeit neu starten
OB 223	Anlaufarten im Mehrprozessorbetrieb vergleichen
OB 224 <sup>2)</sup>	Koppelmerker im Mehrprozessorbetrieb im Block übertragen
OB 226	Wort aus dem Systemprogramm lesen
OB 227	Quersumme des Systemprogramms lesen
OB 228	Statusinformation einer Programmbearbeitungsebene lesen
OB 230 <sup>1)</sup> - 237 <sup>1)</sup>	Funktionen für Standard-Funktionsbausteine
OB 240	Schieberegister initialisieren
OB 241	Schieberegister bearbeiten
OB 242	Schieberegister löschen
OB 250 <sup>1)</sup>	PID-Regler initialisieren
OB 251 <sup>1)</sup>	PID-Regler bearbeiten
OB 254 <sup>1)</sup> , 255 <sup>1)</sup>	Datenbausteine (DB/DX) ins DB-RAM übertragen

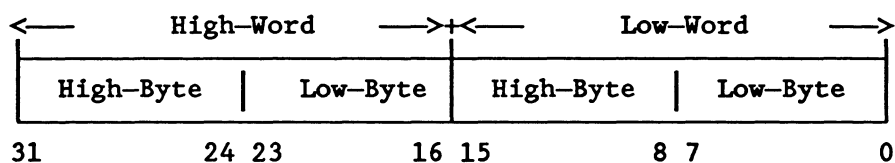
1) Sonderfunktionen mit Pseudobefehlsgrenzen (langlaufend)

2) Die Sonderfunktionen OB 221 und OB 224 wurden aus Kompatibilitätsgründen vom S-Prozessor übernommen und sollten in der CPU 928 nicht programmiert werden. Statt dessen sollte dieses Systemverhalten im DX 0 (siehe Kapitel 7) parametrisiert werden.

**Hinweis:**

Bei den Angaben zur Parametrierung der einzelnen Sonderfunktions-Organisationsbausteine verdeutlichen Sie sich bitte folgende Schreibweise:

Akku 1:	Akku 1,	32 Bit
Akku 1-L:	Akku 1, Low-Wort,	16 Bit
Akku 1-LL:	Akku 1, Low-Wort, Low-Byte,	8 Bit
Akku 1-LH:	Akku 1, Low-Wort, High-Byte,	8 Bit

**Hinweis:**

Unter dem Begriff Parameter sind in der folgenden Beschreibung der einzelnen Sonderfunktionen alle Daten aufgelistet, die der Prozessor benötigt, um die Sonderfunktion korrekt ausführen zu können. Vor Aufruf der Sonderfunktion im STEP5-Programm müssen Sie diese Daten in die Akkus oder in die jeweils angegebenen Speicherzellen laden.

## 6.1 Register-Handling

### 6.1.1 Zugriff auf das Anzeigenbyte (OB 110)

Das Anzeigenregister enthält Informationen über das Ergebnis einer arithmetischen oder logischen Operation und kann über spezielle, anzeigenabhängige Befehle ausgewertet werden.

Mit Hilfe des Sonderfunktions-Organisationsbausteins OB 110 können Sie das Anzeigenregister in den Akku 1 laden oder mit dem Inhalt des Akku 1 beschreiben. Zusätzlich können Sie einzelne Anzeigenbits auf "0" oder "1" setzen.

**Belegung des Akku 1 für den Zugriff auf das Anzeigenregister:**

$2^{31}$	..	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
*)			A1	A0	OV	OS	OR	STA	VKE	$\overline{\text{ER}}$
W O R T - Anzeigen						B I T - Anzeigen				

\*) Bit  $2^8$  bis  $2^{31}$  sind für Erweiterungen reserviert und müssen beim Beschreiben des Anzeigenregisters gleich "0" sein. Sie sollten beim Auslesen des Anzeigenregisters ignoriert werden.

Parameter:

1. Akku 2-L: Funktionskennung  
mögliche Werte: 1, 2 oder 3
2. Akku 1: neues Anzeigenbyte bzw. Maske

Akku 2-L	Akku 1		Funktion:
	vorher	nachher	
1	neues Anzeigenbyte	neues Anzeigenbyte	Der Inhalt von Akku 1 wird in das Anzeigenregister geladen.
2	Maske	neues Anzeigenbyte <sup>1)</sup>	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Anzeigenregister gleich "1" gesetzt. Das neue Anzeigenbyte wird in Akku 1 geladen.
3	Maske	neues Anzeigenbyte <sup>1)</sup>	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Anzeigenregister gleich "0" gesetzt. Das neue Anzeigenbyte wird in Akku 1 geladen.

<sup>1)</sup> Einschränkung:

Die Anzeigenbits OR, STA und  $\overline{\text{ER}}$  können nicht ausgelesen werden, da die Sonderfunktion OB 110 diese Bits immer beeinflusst:  
OR = 0, STA = 1 und  $\overline{\text{ER}}$  = 0

### Fehlerfälle:

- Funktionskennung im Akku 2-L ungleich 1, 2 oder 3.
- Im Akku 1 ist eines der Bits  $2^8$  bis  $2^{31}$  gesetzt.

Im Fehlerfall wird der **OB 31** (Sonstige Laufzeitfehler) aufgerufen und im Akku 1-L die Fehlerkennung 1A49H übergeben.

### **Anwendungsbeispiel**

Der OB 110 ist ein Hilfsmittel zum Testen der Befehle, die das Anzeigenregister auswerten oder beeinflussen. Seine Anwendung ist jedoch nicht auf den Befehlstest begrenzt. Folgendes Beispiel soll einen weiteren Anwendungsfall zeigen.

### **Aufrufverteiler:**

In Abhängigkeit vom Inhalt des Merkerbytes MB0 soll eines von vier Teilprogrammen aufgerufen werden. Den vier Programmteilen werden die Bits M0.0 bis M0.3 zugeordnet. Es darf immer nur eines dieser Bits gesetzt sein.

```
:L    MB0
:SLW  4                ;M0.0 bis M0.3 um vier nach links schieben
:L    KB1              ;Funktionskennung laden
:TAK
:SPA  OB110
:SPS  =M000            ;Sprung falls OS = 1
:SPO  =M001            ;Sprung falls OV = 1
:SPM  =M002            ;Sprung falls ANZ0 = 1
:SPP  =M003            ;Sprung falls ANZ1 = 1
:.
:.
:.                    ;falls kein Bit gesetzt
:.
:BEA
:
M000:.                ;falls M0.0 = 1
:.
:BEA
M001:.                ;falls M0.1 = 1
:.
:BEA
M002:.                ;falls M0.2 = 1
:.
:BEA
M003:.                ;falls M0.3 = 1
:.
:BEA
```

### **6.1.2 Akku 1, 2, 3 und 4 löschen (OB 111)**

Durch den einmaligen Aufruf des Sonderfunktions-Organisationsbausteins OB 111 können Sie die Inhalte der Akkus 1 bis 4 auf einfache Weise löschen: Der OB 111 überschreibt alle vier Register mit '0'.

Parameter: keine

Fehlerfälle: keine



### 6.1.3 Akku Roll Up (OB 112) und Akku Roll Down (OB 113)

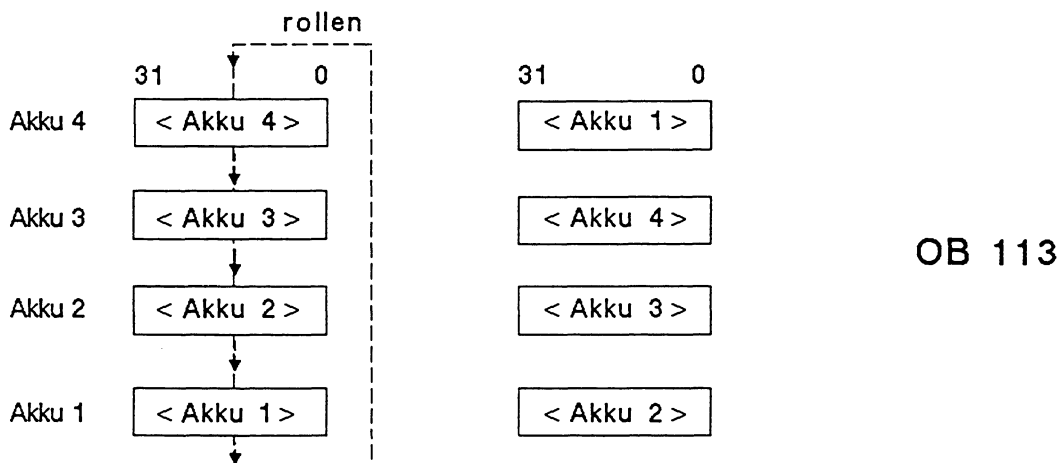
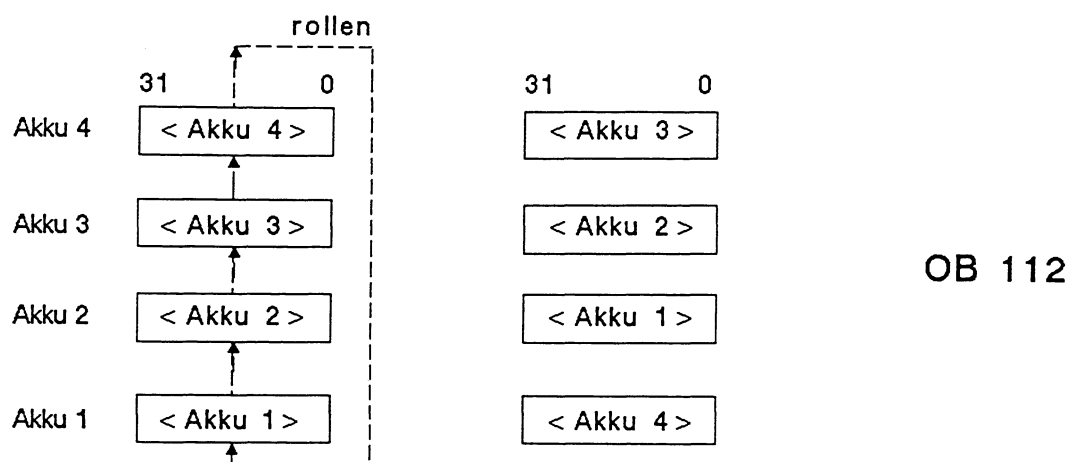
OB 112 und OB 113 bewirken ein 'Rollen' der Akku-Inhalte in aufsteigender bzw. absteigender Richtung:

- Der OB 112 (Roll Up) verschiebt den Inhalt von Akku 1 in den Akku 2, den Inhalt von Akku 2 in den Akku 3 usw..
- Der OB 113 (Roll Down) verschiebt die Akku-Inhalte in entgegengesetzte Richtung: Inhalt von Akku 1 in den Akku 4, Akku 4 in Akku 3 etc..

Parameter: keine

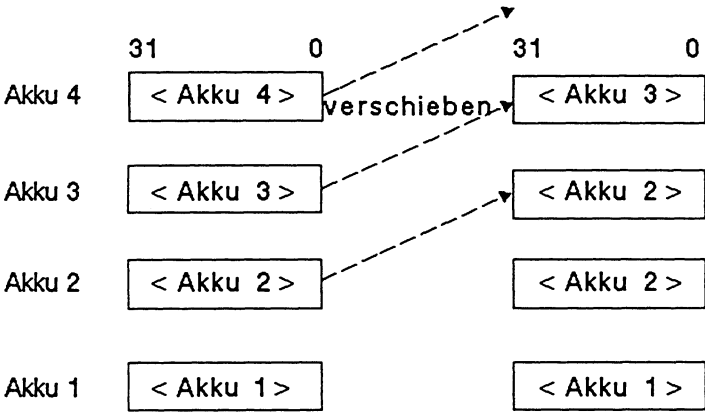
Fehlerfälle: keine

Die nachfolgende Abbildung zeigt die Akku-Inhalte vor und nach dem Aufruf von OB 112 und 113.

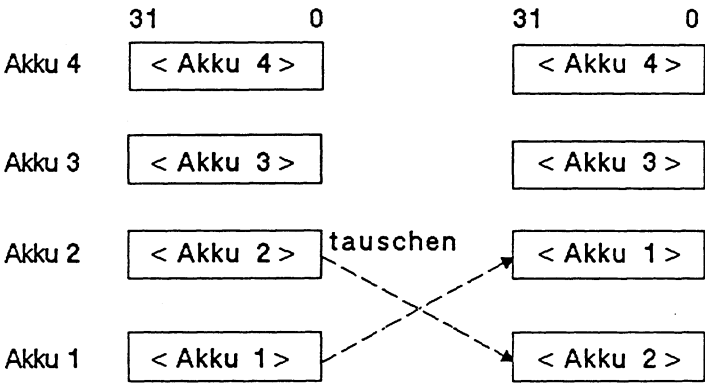


Mit den STEP5-Befehlen ENT (Ergänzender Operationsvorrat) und TAK (Systemoperation) lassen sich die Akku-Inhalte ebenfalls verschieben.

Beachten Sie folgende Unterschiede:



ENT – Befehl



TAK – Befehl

REGISTER.

## 6.2 Strukturbefehle

### 6.2.1 Zählschleifen (OB 160 bis 163)

Mit Hilfe dieser Sonderfunktions-Organisationsbausteine realisieren Sie Programmschleifen mit besonders günstiger Laufzeit.

Jedem der 4 Sonderfunktions-OBs ist ein bestimmtes Systemdatenwort zugeordnet:

BS 60	-->	OB 160
BS 61	-->	OB 161
BS 62	-->	OB 162
BS 63	-->	OB 163

In dieses Systemdatenwort transferieren Sie die erwünschte Anzahl an Schleifendurchläufen. Rufen Sie nun den dazugehörigen Sonderfunktions-OB auf, so wird der Schleifenzähler im Systemdatenwort um eins erniedrigt. Die Schleife wird so lange durchlaufen, bis der Schleifenzähler den Wert Null enthält.

(Enthält der Schleifenzähler bereits vor Aufruf des Sonderfunktions-OBs den Wert Null, so wird er bei Aufruf ebenfalls um eins erniedrigt: Es erfolgen 65 536 Schleifendurchläufe!)

Schleifenzähler im Systemdatenwort > 0 : VKE wird gesetzt (VKE = 1)

Schleifenzähler im Systemdatenwort = 0 : VKE wird gelöscht (VKE = 0)

Die restlichen Bit- und Wortanzeigen werden immer gelöscht!

Die Akkus werden nicht verändert und nicht ausgewertet. Somit stehen diese zu Beginn des nächsten Schleifendurchlaufs noch zur Verfügung und müssen nicht neu hergestellt werden.

Die vier Organisationsbausteine OB 160, 161, 162 und 163 ermöglichen eine vierfache Schachtelung von Schleifen. Sie können damit in den Systemdatenwörtern BS 60 bis 63 vier verschiedene Schleifenzähler einsetzen.

Gegebenenfalls können diese Sonderfunktionen in Verbindung mit dem Befehl B BS (Bearbeite Systemdatum) angewendet werden.

Laufzeit der Sonderfunktionen OB 160 bis 163, falls vor Aufruf der Funktion der Schleifenzähler # 1 war: 12 - 16us

#### Parameter:

1. Systemdatenwort BS 60 - 63: Schleifenzähler  
mögliche Werte: 0 - 65 535 (FFFFH)

Fehlerfälle: keine

### Beispiel:

Im Merkerwort x steht die erwünschte Anzahl an Schleifendurchläufen.

Schleife initialisieren:	:L KB0	
	:L MWx	Schleifenzähler
	:!=F	
	:SPB =M002	
	:T BS62	Schleifenzähler ins Systemdatenwort transferieren
'Schleifenprogramm':	M001: .	
	: .	
	: .	
	: .	
	:	
Schleife verwalten:	:SPA OB162	Zählschleife
	:SPB =M001	bei VKE = 1 wird Schleife erneut durchlaufen
weiter:	M002: .	
	: .	
	: .	
	: .	

(Ein weiteres Beispiel finden Sie im Kapitel 9.2 "TNW und TNB: Speicherblöcke transferieren".)

### 6.3 Bausteinstack (BSTACK) lesen (OB 170)

Im Bausteinstack sind, ausgehend vom OB 1 bzw. FB 0, alle Bausteine eingetragen, die nacheinander aufgerufen worden sind und deren Bearbeitung noch nicht abgeschlossen ist.

Mit Hilfe des Sonderfunktions-Organisationsbausteins OB 170 können Sie die im BSTACK vorhandenen Einträge in einen Datenbaustein einlesen. Auf diese Weise ermitteln Sie die vorhandene Anzahl an BSTACK-Einträgen und damit die Reserve, die Ihnen für weitere Einträge noch zur Verfügung steht.

Zu jedem Eintrag erhalten Sie die jeweilige Rücksprungadresse (Stepadreßzähler = SAZ), die absolute Anfangsadresse des in diesem Baustein gültigen Datenbausteins (DBA) sowie dessen Länge (Anzahl der Datenwörter = DBL).

#### **WICHTIG!**

**Vor Aufruf des OB 170 muß ein ausreichend langer Datenbaustein (DB oder DX) aufgerufen werden! Für jeden gewünschten BSTACK-Eintrag benötigen Sie vier Datenwörter.**

#### Parameter:

1. Akku 2-L: Nummer des Datenworts (DW n), ab dem die Einträge im aufgeschlagenen DB abgelegt werden sollen (Offset)

2. Akku 1-L: gewünschte Anzahl an BSTACK-Elementen

mögliche Werte: 1 - 62

Bsp.: Enthält der Akku 1-L den Wert '1', erhalten Sie den letzten BSTACK-Eintrag, bei '2' den letzten und vorletzten, etc.

#### Nach erfolgreichem Aufruf des OB 170

- steht im Akku 2-L weiterhin der Offset im Datenbaustein,
- steht im Akku 1-L die tatsächlich dargestellte Anzahl an BSTACK-Elementen,

mögliche Werte: 0 - 62, wobei

dargestellte Anzahl  $\leq$  gewünschte Anzahl

0 = 'kein BSTACK-Eintrag vorhanden' oder  
'Fehler'

(Inhalt von Akku 1-L multipliziert mit 4 ergibt die Anzahl der beschriebenen Datenwörter im aufgerufenen DB!)

- wird das VKE gelöscht,
- können die Ergebnisanzeigen ANZ0 und ANZ1 ausgewertet werden (siehe unten),
- sind alle übrigen Bit- und Wortanzeigen gelöscht.

#### Fehlerfälle:

- kein Datenbaustein aufgeschlagen
- aufgeschlagener Datenbaustein nicht vorhanden oder nicht ausreichend lang, um die gewünschte Anzahl von BSTACK-Einträgen aufnehmen zu können
- unzulässige Parameter in Akku 1 und 2

Im Fehlerfall wird das VKE sowie die Ergebnisanzeigen ANZ0 und ANZ1 gesetzt (VKE, ANZ0 und ANZ1 = 1), die übrigen Bit- und Wortanzeigen werden gelöscht. Der Inhalt von Akku 1-L wird zu '0'.

#### **Beeinflussung der Ergebnisanzeigen VKE, ANZ0 und ANZ1**

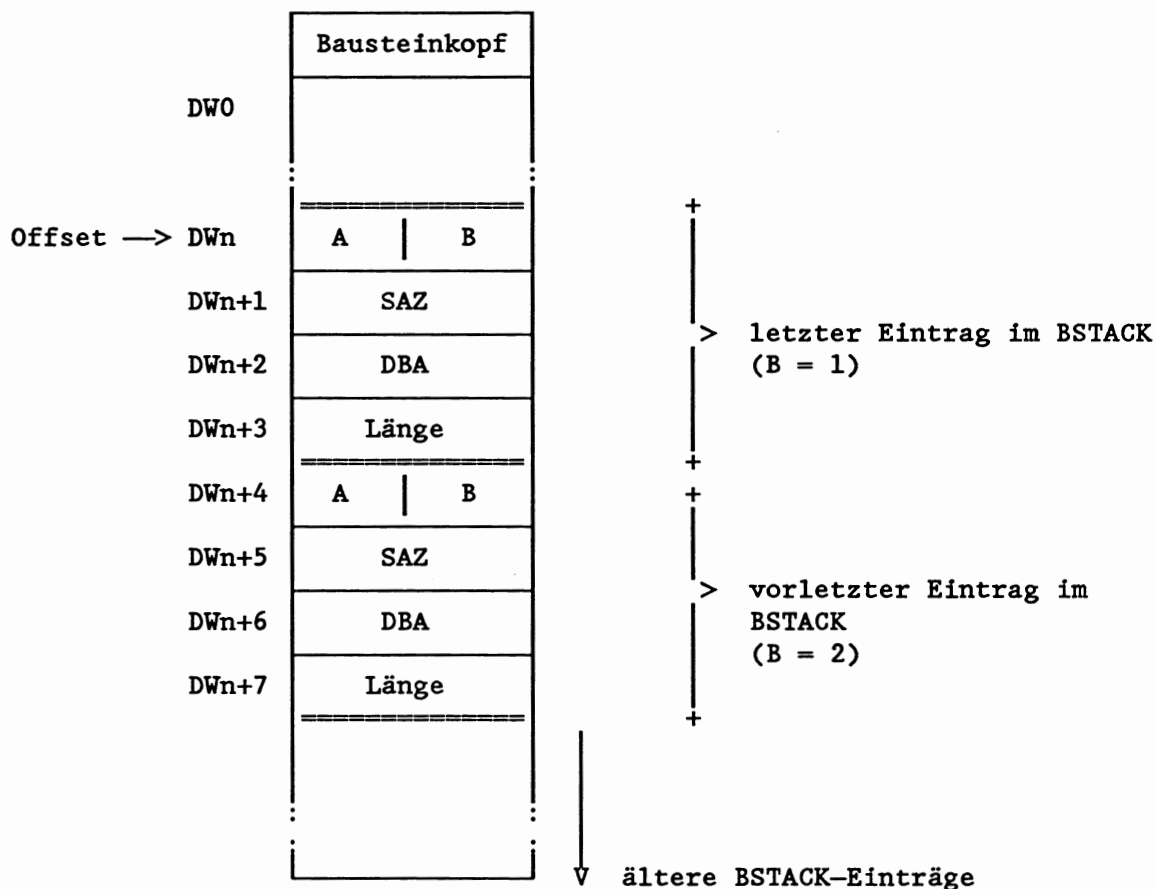
VKE	ANZ0	ANZ1	Abfrage mit	Bedeutung:
0	1	0	SPM	vorhand. Anzahl BSTACK-Elemente < gewünschte Anzahl
0	0	0	SPZ	vorhand. Anzahl BSTACK-Elemente = gewünschte Anzahl
0	0	1	SPP	vorhand. Anzahl BSTACK-Elemente > gewünschte Anzahl
1	1	1	SPB	Fehler

So werden die Inhalte des BSTACKs bei Aufruf des OB 170 im aufgerufenen Datenbaustein abgelegt:

A = BSTACK-Element-Nummer (62 - 1)

(Bereits bei Ausgabe des letzten BSTACK-Elements lässt sich so die Reserve ermitteln: A = 17 --> Reserve = A - 1 = 16)

B = Tiefe des BSTACK-Elementes (1 - 62)

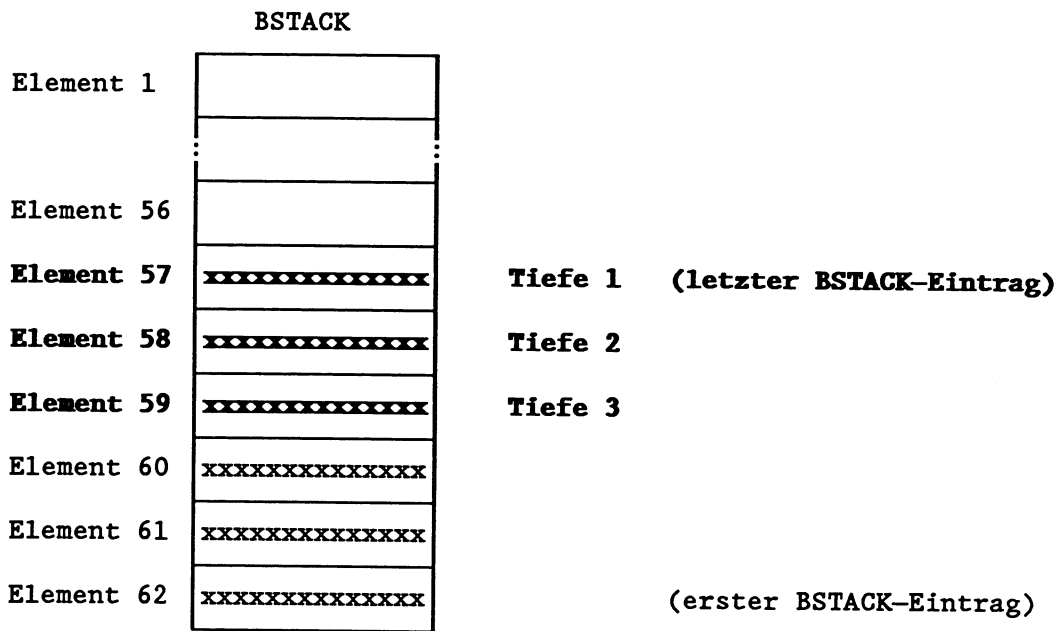


### Beispiel:

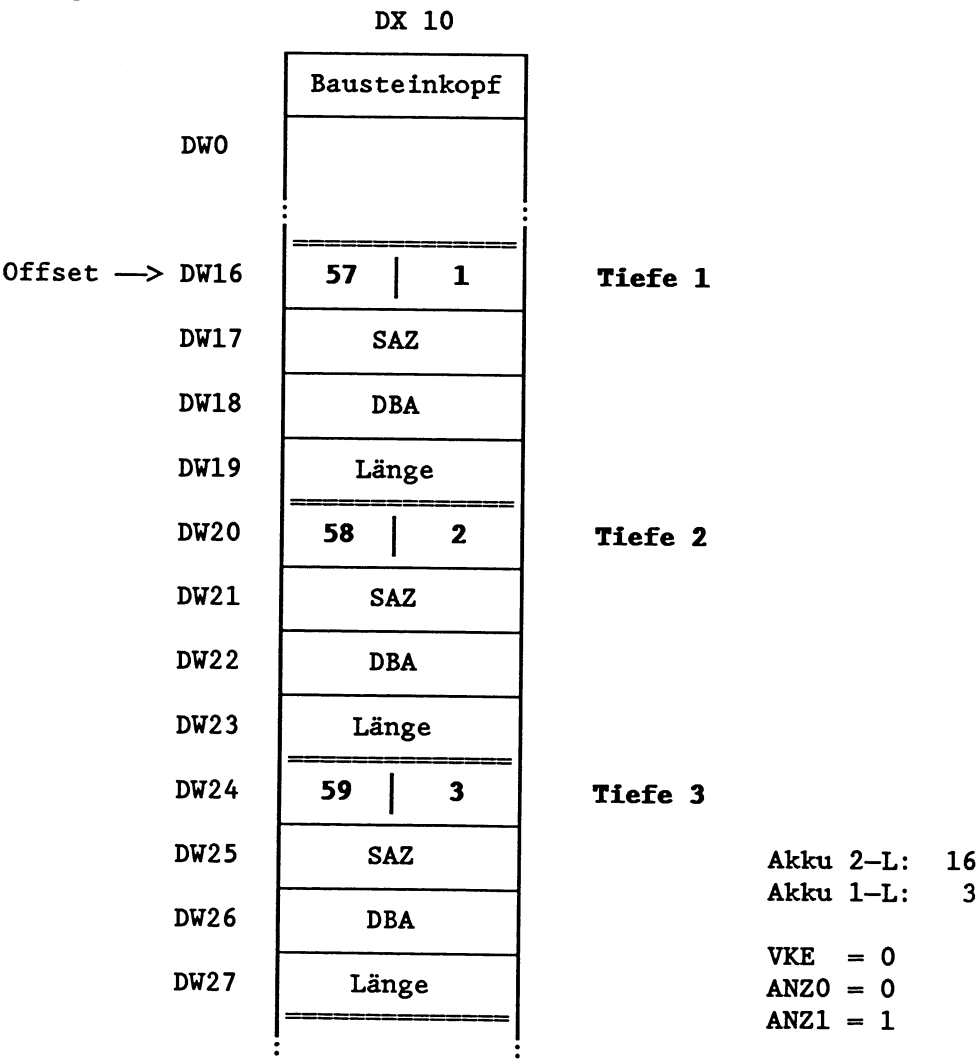
Sie wollen die letzten 3 BSTACK-Einträge in den Datenbaustein DX 10 einlesen. Die Einträge sollen im DX 10 ab Datenwort DW 16 abgelegt werden.

```
:AX DX 10           ;Aufrufen DX 10
:L KY 0,16          ;BSTACK-Einträge sollen ab DW 16 abgelegt werden
:L KY 0,3           ;gewünscht werden die letzten 3 BSTACK-Einträge
:SPA OB 170
```

Im BSTACK sind 6 Bausteine eingetragen:



Nach Aufruf des Sonderfunktions-OBs ist der DX 10 wie folgt belegt:





## 6.4 Baustein-Handling

### 6.4.1 Variabler Datenbaustein-Zugriff (OB 180)

Beim Aufschlagen eines Datenbausteins mit den Befehlen A DB und AX DX wird das 'DBA'-Register (Datenbaustein-Anfangsadresse, Register-Nr. 6) mit der Adresse des Datenwortes DW0 geladen, die im DB0 hinterlegt ist. Der Anwender kann mittels STEP5-Programm (LIR 6, TIR 6) auf das DBA-Register (16 Bit) zugreifen. Zugriffe auf Datenbausteine mit Befehlen wie L DR 60 oder B DW 240 usw. erfolgen immer relativ zur Datenbaustein-Anfangsadresse.

#### WICHTIG!

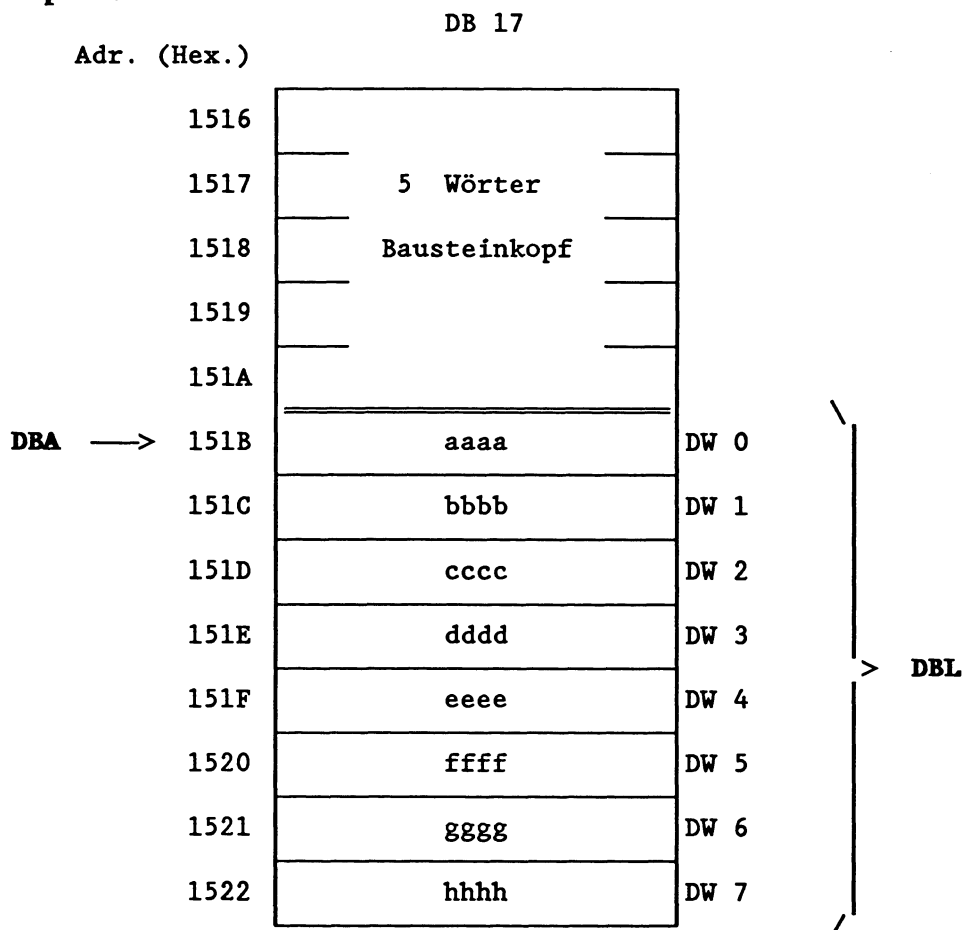
**STEP5-Zugriffe auf Datenwörter sind nur erlaubt bis Datenwort DW 255!**

Zusätzlich zum DBA-Register wird bei jedem Aufruf eines Datenbausteins das 'DBL'-Register (Datenbaustein-Länge, Register-Nr. 8) geladen: Es enthält die Länge (in Wörtern) des aufgeschlagenen DB- oder DX-Datenbausteins ohne Baustein-Kopf. Der Anwender kann mittels STEP5-Programm (LIR 8, TIR 8) auf das DBL-Register (16 Bit) zugreifen.

#### WICHTIG!

**Im DBL-Register kann - je nach Speichergröße des verwendeten PGs - eine maximale Länge von 4091 Datenwörtern eingetragen sein!**

#### Beispiel:



Das DBA-Register beinhaltet die Adresse des Speicherwortes, in welchem das DW 0 hinterlegt ist, im Beispiel: DBA = 151B (Hexadez.).

Das DBL-Register beinhaltet die Anzahl der Datenwörter, im Beispiel: DBL = 8 (DW 0 bis DW 7).

Da der Zugriff auf Datenwörter mittels der STEP5-Befehle L DW, U D, B DW usw. immer relativ zum DBA erfolgt, wird, um z.B. auf das DW 3 zuzugreifen, 3 zu 151B addiert. Die Adresse 151E beinhaltet das DW 3.

Bei schreibenden Zugriffen wird anhand des DBL-Registers geprüft, ob ein Transferfehler vorliegt. Z.B. ist T DW 7 erlaubt, T DW 8 ist fehlerhaft.

Der Sonderfunktions-OB 180 erhöht den DBA-Register-Inhalt um eine vorgegebene Anzahl von Datenwörtern. Damit können Sie mit STEP5-Befehlen auch auf Datenbausteine zugreifen, die länger als 256 Datenwörter sind.

Durch entsprechende Parametrierung und anschließenden Aufruf des OB 180 läßt sich der STEP5-Zugriffsbereich von 256 Datenwörtern beliebig in einem Datenbaustein verschieben.

#### **WICHTIG!**

**Vor Aufruf des OB 180 muß ein ausreichend langer Datenbaustein (DB oder DX) aufgeschlagen sein.**

#### Parameter:

1. Akku 1-L: Versatz (Anzahl der Datenwörter, um die die Datenbausteinanfangsadresse verschoben werden soll)

mögliche Werte: Inhalt Akku 1-L < DBL !

Nach erfolgreichem Aufruf des OB 180

- ist der Wert des DBA-Registers (= Adresse des DW 0) um den Wert des Akku 1-L erhöht,
- ist der Wert des DBL-Registers um den Wert des Akku 1-L erniedrigt,
- ist das VKE gelöscht (VKE = 0),
- sind alle übrigen Bit- und Wortanzeigen gelöscht.

#### Fehlerfälle:

- kein Datenbaustein aufgeschlagen
- Inhalt Akku 1-L  $\geq$  DBL

Im Fehlerfall (Inhalt Akku 1-L  $\geq$  DBL) bleiben DBA- und DBL-Register unbeeinflusst. Das VKE wird gesetzt (VKE = 1). Die übrigen Bit- und Wortanzeigen werden gelöscht.  
 Enthält das DBL-Register den Wert '0', so erkennt der OB 180, daß kein Datenbaustein aufgeschlagen ist. Das VKE wird gesetzt (VKE = 1) und signalisiert so einen Fehler.

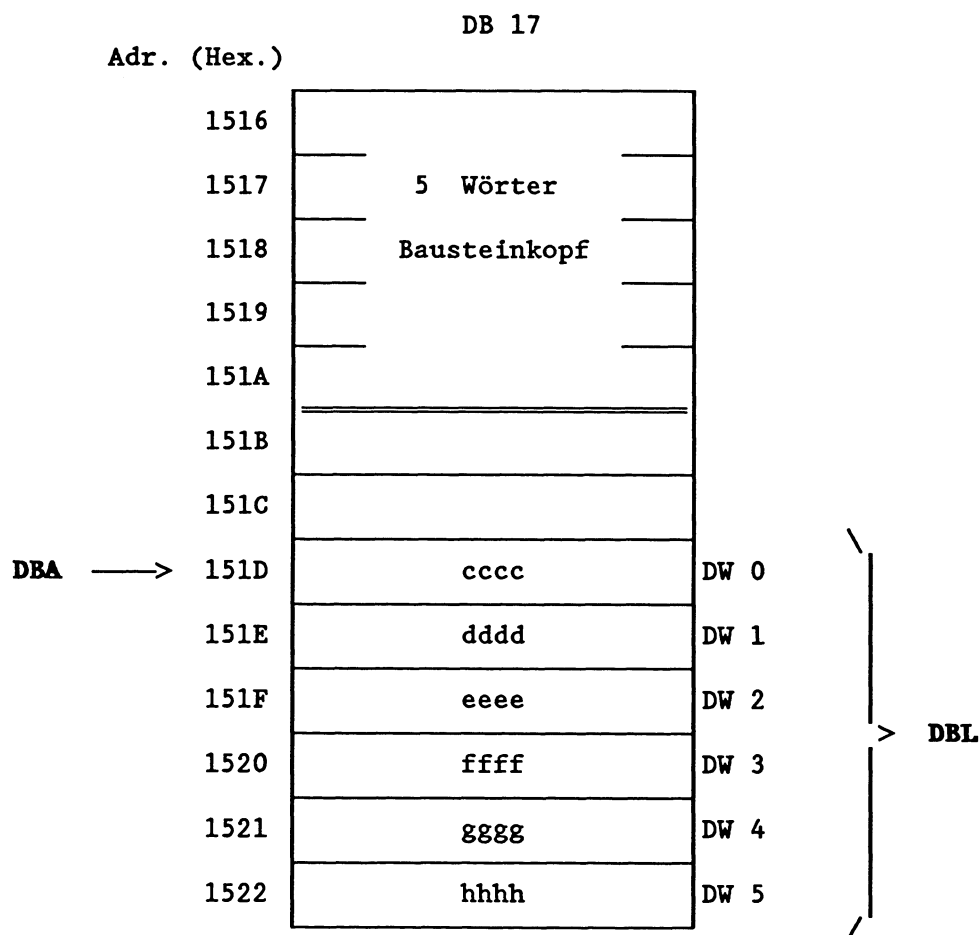
### DBA und DBL auf Anfangswert zurückstellen

Ein erneutes Aufschlagen des Datenbausteins mit den Befehlen A DB oder AX DX stellt den Grundzustand wieder her.

### Beispiel:

Die Datenbausteinanfangsadresse (DBA = 151B) im DB 17 (DBL = 8) soll um zwei Datenwörter verschoben werden.

A DB 17  
 L KB 2  
 SPA OB 180



DBA nach Aufruf des OB 180 = 151D (hex.)

DBL nach Aufruf des OB 180 = 6 (DW 0 bis DW 5)

Nach Aufruf des OB 180 lässt sich z.B. das unter der Adresse 1520 gespeicherte Datenwort mit dem Inhalt 'ffff' nicht mehr mit DW 5, sondern mit DW 3 ansprechen usw..

Aufgrund des gleichzeitig veränderten DBLs bleibt die **Transferfehler-Überwachung gewährleistet**: Der Befehl T DW 5 ist erlaubt, während T DW 6 bereits fehlerhaft ist.

Durch erneute Aufrufe des OB 180 kann das DBA weiter erhöht werden (wobei das DBL weiter verringert wird); der Befehl A DB 17 stellt den Grundzustand (DBA = 151B, DBL = 8) wieder her.

Hätte der DB 17 eine Länge von z.B. 258 Datenwörtern, könnten Sie mit STEP5-Befehlen nicht mehr auf DW 256 und DW 257 zugreifen. Durch Verschieben des DBA um 2 lassen sich die Datenwörter 256 und 257 mit "DW 254" und "DW 255" ansprechen.

### **Mögliche Anwendungen des OB 180**

- Zugriff bei überlangen DBs, d.h. bei DBs mit einer Länge größer 261 Wörter (5 Wörter Kopf, DW 0 - DW 255), siehe oben.
- Handhabung von Datenstrukturen

Falls ein Datensatz, bestehend aus mehreren Datenwörtern, mehrmals innerhalb eines Prozessors existiert, wobei die Belegung (Bedeutung) der Datenwörter immer gleich ist, so spricht man von einer Datenstruktur.

Z.B. könnte die Beschreibung eines (Teil-)Prozesszustandes 20 Datenwörter umfassen, wobei das 1. Datenwort eine Temperatur beinhaltet, das 2. Datenwort einen Druck usw.. Falls nun dieser Prozesszustand **mehrmals und innerhalb eines DBs** hinterlegt werden soll, z.B., weil der Teilprozess mehrfach existiert und/oder weil Vergangenheitswerte gespeichert werden sollen, so kann mittels des Sonderfunktions-OBs OB 180 auf jede Struktur mit den gleichen Befehlen L DD, S D, T DR usw. mit denselben Parametern 0 bis 19 zugegriffen werden.

Im Gegensatz zu anderen Substitutionsmechanismen (Substitution = indizierte Parametrierung) ergeben sich einfachere und laufzeitgünstigere Unterprogramme.

(Zum DBA-Register siehe Kapitel 9 "Speicherzugriffe über absolute Adressen", Register 6)

#### 6.4.2 Datenbausteine (DB/DX) testen (OB 181)

Mit dem Sonderfunktions-Organisationsbaustein OB 181 können Sie prüfen,

- a) ob ein bestimmter DB- oder DX-Datenbaustein vorhanden ist,
- b) unter welcher Adresse das erste Datenwort des Datenbausteins abgelegt ist,
- c) wieviele Datenwörter dieser Datenbaustein enthält
- d) den Speichertyp und Bereich (Anwenderspeicher: RAM oder EPROM, DB-RAM).

Eine Anwendung der Funktion "DB/DX testen" ist sinnvoll vor den Befehlen TNB/TNW, E DB/EX DX und vor Aufruf der Sonderfunktions-Organisationsbausteine OB 254 und OB 255.

So können Sie beispielsweise vor einem Blocktransfer von Datenwörtern den OB 181 aufrufen, um sicherzustellen, daß der Zieldatenbaustein gültig ist und lang genug, alle zu kopierenden Datenwörter aufzunehmen.

##### Parameter:

1. Akku 1-LL: Bausteinnummer  
mögliche Werte: 1 bis 255
2. Akku 1-LH: Bausteinkennung  
mögliche Werte: 1 = DB  
2 = DX

Ist der geprüfte Baustein im Prozessor vorhanden,

- enthält der Akku 1-L die Adresse des ersten Datenwortes (DW 0),
- der Akku 2-L die Länge des Datenbausteins in Wörtern (ohne Bausteinkopf),  
Beispiel: Im Akku 2-L steht der Wert '7' --> Der Datenbaustein besteht aus DW 0 bis DW 6.
- so wird das VKE gelöscht (VKE = 0),
- die Wortanzeigen ANZ0 und ANZ1 werden beeinflusst (siehe folgende Liste),
- die restlichen Bit- und Wortanzeigen werden gelöscht.

Ist der geprüfte Baustein im Speicher nicht vorhanden oder die Parametrierung ist falsch,

- so wird das VKE gesetzt (VKE = 1),
- die Wortanzeigen ANZ0 und ANZ1 werden beeinflusst (siehe folgende Liste),
- die restlichen Bit- und Wortanzeigen werden gelöscht,
- die Akkus werden nicht verändert.

#### Fehlerfälle:

- falsche Bausteinnummer (unzulässig: 0)
- falsche Bausteinkennung (unzulässig: 0, 3 bis 255)
- Speicherfehler

#### **Übersicht: Beeinflussung der Ergebnisanzeigen VKE, ANZ0 und ANZ1**

VKE = 0: DB vorhanden

VKE = 1: DB nicht vorhanden oder Fehler

ANZ1 = 0: DB im Anwendermodul

ANZ1 = 1: DB im DB-RAM

ANZ0 = 0: DB im Schreib-Lese-Speicher

ANZ0 = 1: DB im Nur-Lese-Speicher

VKE	ANZ0	ANZ1	Abfrage	Bedeutung	
0	1	0	SPM	DB in AW-Modul	DB im EPROM (nur lesbar)
0	0	0	SPZ	DB im DB-RAM	DB im RAM (schreib- und lesbar)
0	0	1	SPP	DB im DB-RAM	DB im RAM (schreib- und lesbar)
1	1	1	SPB	DB nicht vorhanden/Speicherfehler/fehlerhafte Parametrierung	

Beispiele siehe

Kapitel 8.2.2 "Bausteinadresslisten im DB-RAM"

Kapitel 9.1 "LIR und TIR: Zugriffe auf Register"

Kapitel 9.2 "TNW und TNB: Speicherblöcke transferieren"

### 6.4.3 Merker in Datenbaustein übertragen (OB 190 und OB 192)

Die Organisationsbausteine OB 190 und OB 192 übertragen eine vom Anwender vorgegebene Anzahl an Merkerbytes in einen dafür vorgesehenen Datenbaustein.

Dies kann z.B. von Vorteil sein vor Bausteinaufrufen, in Fehler-Organisationsbausteinen oder bei Unterbrechung der zyklischen Programmbearbeitung durch eine zeit- oder alarmgesteuerte Programmbearbeitung.

Mit Hilfe des OB 191 und 193 können Sie diese Merkerbytes anschließend wieder aus dem Datenbaustein zurückschreiben lassen.

Hinweis: Verwenden Sie die OBs 190 und 191 für das einfache Retten und Zurücklesen der Merkerbytes, da Sie damit erhebliche Laufzeitvorteile gewinnen.

#### WICHTIG!

**Vor dem eigentlichen Aufruf muß ein Datenbaustein (DB/DX) aufgeschlagen werden!**

Nach Aufruf des OB 190/192 werden im aufgeschlagenen Datenbaustein die Merkerbytes ab der angegebenen Datenwortadresse zwischengespeichert. Der Bereich der zu rettenden Merker entnimmt der OB 190/192 dem Akku.

Die Sonderfunktions-Organisationsbausteine OB 190 und OB 192 sind identisch mit Ausnahme der Art und Weise, in der sie die Merkerbytes übertragen:

- Der OB 190 überträgt die Merker byteweise.
- Der OB 192 überträgt die Merker wortweise.

Dies ist von Belang, wenn die in den Datenbaustein übertragenen Daten anschließend bearbeitet werden sollen und der Datenbaustein nicht nur als einfacher Zwischenspeicher benutzt wird.

Die folgende Abbildung soll diesen Unterschied verdeutlichen:

Merker kopiert mit

**OB190:**

**OB192:**

7    ...    0

0
1
2
3
4
⋮
⋮
255

DL                      DR  
15    ...    8 7    ...    0

1	0
3	2
	4

DL                      DR  
15    ...    8 7    ...    0

0	1
2	3
4	

Hinweis: Falls eine ungerade Anzahl von Merkerbytes übertragen wird, so wird das letzte benutzte Datenwort des Datenbausteins nur zur Hälfte genutzt. Bei OB 190 bleibt das Datum links, bei OB 192 das Datum rechts frei.

**WICHTIG!**

**Bitte beachten Sie folgende Laufzeiten (in  $\mu$ s):**

(n = Anzahl der Merkerbytes)	OB 190	OB 192
Die Nummer des ersten Merkerbytes ist <b>gerade</b> :	$25 + n \cdot 0,32$	$40 + n \cdot 0,57$
Die Nummer des ersten Merkerbytes ist <b>ungerade</b> :	$25 + n \cdot 0,48$	$25 + n \cdot 1,8$

Parameter:

Angaben zur Quelle:

1. Akku 2-LH: Erstes zu übertragendes Merkerbyte  
mögliche Werte: 0 bis 255
2. Akku 2-LL: Letztes zu übertragendes Merkerbyte  
mögliche Werte: 0 bis 255

(Letztes Merkerbyte  $\geq$  Erstes Merkerbyte !)

Angaben zum Ziel:

3. Akku 1-L: Nummer des ersten zu beschreibenden Datenwortes  
im aufgeschlagenen Datenbaustein  
mögliche Werte:
  - orientiert sich an der Länge des Datenbausteins im Speicher
  - es können dabei Nummern  $> 255$  auftreten

Wird der Sonderfunktions-OB 190/192 korrekt bearbeitet, so wird das VKE gelöscht (VKE = 0). Die Akkus bleiben unverändert.

Im Fehlerfall wird das VKE gesetzt (VKE = 1), die Akkus bleiben unverändert.

Fehlerfälle:

- kein DB- oder DX-Datenbaustein aufgeschlagen
- falscher Merkerbereich (Letztes Merkerbyte  $<$  Erstes Merkerbyte)
- Datenwort-Nummer nicht vorhanden
- Länge des DB- oder DX-Datenbausteins nicht ausreichend



#### 6.4.4 Datenblöcke in Merkerbereich übertragen (OB 191 und OB 193)

Mit Hilfe des OB 191 und OB 193 können Sie Daten aus einem Datenbaustein in den Merkerbereich übertragen. So können beispielsweise die zuvor in einen Datenbaustein 'geretteten' Merkerbytes wieder in den Merkerbereich zurückgeschrieben werden.

Die OBs 191/193 unterscheiden sich von den OBs 190/192 nur dadurch, daß Quelle und Ziel vertauscht sind:

OB 190/192: Merkerbereich --> Merker --> Datenbaustein

OB 191/193: Merkerbereich <-- Daten <-- Datenbaustein

#### WICHTIG!

**Vor dem eigentlichen Aufruf muß ein Datenbaustein (DB/DX) aufgeschlagen werden!**

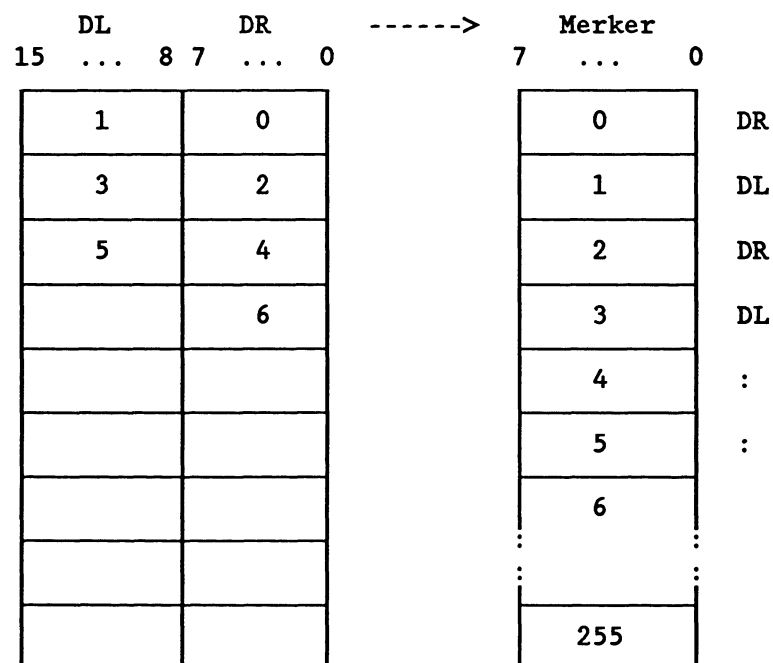
Die Sonderfunktions-Organisationsbausteine OB 191 und OB 193 sind identisch mit Ausnahme der Art und Weise, in der sie die Daten übertragen:

- Der OB 191 überträgt die Datenwörter byteweise.
- Der OB 193 überträgt die Datenwörter wortweise.

Die folgende Abbildung soll diesen Unterschied verdeutlichen:

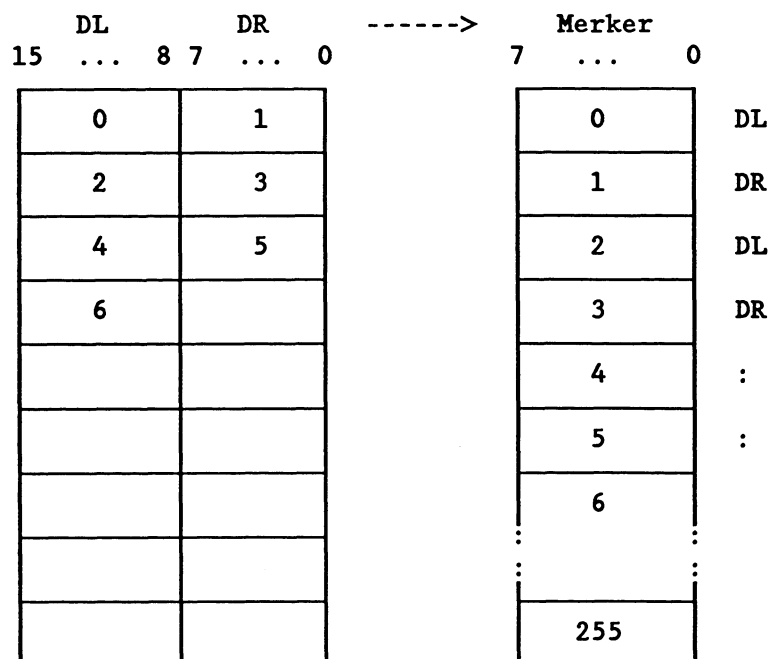
Daten übertragen mit

OB191:



Daten übertragen mit

OB193:



# **WICHTIG!**

**Bitte beachten Sie folgende Laufzeiten (in us):**

(n = Anzahl der Merkerbytes)	OB 191	OB 193
Die Nummer des ersten Merkerbytes ist <b>gerade</b> :	$25 + n \cdot 0,32$	$40 + n \cdot 0,57$
Die Nummer des ersten Merkerbytes ist <b>ungerade</b> :	$25 + n \cdot 0,48$	$25 + n \cdot 1,8$

## Parameter:

Angaben zur Quelle:

1. Akku 2-L: Nummer des ersten zu übertragenden Datenwortes im aufgeschlagenen Datenbaustein

Angaben zum Ziel:

2. Akku 1-LH: Erstes zu beschreibendes Merkerbyte  
mögliche Werte: 0 bis 255
3. Akku 1-LL: Letztes zu beschreibendes Merkerbyte  
mögliche Werte: 0 bis 255

(Letztes Merkerbyte  $\geq$  Erstes Merkerbyte !)

Wird der Sonderfunktions-OB 191/193 korrekt bearbeitet, so wird das VKE gelöscht (VKE = 0). Die Akkus bleiben unverändert.

Im Fehlerfall wird das VKE gesetzt (VKE = 1), die Akkus bleiben unverändert.

Fehlerfälle: siehe OBs 190/192

### Beispiel:

Vor Aufruf des Programmbausteins PB 12 sind alle Merker (MB 0 bis MB 255) in den Datenbaustein DX 37 ab Adresse 100 zu retten und anschließend wieder zurückzuschreiben.

Retten:	:AX DX37	Datenbaustein aufrufen
	:L KY0,255	Merkerbereich MB0 bis MB255
	:L KB100	Nummer des 1. Datenworts
	:SPA OB190	Merker retten

Bausteinwechsel: :SPA PB12

Zurückschreiben:	:	(Datenbaustein bereits aufgerufen)
	:L KB100	Nummer des 1. Datenworts
	:L KY0,255	Merkerbereich MB0 bis MB255
	:SPA OB191	Merker zurückschreiben

### Anwendungsbeispiel zum OB 190 / OB 191

Merker, die vom zyklischen Anwenderprogramm benutzt werden, können nicht zusätzlich durch ein zeit- oder alarmgesteuertes Anwenderprogramm genutzt werden. Jeder Programmbearbeitungsebene muß ein bestimmter Teil des Merkerbereichs zugeordnet sein.

Z.B.: zyklisches Anwenderprogramm: MB0...MB99  
zeitgesteuertes Anwenderprogramm: MB100...MB199  
alarmgesteuertes Anwenderprogramm: MB200...MB255

Falls jedoch das zyklische Anwenderprogramm bereits alle 256 Merkerbytes benutzt und beispielsweise das zeitgesteuerte Anwenderprogramm ebenfalls alle 256 Merkerbytes benötigt, müssen die Merker beim Wechsel der Bearbeitungsebene ausgetauscht und zwischengespeichert werden.

### OB13

A DB100  
MB 0...255 → DW 0...127  
DW 128...255 → MB 0...255

zeitgesteuertes  
Anwenderprogramm

A DB100  
MB 0...255 → DW 128..255  
DW 0...127 → MB 0...255  
BE

### DB100

DW0	:	Merker des zyklischen Anwenderprogramms
:	:	
:	:	
DW127	:	Merker des zeitgesteuerten Anwenderprogramms
DW128	:	
:	:	
:	:	
DW255	:	

Am schnellsten können die Merker mit Hilfe der Sonderfunktionen OB 190 und OB 191 gerettet und geladen werden:

#### STEP5-Programm im OB 13

```

:A   DB100
:L   KY0,255
:L   KB0
:SPA OB190
:L   KB128
:L   KY 0,255
:SPA OB191
: .
: .
: .
:A   DB100
:L   KY0,255
:L   KB128
:SPA OB190
:L   KB0
:L   KY0,255
:SPA OB191
:BE

```

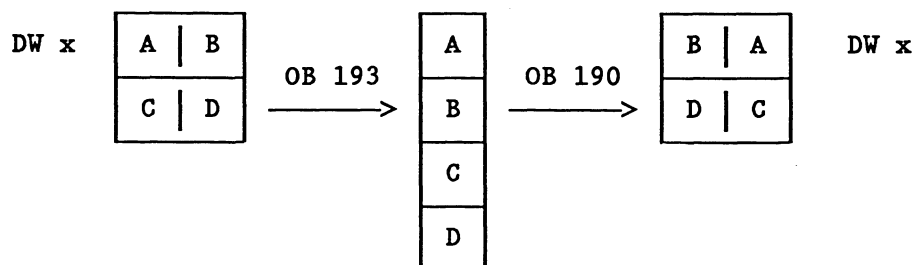
Zum Austauschen und Zwischenspeichern der Merker werden pro OB13-Aufruf 704  $\mu$ s benötigt.

#### Weitere Anwendungen für die Organisationsbausteine OB 190 bis 193

- Bei der CPU 928 werden Befehle zur Einzelbitverarbeitung (U, O, ON, UN, S, R, =), die auf den Merkerbereich zugreifen, um ein Vielfaches schneller bearbeitet als vergleichbare Befehle, die auf Datenbausteine zugreifen (vergleichen Sie hierzu z.B. die Befehle 'U M' <-> 'U D' oder 'S M' <-> 'S D'!).

Aus diesem Grund verbessern Sie die Laufzeit, wenn Sie die Daten in den Merkerbereich kopieren, diese dort bearbeiten lassen und anschließend wieder in den Datenbaustein zurückübertragen.

- Ohne großen Aufwand lassen sich im Datenbaustein High Byte und Low Byte vertauschen, indem mit den entsprechenden OBs die Datenwörter in den Merkerbereich und wieder zurück übertragen werden:



- Sie können Datenblöcke innerhalb eines Datenbausteins 'verschieben', wenn Sie beim Zurückübertragen aus dem Merkerbereich zwar dieselbe DB-Nummer, jedoch ein anderes Datenwort angeben.
- Ebenso einfach lassen sich Datenblöcke (maximal 255 Bytes) in andere Datenbausteine übertragen (eventuell vorher den OB 181 "Datenbausteine (DB/DX) testen" einsetzen).

#### 6.4.5 Datenbausteine ins DB-RAM übertragen (OB 254, OB 255)

Mit den Sonderfunktions-Organisationsbausteinen OB 254 und OB 255 übertragen Sie Datenbausteine vom Anwenderspeicher in das DB-RAM (Datenbausteinspeicher) des Prozessors. Die Sonderfunktionen OB 254 und 255 laufen identisch ab, wobei der OB 254 für DX-Bausteine und der OB 255 für DB-Bausteine zuständig ist.

Beim Übertragen können Sie die Datenbausteine verschieben oder duplizieren lassen:

- **Verschieben** eines Datenbausteins vom Anwenderspeicher ins DB-RAM

Ein Datenbaustein im Anwenderspeicher wird unter Beibehaltung seiner ursprünglichen Bausteinnummer ins DB-RAM verschoben. Die neue Anfangsadresse des Datenbausteins wird in die Adreßliste im DB 0 eingetragen. Die alte Adresse des Bausteins wird überschrieben, d.h., der Datenbaustein wird im Anwenderspeicher für ungültig erklärt.

##### Parameter:

1. Akku 1-L: Nummer des zu verschiebenden Datenbausteins
2. Akku 1-H: 0

##### Fehlerfälle:

- Der zu verschiebende Datenbaustein ist nicht vorhanden (OB19).
- Der Baustein ist bereits im DB-RAM vorhanden (OB31).  
(Funktion nur einmal - vorzugsweise im Anlauf - ausführen)
- Der Speicherplatz im DB-RAM ist nicht ausreichend (OB31).

Im Fehlerfall wird die Funktion nicht ausgeführt. Das Systemprogramm erkennt einen Laufzeitfehler und ruft den **OB 19 oder OB 31** auf. Die weitere Fehlerreaktion hängt von der Programmierung des OB 19 bzw. 31 ab.

Ist der OB 19 bzw. 31 nicht programmiert, geht der Prozessor in den Stoppzustand über. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern.

- **Duplizieren** eines Datenbausteins ins DB-RAM

Ein Datenbaustein im Anwenderspeicher oder im DB-RAM wird ins DB-RAM übertragen, wobei er eine andere Baustein-Nummer erhält. Die Anfangsadresse des neuen Datenbausteins wird in die Adreßliste im DB 0 eingetragen. Die Anfangsadresse des alten Bausteins im DB 0 bleibt erhalten, d.h. der ursprüngliche Datenbaustein ist weiterhin gültig.

Die Eintragung der Anfangsadresse in den DB0 erfolgt erst dann, wenn die Übertragung vollständig abgeschlossen ist und alle Kennungen im Bausteinkopf richtig eingetragen sind. Der duplizierte Baustein wird also vom Systemprogramm erst nach der kompletten Übertragung als gültig bzw. vorhanden erkannt.

Parameter:

1. Akku 1-L: Nummer des zu duplizierenden Datenbausteins
2. Akku 1-H: Nummer des neuen Datenbausteins

Fehlerfälle:

- Der zu duplizierende Baustein ist nicht vorhanden (OB19).
- Der neue Baustein ist bereits vorhanden (OB31).
- Der Speicherplatz im DB-RAM ist nicht ausreichend (OB31).

Im Fehlerfall wird die Funktion nicht ausgeführt. Das Systemprogramm erkennt einen Laufzeitfehler und ruft den **OB 19 oder 31** auf. Die weitere Fehlerreaktion hängt von der Programmierung des OB 19 bzw. OB 31 ab (siehe 'Sonstige Laufzeitfehler'). Ist der OB 19 bzw. 31 nicht programmiert, geht der Prozessor in den Stoppzustand über. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern.

## **6.5 Mehrprozessor-Kommunikation (OB 200 bis OB 205)**

Die Sonderfunktions-Organisationsbausteine OB 200 bis OB 205 ermöglichen im Mehrprozessorbetrieb Datenübertragungen zwischen den einzelnen Prozessoren unter Verwendung des Koordinators KOR C.

### **OB 200: Initialisieren**

Dieser Sonderfunktions-Organisationsbaustein richtet im Koordinator KOR C den Speicher ein, in dem die zu übertragenden Datenblöcke vorübergehend zwischengespeichert werden müssen.

### **OB 202: Senden**

Diese Funktion übergibt einen Datenblock in den Zwischenspeicher des KOR C und gibt an, wieviele Datenblöcke noch gesendet werden können.

### **OB 203: Sende-Test**

Der Sonderfunktions-OB 203 ermittelt die Anzahl der freien Speicherblöcke im Zwischenspeicher des Koordinators KOR C.

### **OB 204: Empfangen**

Diese Funktion übernimmt einen Datenblock vom Zwischenspeicher des KOR C und zeigt an, wieviele Datenblöcke noch empfangen werden können.

### **OB 205: Empfangs-Test**

Der Sonderfunktions-Organisationsbaustein OB 205 ermittelt die Anzahl belegter Speicherblöcke im Zwischenspeicher des KOR C.

Eine detaillierte Bedienungsanleitung zu diesen Sonderfunktions-Organisationsbausteinen befindet sich in Register 8 dieses Gerätehandbuchs.



## 6.6 Kachelzugriffe

Die Organisationsbausteine OB 216 bis 218 ermöglichen den Zugriff auf sogenannte Kacheln.

Die Organisationsbausteine enthalten folgende Funktionen:

- OB 216      Schreiben eines Bytes/Wortes/Doppelwortes auf eine Kachel
- OB 217      Lesen eines Bytes/Wortes/Doppelwortes von einer Kachel
- OB 218      Belegen einer Kachel durch den Prozessor  
(dient der Koordinierung im Mehrprozessorbetrieb)

Diese Funktionen dienen einerseits Testzwecken; zugleich ermöglichen diese Elementarfunktionen die Programmierung von Hantierungsbausteinen oder ähnlichen Funktionen.

Im Mehrprozessorbetrieb ist die Verwendung dieser Sonderfunktions-OBs besonders dann zu empfehlen, wenn Informationen von verschiedenen Prozessoren zu einer Kachel geschrieben bzw. aus einer Kachel gelesen werden sollen.

### Was sind Kacheln?

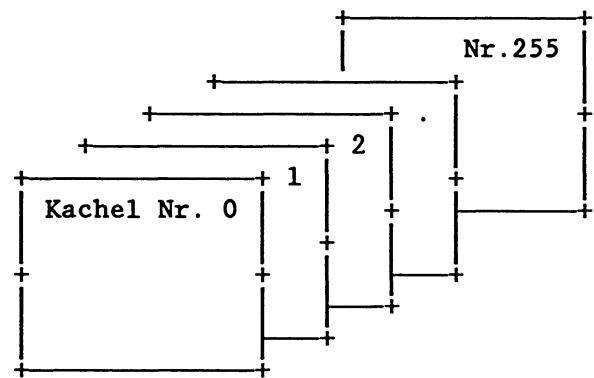
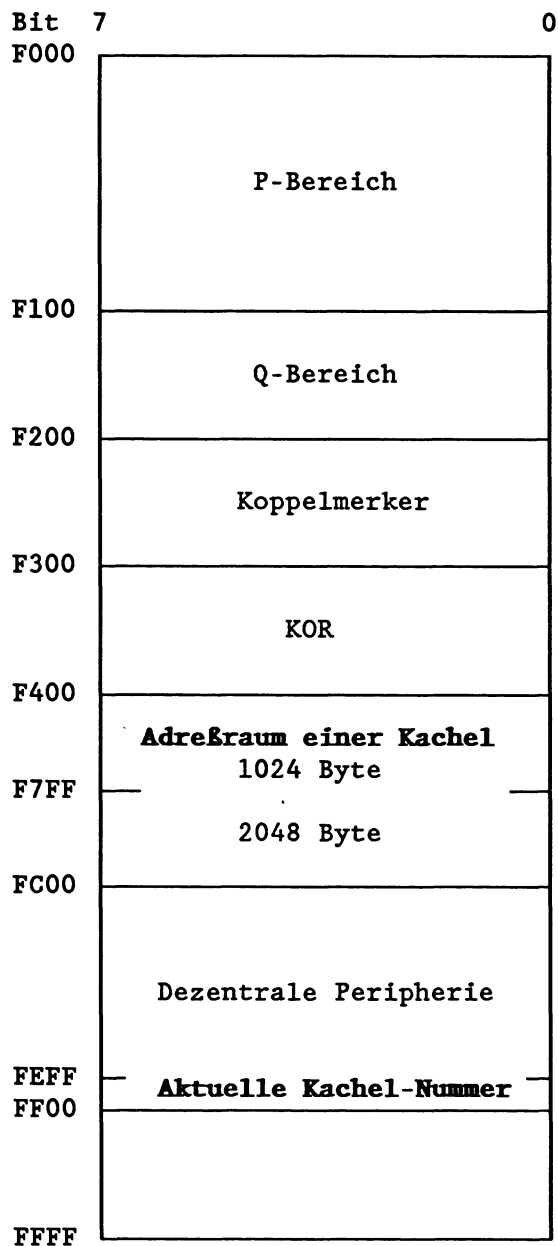
Kacheln sind Speicherbereiche, die auf Kommunikationsprozessoren, auf bestimmten intelligenten Peripheriebaugruppen und bestimmten Koordinatoren für Mehrprozessorbetrieb jeweils einfach oder mehrfach vorhanden sind.

Kacheln sind byteweise organisiert, d.h., jedes Byte ist **einzel**n adressierbar.

Es können bis zu 255 Kacheln im AG sein.

Kachelgröße:	belegter Adreßraum:
1024 Byte	F400H - F7FFH
2048 Byte	F400H - FBFFH

## Adreßbereiche für Peripherie auf dem S5-Bus



mehrfach vorhandener  
Speicherbereich  
Größe: 1024 oder 2048 Bytes

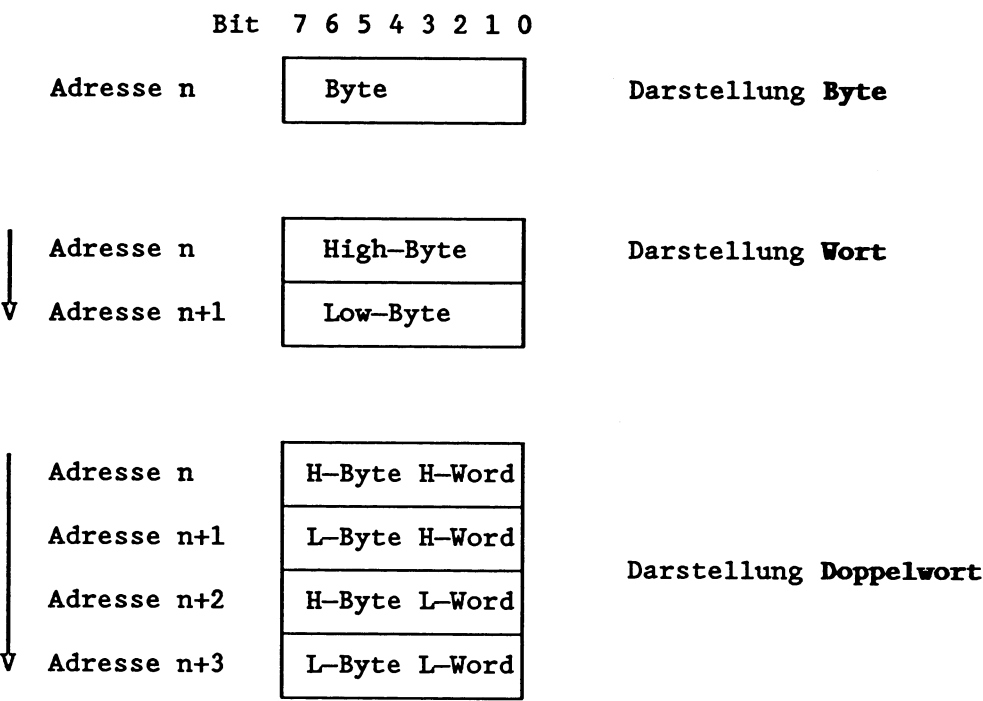
Welche der insgesamt 255 Kacheln verwendet werden soll, geben Sie bei der Parametrierung der Sonderfunktions-OBs 216, 217 und 218 an. Die Nummer der "aktuellen" Kachel wird daraufhin automatisch in eine Zelle mit der Adresse FEFF eingetragen (siehe Bild). Alle Adressierungen beziehen sich dann auf die Kachel, deren Nummer eingetragen wurde.

### WICHTIG!

Die Zelle mit der Adresse FEFF ist nicht lesbar.

**Hinweise zur Parametrierung**

Dem Schreiben (OB216) und dem Lesen (OB217) eines Bytes/Worts/  
Doppelworts liegt folgende Darstellung zugrunde:



### 6.6.1 Schreiben zu einer Kachel (OB 216)

Der Sonderfunktions-Organisationsbaustein überträgt ein Byte, Wort oder Doppelwort vom Akku 1 (rechtsbündig) zu einer bestimmten Kachel.

Das Adressieren der Kachel und das Übertragen des vollständigen Datums (1/2/4 Bytes) bilden eine Programmeinheit, die nicht unterbrochen werden kann.

#### Parameter:

1. Akku 2-L: Zieladresse auf der Kachel  
mögliche Werte: 0 - 2047
2. Akku 3-LL: aktuelle Kachel-Nr.  
mögliche Werte: 0 - 255
3. Akku 3-LH: Kennung des zu übertragenden Datums  
mögliche Werte: 0 = Byte  
1 = Wort  
2 = Doppelwort

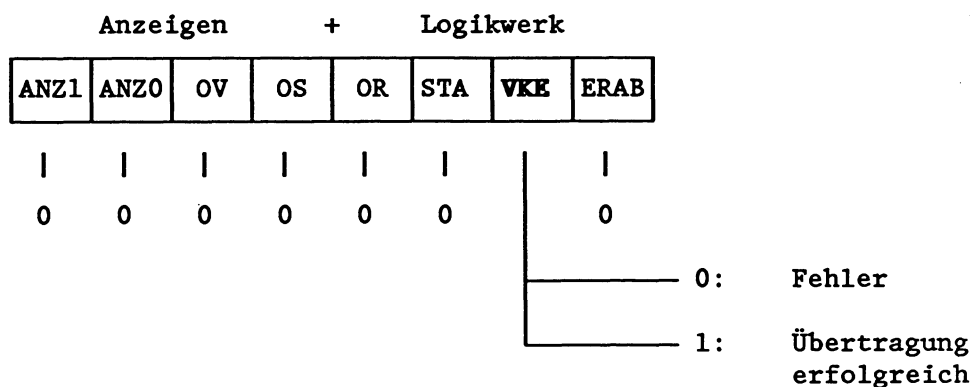
Wenn das Schreiben auf die Kachel erfolgreich verläuft,

- bleiben die Inhalte von Akku 1 und Akku 3 unverändert,
- enthält der Akku 2-L einen um 1/2/4 erhöhten Wert (je nach Länge des übertragenen Datums),
- wird das VKE gesetzt (VKE = 1),
- werden die restlichen Bit- und Wortanzeigen gelöscht (siehe Ergebnisanzeige).

Wenn das Schreiben auf die Kachel nicht möglich ist,

- bleiben die Inhalte aller Akkus unverändert,
- wird das VKE gelöscht,
- werden alle übrigen Bit- und Wortanzeigen ebenfalls gelöscht.

## Ergebnisanzeige:

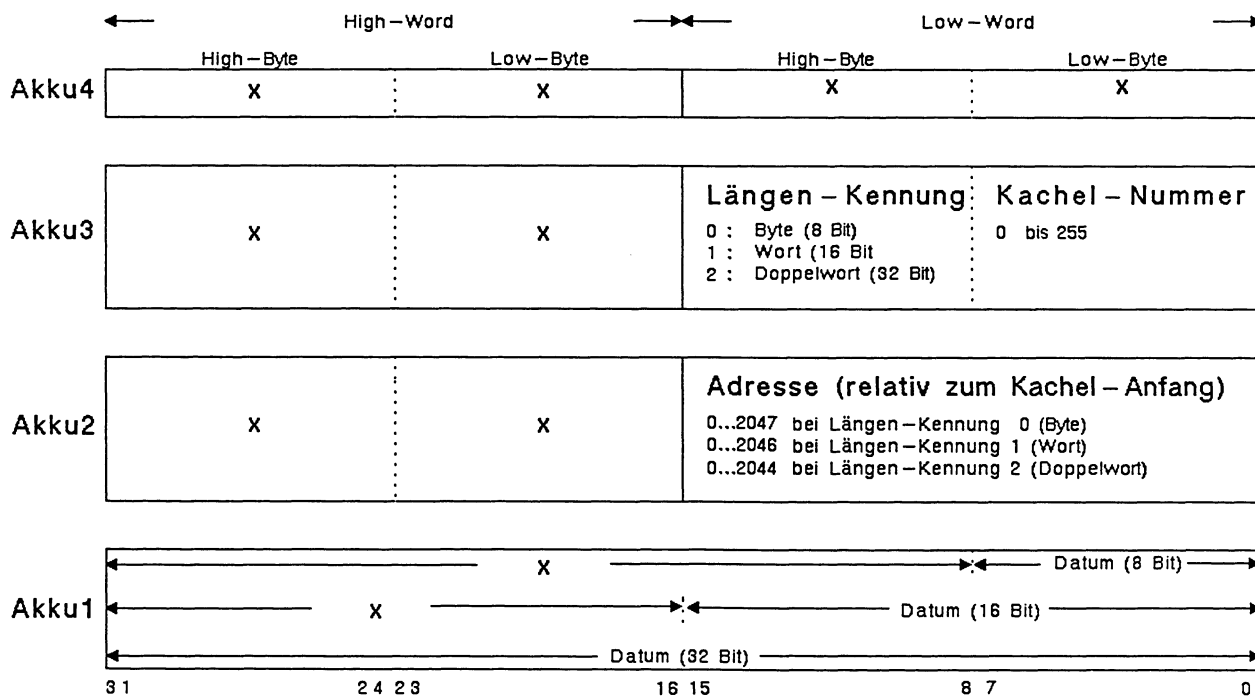


(zur Ergebnisanzeige siehe Kapitel 3)

## Fehlerfälle:

- falsche Längenkennung in Akku 3-LH
- Zieladresse auf der Kachel falsch oder nicht vorhanden
- angegebene Kachel-Nr. nicht vorhanden
- überhaupt keine Kachel vorhanden

## Akku-Belegung v o r dem Schreiben



### 6.6.2 Lesen von einer Kachel (OB 217)

Der Sonderfunktions-Organisationsbaustein überträgt ein Byte, Wort oder Doppelwort von einer bestimmten Kachel zum Akku 1 (rechtsbündig).

Das Adressieren der Kachel und das Übertragen des vollständigen Datums (1/2/4 Byte) bilden eine untrennbare Programmeinheit, die nicht unterbrochen werden kann.

#### Parameter

1. Akku 2-L: Quelladresse auf der Kachel  
mögliche Werte: 0 - 2047
2. Akku 3-LL: aktuelle Kachel-Nr.  
mögliche Werte: 0 - 255
3. Akku 3-LH: Kennung des zu übertragenden Datums  
mögliche Werte: 0 = Byte  
1 = Wort  
2 = Doppelwort

Wenn das Lesen von der Kachel erfolgreich verläuft,

- enthält der Akku 1 (rechtsbündig) den gelesenen Wert (der mögliche Rest der 32 Bits wird gelöscht),
- bleibt der Inhalt von Akku 3 unverändert,
- enthält der Akku 2-L einen um 1/2/4 erhöhten Wert (je nach Länge des übertragenen Datums)
- wird das VKE gesetzt (VKE = 1),
- werden die restlichen Bit- und Wortanzeigen gelöscht.

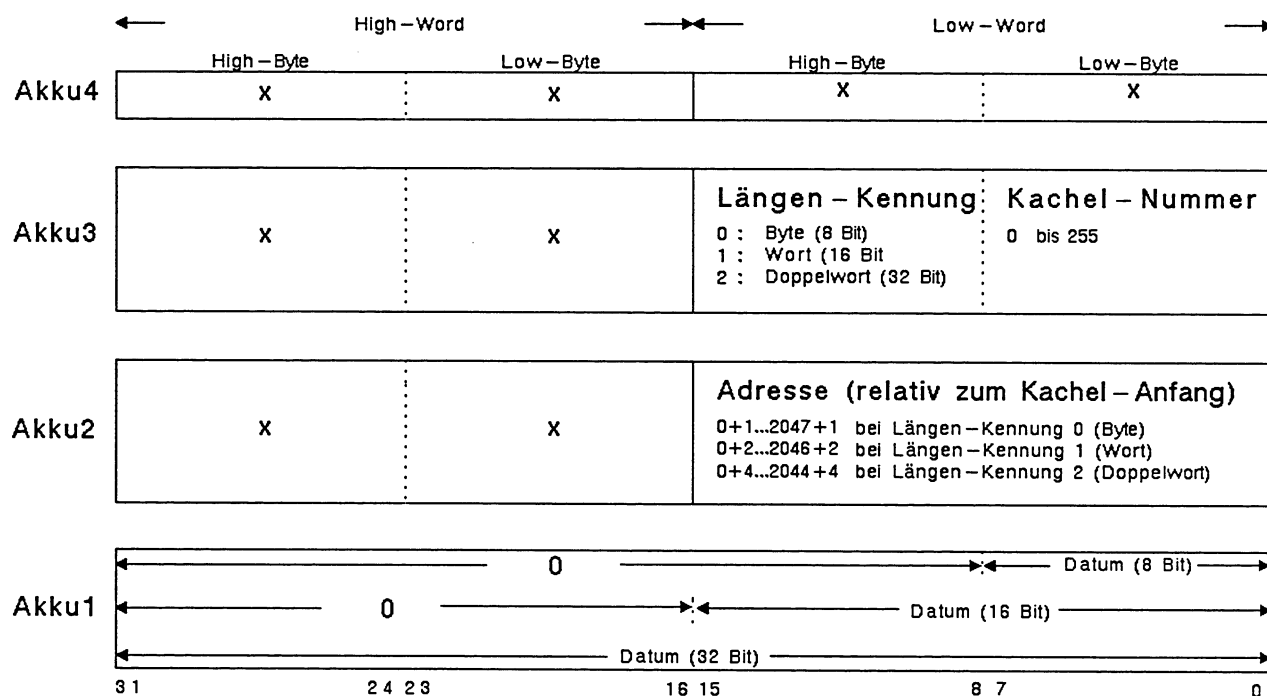
Wenn das Lesen von der Kachel nicht möglich ist,

- bleiben die Inhalte aller Akkus unverändert,
- wird das VKE gelöscht (VKE = 0),
- werden alle übrigen Bit- und Wortanzeigen ebenfalls gelöscht.

### Fehlerfälle:

- falsche Längenkennung im Akku 3-LH
- Quelladresse auf der Kachel falsch oder nicht vorhanden
- angegebene Kachel-Nr. nicht vorhanden
- überhaupt keine Kachel vorhanden

### **Akku-Belegung n a c h dem Lesen**



OB217

### 6.6.3 Belegen einer Kachel (OB 218)

Der Sonderfunktions-Organisationsbaustein überträgt die Steckplatzkennung 'seines' Prozessors zu einer bestimmten Kachel, falls der Inhalt der adressierten Zelle auf dieser Kachel gleich null ist. Solange nun die Steckplatzkennung in der Zelle eingetragen bleibt, ist diese Kachel für einen bestimmten Prozessor reserviert und kann von anderen Prozessoren **nicht** belegt werden.

Der Organisationsbaustein OB 218 dient der Synchronisation des Datentransfers und ist besonders wichtig, wenn **größere, zusammengehörige Datenblöcke** auf einmal gesendet bzw. übertragen werden sollen.

Das Adressieren der Kachel, das Lesen und das eventuelle Schreiben der Steckplatzkennung bilden eine Programmeinheit, die nicht unterbrochen werden kann.

#### Parameter

Akku 1-L: Zieladresse der Zelle auf der Kachel  
mögliche Werte: 0 - 2047

Akku 2-LL: aktuelle Kachel-Nr.  
mögliche Werte: 0 - 255

(Die Inhalte von Akku 3 und 4 sind in diesem Fall irrelevant.)

Wenn das Belegen der Kachel erfolgreich verläuft (Inhalt der Zieladresse = 0),

- bleiben die Inhalte der Akkus unverändert,
- wird das VKE gesetzt (VKE = 1),
- werden die restlichen Bit- und Wortanzeigen gelöscht.

Wenn das Belegen der Kachel nicht möglich ist (Inhalt der Zieladresse  $\neq$  0),

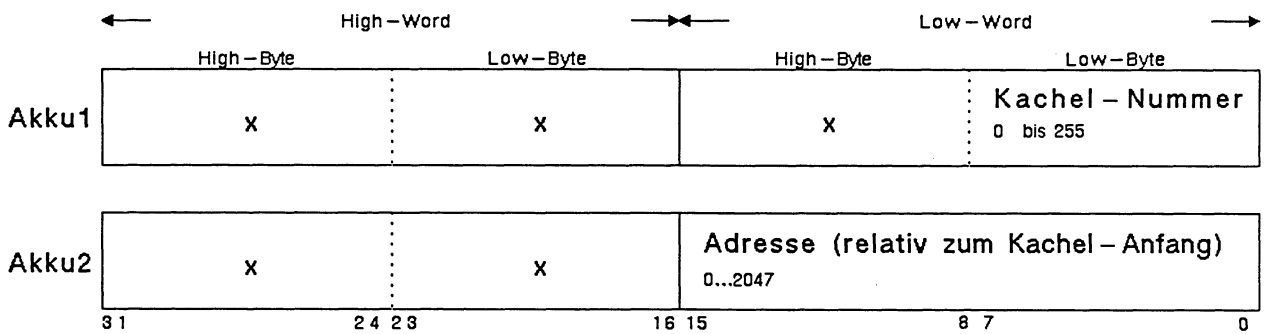
- bleiben die Inhalte der Akkus unverändert,
- wird das VKE gelöscht (VKE = 0),
- werden alle übrigen Bit- und Wortanzeigen gelöscht.



### Fehlerfälle:

- Inhalt der Zieladresse auf der Kachel ist nicht Null
- Zieladresse auf der Kachel falsch oder nicht vorhanden
- angegebene Kachel-Nr. nicht vorhanden
- überhaupt keine Kachel vorhanden

### **Akku vor / nach dem Belegen**



OB218.

## Programmbeispiel

Vom DB 45 einer CPU 928 werden die Datenwörter 4 bis 11 über den KOR C in den DX 45 (Datenwörter 0 bis 7) einer zweiten CPU 928 übertragen. Die Synchronisation zwischen Sender und Empfänger (Mehrprozessorbetrieb!) geschieht mittels OB 218.

Aktuelle Kachel auf dem Koordinator: Nr. 255

Koordinierungszelle auf der Kachel (Belegen): Adr. 53

Datenübergabebereich auf der Kachel (Lesen und Schreiben): Adr. 54-69

### SENDER:

L	KB 255	Kachelnummer	
L	KB 53	Adresse Koordinierungszelle	
SPA	OB 218	Übertragen Steckplatzkennung in Zelle auf Kachel	
SPB	=M001	Wenn VKE = 1 (Übertragen erfolgreich), Sprung auf Marke	
BEA		ansonsten Bausteinende	
M001:	A	DB 45	Quell-Datenbaustein aufschlagen
	L	KY 2,255	2=Längenkennung Doppelwort, Kachelnr.
	L	KB 54	Anfangsadresse auf Kachel
	ENT		Beschreiben von Akku 3
	L	DD 4	Datenwörter 4 und 5 (= 4 Byte)
	SPA	OB 216	Übertragen 1. Doppelwort
	TAK		Adresse um 4 erhöhen (Akku 2-L = 58)
			Retten der Zieladresse
	L	DD 6	
	SPA	OB 216	Übertragen 2. Doppelwort
	TAK		
	L	DD 8	
	SPA	OB 216	Übertragen 3. Doppelwort
	TAK		
	L	DD 10	
	SPA	OB 216	Übertragen 4. Doppelwort
	TAK		
	L	KY 0,255	
	L	KB 53	Adresse mit Steckplatzkennung
	ENT		
	L	KB 0	Akku 1 = 0
	SPA	OB 216	Steckplatzkennung löschen --> Datenübergabebereich freigeben
	BE		

**EMPFÄNGER:**

L	KB 255	Kachelnummer
L	KB 53	Koordinierungszelle
SPA	OB 218	Kachelbelegung durch 2. ZBG
SPB	=M002	Wenn VKE = 1, Sprung auf Marke
BEA		
M002:	AX DX 50	Ziel-Datenbaustein
	L KY 2,255	
	L KB 54	
	ENT	Akku 3 beschreiben
	L KB 0	Akku 2 beschreiben
	SPA OB 217	Lesen 1. Doppelwort
		Adresse um 4 erhöhen (Akku 2-L = 58)
	T DD 0	Transferieren Akku 1 zu Datenwort 0 und 1
	SPA OB 217	Lesen 2. Doppelwort
	T DD 2	
	SPA OB 217	Lesen 3. Doppelwort
	T DD 4	
	SPA OB 217	Lesen 4. Doppelwort
	T DD 6	
	L KY 0,255	
	L KB 53	Adresse mit Steckplatzkennung
	ENT	
	L KB 0	Akku 1 = 0
	SPA OB 216	Steckplatzkennung löschen --> Datenüber-
		gabebereich freigeben
	BE	

## 6.7 Vorzeichenerweiterung (OB 220)

Diese Sonderfunktion erweitert das Vorzeichen einer 16-Bit-Festpunktzahl im Akku 1-L auf das höherwertige Wort (Akku 1-H):

Wenn das Bit  $2^{15} = 0$  (positive Zahl) ist, wird das höherwertige Wort mit KH = 0000 geladen.

Wenn das Bit  $2^{15} = 1$  (negative Zahl) ist, wird das höherwertige Wort mit KH = FFFF geladen.

Diese Vorzeichenerweiterung ist notwendig, um eine negative 16-Bit-Festpunktzahl vor einer Festpunkt-Gleitpunkt-Wandlung (32 Bit, Befehl FDG) zu einer 32-Bit-Festpunktzahl zu erweitern.

Laufzeit der Sonderfunktion OB 220: 16  $\mu$ s

### Parameter:

1. Akku 1-L: 16-Bit-Festpunktzahl

Fehlerfälle: keine.

## 6.8 Systemfunktionen

### 6.8.1 "Alarmer gemeinsam sperren" ein-/ausschalten (OB 120) und "Alarmer gemeinsam verzögern" ein-/ausschalten (OB 122)

Ein STEP5-Programm kann an Baustein- oder Befehlsgrenzen von Programmen höherer Priorität unterbrochen werden. Zu diesen höherprioritären Programmbearbeitungsebenen gehören die Prozeßalarmer und die Weckalarmer. Die Laufzeit des unterbrochenen Programms verlängert sich dabei jeweils um die Laufzeit der eingeschachtelten Programme.

Mit Hilfe der Sonderfunktions-Organisationsbausteine OB 120 und OB 122 können Sie das Einschachteln von Prozeß- und/oder Weckalarmen an einer oder an mehreren aufeinanderfolgenden Baustein- oder Befehlsgrenzen (je nach Einstellung im DX 0) verhindern.

#### OB 120: "Alarmer gemeinsam sperren" ein-/ausschalten

Der Sonderfunktions-Organisationsbaustein OB 120 hat Auswirkungen auf die *Entgegennahme* von Alarmen:

"Alarmer sperren" einschalten" heißt, es werden ab sofort keine Alarme mehr registriert und diejenigen Alarme, die bereits registriert worden sind (die z.B. auf eine Bausteingrenze 'warten'), werden gelöscht. Nur falls der OB 2 (Prozeßalarm) oder ein Weckalarm-OB bereits begonnen wurde, wird dieser vollständig bearbeitet.

"Alarmer sperren" ausschalten" heißt, es werden ab sofort alle auftretenden Alarme wieder registriert, an der nächsten Baustein- oder Befehlsgrenze eingeschachtelt und bearbeitet.

#### OB 122: "Alarmer gemeinsam verzögern" ein-/ausschalten

Dieser Sonderfunktions-Organisationsbaustein hat Auswirkungen auf das Bearbeiten von Alarmen.

"Alarmer verzögern" einschalten" heißt, es werden weiterhin alle auftretenden Alarme registriert und bereits anstehende Alarme bleiben registriert. Eine Bearbeitung der registrierten Alarme findet jedoch zunächst nicht statt.<sup>1)</sup> Vorübergehend werden alle Befehls- bzw. Bausteingrenzen für die Bearbeitung von Alarmen unwirksam gemacht. Nur falls der OB 2 (Prozeßalarm) oder ein Weckalarm-OB bereits begonnen wurde, wird dieser vollständig bearbeitet.

"Alarmer verzögern" ausschalten" heißt, es werden alle registrierten Alarme an der nächsten Baustein- oder Befehlsgrenze eingeschachtelt und bearbeitet.

<sup>1)</sup> Wird innerhalb der 'Alarmer verzögern'-Phase ein bestimmter Weckalarm-OB zum zweiten Mal aufgerufen, kommt es zum Weckfehler.

Die OBs 120 und 122 vermerken die zu sperrenden bzw. zu verzögernden Alarmer in einem **Steuerwort** mit folgender Belegung:

Bit  $2^0$ : Weckalarmer

Bit  $2^2$ : Prozeßalarmer

Bit  $2^1$ ,  $2^3$  bis  $2^{31}$ : reserviert; diese Bits müssen "0" sein!

Das heißt:

Solange das Bit  $2^0 = 1$  ist, werden *alle* auftretenden Weckalarmer gesperrt bzw. verzögert.

Solange das Bit  $2^2 = 1$  ist, werden *alle* auftretenden Prozeßalarmer gesperrt bzw. verzögert.

Sind sowohl Bit  $2^0$  als auch Bit  $2^2$  auf "1" gesetzt, kommen weder Weck- noch Prozeßalarmer durch!

#### Parameter:

1. Akku 2-L: Funktionskennung  
mögliche Werte: 1, 2 oder 3
2. Akku 1: neues Steuerwort bzw. Maske

Akku 2-L	Akku 1		Funktion:
	vorher	nachher	
1	Steuerwort	Steuerwort	Der Inhalt von Akku 1 wird ins Steuerwort geladen.
2	Maske	neues Steuerwort	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Steuerwort gleich "1" gesetzt. Das neue Steuerwort wird in Akku 1 geladen.
3	Maske	neues Steuerwort	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Steuerwort gleich "0" gesetzt. Das neue Steuerwort wird in Akku 1 geladen.

#### Fehlerfälle:

- unzulässige Funktionskennung in Akku 2-L
- eines der reservierten Bits in Akku 1 ( $2^1$ ,  $2^3$  bis  $2^{31}$ ) ist gleich "1"

Im Fehlerfall wird der OB 31 (Sonstige Laufzeitfehler) aufgerufen und im Akku 1 eine Fehlerkennung übergeben:

1A47H bei OB 120  
1A48H bei OB 122

## Hinweise

- Der Zustand des Steuerworts läßt sich durch folgende Programmsequenz abfragen:
  1. Funktionskennung 2 oder 3 in Akku 2-L laden
  2. Wert "0" in Akku 1 laden
  3. Sonderfunktions-OB 120/122 aufrufen
  4. Akku 1 auslesen
- Der Zustand der Alarmverarbeitung kann auch durch Auslesen der Systemdatenwörter BS 131 und BS 132 ermittelt werden.

BS 131    Anzeigenwort "Alarmer gemeinsam sperren" (OB 120)

BS 132    Anzeigenwort "Alarmer gemeinsam verzögern" (OB 122)

- Zum Sperren und Freigeben des Prozeßalarms können statt des OB 120 die Befehle AS und AF verwendet werden:

AS            entspricht        :L    KB2  
                                  :L    KM00000000 00000100  
                                  :SPA OB120

AF            entspricht        :L    KB3  
                                  :L    KM00000000 00000100  
                                  :SPA OB120

### **6.8.2 "Weckalarmer einzeln sperren" ein-/ausschalten (OB 121) und "Weckalarmer einzeln verzögern" ein-/ausschalten (OB 123)**

Mit Hilfe der Sonderfunktions-Organisationsbausteine OB 121 und OB 123 können Sie das Einschachteln von *bestimmten* Weckalarm-OBs an einer oder an mehreren aufeinanderfolgenden Baustein- oder Befehlsgrenzen verhindern. Beispielsweise können Sie für einen bestimmten Programmteil festlegen, daß er nicht unterbrochen werden kann durch einen OB 10 (10ms) und einen OB 11 (20ms). Alle übrigen programmierten Weckalarmer hingegen werden wie üblich bearbeitet.

#### **OB 121: "Weckalarmer einzeln sperren" ein-/ausschalten**

Der Sonderfunktions-Organisationsbaustein OB 121 hat Auswirkungen auf die *Entgegennahme* von Weckalarmen:

"'Weckalarmer einzeln sperren' einschalten" heißt, es werden ab sofort keine der angegebenen Weckalarmer mehr registriert und diejenigen Alarmer, die bereits registriert worden sind (die z.B. auf eine Bausteingrenze 'warten'), werden gelöscht. Nur falls ein Weckalarm-OB bereits begonnen wurde, wird dieser vollständig bearbeitet.

"'Weckalarmer einzeln sperren' ausschalten" heißt, es werden ab sofort alle auftretenden Weckalarmer wieder registriert, an der nächsten Baustein- oder Befehlsgrenze (je nach Einstellung im DX 0) eingeschachtelt und bearbeitet.

### OB 123: "Weckalarme einzeln verzögern" ein-/ausschalten

Dieser Sonderfunktions-Organisationsbaustein hat Auswirkungen auf das Bearbeiten von bestimmten Weckalarmen.

"'Weckalarme einzeln verzögern' einschalten" heißt, es werden weiterhin alle auftretenden Alarme registriert und bereits anstehende Weckalarme bleiben registriert. Eine Bearbeitung der im Steuerwort angegebenen Weckalarme findet jedoch zunächst nicht statt. Vorübergehend werden alle Befehls- bzw. Bausteingrenzen für die Bearbeitung dieser Weckalarme unwirksam gemacht. Nur falls einer dieser Weckalarm-OBs bereits begonnen wurde, wird er vollständig bearbeitet.

"'Weckalarme einzeln verzögern' ausschalten" heißt, es werden alle registrierten Alarme an der nächsten Baustein- oder Befehlsgrenze (je nach Einstellung im DX 0) eingeschachtelt und bearbeitet.

Die OBs 121 und 123 vermerken die zu sperrenden bzw. zu verzögernden Weckalarme in einem **Steuerwort** mit folgender Belegung:

Bit 2<sup>0</sup>: muß "0" sein!  
Bit 2<sup>1</sup>: muß "0" sein!  
Bit 2<sup>2</sup>: muß "0" sein!  
Bit 2<sup>3</sup>: Weckalarm 10ms (OB 10)  
Bit 2<sup>4</sup>: Weckalarm 20ms (OB 11)  
Bit 2<sup>5</sup>: Weckalarm 50ms (OB 12)  
Bit 2<sup>6</sup>: Weckalarm 100ms (OB 13)  
Bit 2<sup>7</sup>: Weckalarm 200ms (OB 14)  
Bit 2<sup>8</sup>: Weckalarm 500ms (OB 15)  
Bit 2<sup>9</sup>: Weckalarm 1sec (OB 16)  
Bit 2<sup>10</sup>: Weckalarm 2sec (OB 17)  
Bit 2<sup>11</sup>: Weckalarm 5sec (OB 18)  
Bit 2<sup>12</sup>: muß "0" sein!  
Bit 2<sup>13</sup>: muß "0" sein!  
Bit 2<sup>14</sup>: muß "0" sein!  
Bit 2<sup>15</sup>: muß "0" sein!

Solange ein Bit auf "1" gesetzt ist, ist der betreffende Weckalarm gesperrt bzw. verzögert.

#### Parameter:

1. Akku 2-L: Funktionskennung  
mögliche Werte: 1, 2 oder 3
2. Akku 1: neues Steuerwort bzw. Maske



Akku 2-L	Akku 1		Funktion:
	vorher	nachher	
1	Steuerwort	Steuerwort	Der Inhalt von Akku 1 wird ins Steuerwort geladen.
2	Maske	neues Steuerwort	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Steuerwort gleich "1" gesetzt. Das neue Steuerwort wird in Akku 1 geladen.
3	Maske	neues Steuerwort	Alle in der Maske in Akku 1 mit einer "1" gekennzeichneten Bits werden im Steuerwort gleich "0" gesetzt. Das neue Steuerwort wird in Akku 1 geladen.

#### Fehlerfälle:

- unzulässige Funktionskennung in Akku 2-L
- eines der reservierten Bits in Akku 1 ist gleich "1"

Im Fehlerfall wird der OB 31 (Sonstige Laufzeitfehler) aufgerufen und im Akku 1 eine Fehlerkennung übergeben:

1A4AH bei OB 121  
1A4BH bei OB 123

#### Hinweise

- Der Zustand des Steuerworts läßt sich durch folgende Programmsequenz abfragen:
  1. Funktionskennung 2 oder 3 in Akku 2-L laden
  2. Wert "0" in Akku 1 laden
  3. Sonderfunktions-OB 121/123 aufrufen
  4. Akku 1 auslesen
- Der Zustand der Weckalarmverarbeitung kann auch durch Auslesen der Systemdatenwörter BS 135 und BS 137 ermittelt werden.

BS 135 Anzeigenwort "Weckalarme einzeln sperren" (OB 121)  
BS 137 Anzeigenwort "Weckalarme einzeln verzögern" (OB 123)

### **6.8.3 Zykluszeit einstellen (OB 221)**

Durch Aufruf dieser Sonderfunktion können Sie die Überwachung der maximalen Zykluszeit, die standardmäßig auf 150 ms eingestellt ist, auf einen neuen Wert ändern.

Gleichzeitig wird mit dem Aufruf der Timer für die Überwachung neu gestartet: Der aktuelle Zyklus, d.h. der Zyklus, in dem der OB 221 erstmals aufgerufen wird, verlängert sich um den neu eingestellten Wert, vom Zeitpunkt des Sonderfunktionsaufrufes an gerechnet. Die Zykluszeit aller folgender Zyklen entspricht dem neu eingestellten Wert (= dem Zeitwert, den Sie im Akku 1 übergeben).

#### Parameter:

1. Akku 1: Neue Zykluszeit (in Millisekunden)  
mögliche Werte: 1 ms - 6000 ms

#### Fehlerfälle:

- angegebene Zykluszeit nicht im Bereich 1 ms - 6000 ms.

#### Hinweis:

Die Sonderfunktion OB 221 wurde aus Kompatibilitätsgründen vom S-Prozessor übernommen und sollte in der CPU 928 nicht programmiert werden. Statt dessen sollte dieses Systemverhalten im DX 0 parametrisiert werden (siehe Kapitel 7).

### **6.8.4 Zykluszeit neu starten (OB 222)**

Die Sonderfunktion OB 222 bewirkt ein Nachtriggern der Zykluszeitüberwachung, d.h., der Timer für die Überwachung wird neu gestartet. Durch Aufruf dieser Sonderfunktion wird die maximal zulässige Zykluszeit um den eingestellten Wert - standardmäßig 150 ms oder im DX 0 festgelegt - vom Zeitpunkt des Aufrufs an verlängert.

Parameter: keine

Fehlerfälle: keine.

### 6.8.5 Anlaufarten vergleichen (OB 223)

Durch Aufruf des OB 223 - z.B. im Anlauf oder zu Beginn der zyklischen Programmbearbeitung - wird im Mehrprozessorbetrieb überprüft, ob die Anlaufarten aller beteiligten Prozessoren gleich sind.

Ist dies nicht der Fall, so erkennt derjenige Prozessor, in dem der OB 223 bearbeitet wurde, einen Laufzeitfehler. Es wird der **OB 31** aufgerufen. Im Akku 1 steht die Fehlerkennung 1A3B (hex.). Ist der OB 31 nicht programmiert, geht der Prozessor mit der Fehlermeldung LZF in den Stoppzustand. Seine STOP-LED zeigt langsames Blinken. Die übrigen Prozessoren gehen mit Dauerlicht in den Stop.

Parameter: keine

Fehlerfälle: keine

### 6.8.6 Koppelmerker im Block übertragen (OB 224)

Im Mehrprozessorbetrieb werden die im DB 1 angegebenen Koppelmerker dann übertragen, wenn der Prozessor das Signal erhält, daß er auf den Peripheriebus zugreifen darf.

Wollen mehrere Prozessoren gleichzeitig zugreifen, so erteilt der Koordinator reihum jedem Prozessor das Busfreigabesignal. Dabei darf ein Prozessor jeweils nur ein Byte übertragen. Durch diese verzahnte Übertragung kann es vorkommen, daß zusammengehörige Koppelmerkerinformationen auseinandergerissen werden und daraufhin mit falschen Werten gearbeitet wird.

Durch Aufruf des Organisationsbausteines OB 224 erreichen Sie eine blockweise Übertragung aller im DB 1 des jeweiligen Prozessors angegebenen Koppelmerker: Solange ein Prozessor mit der Koppelmerkerübertragung beschäftigt ist, kann er von einem anderen Prozessor nicht unterbrochen werden. Da der nächste Prozessor mit seiner Übertragung warten muß, wird die zyklische Programmbearbeitung solange verzögert (Zykluszeit!).

Der OB 224 gewährleistet somit eine Zusammengehörigkeit der gesamten Koppelmerkerinformation. Er muß im Anlaufprogramm aufgerufen werden, und zwar

- a) in allen am Koppelmerkertransfer beteiligten Prozessoren und
- b) in jeder verwendeten Anlaufart.

Parameter: keine

Fehlerfälle: keine

Hinweis:

Die Sonderfunktion OB 224 wurde aus Kompatibilitätsgründen vom S-Prozessor übernommen und sollte in der CPU 928 nicht programmiert werden. Statt dessen sollte dieses Systemverhalten im DX 0 parametrisiert werden (siehe Kapitel 7).

**6.8.7 Wort aus dem Systemprogramm lesen (OB 226)**

Das Systemprogramm des Prozessors hat eine Länge von  $64 \times 2^{10}$  Wörtern und liegt in einem Speicherbereich, auf den Sie mit STEP5-Anweisungen keinen Zugriff haben. Mit Hilfe des OB 226 kann auf diesen Speicherbereich zugegriffen werden.

Parameter:

1. Akku 1-L: Adresse der zu lesenden Systemprogramm-Speicherzelle

Fehlerfälle: keine

Nach dem Aufruf des OB 226

- steht das gelesene Wort rechtsbündig im Akku 1,
- ist der restliche Inhalt des Akku 1 gelöscht,
- steht der vorherige Inhalt des Akku 1 (d.h. die Wortadresse) im Akku 2,
- geht der vorherige Inhalt des Akku 2 verloren.

Zur Verwendung des OB 226 beachten Sie bitte die Beschreibung des OB 227 und das dazugehörige Programmbeispiel.

### 6.8.8 Quersumme des Systemprogramms lesen (OB 227)

Der Sonderfunktions-Organisationsbaustein OB 227 lädt die Quersumme aus dem Speicherbereich des Systemprogramms in den Akku 1.

Parameter: keine

Fehlerfälle: keine

Nach dem Aufruf des OB 227

- steht die gelesene Quersumme (1 Wort) rechtsbündig im Akku 1,
- ist der restliche Inhalt des Akku 1 gelöscht,
- steht der vorherige Inhalt des Akku 1 im Akku 2,
- geht der vorherige Inhalt des Akku 2 verloren.

#### Anwendung:

Sie können während der zyklischen Programmbearbeitung den Inhalt des Systemprogramms prüfen, indem Sie

- die einzelnen Speicherzellen des Systemprogramms mit Hilfe des OB 226 lesen,
- alle Speicherzellen mit Festpunktaddition (Befehl +F) addieren,
- mit Hilfe des OB 227 die Quersumme lesen und anschließend
- die durch Festpunktaddition gebildete Endsumme mit der gelesenen Quersumme vergleichen.

## Programmbeispiel

FB111

NAME: CHECKSUM

:L	KH0000	
:T	MW254	Quersummenmerker löschen
:T	MW252	Adreßzähler löschen
:		
M001	:SPA OB222	Zykluszeit nachtriggern
:	:L MW252	Adresse der zu lesenden Speicherzelle laden
:	:SPA OB226	Wort lesen
:	:L MW254	Quersummenmerker laden
:	:+F	Addieren
:	:T MW254	Quersummenmerker speichern
:		
:	:L MW252	Adreßzähler inkrementieren
:	:L KF+1	
:	:+F	
:	:T MW252	
:		
:	:L KH0000	Falls Adreßzähler ungleich '0',
:	:>< F	
:	:SPB =M001	auf Marke M001 springen
:		
:	:SPA OB227	Falls Adreßzähler gleich '0', Quersumme lesen
:	:L MW254	Quersummenmerker laden
:	:!=F	Falls gleich, Bausteinende
:	:BEB	
:		
:	:STP	Falls ungleich, Stoppbefehl
:	:BE	

### 6.8.9 Statusinformation einer Programmbearbeitungsebene lesen (OB 228)

Beim Auftreten eines bestimmten Ereignisses ruft das Systemprogramm die dazugehörige Programmbearbeitungsebene auf. Die Programmbearbeitungsebene ist damit 'aktiviert'.

Mit Hilfe des Organisationsbausteins OB 228 können Sie nun feststellen, ob zu einem Zeitpunkt eine bestimmte Programmbearbeitungsebene aktiviert ist oder nicht. Im Akku 1 übergeben Sie die Nummer derjenigen Programmbearbeitungsebene, deren Status abgefragt werden soll. (Die Nummern entsprechen den im USTACK unter 'EBENE' eingetragenen Nummern.)

Bei Aufruf der Sonderfunktion überträgt der OB 228 die Statusinformation der angegebenen Programmbearbeitungsebene in den Akku 1.

#### Parameter:

1. Akku 1-L: Nummer der Programmbearbeitungsebene (siehe USTACK, EBENE)

mögliche Werte (hexadezimal):

02	-	Neustart
04	-	Zyklus
06	-	Weckalarm 5sec
08	-	Weckalarm 2sec
0A	-	Weckalarm 1sec
0C	-	Weckalarm 500ms
0E	-	Weckalarm 200ms
10	-	Weckalarm 100ms
12	-	Weckalarm 50ms
14	-	Weckalarm 20ms
16	-	Weckalarm 10ms
18	-	nicht belegt
1A	-	nicht belegt
1C	-	Regelung
1E	-	nicht belegt
20	-	nicht belegt
22	-	nicht belegt
24	-	Prozeßalarm
26	-	nicht belegt
28	-	nicht belegt
2A	-	nicht belegt
2C	-	Abbruch
2E	-	nicht belegt
30	-	Weckfehler
32	-	Reglerfehler
34	-	Zyklusfehler
36	-	nicht belegt
38	-	Befehlscodefehler
3A	-	Laufzeitfehler
3C	-	Adressierfehler
3E	-	Quittungsverzug
40	-	nicht belegt
42	-	nicht belegt
44	-	Manueller Wiederanlauf
46	-	Automatischer Wiederanlauf

Fehlerfälle: keine

Nach dem Aufruf des OB 228

- steht die Statusinformation im Akku 1:

Inhalt von Akku 1 = 0 :    Programmbearbeitungsebene ist nicht  
   aufgerufen

Inhalt von Akku 1 # 0 :    Programmbearbeitungsebene ist aktiviert

- steht der vorherige Inhalt des Akku 1 im Akku 2,
- geht der vorherige Inhalt des Akku 2 verloren.

Dadurch können Sie Ihre Programmbearbeitung abhängig machen vom Status einer anderen Programmbearbeitungsebene.

**Beispiel:**

Ein Quittungsverzug soll im Neustart ignoriert werden, nicht jedoch in den übrigen Programmbearbeitungsebenen.

Sie rufen zu Beginn des OB 23 den Sonderfunktions-Organisationsbaustein OB 228 auf, um festzustellen, ob beim Auftreten des QVZ die Programmbearbeitungsebene NEUSTART (Nummer 02) aktiviert ist oder nicht. Die weitere Fehlerbehandlung machen Sie abhängig von der Statusinformation, die Sie erhalten:

Akku 1 = 0:    NEUSTART passiv    -> QVZ ist nicht im Neustart  
   aufgetreten  
   -> Fehlerprogramm muß bearbeitet  
   werden

Akku 1 # 0:    NEUSTART aktiviert    -> QVZ ist im Neustart aufgetreten  
   -> QVZ darf ignoriert werden

Der OB 228 ermöglicht Ihnen somit u.a. eine differenziertere Fehlerbehandlung.



## 6.9 Funktionen für Standard-Funktionsbausteine (OB 230 bis OB 237)

Die Sonderfunktions-Organisationsbausteine OB 230 bis OB 237 sind für Hantierungsfunktionen reserviert und können nur innerhalb der Standard-Funktionsbausteine FB 120 bis FB 127 aufgerufen werden.

Diese Standard-Funktionsbausteine - die sog. "Hantierungsbausteine" - steuern im Einzel- und im Mehrprozessorbetrieb den Datenverkehr über den Kachelbereich: Sie werden eingesetzt, wenn Daten oder Parameter sowie Steuerungsinformationen von den Kommunikationsprozessoren übernommen bzw. an die Kommunikationsprozessoren übergeben werden sollen.

### Zuordnungshilfe

Standard-Funktionsbaustein	Sonderfunktions-OB	Hantierungsbaustein
FB 120	SF-OB 230	SEND
FB 121	SF-OB 231	RECEIVE
FB 122	SF-OB 232	FETCH
FB 123	SF-OB 233	CONTROL
FB 124	SF-OB 234	RESET
FB 125	SF-OB 235	SYNCHRON
FB 126	SF-OB 236	SEND ALL
FB 127	SF-OB 237	RECEIVE ALL

Zur Anwendung der Hantierungsbausteine, die als Software-Produkt auf Diskette zu beziehen sind, gibt es eine detaillierte Betriebsanleitung mit dem Titel "Automatisierungsgerät S5-135U Hantierungsbausteine für R-Prozessor und CPU 928" (Bestell-Nr. C79000-B8500-C366-02).

## 6.10 Schieberegister

Ein Software-Schieberegister besteht aus nebeneinander aufgereihten, 8-Bit-breiten Speicherzellen und kann zwischen 2 und maximal 256 Speicherzellen lang sein.

Die Daten eines Schieberegisters liegen im Datenbaustein-RAM des Prozessors. Jedes Schieberegister ist einem bestimmten Datenbaustein fest zugeordnet: Beide haben dieselbe Nummer (zulässig: 192 bis 255). Wenn Sie z.B. ein Schieberegister mit der Nummer 210 eingerichtet haben, so stehen die dazugehörigen Daten im Datenbaustein DB 210.

Das DB-RAM umfaßt ca. 23k Wörter (Adresse KH 8000 bis KH DD7F). In diesem Bereich liegen die mit OB 254 und 255 kopierten Datenbausteine (von KH 8000 an aufsteigend) und die vom Anwender eingerichteten Schieberegister (von KH DD7F an abfallend). Falls beim Kopieren von DBs oder Einrichten von Schieberegistern der Speicherbereich des DB-RAMs nicht ausreicht, erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Die weitere Reaktion hängt von der Programmierung des OB 31 ab (siehe 'Sonstige Laufzeitfehler').

Sie können Daten in das Schieberegister schreiben und Daten aus dem Schieberegister lesen. Dies erfolgt über die sog. "Zeiger": Zeiger sind Merkerbytes, die den Inhalt einzelner Zellen eines Schieberegisters enthalten.

Die nachfolgenden Abbildungen zeigen das Prinzip des Software-Schieberegisters.

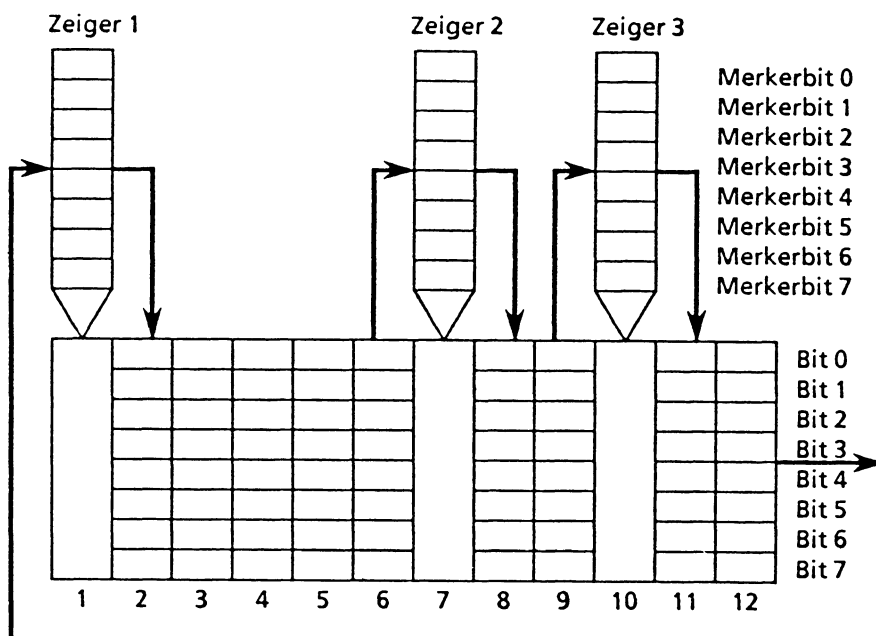


Abb. 6-1: Prinzipskizze des Schieberegisters mit 3 Zeigern und 12 Speicherzellen

Der erste Zeiger (= Basiszeiger) ist fest auf die erste Speicherzelle des Schieberegisters eingestellt. Die Nummer dieses Merkerbytes ist vom Anwender festzulegen. Alle weiteren Zeiger werden nun relativ zum Basiszeiger positioniert, wobei pro Schieberegister zwischen 1 und maximal 6 Zeiger verwendet werden können.

Bei der Bearbeitung eines Schieberegisters wird - wie bei einem Hardware-Schieberegister - die Information byteweise von einer Speicherzelle zur nächsten Speicherzelle übertragen (siehe Bild). Jeder Aufruf der Schieberegisterfunktion bewirkt also ein Verschieben der Information um genau 1 Speicherzelle ( $\hat{=}$  1 Takt). Die Zeiger werden dabei mit neuen Inhalten versorgt. Entsprechend den eingezeichneten Pfeilen wird die Information durch das gesamte Schieberegister bis in die letzte Speicherzelle 'durchgeschoben', von wo aus sie wiederum in die Speicherzelle 1 gelangt. (Beim abgebildeten Schieberegister ist dies nach 12 Takten der Fall.)

### Beispiel:

Die nachfolgenden Abbildungen zeigen das Verschieben der Information innerhalb eines Schieberegisters.

Vor Aufruf der Sonderfunktion werden die Merkerbits in den Zeigern gesetzt:

Merkerbit 0 von Zeiger 1 setzen	:S M0.0
Merkerbit 3 von Zeiger 2 setzen	:S M1.3
Merkerbit 2 von Zeiger 3 setzen	:S M2.2

Dann wird die Schieberegister-Funktion aufgerufen :SPA OB 241

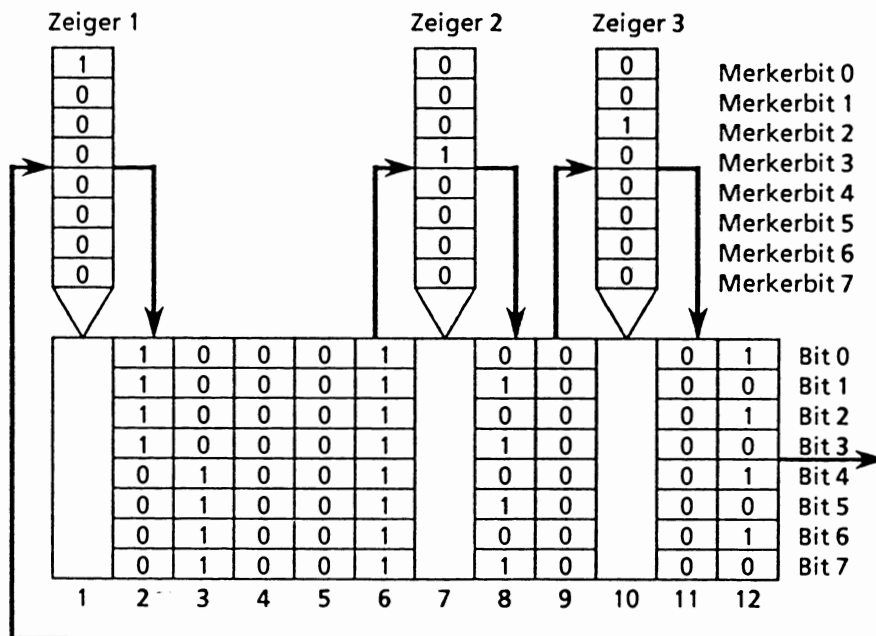


Abb. 6-2: Prinzipskizze des Schieberegisters mit 3 Zeigern und 12 Speicherzellen vor dem ersten Takt

Nach Aufruf der Sonderfunktion ist die 8-Bit-breite Information der Speicherzellen um eine Zelle verschoben:

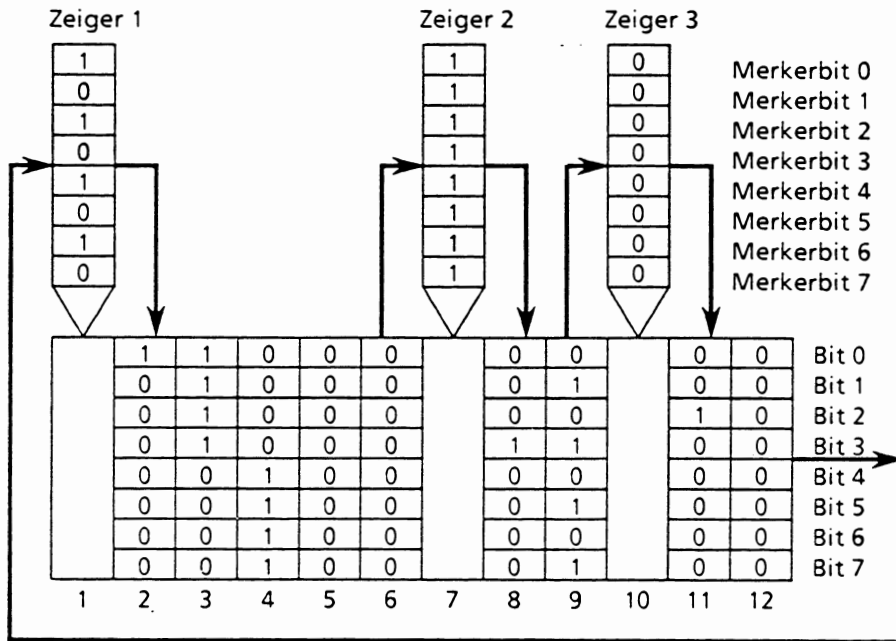


Abb. 6-3: Prinzipskizze des Schieberegisters mit 3 Zeigern und 12 Speicherzellen nach dem ersten Takt

Die Information, die sich jetzt in den Zeigern befindet, können Sie auswerten mit :L MB 0 etc.

Merkerbit 0, 3 und 2 lassen sich am Basiszeiger abfragen: Auf diese Weise kann die gesamte Information, die aus den Einträgen in allen Zeigern stammt, am Basiszeiger ausgewertet werden (im Beispiel erstmals nach 12 Takten möglich).

Wenn Sie ein Schieberegister verwenden wollen, so stehen Ihnen dazu 3 Sonderfunktions-Organisationsbausteine zur Verfügung:

**OB 240:** Diese Funktion initialisiert ein Schieberegister.

**OB 241:** Diese Funktion bearbeitet ein Schieberegister.

**OB 242:** Diese Funktion löscht ein Schieberegister.

### 6.10.1 Schieberegister initialisieren (OB 240)

Vor der Bearbeitung eines Schieberegisters muß dieses zuerst initialisiert werden. Dazu wird einmalig - zweckmäßigerweise in einem Anlauf-Organisationsbaustein - der OB 240 aufgerufen.

Die Parameter, die der OB 240 zum Einrichten eines bestimmten Schieberegisters benötigt, stehen in einem Datenbaustein mit der Nummer des zu initialisierenden Schieberegisters. Möglich sind damit DB-Nummern zwischen 192 und 255.

Der Datenbaustein ist nach einem festen Schema aufgebaut, das unbedingt eingehalten werden muß. Er kann maximal 9 Datenwörter lang sein (DW 0 bis DW 8).

Die einzelnen Datenwörter müssen folgendermaßen belegt werden:

0	DW 0
Schieberegisterlänge (Bytes)	DW 1
Nummer des 1. Merkerbytes	DW 2
Abstand $n_2$	DW 3
Abstand $n_3$	DW 4
Abstand $n_4$	DW 5
Abstand $n_5$	DW 6
Abstand $n_6$	DW 7
0	DW 8

Abb. 6-4: Aufbau des Datenbausteins zur Initialisierung eines Schieberegisters

Datenwort 0: Muß immer den Inhalt 0 haben.

Datenwort 1: Die Schieberegisterlänge ist die Anzahl (in Bytes) der Speicherzellen des Schieberegisters. Sie kann im Bereich  $2 \leq L \leq 256$  liegen.

Datenwort 2: Die Nummer des ersten Merkerbytes legt den Basiszeiger fest und damit den Merkerblock, der den Zeigern zugeordnet wird. Wenn Sie z.B. zwei Zeiger parametrieren, sind dies zusammen mit dem Basiszeiger drei Zeiger. Dann werden das im Datenbaustein festgelegte und die zwei darauffolgenden Merkerbytes reserviert. Achten Sie darauf, daß bis zum Ende des Merkerblocks noch genügend Merker für alle parametrierten Zeiger zur Verfügung stehen.

Datenwort 3

bis maximal 7: Abstände (in Bytes) der Zeiger zum Basiszeiger:

$n_2$  = Abstand von Zeiger 2 zum Basiszeiger  
 $n_3$  = Abstand von Zeiger 3 zum Basiszeiger  
 $n_4$  = Abstand von Zeiger 4 zum Basiszeiger  
etc.  
(maximal 5 Einträge)

Datenwort

nach letztem

Zeigerabstand: (Im Beispiel DW 8) Muß immer den Inhalt 0 haben.

Werden zum Basiszeiger nur zwei weitere Zeiger parametriert, so steht die '0' im Datenwort DW 5 etc..

Alle Angaben erfolgen als Festpunktzahlen.

**WICHTIG!**

- Die Anzahl der Zeiger (inklusive Basiszeiger) darf nicht größer sein als die Länge des Schieberegisters!
- Der Abstand eines Zeigers zum Basiszeiger darf nicht größer sein als die Länge des Schieberegisters.
- Das Datenwort DW 0 und das Datenwort nach dem letzten Zeigerabstand müssen immer den Inhalt 0 haben.
- Der Datenbaustein muß vor dem Aufruf des OB 240 programmiert und aufgerufen werden!

Durch Aufruf des OB 240 wird nun mit den Informationen aus diesem Datenbaustein ein bestimmter Speicherbereich am Ende des Datenbaustein-RAMs reserviert und initialisiert.

### Speicherplatz:

Für jedes Schieberegister werden

$$n = \text{Schieberegisterlänge} / 2 + 8 \text{ Datenwörter}$$

benötigt, d.h., die Länge des DB-RAMs verringert sich um n Datenwörter, wobei sich die Datenbaustein-RAM-Endadresse zu niedrigeren Adressen verschiebt.

Wenn ein Schieberegister, das initialisiert werden soll, schon vorhanden ist, wird bei gleicher Länge des neuen und des bereits vorhandenen Schieberegisters der schon belegte Bereich neu initialisiert. Andernfalls wird der alte Bereich für ungültig erklärt und ein neuer Bereich eröffnet.

### Parameter:

1. aufgerufener Datenbaustein  
mögliche Werte: DB-Nr. 192 - 255

### Fehlerfälle:

- unzulässige Datenbausteinnummer (<192, >255)
- vorhandener Speicherplatz im DB-RAM nicht ausreichend
- formaler Fehler im Aufbau des Datenbausteins
- unzulässige Längenangabe für das Schieberegister
- Parametrierungsfehler bei den Zeigern

Im Fehlerfall erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Die weitere Reaktion hängt von der Programmierung des OB 31 ab (siehe 'Sonstige Laufzeitfehler').

Ist der OB 31 nicht programmiert, geht der Prozessor in Stop. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher beschreiben.

### 6.10.2 Schieberegister bearbeiten (OB 241)

Der Sonderfunktions-Organisationsbaustein OB 241 bearbeitet ein Schieberegister, das zuvor durch den OB 240 initialisiert werden muß.

In der CPU 928 können maximal 64 Schieberegister aufgerufen werden.

#### Parameter:

1. Akku 1-L: Nummer des zu bearbeitenden Schieberegisters  
mögliche Werte: 192 - 255

Vor dem Aufruf des OB 241 werden im Normalfall bestimmte Merkerbits in den Zeigern gesetzt/rückgesetzt.

Bei jedem Aufruf des OB 241 wird die Information byteweise von einer Speicherzelle in die nächst höhere Speicherzelle verschoben. Die Zeiger werden dabei entsprechend mit neuen Inhalten versorgt. Durch wiederholtes Aufrufen des OB 241 kann die Information durch das gesamte Schieberegister bis in die letzte Speicherzelle 'durchgeschoben' werden, von wo aus sie wiederum in die Speicherzelle 1 gelangt.

Nach dem Aufruf des OB 241 enthalten die Zeiger (maximal 6 pro Schieberegister, mit Ausnahme des Basiszeigers beliebig positionierbar) die Information der davorliegenden Speicherzelle. Diese Information kann dann ausgewertet werden.

#### Fehlerfälle:

- unzulässige Schieberegister-Nummer im Akku 1
- Schieberegister nicht initialisiert

Im Fehlerfall erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Die weitere Reaktion hängt von der Programmierung des OB 31 ab (siehe 'Sonstige Laufzeitfehler').

Ist der OB 31 nicht programmiert, geht der Prozessor in Stop. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher beschreiben.



### 6.10.3 Schieberegister löschen (OB 242)

Mit dieser Sonderfunktion wird ein Schieberegister im Datenbaustein-RAM 'gelöscht': der Eintrag in der Adressliste DB 0 wird gelöscht und das entsprechende Schieberegister im DB-RAM für ungültig erklärt (Achtung: Auch gelöschte Schieberegister belegen weiterhin Speicherplatz!)

#### Parameter:

1. Akku 1-L: Nummer des zu löschenden Schieberegisters  
mögliche Werte: 192 bis 255

Nach Aufruf des OB 242 ist das Schieberegister gelöscht und kann nicht mehr verwendet werden; soll es wieder bearbeitet werden, muß es erneut initialisiert werden.

#### Fehlerfälle:

- unzulässige Schieberegister-Nummer im Akku 1
- Schieberegister nicht initialisiert

Im Fehlerfall erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Die weitere Reaktion hängt von der Programmierung des OB 31 ab (siehe 'Sonstige Laufzeitfehler'). Ist der OB 31 nicht programmiert, geht der Prozessor in Stop. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher beschreiben.

## 6.11 Regelung: PID-Algorithmus

Dieses Kapitel ist nur für Anwender, die mit PID-Reglern arbeiten wollen!  
Ansonsten werden die hier aufgeführten Informationen nicht benötigt!

In der CPU 928 des AG 135U können Sie einen oder mehrere PID-Regler aufrufen.

Jeder Regler muß im Anlauf-Organisationsbaustein initialisiert werden. Zur Übergabe von Parametern wird ein Datenbaustein verwendet.

Der eigentliche Regel-Algorithmus ist im Systemprogramm integriert und vom Anwender lediglich als Organisationsbaustein aufrufbar. Als Datenschnittstelle zwischen Regel-Algorithmus und dem Anwenderprogramm dient wieder ein Datenbaustein.

### Funktionsbeschreibung des PID-Reglers

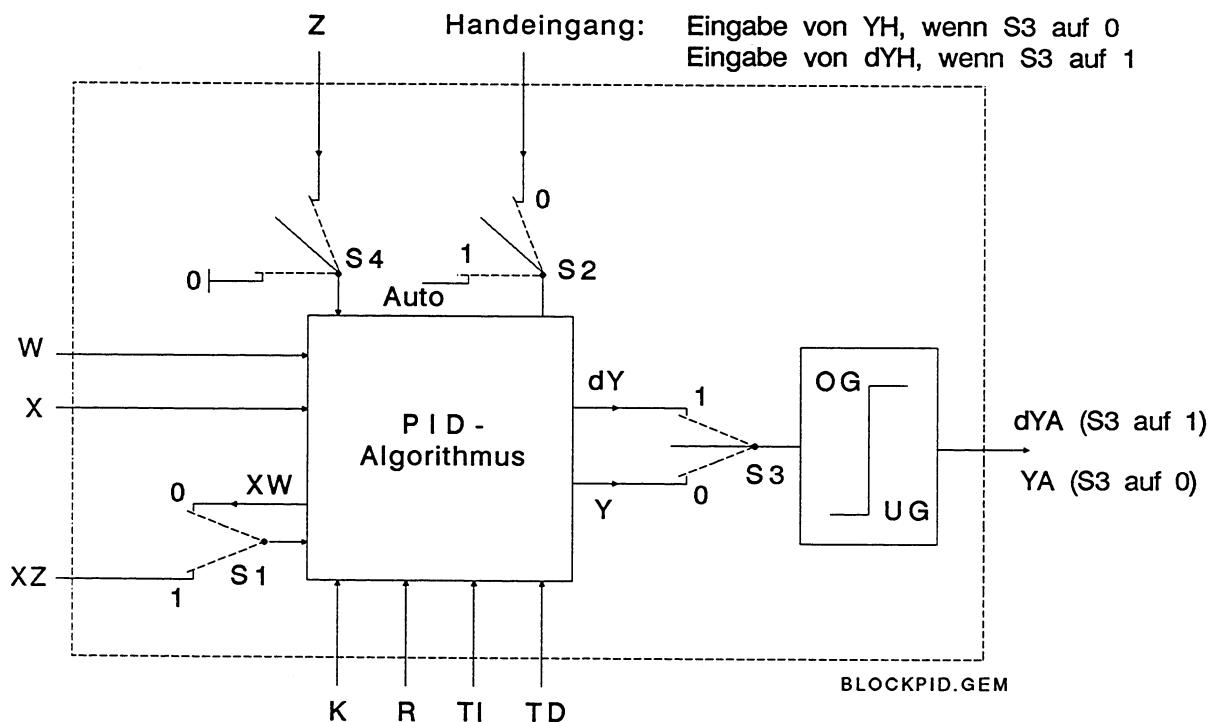


Abb. 6-5: Blockschaltbild des PID-Reglers

Index k: k-te Abtastung

Schalter	Stellung	Wirkung
S1 STEU-Bit 1	1	Dem Differenzierer wird die Regeldifferenz $XW_k$ zugeführt. Dem Differenzierer kann über XZ ein anderes Signal zugeführt werden.
S2 STEU-Bit 0	0 1	Handbetrieb Automatik
S3 STEU-Bit 3	0 1	Stellungs-Algorithmus Geschwindigkeits-Algorithmus
S4 STEU-Bit 5	0 1	Mit Störgrößenaufschaltung Ohne Störgrößenaufschaltung

Eine den Schalterstellungen dieses Blockschaltbildes entsprechende Funktion wird bei der Parametrierung des PID-Reglers dadurch erzielt, daß im Steuerwort STEU die Steuerbits passend gesetzt werden. Der kontinuierliche Regler ist für schnelle Regelstrecken, wie sie z.B. in der Verfahrenstechnik als Druck-, Temperatur- oder Durchflußregelungen auftreten, ausgelegt.

Dem Regler selbst liegt ein PID-Algorithmus zugrunde. Sein Ausgangssignal kann wahlweise als Stellgröße (Stellungs-Algorithmus) oder als Stellgrößenänderung (Geschwindigkeits-Algorithmus) ausgegeben werden.

Die einzelnen P-, I- und D-Anteile sind über ihre jeweiligen Parameter R, TI und TD abschaltbar, indem die betreffenden Zellen mit Null vorbesetzt werden. Damit können alle gewünschten Reglerstrukturen, z.B. PI-, PID- oder PD-Regler, leicht realisiert werden.

Dem Differenzierer kann wahlweise die Regeldifferenz XW oder - über den XZ-Eingang - eine beliebige Störgröße oder der invertierte Istwert  $-x$  zugeführt werden.

Für den Fall, daß zur Kompensation eines Störgrößeneinflusses eine Vorsteuerung des Stellgliedes ohne Zeitverhalten erwünscht ist, kann dem Regel-Algorithmus eine im Prozeß meßtechnisch erfaßbare Störgröße Z aufgeschaltet werden. Im Handbetrieb wird an dieser Stelle die vorgewählte Stellgröße YN übernommen.

Wenn ein invertierter Reglersinn gefordert wird, ist ein negativer K-Wert vorzugeben.

Wenn die Stellinformation (dY oder Y) an einer Begrenzung liegt, wird der I-Anteil automatisch abgeschaltet, um eine Verschlechterung des Reglerverhaltens zu vermeiden.

Das Reglerprogramm kann durch Vorgabe von Festwerten oder über adaptive (dynamische) Vorgabe von Parametern (K, R, TI, TD) versorgt werden. Die Eingabe erfolgt über die den einzelnen Parametern zugeordneten Speicherzellen.

## PID-Algorithmus

Dem PID-Regler liegt ein Geschwindigkeits-Algorithmus zugrunde, nach dem zu einem bestimmten Zeitpunkt  $t = k \cdot T_A$  das jeweilige Stellinkrement  $dY_k$  nach folgender Formel berechnet wird:

$$dY_k = K[(XW_k - XW_{k-1})R + \frac{T_A}{2TN}(XW_k + XW_{k-1}) +$$

$$\frac{1}{2}(\frac{TV}{T_A}(XU_k - 2XU_{k-1} + XU_{k-2}) + dD_{k-1})]$$

$$= K(dPW_k R + dI_k + dD_k)$$

P-Anteil    I-Anteil    D-Anteil

$dXXX_k$ : Änderung der Größe XXX zum Zeitpunkt t.

U kann W oder Z sein, je nachdem, ob dem Differenzierer XW oder XZ zugeführt wird. Entsprechend gilt:

Bei  $XW_k$ -Zuführung:

Bei  $XZ$ -Zuführung:

$$PW_k = W_k - X_k$$

$$PW_k = XW_k - XW_{k-1}$$

$$QW_k = PW_k - PW_{k-1}$$

$$QW_k = XW_k - 2XW_{k-1} + XW_{k-2}$$

$$PZ_k = XZ_k - XZ_{k-1}$$

$$QZ_k = PZ_k - PZ_{k-1}$$

$$QZ_k = XZ_k - 2XZ_{k-1} + XZ_{k-2}$$

$$dPW_k = (XW_k - XW_{k-1})R$$

$$dI_k = TI \cdot XW_k \quad TI = \frac{T_A}{TN}$$

$$dD_k = \frac{1}{2}(TD \cdot QU_k + dD_{k-1}) \quad TD = \frac{TV}{T_A}$$

Wenn als Reglerausgang zum Zeitpunkt  $t_k$  die Stellgröße  $Y_k$  gewünscht wird, wird sie nach folgender Formel gebildet:

$$Y_k = \sum_{m=0}^{m=k} dY_m$$

Bei den meisten Reglerentwurfsverfahren geht man davon aus, daß  $R = 1$  ist, wenn ein P-Verhalten erwünscht ist.

Mit der Größe R kann der Proportionalanteil des PID-Reglers eingestellt werden.

## **Datenbausteine für den PID-Regler**

Die reglerspezifischen Daten werden mit Hilfe eines Übergabe-Datenbausteins eingegeben (Initialisierung und Bearbeitung des PID-Reglers siehe Kapitel 6.11.1 und 6.11.2).

Diese Daten müssen Sie im Übergabe-Datenbaustein **x** vorgeben:

**K, R, TI, TD, W, STEU, YH, BGOG, BGUG**

Im folgenden wird der Aufbau des Übergabe-Datenbausteins näher erläutert, der aus 49 Datenwörtern mit den Nummern 0 bis 48 bestehen muß.

## Aufbau des Übergabe-Datenbausteins

Adr. im DB	Name	E/A	Zahlen- format	PG- Format	Bemerkung
DW 0	-	-	-	-	Reserve
DD 1	K	E	GP	KG	Proportionalbeiwert K > 0: Positiver Regelsinn d.h. gleichsinnige Änderung von Sollwert und Stellgröße K < 0: Negativer Regelsinn; Gleitpunktzahlenbereich
DD 3	R	E	GP	KG	R-Parameter, gewöhnlich = 1 bei Reglern mit P-Anteil; Gleitpunktzahlenbereich
DD 5	TI	E	GP	KG	TI = TA/TN; Gleitpunktzahlenbereich
DD 7	TD	E	GP	KG	TD = TV/TA; Gleitpunktzahlenbereich
DD 9	$W_k$	E	GP	KG	Sollwert-Eingabe hier, wenn STEU-Bit 6 = 1, ansonsten in Wort Nr. 19 ( $-1 \leq W_k < 1$ )
DW 11	STEU	E	BM	KM	Steuerwort
DD 12	$YH_k$	E	GP	KG	Handwert-Eingabe hier, wenn STEU-Bit 6 = 1; ansonsten in Wort Nr. 18 ( $-1 \leq YH_k < 1$ ). Bei Geschwindigkeits- Algorithmus sind hier Stellwert-Inkrement anzugeben.
DD 14	BGOG	E	GP	KG	Oberer Begrenzungswert $-1 \leq BGOG \leq 1 (YA_k \text{ max})$ ; !! BGUG < BGOG !!
DD 16	BGUG	E	GP	KG	Unterer Begrenzungswert $-1 \leq BGUG \leq 1 (YA_k \text{ min})$ ;
DW 18	$YH_k$	E	LP	KF	Handwert-Eingabe hier, wenn STEU-Bit 6 = 0 ( $-1 \leq YH < 1$ ). Bei Geschwindigkeits- Algorithmus sind hier Stellwert-Inkrement anzugeben.
DW 19	$W_k$	E	LP	KF	Sollwert-Eingabe hier, wenn STEU-Bit 6 = 0 ( $-1 \leq W_k < 1$ )

Adr. im DB	Name	E/A	Zahlen- format	PG- Format	Bemerkung
DW 20	MERK		BM	KM	Bit 0 = 1: positive Begrenzung überschritten; Bit 1 = 1: negative Begrenzung unterschritten
DW 21	$X_k$	E	LP	KF	Istwert-Eingabe für STEU-Bit 7 = 0 ( $-1 \leq X_k < 1$ )
DD 22	$X_k$	E	GP	KG	Istwert-Eingabe für STEU-Bit 7 = 1 ( $-1 \leq X_k < 1$ )
DW 24	$Z_k$	E	LP	KF	Störgröße ( $-1 \leq Z_k < 1$ )
DD 25	$Z_k$	E	GP	KG	Störgrößeneingabe hier, wenn STEU-Bit 7 = 1 ( $-1 \leq Z_k < 1$ )
DD 27	$Z_{k-1}$		GP	KG	Vergangenheitswert der Störgröße
DW 29	$XZ_k$	E	LP	KF	Über den Eingang XZ dem Differenzierer zugeführte Größe ( $-1 \leq XZ_k < 1$ ); Eingabe hier, wenn STEU-Bit 7 = 0
DD 30	$XZ_k$	E	GP	KG	XZ-Eingabe hier, wenn STEU- Bit 7 = 1 ( $-1 \leq XZ_k < 1$ )
DD 32	$XZ_{k-1}$		GP	KG	Vergangenheitswert von $XZ_k$
DD 34	$PZ_{k-1}$		GP	KG	$XZ_{k-1} - XZ_{k-2}$
DD 36	$dD_{k-1}$		GP	KG	Differentialanteil
DD 38	$XW_{k-1}$		GP	KG	Vergangenheitswert der Regeldifferenz
DD 40	$PW_{k-1}$		GP	KG	$XW_{k-1} - XW_{k-2}$
DW 42	-	-	-	-	Reserve
DD 44	$Y_{k-1}$		GP	KG	Vergangenheitswert der be- rechneten Stellgröße $Y_{k-1}$ bzw. $dY_{k-1}$ vor dem Begrenzer
DD 46	$YA_k$	A	GP	KG	Ausgangsgröße
DW 48	$YA_k$	A	LP	KF	Ausgangsgröße $BGUG \leq YA \leq BGOG$

Vorgeschlagenes Format  
(KH, KM ebenfalls zulässig)

GP = Gleitpunkt-, LP = Linkspunktzahl

E = Eingabe, A = Ausgabe

# Belegung des Steuerwortes STEU (Datenwort DW 11 im Übergabe-DB)

DW 11 BitNr	Name	Bedeutung
11.0	AUTO	= 1: Automatikbetrieb = 0: Handbetrieb
11.1	XZ_EIN	= 1: Dem Differenzierer wird über den XZ-Eingang eine andere Größe zugeführt, die nicht $XW_k$ sein darf. = 0: Dem Differenzierer wird $XW_k$ zugeführt. Der XZ-Eingang bleibt unberücksichtigt.
11.2	REG_AUS	= 1: Beim Aufruf des Reglers (OB 251) werden mit Ausnahme von K, R, TI, TD, BGOG, BGUG, STEU, $YH_k$ , $W_k$ , $Z_k$ und $Z_{k-1}$ alle anderen Größen (DW 20 bis DW 48) im Regler-DB einmal gelöscht. Der Regler ist ausgeschaltet. Der Vergangenheitswert der Störgröße wird aktualisiert. = 0: Regeln
11.3	GESCHW	= 1: Geschwindigkeits-Algorithmus = 0: Stellungs-Algorithmus
11.4 <sup>1)</sup>	HANDART	= 1: Bei GESCHW = 0 (Stellungs-Algorithmus) wird die zuletzt ausgegebene Stellgröße beibehalten. Bei GESCHW = 1 (Geschwindigkeits-Algorithmus) wird das Stellinkrement $dY_k = 0$ gesetzt. = 0: Bei GESCHW = 0 wird nach dem Umschalten auf Handbetrieb der ausgegebene Stellwert YA in 4 Abtastschritten exponentiell auf den eingestellten Handwert geführt. Danach werden weitere Handwerte sofort am Reglerausgang übernommen. Bei GESCHW = 1 werden die Handwerte sofort auf den Reglerausgang durchgeschaltet. Im Handbetrieb sind die Begrenzungen wirksam. Im Handbetrieb werden folgende Größen aktualisiert: 1) $X_k$ , $XW_{k-1}$ und $PW_{k-1}$ 2) $XZ_k$ , $XZ_{k-1}$ und $PZ_{k-1}$ , wenn STEU-Bit 1 = 1 3) $Z_k$ und $Z_{k-1}$ , wenn STEU-Bit 5 = 0 Die Größe $dd_{k-1}$ wird = 0 gesetzt. Der Algorithmus wird nicht berechnet.
11.5	NO_Z	= 1: Keine Störgrößenaufschaltung = 0: Mit Störgrößenaufschaltung
11.6	PGDG	= 1: $W_k$ -, $YH_k$ -Eingabe als Gleitpunktzahl = 0: Eingabe als Linkspunktzahl
11.7	VAR_GP	= 1: Die Variablen $X_k$ , $XZ_k$ und $Z_k$ werden als Gleitpunktzahl eingegeben. = 0: Eingabe der Variablen als Linkspunktzahl
11.8	STOS	= 1: Keine stoßfreie Hand-Automatik-Umschaltung = 0: Stoßfreie Hand-Automatik-Umschaltung
11.9 bis 11.15		Ohne Bedeutung

<sup>1)</sup> Nur bei Handbetrieb (AUTO = 0) relevant.



### 6.11.1 PID-Algorithmus initialisieren (OB 250)

Der OB 250 initialisiert den PID-Algorithmus und wird in den Anlauf-OBs 20/21/22 aufgerufen.

Die für das Initialisieren erforderlichen Parameter stehen im Übergabe-Datenbaustein (DB x).

#### **WICHTIG!**

**Der Übergabe-Datenbaustein muß vor Aufruf des OB 250 aufgeschlagen werden.**

Für jeden Regler muß zur Datenübergabe ein eigener DB x verwendet werden ( $x \leq 254$ ). Das Systemprogramm erzeugt daraus automatisch einen weiteren DB x + 1 im Datenbaustein-RAM, den der Regler im zyklischen Betrieb als Datenfeld verwendet; die entsprechenden DB-Nummern müssen also noch frei sein. Die Datenbausteine DB x + 1 sind die jeweilige Datenschnittstelle zwischen den Reglern und dem Anwender bzw. der Peripherie.

Der OB 250 verwendet intern den OB 254 bzw. OB 255 (Duplizieren von Datenbausteinen). Im Fehlerfall erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Wenn dieser nicht programmiert ist, geht der Prozessor in den Stopp. Die im Akku 1 hinterlegten Fehlerkennungen beziehen sich dann auf den OB 250.

**Achtung!** Wenn bei der Initialisierung der DB x + 1 nicht freigehalten war, wird dieser ohne Meldung vom Systemprogramm als Reglerdatenfeld verwendet, sofern er die gleiche Länge hat wie ein Regler-DB (48 Datenwörter); dabei werden die Datenwörter 20 bis 48 gelöscht. Ansonsten geht der Prozessor in den Stoppzustand.

Statt Datenbausteinen DB können auch erweiterte Datenbausteine DX verwendet werden. Die Initialisierung verläuft dabei analog zu der bei Datenbausteinen DB.

### 6.11.2 PID-Algorithmus bearbeiten (OB 251)

Der OB 251 wird während der zyklischen Programmbearbeitung aufgerufen und bearbeitet den PID-Algorithmus.

Nach Ablauf der Abtastzeit soll der Regler aufgerufen werden. Halten Sie dabei folgende Reihenfolge ein:

- Datenbaustein DB  $x + 1$  aufrufen
- Laden der Eingangsdaten  $X_k$ ,  $XZ_k$ ,  $Z_k$  und  $YH_k$  oder eine Unter-  
menge davon
- formatrichtige Wandlung der Eingangsdaten und Transfer in den  
DB  $x + 1$
- Aufruf des OB 251 (PID-Regler bearbeiten)
- Laden des Ausgangsdatums  $YA_k$  aus dem DB  $x + 1$
- Wandlung des Datums und Transfer zur Prozeßperipherie.

#### Format der Reglereingänge und -ausgänge

Der PID-Regelalgorithmus verwendet intern zur Zahlendarstellung das Gleitpunktformat und kann mit Gleitpunktwerten versorgt werden. Eine Versorgung des PID-Regelalgorithmus im Linkspunktformat ist ebenfalls möglich (siehe dazu Bit 6 und 7 im Steuerwort STEU). In diesem Fall wandelt der Regler bei jedem Aufruf selbständig die Wörter ins Gleitpunktformat um.

Die Anpassung der Wörter von den Eingabe- und Ausgabebaugruppen im STEP-5-Programm ist lauffzeitgünstiger bei Verwendung des Linkspunktformates.

#### Eingänge

W, YH, X, Z und XZ können wahlweise als Gleit- oder Linkspunktzahl eingegeben werden. Im Datenübergabebaustein sind für jede Größe jeweils unterschiedliche Speicherplätze vorgesehen.

### Eingabe als Linkspunktzahl

(Erläuterungen zur Linkspunktzahl: Siehe Ende des Kapitels 6.11)

**Achtung!** Bei Einhaltung der Eingangsnennbereiche der Analogeingabebaugruppen ist zu beachten, daß das Bitmuster für einen bestimmten Eingangswert anders ist als bei Ausnutzung des vollen Eingangsbereichs. Diese Tatsache muß insbesondere bei der Sollwerteinstellung berücksichtigt werden, da es sonst vorkommen kann, daß ein über das PG eingegebener Sollwert nicht erreicht werden kann, obwohl der Istwert weit über dem gewünschten Wert liegt.

Wenn der eingesetzte Analog-Digital-Umsetzer die negativen Zahlen als Betrag und Vorzeichen liefert, muß man daraus, bevor sie in den Regler-DB transferiert werden, das Zweierkomplement bilden. Anschließend muß die Binärstelle 15 = 1 gesetzt werden.

Wenn die Zahl -0 als Betrag und Vorzeichen in der Form

1000000000000000

beim verwendeten Analog-Digital-Umsetzer möglich ist, darf davon kein Zweierkomplement gebildet werden, sondern die Zahl muß als +0 in den Regler-DB gelangen:

0000000000000000

### Ausgang

Der Reglerausgang YA liegt im DB als Links- und als Gleitpunktzahl vor. Linkspunkteingänge und -ausgänge müssen unter Berücksichtigung der verwendeten Ein- und Ausgabebaugruppen (Analog-Digital-Umsetzer, Digital-Analog-Umsetzer) vor und nach dem Regleraufruf im STEP5-Anwenderprogramm formatgewandelt werden, bevor sie in den bzw. aus dem Regler-DB transferiert werden.

## Allgemeine Hinweise

Wenn STOS (STEU-Bit 8) auf Null steht, verläuft die Umschaltung von Hand- auf Automatikbetrieb stoßfrei; d.h., eine anstehende, beliebig große Regeldifferenz wird nur über den I-Anteil ausgeregelt. Wenn jedoch  $TI = TA/TN = 0$  gewählt wird (P- oder PD-Regler), verursacht die Regeldifferenz bei der Umschaltung keine Änderung der Stellgröße.

Dies kann vermieden werden, indem man  $STOS = 1$  setzt. Eine Regeldifferenz wird dann bei Hand-Automatik-Umschaltung schnell ausgeregelt, gleichgültig ob  $TI = 0$  ist oder nicht. Der dabei entstehende Stellgrößensprung entspricht der Größe der Regeldifferenz, ist also nicht willkürlich im Sinn einer Störung des Reglerbetriebs.

Bit 0 und 1 von MERK können, falls gewünscht, zur Anzeige gebracht werden, um anzuzeigen, daß die Stellgröße (bei Geschwindigkeitsalgorithmus das Stellinkrement) in der oberen oder unteren Begrenzung ist. Da diese Bits vom Algorithmus zur Abschaltung des I-Anteils ausgewertet werden, dürfen sie nicht überschrieben werden.

Regler-Datenbausteine  $DB\ x + 1$  dürfen während des zyklischen Betriebs nicht neu geladen werden.

Wenn mit zwei oder mehr Reglern eine Kaskadenregelung aufgebaut wird, ist folgendes zu beachten:

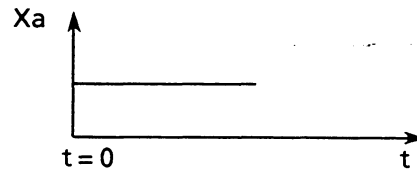
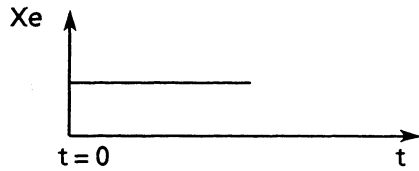
- Wenn die Kaskade aufgetrennt wird, müssen entweder alle Regler gleichzeitig in den Handbetrieb gehen, damit kein Regler infolge seines I-Anteils abdriften kann, oder es muß zumindest der Regler des äußeren Kreises im Handbetrieb arbeiten, damit die letzte Stellgröße, die dem Sollwert des inneren Kreises entspricht, beibehalten oder auf einen Sicherheitswert gefahren werden kann.
- Wenn die Kaskade geschlossen werden soll, sollten beide Kreise gleichzeitig oder wenigstens der innere Kreis im Automatikbetrieb arbeiten, damit die Stellgröße des äußeren Kreises als Sollwert übernommen werden kann.

Wenn beim Umschalten auf Handbetrieb die Regelstrecke vom Regler abgetrennt und am Stellglied direkt verstellt wird, ist die so gewonnene Stellgröße über den Handeingang dem Regler zuzuführen. Dies bewirkt, daß beim Umschalten von Hand- auf Automatikbetrieb der Reglerausgang mit der im Handbetrieb eingestellten Stellgröße übereinstimmt. Beim Geschwindigkeits-Algorithmus handelt es sich um die Stellgrößenänderung.

## Reglerkenngrößen

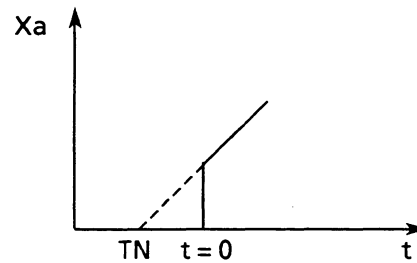
### P-Regler

Die Kenngröße für einen P-Regler ist  $K$ . Sie ist der Quotient aus Ausgangs- und Eingangsgröße:  $K = X_a/X_e$ .



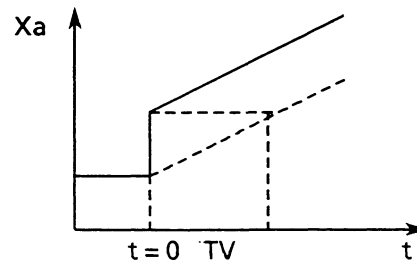
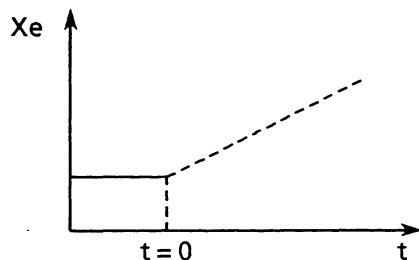
### PI-Regler

Die Kenngrößen für einen PI-Regler sind der Proportionalbeiwert  $K$  und die Nachstellzeit  $T_N$ . Der Proportionalbeiwert  $K$  ist der Quotient aus Ausgangsgröße und Eingangsgröße und bestimmt das P-Verhalten. Die Nachstellzeit  $T_N$  ist diejenige Zeit, die bei der Antwort benötigt wird, um infolge der I-Wirkung eine gleichgroße Stellgrößenänderung zu erzielen, wie sie infolge des P-Anteils entsteht.



### PD-Regler

Die Kenngrößen für einen PD-Regler sind der Proportionalbeiwert  $K$  (siehe oben) und die Vorhaltezeit  $T_V$ . Die Vorhaltezeit ist diejenige Zeit, die ein P-Regler bei konstanter Änderungsgeschwindigkeit der Eingangsgröße benötigen würde, um die gleiche Änderung der Ausgangsgröße zu bewirken, die ein PD-Regler infolge seines D-Anteils sofort bewirkt. Um die Vorhaltezeit  $T_V$  zu bestimmen, geht man nicht von einer Sprungfunktion aus, sondern von einer linearen Änderung der Eingangsgröße.



## PID-Regler

Die Kenngröße für einen PID-Regler sind der Proportionalbeiwert  $K$ , die Nachstellzeit  $T_N$  und die Vorhaltezeit  $T_V$ . Sie bestimmen jeweils das P-, I- und D-Verhalten.

## Parameteränderung

Der P-Anteil der Stellgröße wird nach folgender Formel gebildet:

$$\text{P-Anteil} = \quad \quad \quad K_P \cdot (XW_k - XW_{k-1})$$

Wird  $K_P$  oder  $R$  im Automatikbetrieb geändert, wirkt sich dies nur auf nachfolgende Änderungen der Regeldifferenz  $XW_k$  aus. Der momentane Wert der Stellgröße bleibt von der Parameteränderung unbeeinflusst. Dieses Verhalten ermöglicht eine stoßfreie Parameteränderung.

Ist dieses Verhalten jedoch nicht erwünscht, so kann es durch folgende Berechnung, die nur einmal bei jeder Parameteränderung vorzunehmen ist, beseitigt werden (Beispiel für  $K_P$ -Änderung):

$$Y_{k-1} = Y_{k-1} + XW_{k-1}(K_{P_{\text{neu}}} - K_{P_{\text{alt}}})$$

Wird bei einer Parameteränderung folgendes Programm verwendet, verhält sich der Regler wie ein analoger Regler:

```
:L   KP_neu           KP_neu laden
:L   KP_alt           KP_alt laden
:-G
:L   DD38             XW_k-1
:xG
:L   DD44             Y_k-1
:+G
:T   DD44             = Y_k-1
```

## Abkürzungen für PID-Regler

$dY_k$	Berechnetes Stellinkrement
$dZ_k$	Störinkrement
GP	Gleitpunktdarstellung
k	k-te Abtastung
K	Proportionalbeiwert
LP	Linkspunktdarstellung
OG	Obere Grenze (Begrenzer)
R	R-Parameter
TA	Abtastzeit
TD	$T_V/TA$
TI	$TA/T_N$
t	Abtastzeitpunkt = $k \cdot TA$
$T_N$	Nachstellzeit
$T_V$	Vorhaltzeit
UG	Untere Grenze (Begrenzer)
$W_k$	Sollwert
$X_k$	Istwert
$XW_k$	Regeldifferenz
$Y_k$	Berechnete Stellgröße
$YA_k$	Stellwert (Stellinkrement oder Stellgröße)
$Z_k$	Störgröße

## Linkspunktzahl

Zur Darstellung einer Linkspunktzahl im Datenbaustein wird ein Wort benötigt. Die Zuordnung zwischen dezimal dargestellter Linkspunktzahl, dual dargestellter Linkspunktzahl und der Darstellung im Format KF am Programmiergerät ist im folgenden Beispiel dargestellt.

Linkspunktzahl in		Festpunktzahl
Dezimaldarstellung	Dualdarstellung	
-0.999...	1000000000000001	-32767
-0.75	1010000000000000	-24576
-0.5	1100000000000000	-16384
-0.25	1110000000000000	- 8192
0	0000000000000000	0
+0.25	0010000000000000	+ 8192
+0.5	0100000000000000	+16384
+0.75	0110000000000000	+24576
+0.999...	0111111111111111	+32767

Negative Linkspunktzahlen ergeben sich in der Dualdarstellung durch Zweierkomplementbildung aus positiven Linkspunktzahlen.

Linkspunktzahlen (LF) lassen sich nach der folgenden Beziehung umrechnen in die am Programmiergerät dargestellten Werte (KF):

$$LP \cdot 32767 = KF$$

$$\text{mit } -1 < LP < +1 \quad \text{und} \quad -32767 \leq KF \leq +32767$$

## **7 Erweiterter Datenbaustein DX 0**

Als Anwender können Sie bestimmte Leistungen des Systemprogramms Ihren Erfordernissen anpassen, indem Sie im DX 0 alternativ zu den Standardvoreinstellungen (in der folgenden Tabelle mit "V" gekennzeichnet) andere Einstellungen eingeben.

Die Standardvoreinstellungen des Systemprogramms (V) werden automatisch bei jedem Neustart gesetzt. Danach wird der DX 0 ausgewertet. Ist kein DX 0 programmiert, gelten weiterhin die Standardvoreinstellungen, ansonsten werden die Einstellungen des Anwenders gültig.

### **WICHTIG!**

Eine Eingabe oder eine Änderung des DX 0 wird nur über einen Neustart wirksam.



## Aufbau des DX 0

Der DX 0 setzt sich aus drei Teilen zusammen:

1. der Anfangskennung für den DX 0 (DW 0, 1 und 2)
2. mehreren Blöcken unterschiedlicher Länge (je nach Parameteranzahl)
3. der Endekennung EEEE.

Die angegebenen Zahlenwerte entsprechen dem Hexadezimalformat.

Formaler Aufbau:

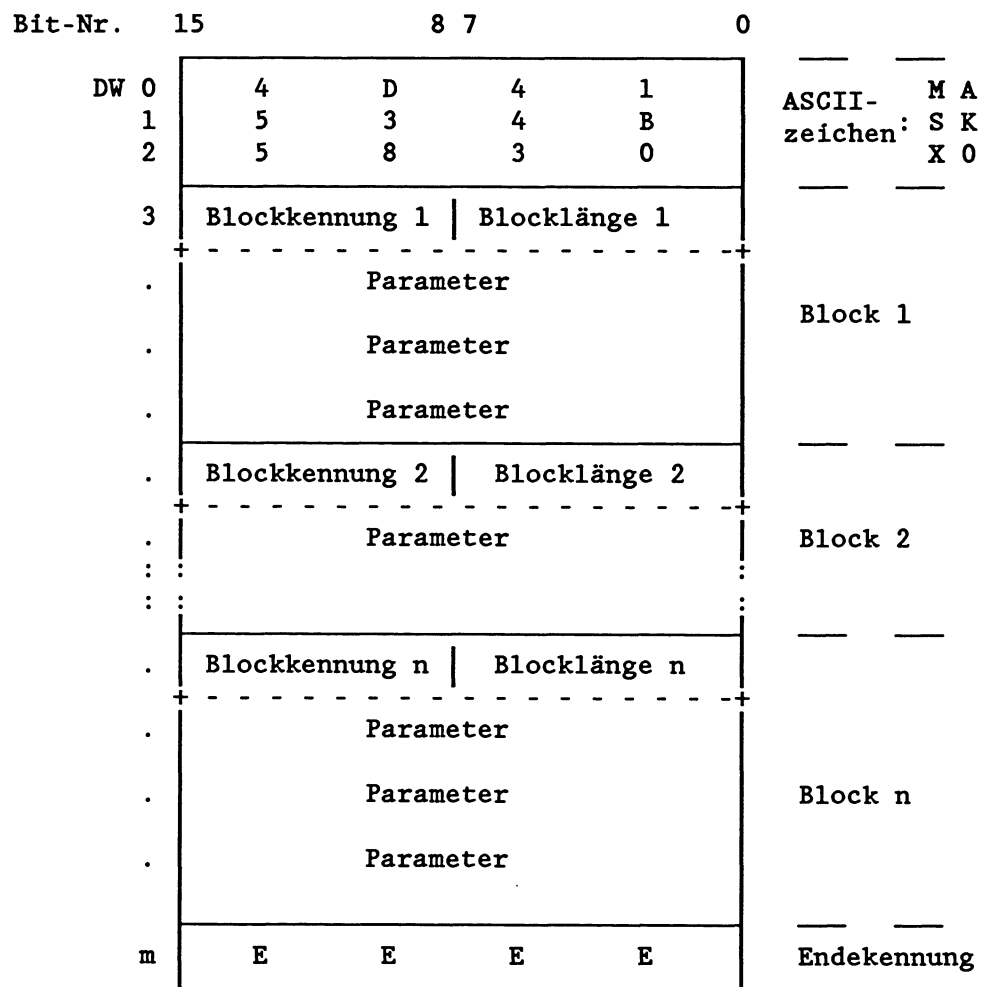


Abb. 7-1: Aufbau des DX 0

## Beispiel für die Eingabe des DX 0

Anfangskennung	DW 0:	KH= 4D41	
	DW 1:	KH= 534B	
	DW 2:	KH= 5830	
Blockkennung/-länge	DW 3:	KH= 0203	
Parameter (belegt 1 DW)	DW 4:	KH= 3001	Block 1
Parameter (belegt 2 DWs)	DW 5:	KH= BB00	
	DW 6:	KH= 0000	
Blockkennung/-länge	DW 7:	KH= 0402	
Parameter (belegt 2 DWs)	DW 8:	KH= 1000	Block 2
	DW 9:	KF= 4000	
Endekennung	DW10:	KH= EEEE	

Ein **Block** im DX 0 besteht aus 1 bis n Datenwörtern.

Diese enthalten

- die Blockkennung
- die Blocklänge
- die Blockparameter.

Die **Blockkennung** gibt an, welche *Bedeutung* die folgenden Parameter haben. Jeder Block ist einem bestimmten Systemprogrammteil oder einer bestimmten Systemfunktion zugeordnet (z.B. Blockkennung '04' --> zyklische Programmbearbeitung).

Die **Blocklänge** gibt an, *wieviele Datenwörter* die nachfolgenden Parameter belegen.

Die möglichen **Parameter** sind auf den folgenden Seiten angegeben.

Beachten Sie bei der Parametrierung des DX 0 folgende Punkte:

- Die Reihenfolge, in der Sie die einzelnen Blöcke eingeben, ist beliebig.
- Nicht benötigte Blöcke brauchen nicht angegeben zu werden.
- Ist ein bestimmter Block mehrfach vorhanden, gilt der Block, den Sie zuletzt eingeben haben.
- Die Reihenfolge, in der Sie die einzelnen Parameter eingeben, ist beliebig.
- Nicht benötigte Parameter brauchen nicht angegeben zu werden.
- Ist ein bestimmter Parameter mehrfach genannt, gilt die letzte Angabe.

### WICHTIG!

**Nach Eingabe des letzten Blocks müssen Sie den DX 0 unbedingt mit der Endekennung EEEE abschließen!**

## Mögliche Angaben im DX 0

Block- kennung /-länge	Parameter	Bedeutung <sup>1)</sup>
ANLAUF und RUN:		
02xx <sup>2)</sup>	1000	V Automatischer Wiederanlauf nach "Netz ein"
	1001	Automatischer Neustart nach "Netz ein"
	2000	V Synchronisation des Anlaufs im Mehrprozessorbetrieb
	2001	Keine Synchronisation des Anlaufs im Mehrprozessorbetrieb
	3000	V Adressierfehlerüberwachung
	3001	Keine Adressierfehlerüberwachung
	6000	V Gleitpunktarithmetik mit 16-Bit-Mantisse (geschwindigkeitsoptimiert)
	6001	Gleitpunktarithmetik mit 24-Bit-Mantisse (genauigkeitsoptimiert)
	BB00 yyyy	Anzahl der zu aktualisierenden Zeitzellen <sup>3)</sup>
		Voreinstellung: yyyy = 256 Zeitzellen, d.h. Zeitzelle 0 bis 255 zulässig: 0...256

<sup>1)</sup> V = Voreinstellung bei nicht geladenem DX 0 oder fehlendem Block

<sup>2)</sup> xx = Blocklänge (Anzahl der von den Parametern belegten Datenwörter)

<sup>3)</sup> Zur Aktualisierung der Zeitzellen:

- Standardmäßig werden die Zeitzellen T 0 bis T 255 bearbeitet.
- Wenn Sie im DX 0 die Anzahl '0' eintragen, werden keine Zeitzellen bearbeitet.
- Bei einer Anzahl zwischen '1' und '128' werden alle Zeitzellen von T 0 bis T 127 bearbeitet, bei einer Anzahl zwischen '129' und '256' werden alle Zeitzellen von T 0 bis T 255 bearbeitet.

Block- kennung /-länge	Parameter	Bedeutung <sup>1)</sup>
------------------------------	-----------	-------------------------

#### Zyklische Programmbearbeitung:

04xx <sup>2)</sup>	1000 yyyy	Länge der Zykluszeit in Millisekunden; Voreinstellung: yyyy = 150 ms, zulässig: $1 \leq yyyy \leq 1770$ (hex) 1 ms bis 6000 ms (dez)
	4000	V Aktualisierung des Prozeßabbildes der Koppelmerker ohne Semaphorschutz
	4001	Aktualisierung des Prozeßabbildes der Koppelmerker unter Semaphorschutz (im Block)

#### Alarmgesteuerte Programmbearbeitung:

06xx <sup>2)</sup>	1006	Bearbeitung des Prozeßalarms, Regleralarms und Weckalarms an STEP5-Befehlsgrenzen <sup>3)</sup>
	1008	Bearbeitung des Prozeß- und Regleralarms an STEP5-Befehlsgrenzen, Bearbeitung des Weckalarms an Bausteingrenzen
	100A	Bearbeitung des Prozeßalarms an STEP5- Befehlsgrenzen, Bearbeitung des Regleralarms und Weckalarms an Bausteingrenzen
	100C	V Bearbeitung des Prozeßalarms, Regleralarms und Weckalarms an Bausteingrenzen

Die obigen Parameter gelten für die Version der CPU 928, die nur den 100ms-Weckalarm kennt. Mit der neuen Version der CPU 928 lassen sich jetzt bis zu 9 verschiedene Weckalarm-OBs programmieren. Für diese gibt es im DX 0 neue Kennungen (siehe folgende Seite).

<sup>1)</sup> V = Voreinstellung bei nicht geladenem DX 0 oder fehlendem Block

<sup>2)</sup> xx = Blocklänge (= Anzahl der von den Parametern belegten Datenwörter)

<sup>3)</sup> **Hinweis:**

Wenn Sie im DX 0 eine Bearbeitung von Alarmen an STEP5-Befehlsgrenzen vorsehen, sollten Sie beachten, daß bei Unterbrechungen die Befehle 'TNB' und 'TNW' u.U. nicht fertig bearbeitet sind, da sie Pseudobefehlsgrenzen enthalten. Dies gilt ebenso für einige wenige Sonderfunktions-Organisationsbausteine, Standardfunktionsbausteine und Regler-FBs.

Prozeßalarm = prozeßalarmgesteuerte Programmbearbeitung  
Weckalarm = zeitgesteuerte Programmbearbeitung

Durch Programmierung der neuen Parameter im DX 0 ergeben sich für die Alarmbearbeitung folgende Einstellmöglichkeiten:

		Weckalarme									Prozeßalarm			
Parameter		5 s	2 s	1 s	500 ms	200 ms	100 ms	50 ms	20 ms	10 ms	Regler		Parameter alt	
122C	V	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	100C	V
1224		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	100A	
121C		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	1008	
1216		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
1214		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
1212		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
1210		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
120E		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
120C		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
120A		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
1208		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	-	
1206		<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	1006	

V = Voreinstellung

ALARME.GEM

- ☐ Unterbrechung an Bausteingrenzen
- ☐ Unterbrechung an Befehlsgrenzen

Block-kennung	Parameter	Bedeutung <sup>1)</sup>
06xx <sup>2)</sup>	2000	V Prozeßalarm-Signal, pegelgetriggert
	2001	Prozeßalarm-Signal, flankengetriggert

Fehlerbehandlung :

10xx <sup>2)</sup>	1000	Weckfehlerbearbeitung V Systemstopp, wenn Ereignis eintritt und der OB 33 nicht geladen ist.
	1001	Kein Systemstopp, wenn Ereignis eintritt und der OB 33 nicht geladen ist.
	1200	Reglerfehlerbearbeitung V Systemstopp, wenn Ereignis eintritt und der OB 34 nicht geladen ist.
	1201	Kein Systemstopp, wenn Ereignis eintritt und der OB 34 nicht geladen ist.
	1400	Zyklusfehlerbearbeitung V Systemstopp, wenn Ereignis eintritt und der OB 26 nicht geladen ist.
	1401	Kein Systemstopp, wenn Ereignis eintritt und der OB 26 nicht geladen ist.
	1800	Befehlscodefehlerbearbeitung V Systemstopp, wenn Ereignis eintritt und der OB 27/29/30 nicht geladen ist.
	1801	Kein Systemstopp, wenn Ereignis eintritt und der OB 27/29/30 nicht geladen ist.

<sup>1)</sup> V = Voreinstellung bei nicht geladenem DX 0 oder fehlendem Block

<sup>2)</sup> xx = Blocklänge (= Anzahl der von den Parametern belegten Datenwörter)

Befehlscodefehler = Substitutions-, Opcode-, Parameterfehler

Laufzeitfehler = Aufruf eines nicht geladenen Bausteins, Transferfehler oder sonst. Laufzeitfehler

Block- kennung	Parameter	Bedeutung <sup>1)</sup>
10xx <sup>2)</sup>	1A00	Laufzeitfehlerbearbeitung  V Systemstopp, wenn Ereignis eintritt und der OB 19/31/32 nicht geladen ist.
	1A01	Kein Systemstopp, wenn Ereignis eintritt und der OB 19/31/32 nicht geladen ist.
	1C00	Adressierfehlerbearbeitung  V Systemstopp, wenn Ereignis eintritt und der OB 25 nicht geladen ist.
	1C01	Kein Systemstopp, wenn Ereignis eintritt und der OB 25 nicht geladen ist.
	1E00	Quittungsverzugsfehler  Systemstopp, wenn Ereignis eintritt und der OB 23/24 nicht geladen ist.
	1E01	V Kein Systemstopp, wenn Ereignis eintritt und der OB 23/24 nicht geladen ist.
EEEE		Endekennung

<sup>1)</sup> V = Voreinstellung bei nicht geladenem DX 0 oder fehlendem Block

<sup>2)</sup> xx = Blocklänge (= Anzahl der von den Parametern belegten Datenwörter)

### Beispiel A für die Parametrierung des DX 0:

Sie wollen im Mehrprozessorbetrieb drei Prozessoren einsetzen: Prozessor A, B und C. Prozessor A und B arbeiten eng miteinander zusammen, tauschen häufig Daten aus und bearbeiten ein umfangreiches Anlaufprogramm. Prozessor C bearbeitet weitgehend unabhängig davon ein kurzes, zeitkritisches Programm.

Standardmäßig beginnen im Mehrprozessorbetrieb alle Prozessoren gemeinsam mit der zyklischen Programmbearbeitung, d.h., die Prozessoren warten solange aufeinander, bis alle ihren Anlauf beendet haben, und gehen dann gemeinsam in die zyklische Programmbearbeitung.

Da Prozessor C sein Programm unabhängig von den anderen Prozessoren ausführt und ein sehr kurzes Anlaufprogramm bearbeitet, ist bei ihm keine Synchronisation des Anlaufs notwendig. Durch Parametrierung des DX 0 erreichen Sie, daß Prozessor C nach beendetem Anlauf sofort in die zyklische Programmbearbeitung geht, ohne auf Prozessor A und B zu warten.

Programmieren Sie den DX 0:

<u>DX 0</u>	Anfangskennung	DW 0:	KH= 4D41
		DW 1:	KH= 534B
		DW 2:	KH= 5830
	1. Blockkennung/-länge	DW 3:	KH= 0201
	Parameter 1	DW 4:	KH= 2001
	Endekennung	DW 5:	KH= EEEE

Haben Sie diesen DX0 in den Programmspeicher geladen, wird er mit dem nächsten Neustart wirksam. Da Prozessor C ein sehr kurzes Anlaufprogramm bearbeitet und nicht auf A und B wartet, geht bei ihm sofort die grüne RUN-Led an. Das BASP-Signal (Befehlsausgabesperre) wird jedoch erst zurückgenommen, wenn alle drei Prozessoren ihren Anlauf beendet haben. Dies bedeutet, daß Prozessor C nicht auf die Digitalperipherie zugreifen darf.



### Beispiel B für die Parametrierung des DX 0:

Mit der folgenden Parametrierung des DX 0 wird

- a) die Adressierfehler-Überwachung abgeschaltet,
- b) die Zeitzellenaktualisierung abgeschaltet,
- c) die Zykluszeit auf 4 sec eingestellt.

<u>DX 0</u>	Anfangskennung	DW 0:	KH= 4D41
		DW 1:	KH= 534B
		DW 2:	KH= 5830
	1. Blockkennung/-länge	DW 3:	KH= 0203
	Parameter	DW 4:	KH= 3001
	Parameter *)	DW 5:	KH= BB00
		DW 6:	KH= 0000
	2. Blockkennung/-länge	DW 7:	KH= 0402
	Parameter *)	DW 8:	KH= 1000
		DW 9:	KF= 4000
	Endekennung	DW10:	KH= EEEE

- \*) Parameter, die zwei Datenwörter belegen, müssen bei der Angabe der Blocklänge mit '2' berücksichtigt werden!

Diese Parametrierung des DX 0 hat folgende Auswirkungen auf die Programmbearbeitung:

- Derjenige Teil des Prozeßabbilds, dem keine Peripheriebaugruppen zugeordnet sind, kann als zusätzlicher "Merkerbereich" benutzt werden.
- Die Laufzeit des Systemprogramms verkürzt sich, da keine Zeitzellen aktualisiert werden.
- Ein Zyklusfehler wird erst dann erkannt, wenn die Laufzeit des zyklischen Anwenderprogramms und des Systemprogramms zusammen 4 sec übersteigt.

## 8 Speicherbelegung und Speicherorganisation

Der gesamte Speicherbereich in der CPU 928 untergliedert sich im wesentlichen in folgende Bereiche:

	Breite:
1. Anwenderspeicher für OB, FB, FX, PB, SB, DB, DX	(16 Bits)
2. DB-RAM für Datenbausteine, Schieberegister	(16 Bits)
3. - Bereich Anschaltung: BA, BB	(16 Bits)
- Bereich System: BS, BT	(16 Bits)
- Zähler: Z	(16 Bits)
- Zeiten: T	(16 Bits)
4. - Merker: M	(8 Bits)
- Prozeßabbild der Ein- und Ausgänge: PAE, PAA	(8 Bits)
5. Peripheriebereich:	(8 Bits)
- P-Peripherie	
- Q-Peripherie	
- Koppelmerker	
- KOR	
- Kacheln	
- Dezentrale Peripherie	

Die genauen Adressen dieser Bereiche entnehmen Sie dem Speicherbelegungsplan auf der nächsten Seite.

### WICHTIG!

Der STEP5-Zugriff auf eine Speicherzelle innerhalb eines Operandenbereiches (z.B. Merker) sollte nie direkt über die absolute Adresse dieser Speicherzelle erfolgen, sondern ausschließlich relativ zur Basisadresse des jeweiligen Operandenbereichs.

Die Basisadressen aller Operandenbereiche sind im Bereich der Systemdaten (BS-Bereich) abgelegt (siehe "Systemdatenbelegung").

## 8.1 Adreßbraumaufteilung in der CPU 928

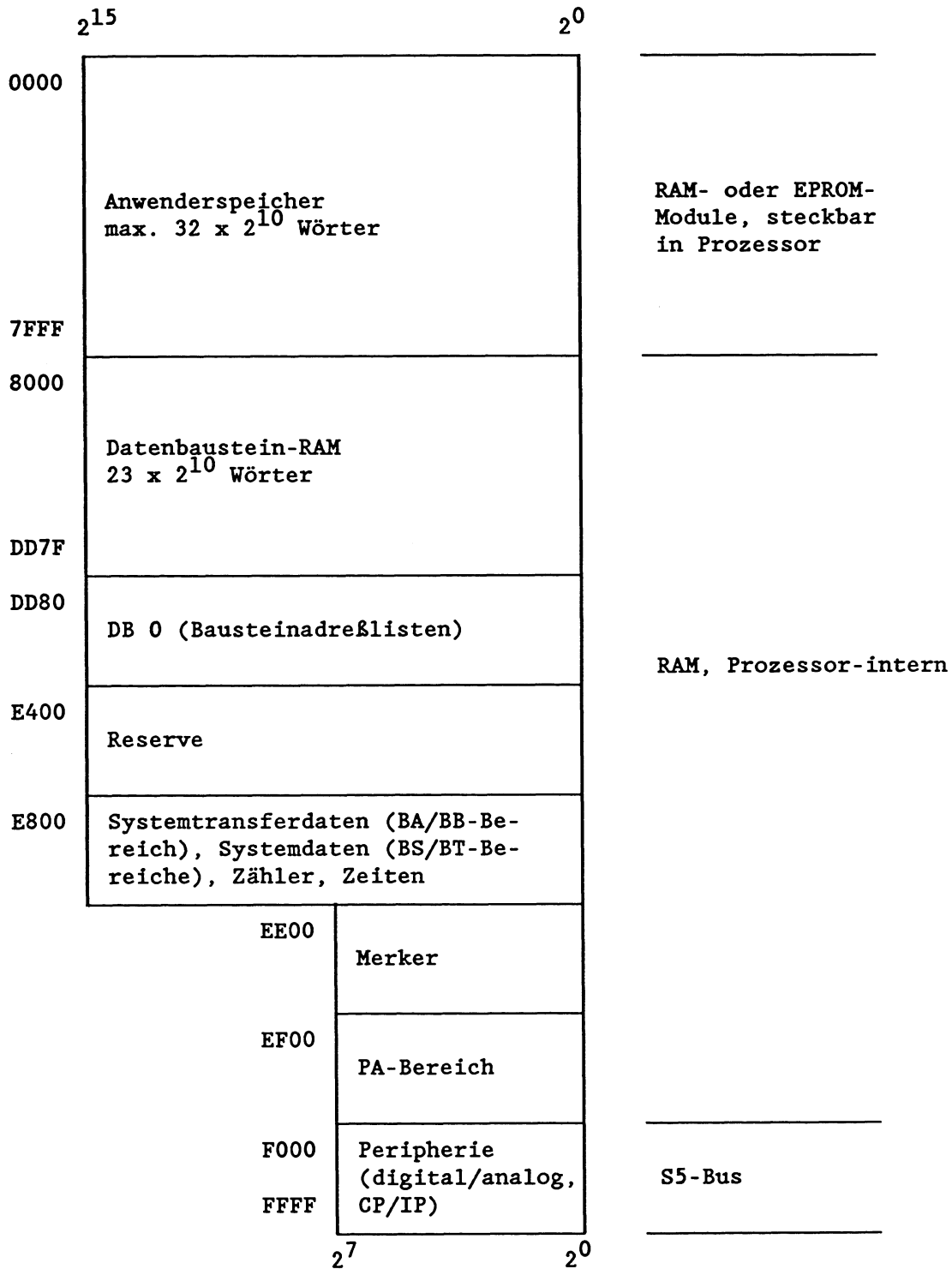


Abb. 8-1: Adreßbraumaufteilung in der CPU 928

### 8.1.1 Adreßraumaufteilung System-RAM

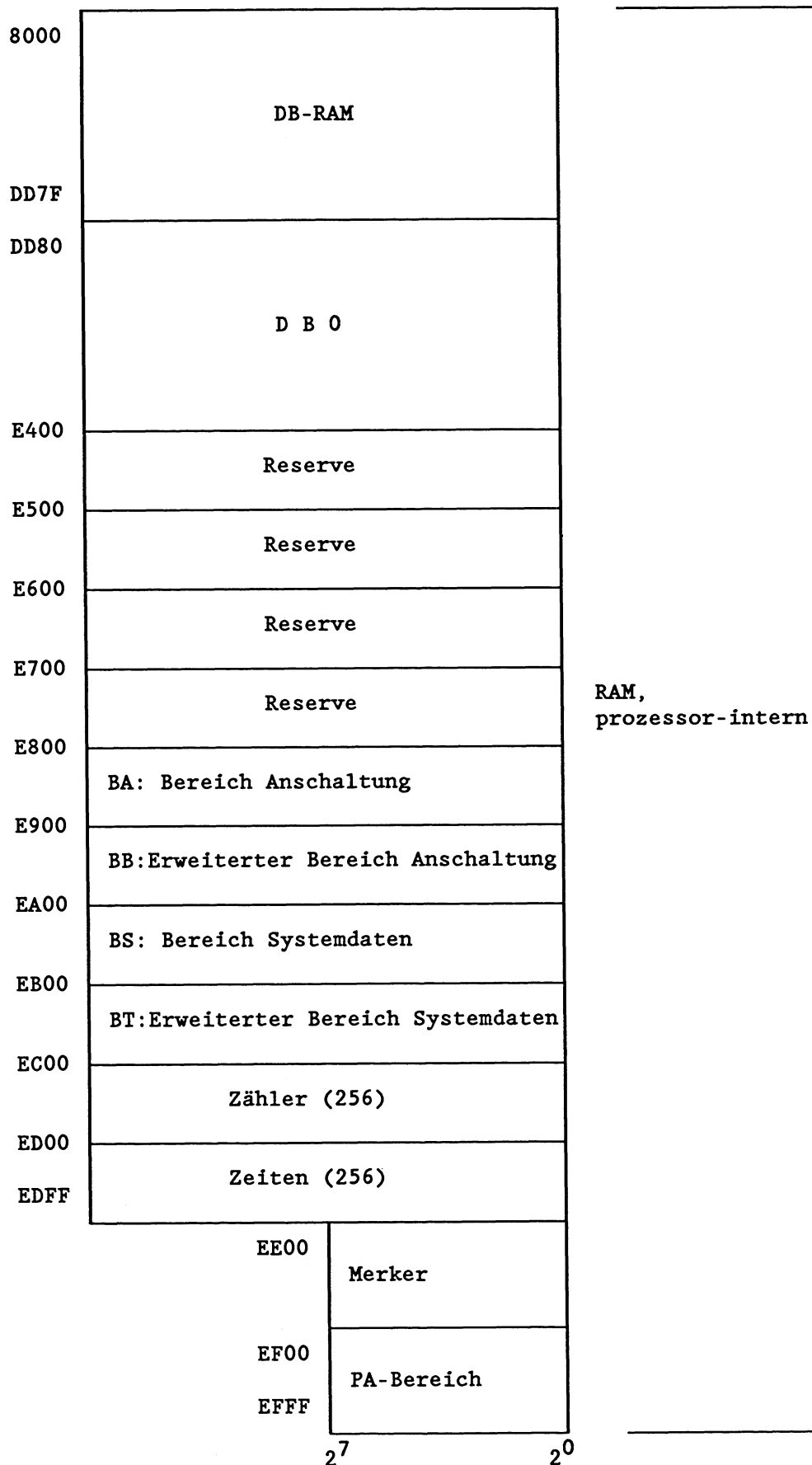


Abb. 8-2: Adreßraumaufteilung System-RAM (16 Bits)

### 8.1.2 Adreßraumaufteilung Peripherie

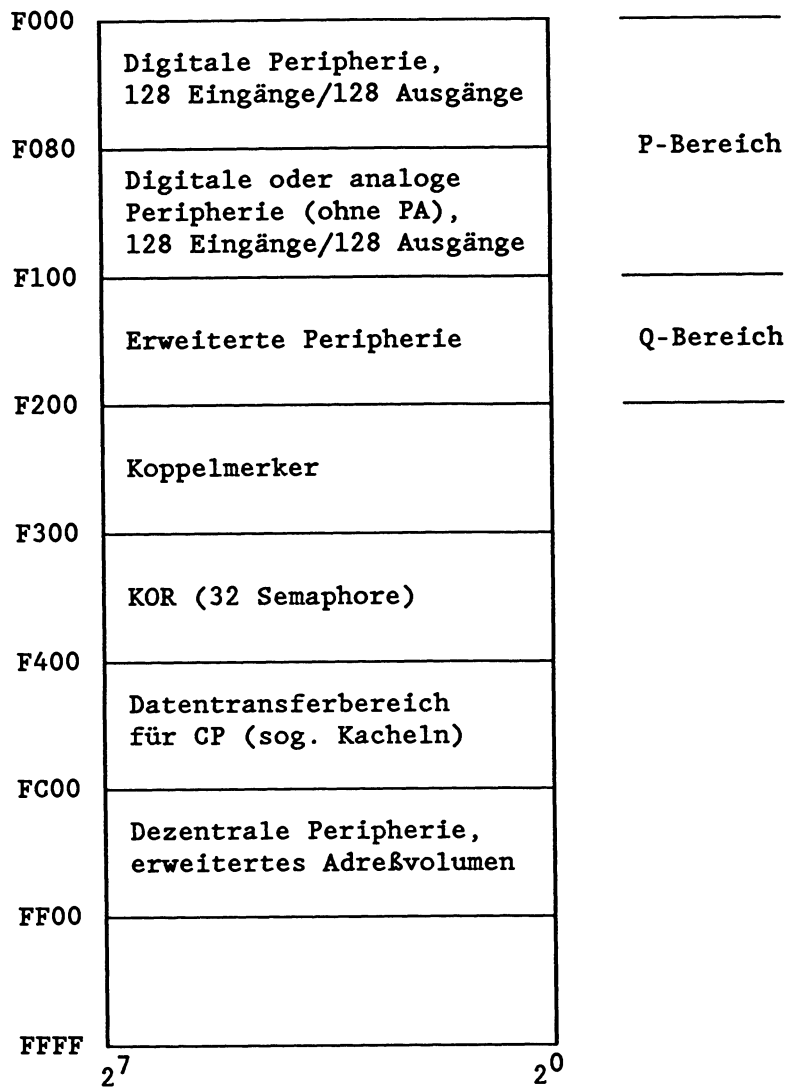


Abb. 8-3: Adreßraumaufteilung Peripherie (8 Bits)

## Adreßbereiche für Peripherie/Programmierung

Bereich (absolute Adresse)	Wird angesprochen mit	Parameter
EF00 PAA (Prozeßabbild Eingänge) EF7F	L EB/T EB L EW/T EW L ED/T ED U E/UN E/O E/ON E S E/R E/= E	0 bis 127 0 bis 126 0 bis 124 0.0 bis 127.7
EF80 PAA (Prozeßabbild Ausgänge) EFFF	L AB/T AB L AW/T AW L AD/T AD U A/UN A/O A/ON A S A/R A/= A	0 bis 127 0 bis 126 0 bis 124 0.0 bis 127.7
F000 Digitale Peripherie, Eingänge/Ausgänge F07F	L PY/T PY L PW/T PW	0 bis 127 0 bis 126
P-Peripherie mit Prozeßabbild		
F080 Digitale oder ana- loge Peripherie, Eingänge/Ausgänge F0FF	T PY/T PY T PW/T PW	128 bis 255 128 bis 254
P-Peripherie ohne Prozeßabbild		
F100 Erweiterte Peripherie, Eingänge/Ausgänge F1FF	L QB/T QB L QW/T QW	0 bis 255 0 bis 254
Q-Peripherie		

Abb. 8-4: Adreßraumaufteilung für Peripherie/Programmierung

## **Zugriffe auf die Peripherie**

Mit STEP5-Befehlen können Sie entweder direkt oder über das Prozeßabbild auf die Peripherie zugreifen. Beachten Sie dabei, daß ein Prozeßabbild nur für Ein- und Ausgabebytes der P-Peripherie mit Byteadressen von 0 bis 127 existiert!

Direkte Peripheriezugriffe: L/T PY, L/T PW, L/T QB, L/T QW

Die Ein- und Ausgänge werden zum Zeitpunkt der Befehlsbearbeitung gelesen bzw. gesetzt. Für die digitale Peripherie (0 bis 127) werden die Ausgänge im Prozeßabbild nachgeführt.

Peripheriezugriffe über das Prozeßabbild: L/T EB, L/T EW, L/T ED,  
L/T AB, L/T AW, L/T AD,  
U/UN/O/ON E, =A etc.

Zum Zeitpunkt der Befehlsbearbeitung wird nur das Prozeßabbild verändert. Erst am Ende des Zyklus wird der neue gesamte Zustand des Prozeßabbilds an die Peripherie ausgegeben.

## 8.2 Speicherorganisation in der CPU 928

Der Anwenderspeicher umfaßt den Speicherbereich von 0000H bis 7FFFH. Beim Laden der einzelnen Bausteine des Anwenderprogramms werden diese in beliebiger Reihenfolge im Speicher abgelegt (aufsteigende Adressen). Ist der Anwenderspeicher gefüllt, werden Datenbausteine im internen DB-RAM (8000H bis DD7FH) abgelegt.

Mit der On-line-Funktion "SPAUS" (Speicherausbau) erhalten Sie die Adresse (hexadezimal) der Speicherzelle, die den Bausteinde-Befehl des letzten im Speicher vorhandenen Bausteins enthält, sowie die Größe des RAM-Moduls.

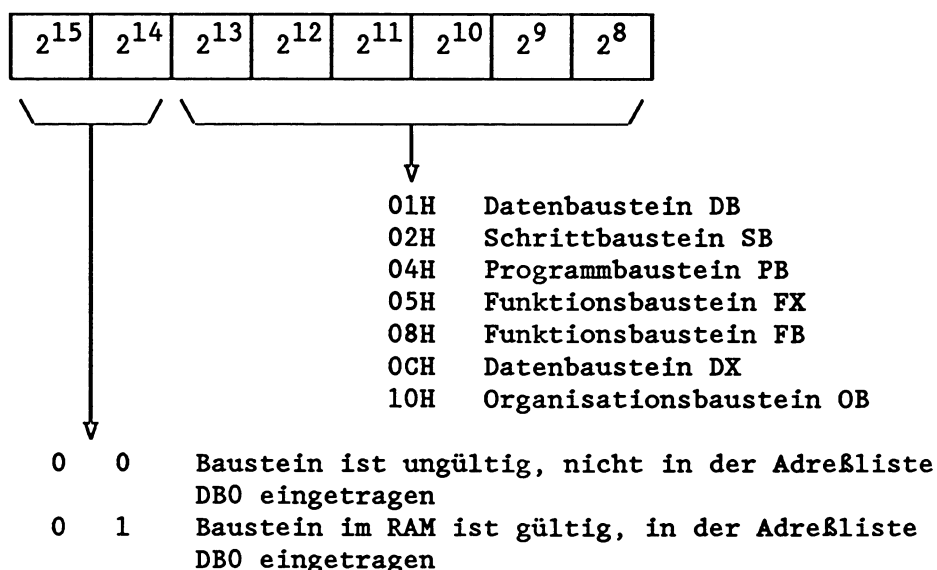
Beim Korrigieren von Bausteinen wird der 'alte' Baustein im Speicher für ungültig erklärt (d.h., die Anfangskennung wird überschrieben), und ein neuer Baustein im Speicher eingetragen. Ebenso werden beim Löschen von Bausteinen die Bausteine im Speicher nicht wirklich gelöscht, sondern nur für ungültig erklärt. Gelöschte und korrigierte Bausteine belegen also weiterhin Speicherplatz.

Die On-line-Funktion "Speicher KOMPRIMIEREN" beseitigt alle ungültigen Bausteine im Speicher und schiebt die gültigen zusammen.

### 8.2.1 Bausteinköpfe im Anwenderspeicher und DB-RAM

Jeder Baustein im Speicher beginnt mit einem 5 Wörter langen Bausteinkopf.

1. Wort: Baustein-Anfangskennung: 7070H
2. Wort: High-Byte = Baustein-Art



**Low-Byte = Baustein-Nummer**

Die Baustein-Nummer (0 bis 255) liegt im Low-Byte des 2. Kopfwortes und wird als Dualzahl codiert: 00 bis FFh.



3. Wort: Im High-Byte des 3. Wortes stehen die Kennungen für das Programmiergerät, im Low-Byte ein Teil der Bibliotheksnummer.
4. Wort: Das vierte Wort enthält den Rest der Bibliotheksnummer.
5. Wort: Im 5. Wort (Low- und High-Byte) steht die Länge des Bausteins inklusive Bausteinkopf. Die Angabe erfolgt in Wörtern.

### **8.2.2 Bausteinadreßlisten im Datenbaustein DB 0**

Der Datenbaustein DB 0 enthält die Adreßliste mit den Anfangsadressen aller Bausteine, die sich im Anwenderspeicher oder im DB-RAM des Prozessors befinden. Diese Adreßliste wird nach Netzein vom Systemprogramm erzeugt und bei der Eingabe oder Änderung von Bausteinen mit dem Programmiergerät automatisch aktualisiert.

Für jeden Bausteintyp gibt es im DB 0 eine eigens reservierte, 256 Wörter lange Adreßliste. Nicht geladene Bausteine haben die Anfangsadresse '0'.

Die Anfangsadressen der einzelnen Bausteinadreßlisten stehen außerdem in den Systemdaten BS 32 bis BS 38:

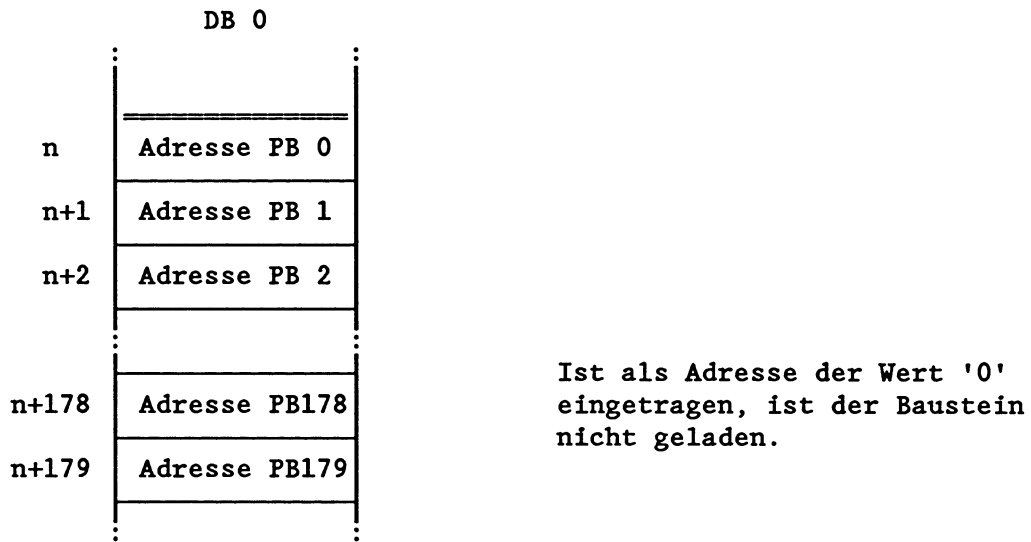
- BS 32: Anfangsadresse der DX-Adreßliste
- BS 33: Anfangsadresse der FX-Adreßliste
- BS 34: Anfangsadresse der DB-Adreßliste
- BS 35: Anfangsadresse der SB-Adreßliste
- BS 36: Anfangsadresse der PB-Adreßliste
- BS 37: Anfangsadresse der FB-Adreßliste
- BS 38: Anfangsadresse der OB-Adreßliste (nur 48 Wörter lang)

Die Anfangsadressen zeigen immer auf das erste Datenwort nach dem Bausteinkopf:

- bei Datenbausteinen jeweils auf das Datenwort DW 0!
- bei Codebausteinen jeweils auf die erste STEP5-Anweisung!  
(bei FBs auf den 'SPA'-Befehl)

### Ablage der Bausteinadressen im DB 0:

n = Anfangsadresse der PB-Adreßliste (= Inhalt von BS 36)



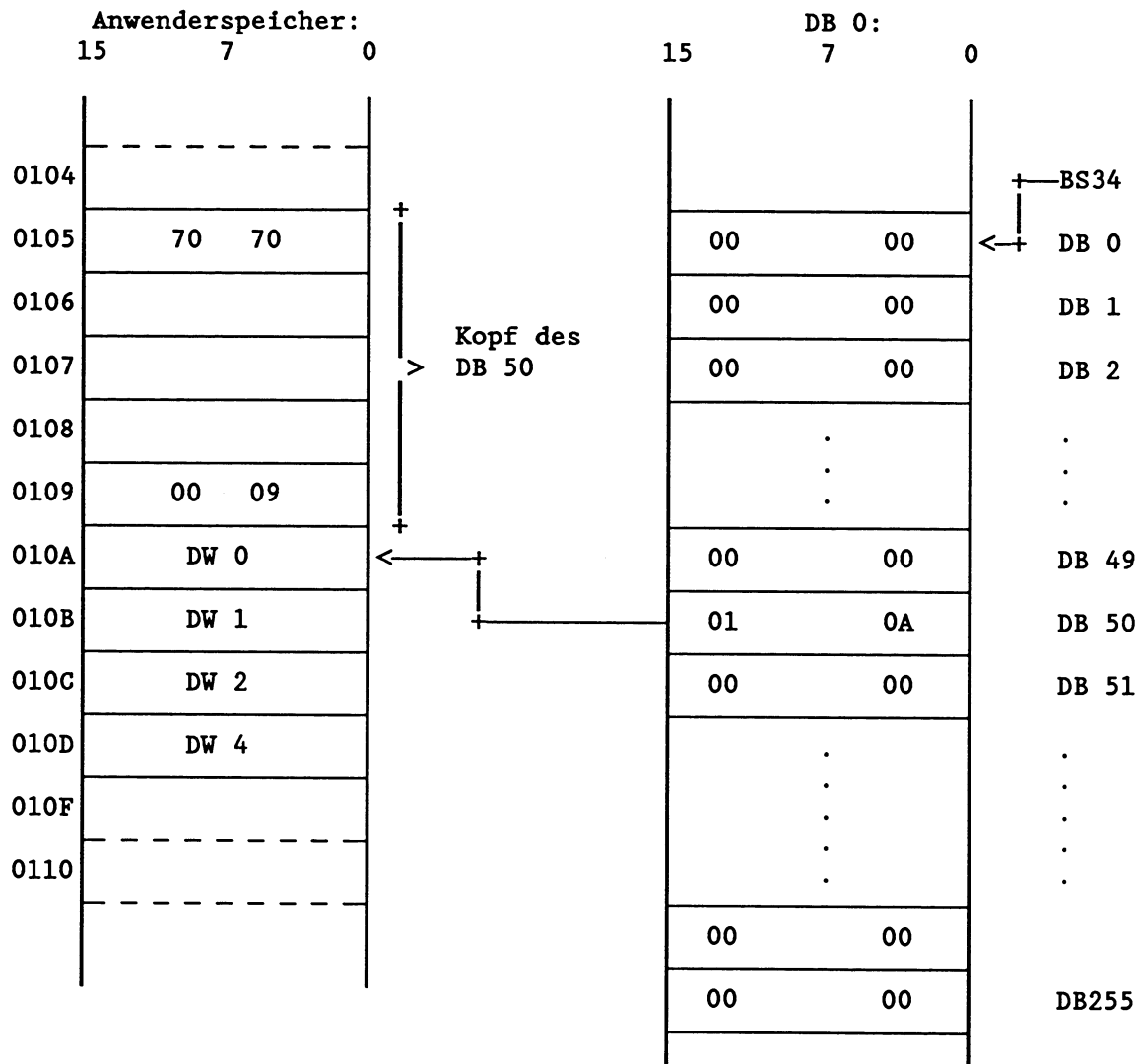
So ermitteln Sie die Adresse eines beliebigen Bausteins im Speicher:

#### Beispiel: Anfangsadresse des FB 40

```
:L BS 37      Basisadresse FB-Adressliste
:L KB 40      + FB-Nummer
:+F           = Adresse der Speicherzelle, die
:             die Anfangsadresse des FB 40
:             enthält
:LIR 1        Anfangsadresse des FB 40 in
:             den Akku 1 laden.
:             (Baustein ist nicht vorhanden,
:             falls Anfangsadr.= 0)
```

**Beispiel: Anfangsadresse und Länge des Datenbausteins DB 50**

a) über indirekten Speicherzugriff



:L BS34	Basisadresse der DB-Adressliste laden
:L KB50	Die Adresse des Eintrags für den
:+F	DB 50 errechnen und die Anfangs-
:LIR 1	adresse in den Akku 1 laden
:L KB0	Bei nicht vorhandenem Baustein
:!=F	zur Marke NIVO springen
:SPB =NIVO	
:ENT	Anfangsadresse des DB 50 in Akku 3
:TAK	und Akku 1 laden
:L KF-1	Anfangsadresse um Eins vermindern
:+F	und die Bausteinlänge in den
:LIR 1	Akku 1 laden.
.	
.	
NIVO:.....	

Ergebnis: Akku 1-L: Länge des DB 50  
Akku 2-L: Anfangsadresse des DB 50

b) mit dem Sonderfunktions-OB 181 "Datenbausteine (DB/DX) testen"

Der OB 181 enthält dieselbe Funktion wie unter a) beschrieben. Er testet jedoch zusätzlich, ob der Datenbaustein im Anwenderspeicher (RAM oder EPROM) oder im DB-RAM liegt.

```
      :L   KY1,50      Datenbaustein DB 50
      :SPA OB181      "Datenbausteine (DB/DX) testen"
      :SPB NIVO
      :SPM =PROM
      :SPZ =ANWE
      :SPP =DBRA

NIVO:      Datenbaustein nicht vorhanden
:
:BEA

PROM:      Datenbaustein liegt im
:           Anwenderspeicher (EPROM-Modul)
:BEA

ANWE:      Datenbaustein liegt im
:           Anwenderspeicher (RAM-Modul)
:BEA

DBRA:      Datenbaustein liegt im DB-RAM
:
:BE
```

Ergebnis:    Akku 1-L:    Anfangsadresse des DB 50  
                 Akku 2-L:    Länge des DB 50  
                 VKE    = 1 falls DB 50 nicht vorhanden

### 8.2.3 BA-/BB-Bereich

Der BA-Bereich ist ein 256 Wörter langer Bereich im internen System-RAM des Prozessors. Er belegt die Adressen E800H bis E8FFH.

Der BB-Bereich ist ein 256 Wörter langer Bereich im internen System-RAM des Prozessors. Er belegt die Adressen E900H bis E9FFH.

Der gesamte BA-Bereich (BA 0 bis BA 255) und der gesamte BB-Bereich (BB 0 bis BB 255) kann vom Anwender für eigene Zwecke genutzt werden.

Der BA-/BB-Bereich wird nur bei Urlöschen initialisiert!

### 8.2.4 BS-/BT-Bereich

Der BS-Bereich ist ein 256 Wörter langer Bereich im internen System-RAM des Prozessors. Er belegt die Adressen EA00H bis EAFFH.

#### **WICHTIG!**

**Es dürfen ausschließlich die Systemdatenwörter BS 1, BS 60 bis BS 63 und BS 133 beschrieben werden:**

- BS 60 bis BS 63 stehen für eigene Zwecke zur Verfügung.
- BS 1 und BS 133 haben eine festgelegte Bedeutung und beeinflussen die Programmbearbeitung. Sie dürfen nur mit *gültigen Kennungen* beschrieben werden!

#### **WICHTIG!**

**Alle übrigen Systemdaten dürfen nur gelesen werden:**

- Sie enthalten teils Informationen für den Systemprogrammierer, teils Systemvariablen, deren Bedeutung nicht veröffentlicht wird.
- Ein Beschreiben dieser Systemdaten kann Rückwirkungen auf die Funktionsfähigkeit des Automatisierungsgerätes sowie angeschlossener Programmiergeräte zur Folge haben!

Der BT-Bereich ist ein 256 Wörter langer Bereich im internen System-RAM des Prozessors. Er belegt die Adressen EB00H bis EBFFH.

Der gesamte BT-Bereich (BT 0 bis BT 255) kann vom Anwender für eigene Zwecke genutzt werden.

Der BS-/BT-Bereich wird nur bei Urlöschen initialisiert!

# Systemdatenbelegung des BS-Bereichs

BS	Bezeichnung		Adr.
0	Unterbrechungsanzeigenwort (STOER.URS.)		EA00
1	<b>Unterbrechungsanzeigenloeschwort (UALW)</b>		<b>EA01</b>
2	Unterbrechungsanzeigensammelwort (UAMK)		EA02
3	Anlauf-Fehlerkennung-Anzeige		EA03
4			EA04
5	Stopp-Kennungen	Anlauf-Kennungen	EA05
6	Zyklus-Kennungen	Modul-Kennungen	EA06
7	Urloesch-Kennungen	Fehlerkennungen (H)	EA07
8	Fehlerkennungen (M)	Fehlerkennungen (L)	EA08
9	XXXXXXXXXXXXXXXXXXXXX	Akt. Ident.-Nr.	EA09
10	Basisadresse der Eingangssignalformer		EA0A
11	Basisadresse der Ausgangssignalformer		EA0B
12	Basisadresse Prozessabbild d. Eingaenge		EA0C
13	Basisadresse Prozessabbild d. Ausgaenge		EA0D
14	Basisadresse Merkerbereich		EA0E
15	Basisadresse Zeitbereich		EA0F
16	Basisadresse Zaehlerbereich		EA10
17	Basisadresse Bereich Anschaltung		EA11
18	XXXXXXXXXXXXXXXXXXXXX	AG-SW-Stand	EA12
19	Endadresse des Anwendermodulspeichers		EA13
20	Basisadresse Bereich System		EA14
21	Laenge der DB-Adressliste		EA15
22	Laenge der SB-Adressliste		EA16
23	Laenge der PB-Adressliste		EA17

XXXXXXXXXX: Belegung nicht freigegeben

24	Laenge der FB-Adressliste	EA18
25	Laenge der OB-Adressliste	EA19
26	Laenge der FX-Adressliste	EA1A
27	Laenge der DX-Adressliste	EA1B
28	Länge des Adreßlisten-DBs (DB 0)	EA1C
29	Steckplatzkennung	CPU-Kennung 2 (Typ) EA1D
30	Laenge der Bausteinkopf-Information	EA1E
31	CPU-Kennung 1	SW-Stand PG-Ansch. EA1F
32	Basisadresse DX-Adressliste	EA20
33	Basisadresse FX-Adressliste	EA21
34	Basisadresse DB-Adressliste	EA22
35	Basisadresse SB-Adressliste	EA23
36	Basisadresse PB-Adressliste	EA24
37	Basisadresse FB-Adressliste	EA25
38	Basisadresse OB-Adressliste	EA26
39	XX	EA27
:	XX	:
:	XX	:
:	XX	:
:	XX	:
:	XX	:
:	XX	:
54	XX	EA36
55	Zaehler für 1 Std. (bis 3599sec, hex.)	EA37
56	reserviert für Hantierungsbausteine	EA38
59		EA3B
60	reserviert für Anwenderzwecke	EA3C
63		EA3F
64	reserviert fürs Systemprogramm	EA40
127		EA7F
128	XX	EA80
129	XX	EA81

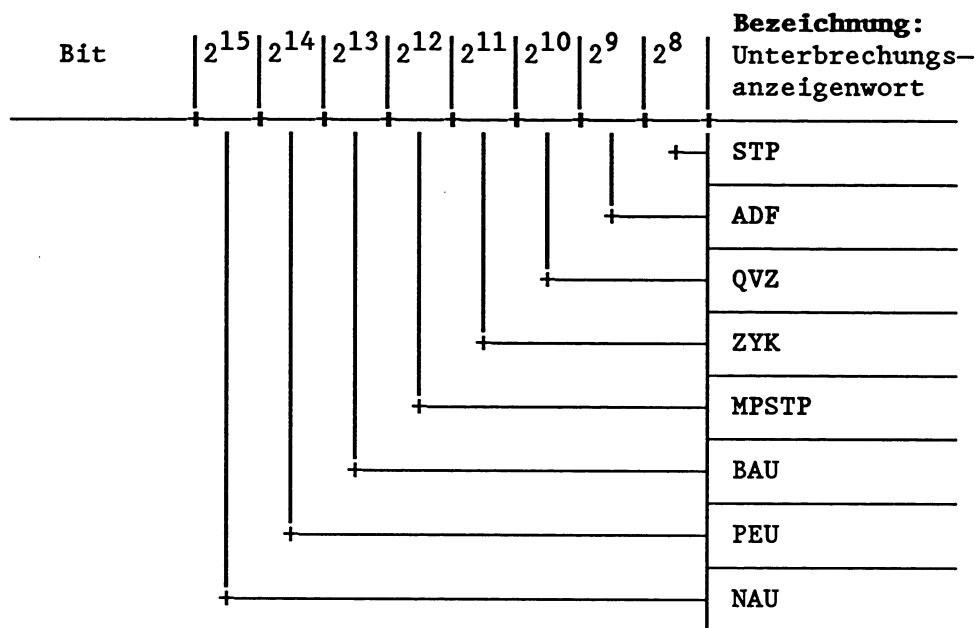
130	Kennung 'Regelung'	EA82
131	Anzeigenwort 'Alarmer gemeinsam sperren'	EA83
132	Anzeigenwort 'Alarmer gemeinsam verzögern'	EA84
133	Kennung 'Prozessabbildaktualisierung'	EA85
134	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	EA86
135	Anzeigenwort 'Weckalarmer einzeln sperren'	EA87
136	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	EA88
137	Anzeigenwort 'Weckalarmer einz. verzögern'	EA89
138	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	EA8A
:	:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	:
:	:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	:
:	:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	:
:	:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	:
:	:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:	:
255	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	EAFF

Die Informationen einiger Systemdaten (über den internen Aufbau des Prozessors, den Ausgabestand der Software, die CPU-Kennung etc.) erhalten Sie außerdem über die On-line-Funktion 'SYSTEM-PARAMETER'.

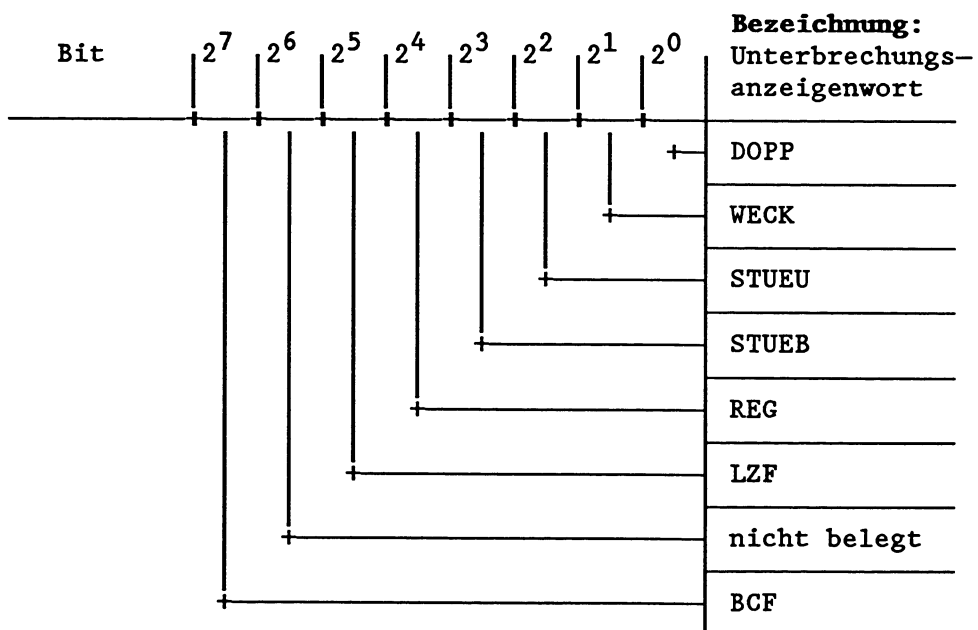
Ergänzend zu obiger Liste ist im folgenden die Bit-Belegung einiger Systemdaten angegeben, die der Anwender über STEP5-Befehle oder mit dem PG auswerten kann (Abkürzungen siehe Kapitel 5.3).



Systemdatum: BS 0      Adresse: EA00 (HIGH)

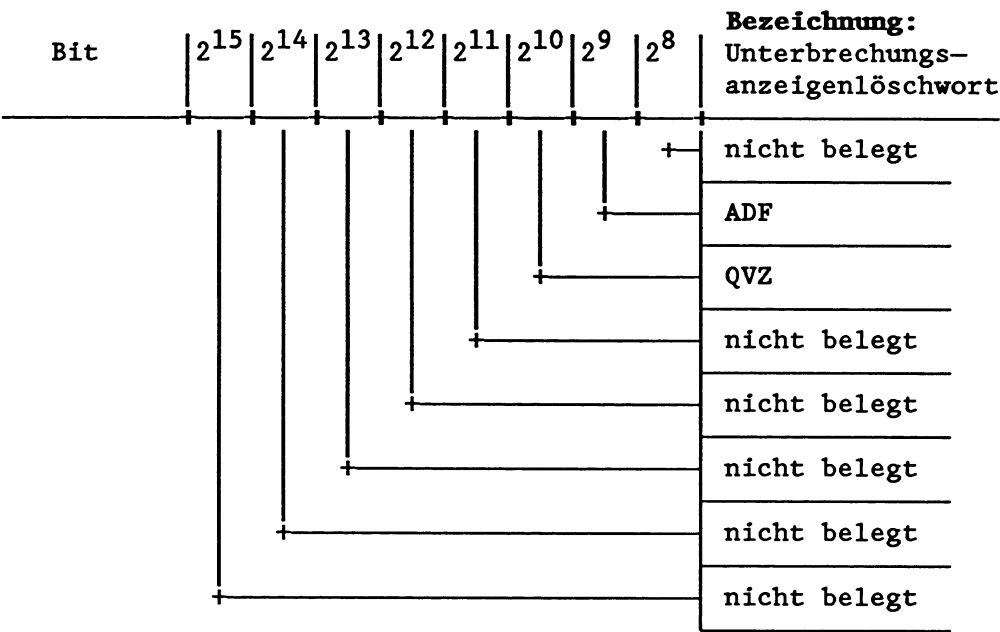


Systemdatum: BS 0      Adresse: EA00 (LOW)

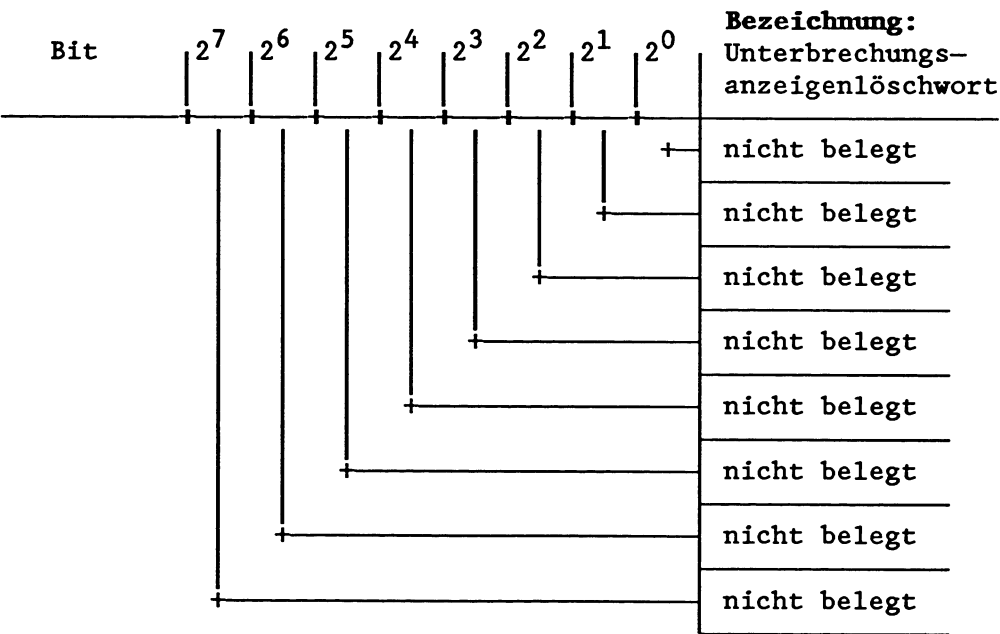


Das Systemdatum BS0 entspricht der "STOERUNGSURSACHE" im USTACK. Tritt bei der Programmbearbeitung z.B ein Laufzeitfehler auf, wird das Bit 2<sup>5</sup> gesetzt. Ist die Programmbearbeitungsebene LZF vollständig bearbeitet, wird das Bit 2<sup>5</sup> rückgesetzt.

Systemdatum: BS 1      Adresse: EA01 (HIGH)



Systemdatum: BS 1      Adresse: EA01 (LOW)



BS 1: Aktive Schnittstelle, für Anwender freigegeben (siehe folgende Seite)!

## Unterbrechungsanzeigen-Löschwort (Systemdatum BS 1):

Durch Setzen von Bit 9 bzw. Bit 10 des UALW erreichen Sie, daß der nächstfolgende ADF bzw. QVZ ignoriert wird und die laufende Programmbearbeitung dadurch nicht beeinflußt wird. Nach Auftreten eines QVZ bzw. ADF setzt das Systemprogramm das betreffende Bit zurück.

Jede Programmbearbeitungsebene hat ihr eigenes UALW!

Im folgenden Beispiel wird getestet, ob unter einer bestimmten Peripheriadresse eine Baugruppe ansprechbar ist. Ist die Baugruppe nicht vorhanden, wird mit Hilfe des UALWs ein Quittungsverzug verhindert und ein für diesen Fall vorgesehenes Programm bearbeitet. Ebenso wird getestet, ob eine bestimmte Peripherieadresse im DB 1 eingetragen ist. Falls nicht, wird mit Hilfe des UALWs ein Adressierfehler verhindert und ein spezielles Programm bearbeitet.

FB 10

NAME:PERITEST

BEZ :PADR E/A/D/B/T/Z: E BI/BY/W/D: BY

BEZ :MASK E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KM

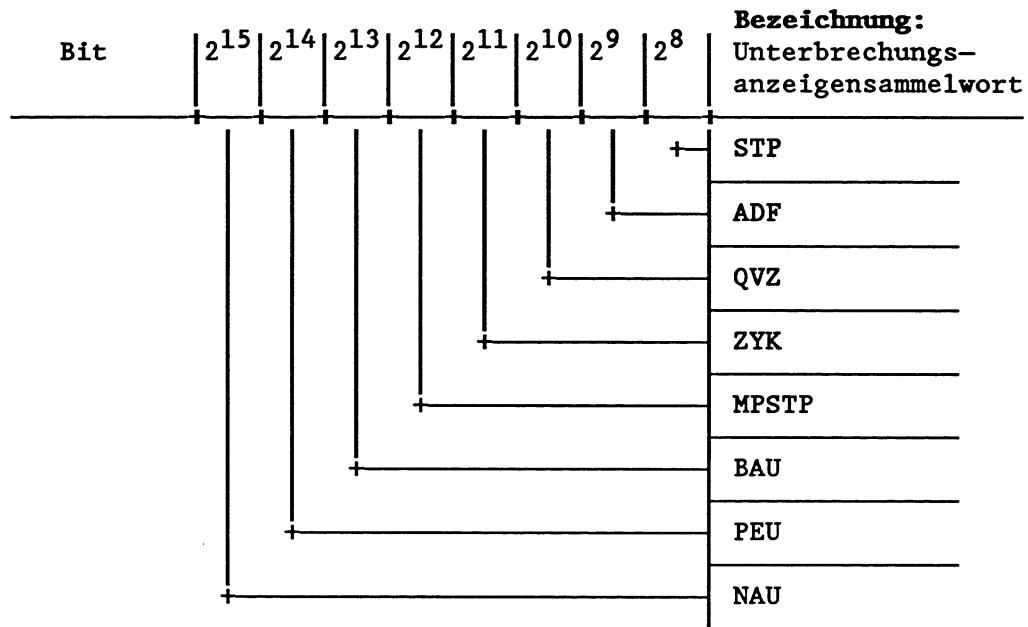
:L BS1 UALW laden  
:T BS60 und sichern  
:LW =MASK QVZ- bzw. ADF-Bit setzen  
:OW  
:T BS1 UALW zurückschreiben  
:L =PADR Einzel-Peripheriezugriff bzw. Zugriff auf  
:L BS1 das Prozeßabbild  
:LW =MASK QVZ- bzw. ADF-Bit maskieren  
:UW  
:L BS60 Altes UALW zurückschreiben, damit der nächste  
:T BS1 QVZ bzw. ADF erkannt wird  
:TAK  
:BE

FBO

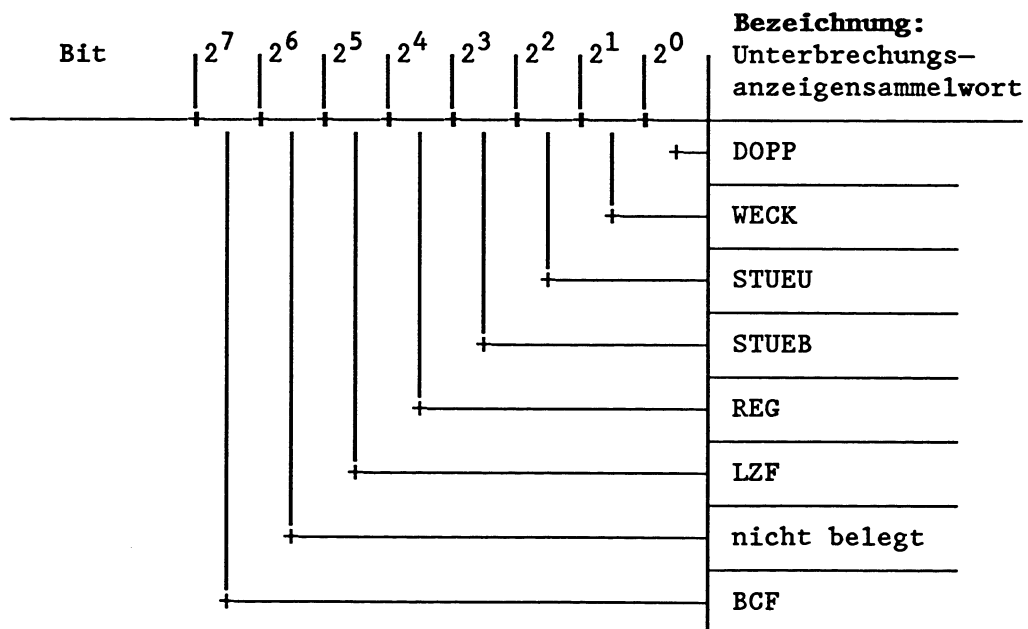
NAME:L

:SPA FB10  
NAME:PERITEST Testen, ob  
PADR: PB128 unter der Peripherieadr. 128 eine  
MASK: KM00000100 00000000 Baugruppe ansprechbar ist  
:SPN =M001  
:... Dieser Programmteil wird bearbeitet,  
:... falls Baugruppe nicht ansprechbar  
:... ist  
M001:  
:SPA FB10  
NAME:PERITEST Testen, ob  
PADR: AB4 im DB 1 eine Baugruppe mit der  
MASK: KM00000010 00000000 Peripherieadr. 4 eingetragen ist  
:SPN =M002  
:... Dieser Programmteil wird bearbeitet,  
:... falls Peripherieadresse nicht  
:... eingetragen ist  
M002:  
:BE

Systemdatum: **BS 2**      Adresse: **EA02 (HIGH)**



Systemdatum: **BS2**      Adresse: **EA02 (LOW)**



Das Unterbrechungsanzeigen-Sammelwort (UAMK im USTACK, siehe folgende Seite) darf nur gelesen werden!

### **Unterbrechungsanzeigen-Sammelwort (Systemdatum BS 2):**

Die 16 Bits des Unterbrechungsanzeigen-Sammelworts entsprechen den unter "STOERUNGSURSACHE" im USTACK aufgeführten möglichen Fehlerursachen.

Bei Auftreten eines bestimmten Fehlers wird das dazugehörige Bit gesetzt.

#### **Beispiel:**

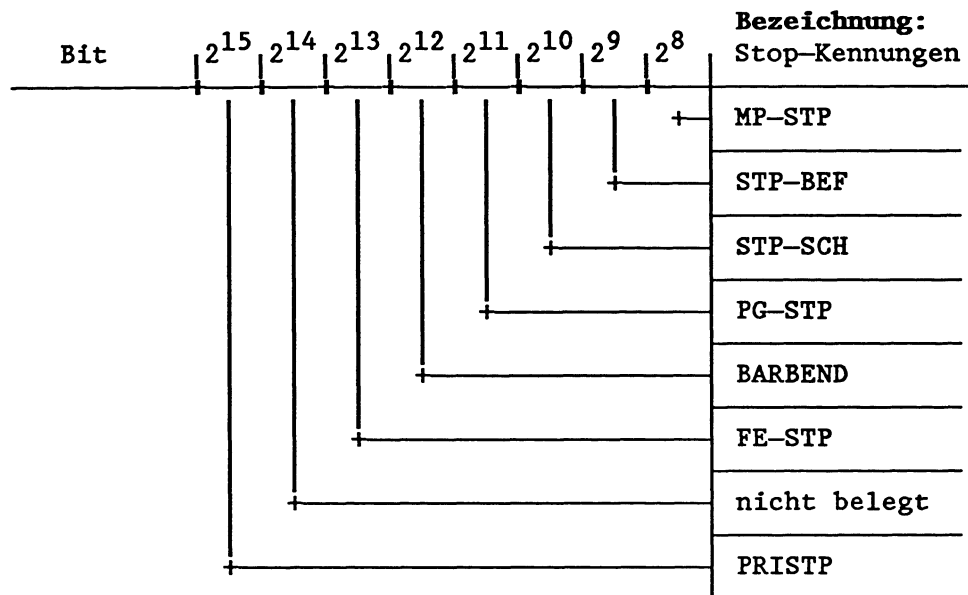
Geht der Prozessor aufgrund eines Adressierfehlers (ADF) in Stop, so wird im UAMK Bit 9 gesetzt. Wenn nun bei der Bearbeitung des ADF ein Befehlscodefehler (BCF) auftritt, wird im UAMK außerdem Bit 7 gesetzt.

Inhalt des UAMK (binär):	00000010 10000000
Darstellung (hexadezimal) im USTACK:	0280

Während im USTACK unter STOERUNGSURSACHE nur der jeweils zuletzt aufgetretene Fehler angekreuzt ist, sind im UAMK alle bis dahin aufgetretenen Fehler aufsummiert (USTACK Tiefe 05: im UAMK sind 5 Bits gesetzt). Durch Umwandlung des Hexadezimalcodes in den Binärcode läßt sich der Inhalt des UAMK auswerten. Auf diese Weise können Sie feststellen, welche Fehler den Stoppzustand verursacht haben.

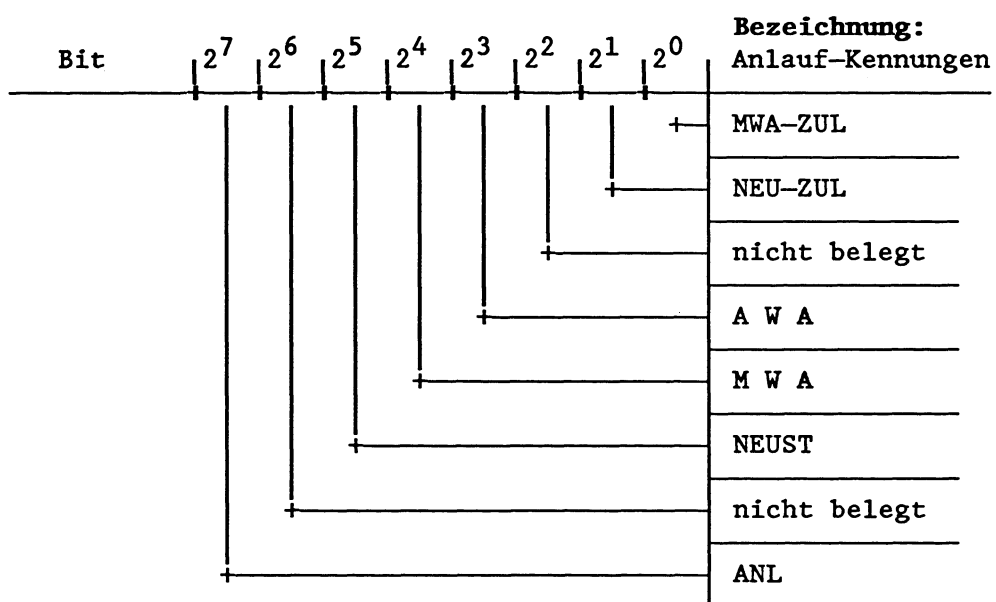
Die Fehler-Bits werden rückgesetzt, sobald die entsprechende Fehler-Programmbearbeitungsebene vollständig bearbeitet ist und damit verlassen wird.

Systemdatum: BS 5      Adresse: EA05 (HIGH)



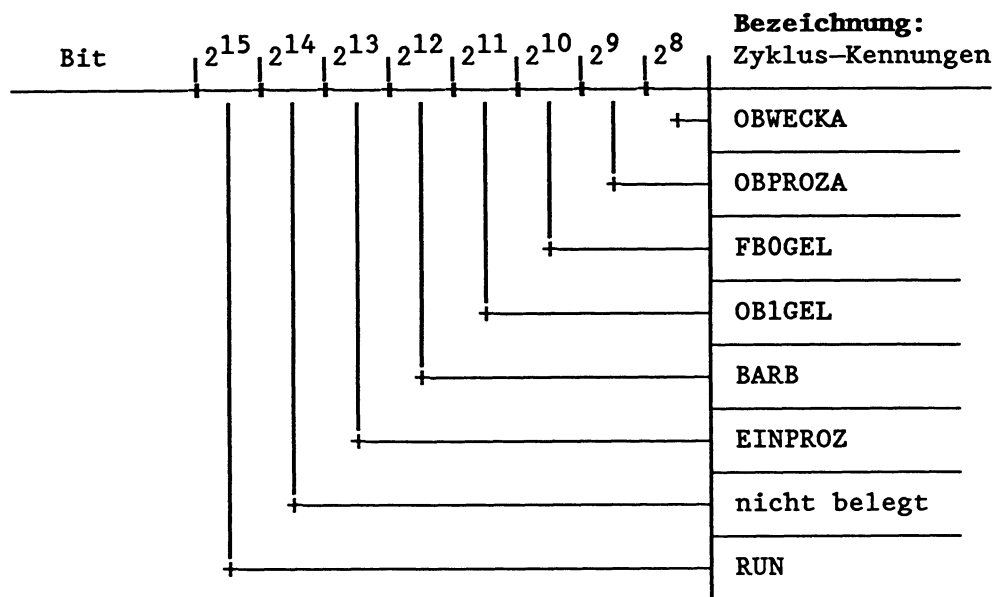
---> 1. Zeile Steuerbits

Systemdatum: BS 5      Adresse: EA05 (LOW)



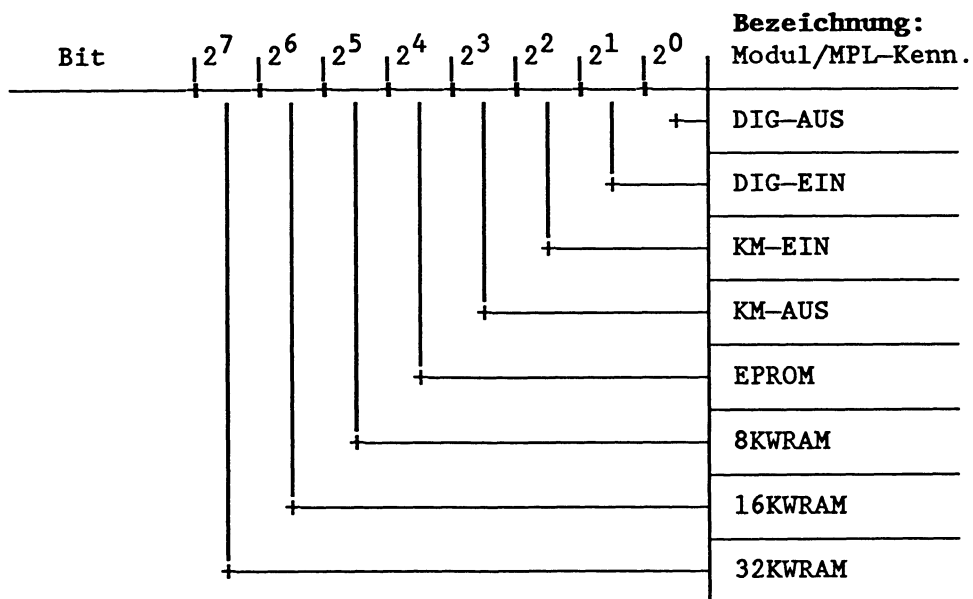
---> 2. Zeile Steuerbits

Systemdatum: **BS 6**      Adresse: EA06 (HIGH)



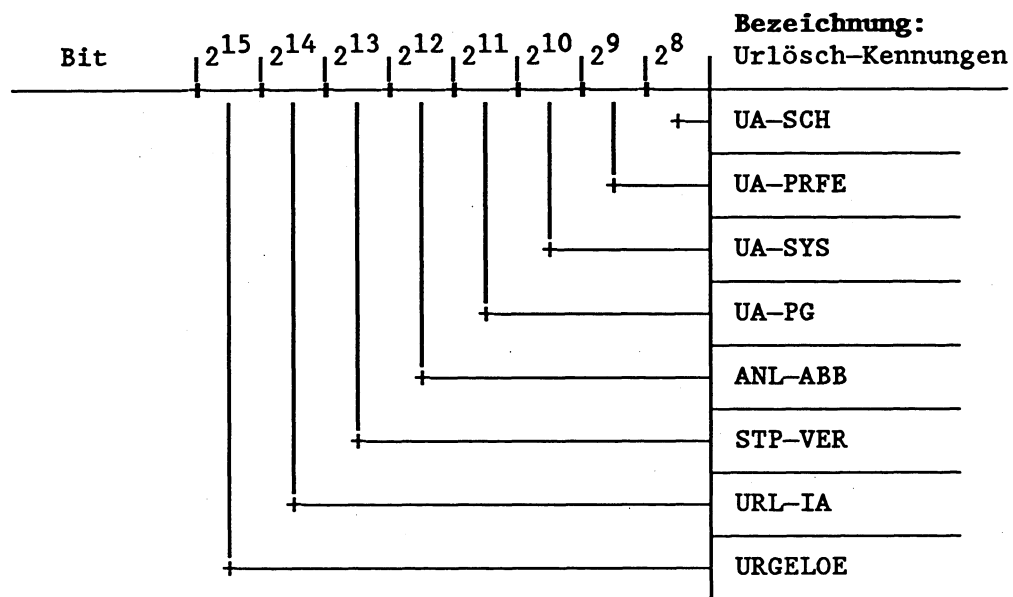
---> 3. Zeile Steuerbits

Systemdatum: **BS 6**      Adresse: EA06 (LOW)



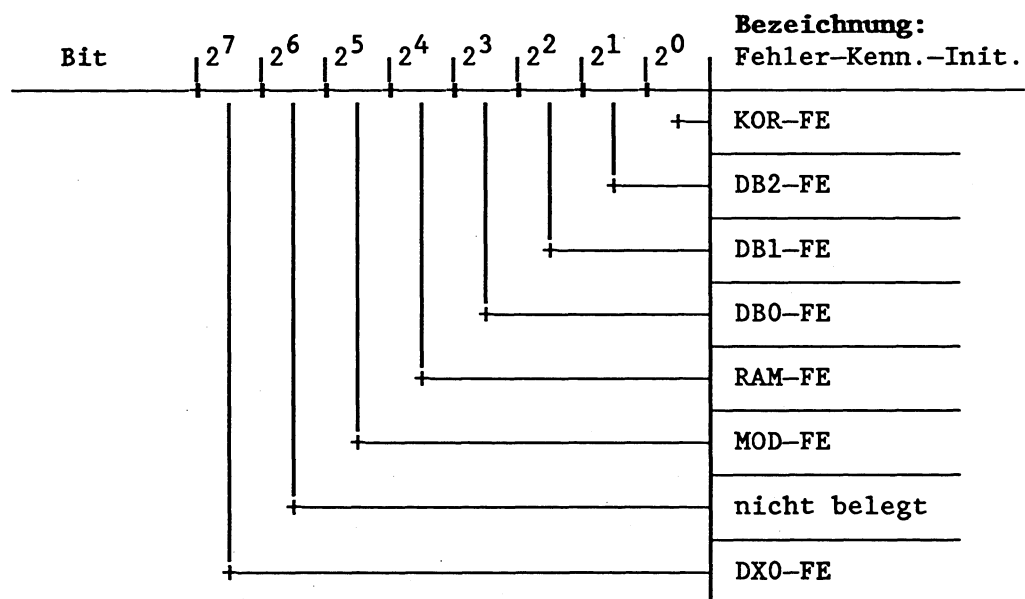
---> 4. Zeile Steuerbits

Systemdatum: BS 7 Adresse: EA07 (HIGH)



---> 5. Zeile Steuerbits

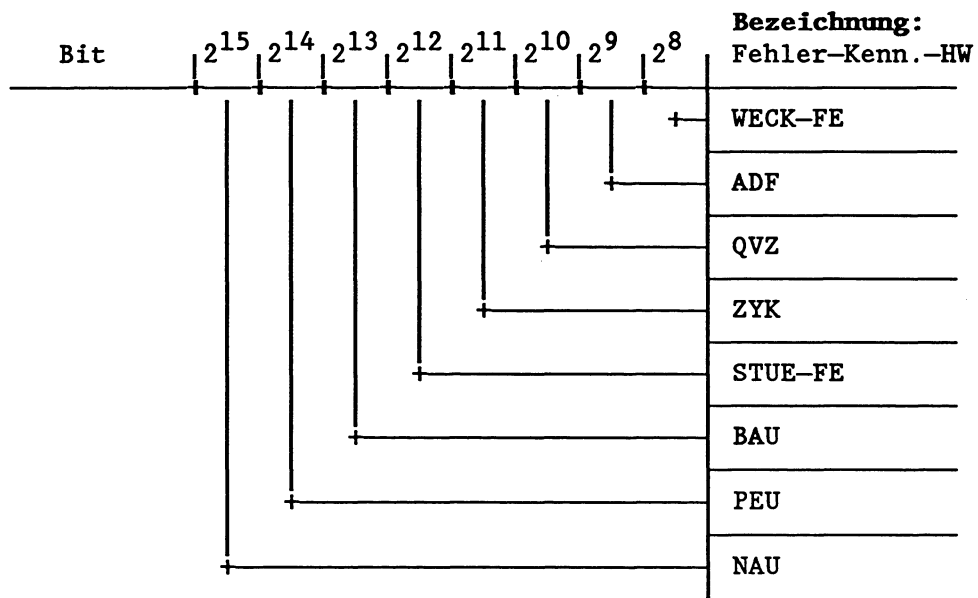
Systemdatum: BS 7 Adresse: EA07 (LOW)



---> 6. Zeile Steuerbits

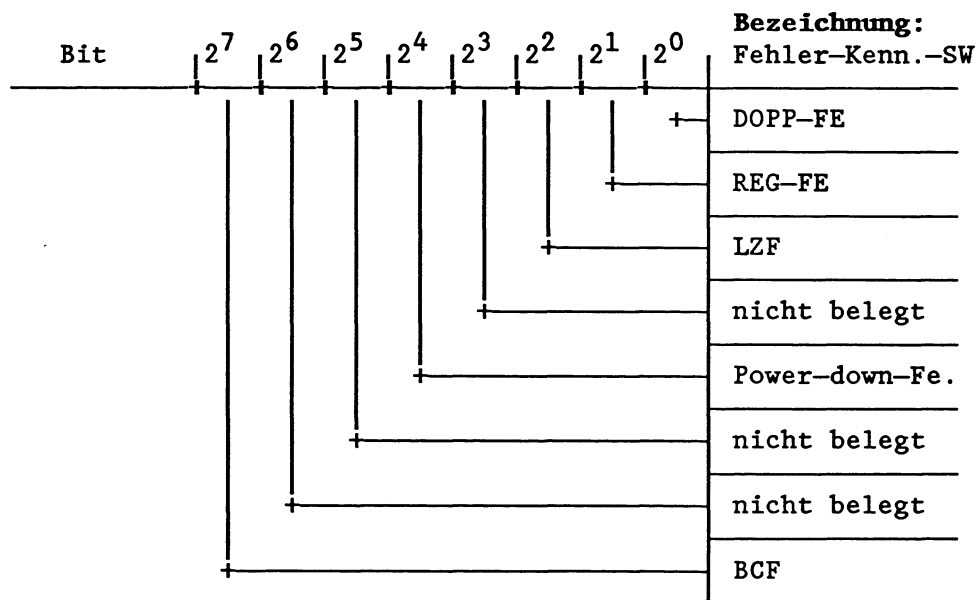


Systemdatum: BS 8 Adresse: EA08



---> 7. Zeile Steuerbits

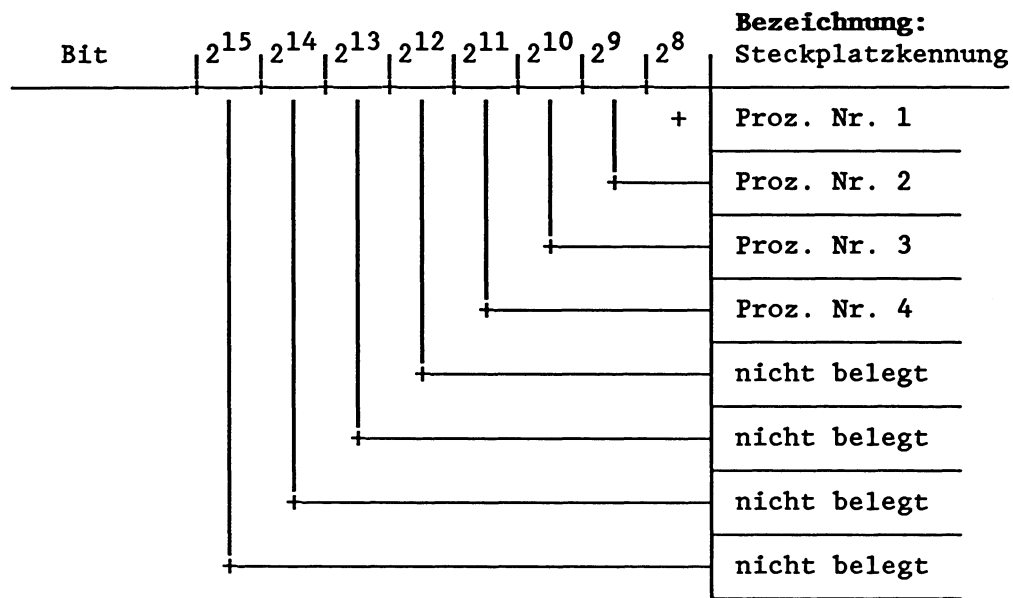
Systemdatum: BS 8 Adresse: EA08 (LOW)



---> 8. Zeile Steuerbits

Systemdatum: **BS 29**

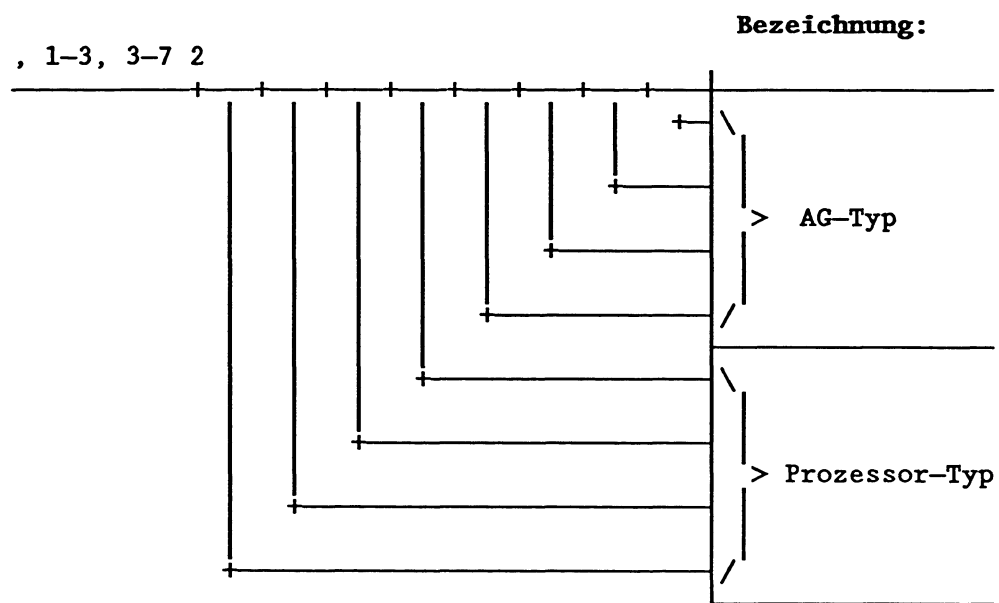
Adresse: **EA1D (HIGH)**



**BS 29 (HIGH):** Aktive Schnittstelle, wird von den Hantierungsbausteinen und bei der Mehrprozessorkommunikation sowie vom OB 218 und den Befehlen SES und SEF benutzt.

Systemdatum: BS 29

Adresse: EA1D (LOW)



AG-Typ:

0 1 1 1 AG S5-135U

Prozessor-Typ: 0 0 1 1 CPU 928

Systemdatum: **BS 130**      Adresse: **EA82 (LOW)**

Das Systemdatum BS 130 hat eine rein anzeigende Funktion:

Bit  $2^0 = 0$  :    Programmbearbeitungsebene "Regelung" aktiviert  
Bit  $2^0 = 1$  :    Programmbearbeitungsebene "Regelung" unterdrückt

Vor Aufruf eines Anlauf-Organisationsbausteins (OB 20, 21 oder 22) wertet das Systemprogramm den Datenbaustein DB 2 aus (falls vorhanden). Je nach Ergebnis der Auswertung wird das BS 130 vom Systemprogramm gesetzt bzw. rückgesetzt. Danach ruft das Systemprogramm einen Anlauf-OB auf.

Ist das BS 130 (LOW) rückgesetzt, wird die Reglerbearbeitung im zyklischen Betrieb entsprechend der Reglerliste im DB 2 durchgeführt.

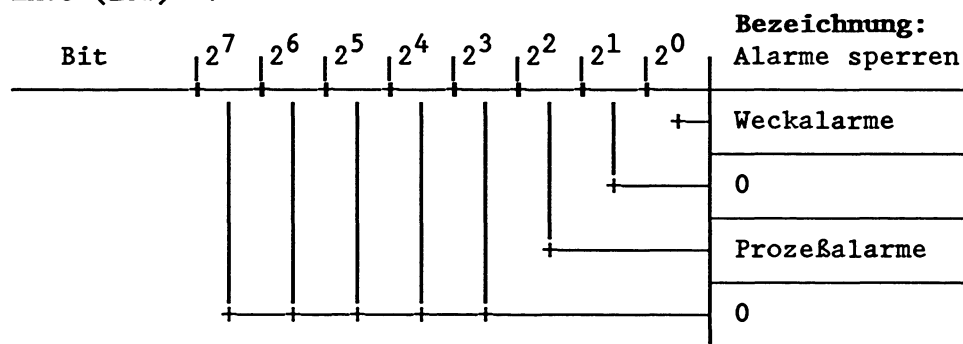
Systemdatum: **BS 131**      Adresse: **EA83**

Das Systemdatum BS 131 hat eine rein anzeigende Funktion.

**Anzeigenwort "Alarmer gemeinsam sperren"** : siehe Kapitel 6.8.1 (OB 120)

EA83 (HIGH) = 0

EA83 (LOW) :



Bit = 1 bedeutet: Diese Alarmer sind gesperrt!

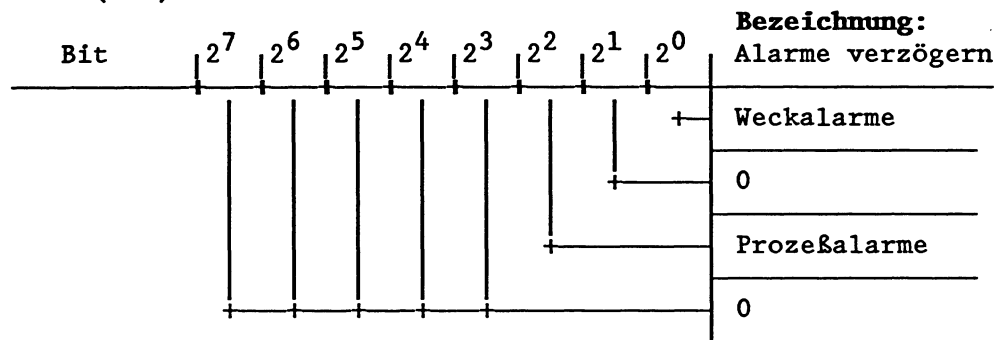
Systemdatum: BS 132      Adresse: EA84

Das Systemdatum BS 132 hat eine rein anzeigende Funktion.

**Anzeigenwort "Alarmer gemeinsam verzögern":** siehe Kapitel 6.8.1 (OB 122)

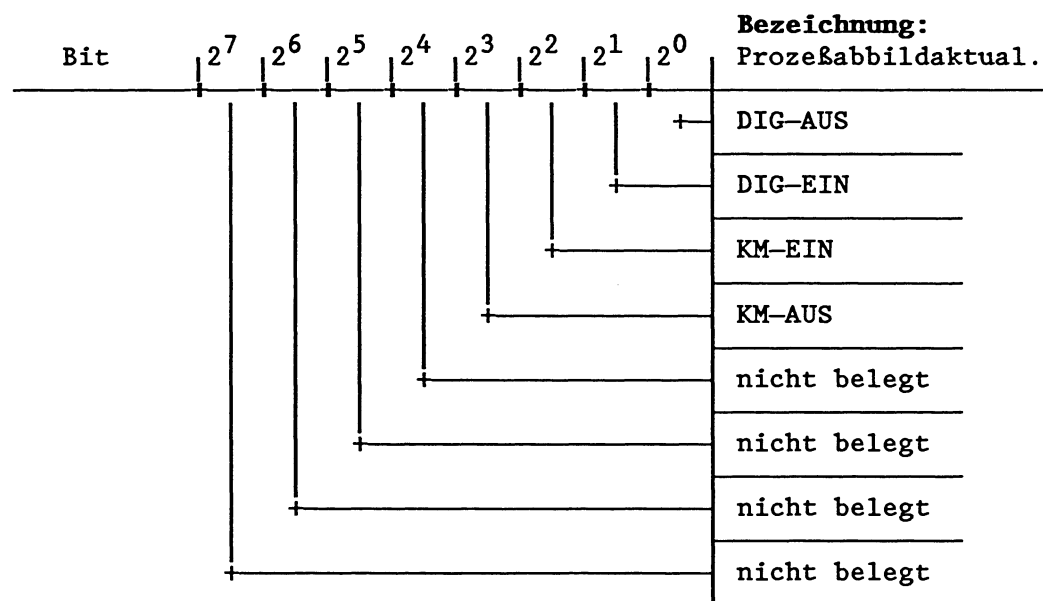
EA84 (HIGH) = 0

EA84 (LOW) :



Bit = 1 bedeutet: Diese Alarmer sind verzögert!

Systemdatum: BS 133      Adresse: EA85 (LOW)



Bit 2<sup>0</sup> = 0 : nächstes Prozeßabbild der digitalen Ausgänge wird ausgegeben

Bit 2<sup>0</sup> = 1 : nächste Prozeßabbildaktualisierung der digitalen Ausgänge wird unterdrückt

Bit  $2^1 = 0$  : nächstes Prozeßabbild der digitalen Eingänge wird  
eingelesen  
Bit  $2^1 = 1$  : nächste Prozeßabbildaktualisierung der digitalen  
Eingänge wird unterdrückt

Bit  $2^2 = 0$  : nächstes Prozeßabbild der Koppelmerkereingänge  
wird eingelesen  
Bit  $2^2 = 1$  : nächste Prozeßabbildaktualisierung der Koppelmer-  
kereingänge wird unterdrückt

Bit  $2^3 = 0$  : nächstes Prozeßabbild der Koppelmerkerausgänge  
wird ausgegeben  
Bit  $2^3 = 1$  : nächste Prozeßabbildaktualisierung der Koppelmer-  
kerausgänge wird unterdrückt

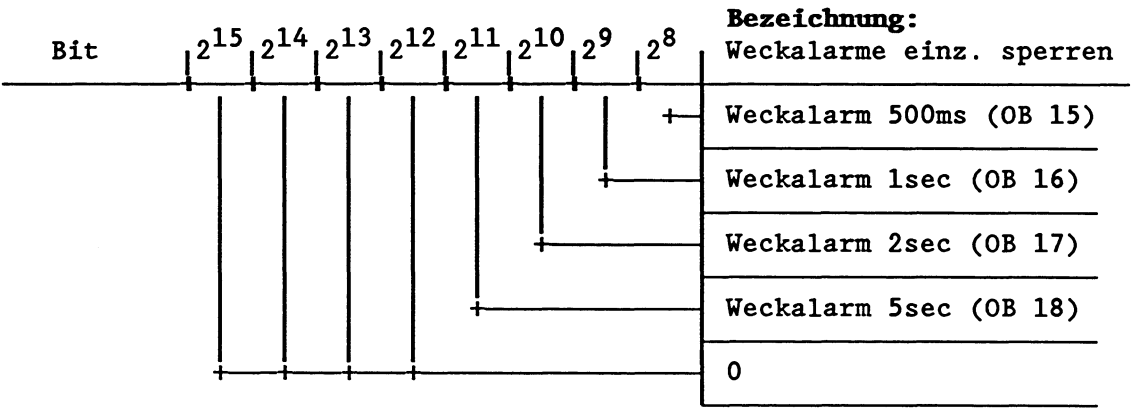
Hinweis: Jedes Bit verhindert, falls es gesetzt ist, die Prozeß-  
abbildaktualisierung einmal, anschließend wird es vom  
Systemprogramm sofort wieder auf "0" gesetzt.

Systemdatum: **BS 135**      Adresse: **EA87**

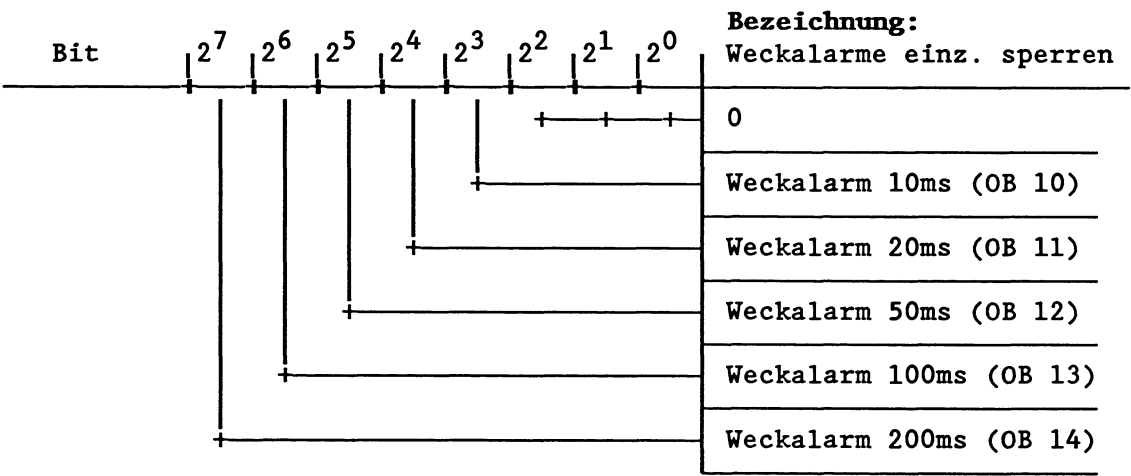
Das Systemdatum BS 135 hat eine rein anzeigende Funktion.

**Anzeigewort "Weckalarme einzeln sperren"**: siehe Kapitel 6.8.2 (OB 121)

EA87 (HIGH) :



EA87 (LOW) :



Bit = 1 bedeutet: Dieser Weckalarm ist gesperrt!

Systemdatum: **BS 137**      Adresse: **EA89**

Das Systemdatum BS 137 hat eine rein anzeigende Funktion.

**Anzeigewort "Weckalarme einzeln verzögern"**: siehe Kapitel 6.8.2 (OB 123)

Das Systemdatum BS 137 ist gleich aufgebaut wie BS 135. Wenn ein Bit auf "1" gesetzt ist, wird die Bearbeitung des entsprechenden Weckalarms verzögert. Ist das Bit auf "0" gesetzt, wird der betreffende Weckalarm bearbeitet.

## 9 Speicherzugriffe über absolute Adressen

Die Programmiersprache STEP 5 enthält Operationen, mit denen Zugriffe auf den gesamten Adreßraum möglich sind.

### **WICHTIG!**

Bei einer nicht sachgerechten Anwendung dieser Befehle können STEP5-Bausteine und Systemdaten überschrieben werden. Dies kann unerwünschte Betriebszustände zur Folge haben. Operationen, die mit absoluten Adressen arbeiten, sollten deshalb nur von Anwendern mit sehr guten Systemkenntnissen benutzt werden.

### **Lokaler Speicher**

Als lokaler Speicher wird ein Speicherbereich bezeichnet, der auf jedem Prozessor vorhanden ist (Anwender-Modul, DB-RAM, BA-, BB-, BS-, BT-Bereich, Zähler, Zeiten, Merker, Prozeßabbild).

### **Globaler Speicher**

Ein globaler Speicher ist nur einmal für alle Prozessoren vorhanden und wird über den S5-Bus adressiert.

### **Speicherorganisation**

Speicherbereiche sind byteweise oder wortweise organisiert.

Byteweise Organisation: Jede Adresse adressiert ein Byte.

Wortweise Organisation: Jede Adresse adressiert ein Wort.

Die Organisation des lokalen Speichers ist vorgegeben (siehe Kapitel 8 "Speicherbelegung und Speicherorganisation"). Die Organisation des globalen Speichers hängt vom Typ der gesteckten Baugruppen ab.



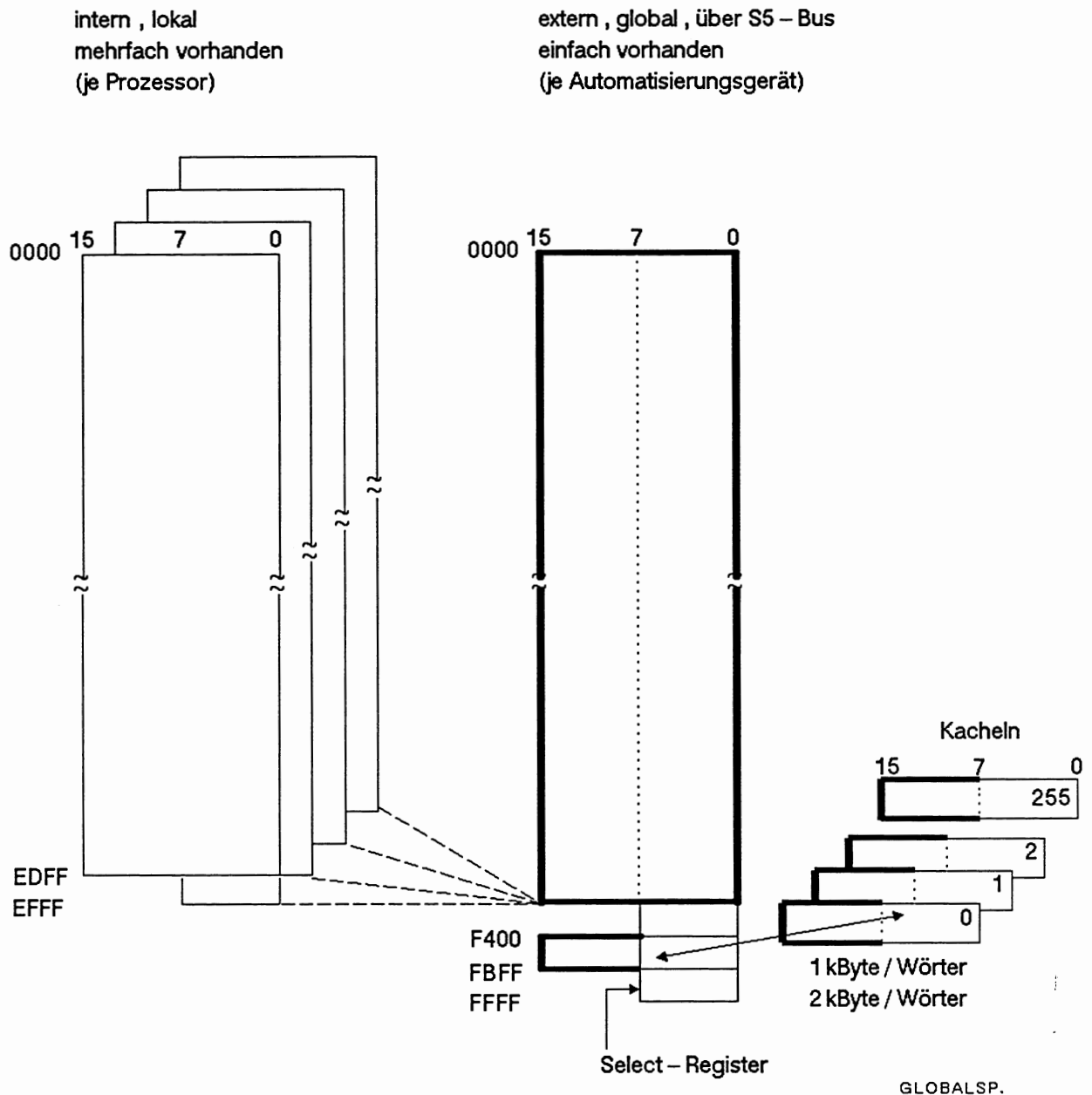


Abb. 9-1: Globaler und lokaler Speicher

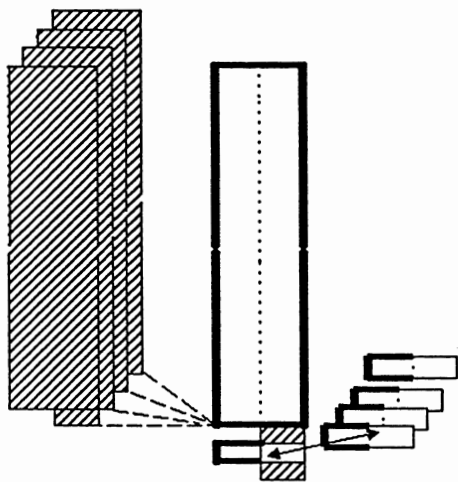
Die folgenden Befehle ermöglichen Zugriffe auf lokale bzw. globale Speicherbereiche über *absolute Adressen*.

- auf den Lokalbereich (0000 bis EFFF) und den byteweise organisierten Teil des Globalbereichs (F000 bis F3FF, FC00 bis FFFF):  
TNB, TNW, LIR, TIR
- auf den wortweise organisierten Teil des Lokalbereiches (0000 bis EDFF):  
LRW, TRW, LRD, TRD
- auf den byteweise organisierten Teil des Globalbereiches (0000 bis EFFF):  
LB GB, LB GW, LB GD, TB GB, TB GW, TB GD, TSG

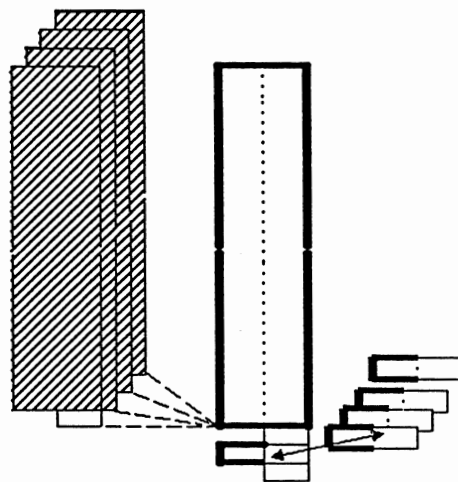
- d) auf den wortweise organisierten Teil des Globalbereichs (0000 bis EFFF):  
LW GW, LW GD, TW GW, TW GD, TSG
- e) auf den byteweise organisierten Teil des Globalbereichs (F400 bis FBFF, = Kachelbereich):  
LB CB, LB CW, LB CD, TB CB, TB CW, TB CD, TSC
- f) auf den wortweise organisierten Teil des Globalbereichs (F400 bis FBFF, = Kachelbereich):  
LW CW, LW CD, TW CW, TW CD, TSC

**Zugriffe auf lokale bzw. globale Speicherbereiche über absolute Adressen**

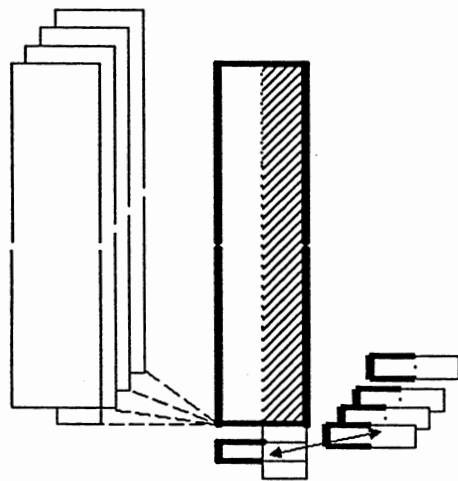
kein Zugriff möglich
  Zugriff möglich



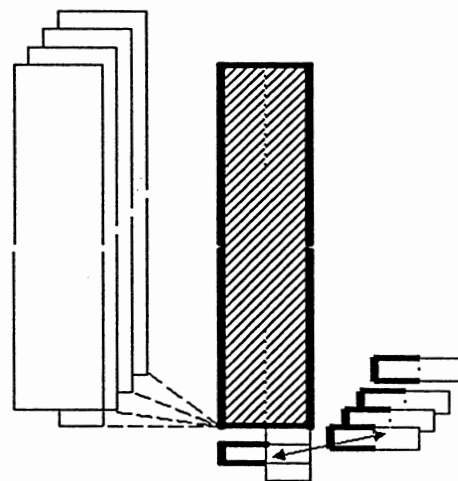
a) LIR, TIR, TNB, TNW



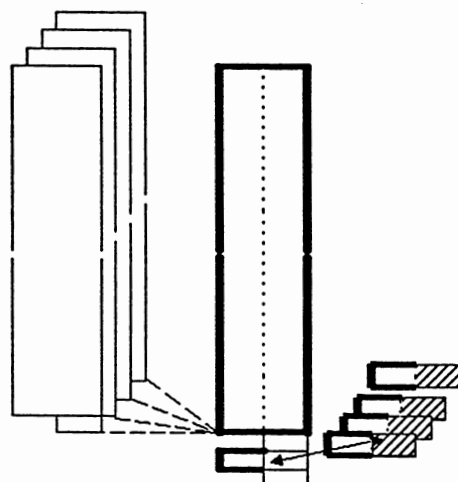
b) LRW, TRW, LRD, TRD



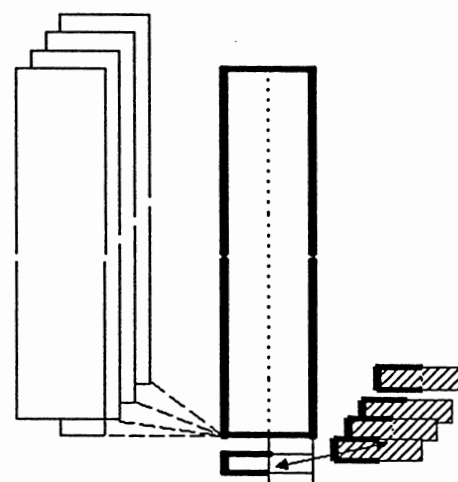
c) LB GB, LB GW, LB GD  
TB GB, TB GW, TB GD, TSG



d) LW GW, LW GD  
TW GW, TW GD, TSG



e) LB CB, LB CW, LB CD  
TB CB, TB CW, TB CD, TSC



f) LW CW, LW CD,  
TW CW, TW CD, TSC

## 9.1 Zugriffe auf Register und den Speicher über eine Adresse im Akku 1

Die Register sind besondere Speicherzellen, die der Prozessor zum Bearbeiten des STEP5-Programms benötigt. Jedes Register ist 16 Bit breit. Mit den Systemoperationen LIR (Lade indirekt Register) und TIR (Transferiere indirekt Register) können Sie auf die Inhalte der Register zugreifen.

**LIR 0...15** Lädt in das Register 0...15 den Inhalt der durch den Akku 1-L adressierten Speicherzelle.

**TIR 0...15** Transferiert in die durch den Akku 1-L adressierte Speicherzelle den Inhalt des Registers 0...15.

Die Speicherzelle liegt entweder im Lokalbereich (0000 bis EFFF) oder im byteweise organisierten Teil des Globalbereichs (F000 bis F3FF, FC00 bis FFFF).

### LIR und TIR auf den Kachelbereich

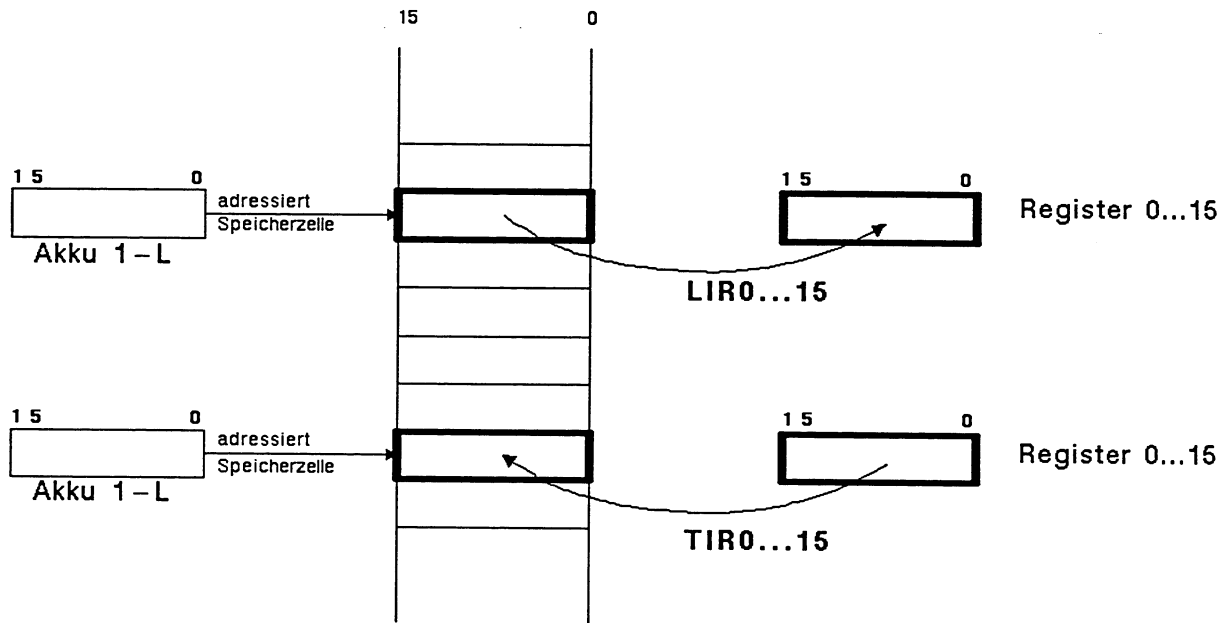
Die Befehle LIR und TIR sind im Mehrprozessor-Automatisierungsgerät AG S5-135U nicht für den Zugriff auf den Kachelbereich (F400 - FBFF) geeignet. Verwenden Sie stattdessen die Befehle aus Kapitel 9.3.5 "Zugriff auf den Kachelspeicher" oder die Sonderfunktionen aus Kapitel 6.6 "Kachelzugriffe".

Die folgende Darstellung zeigt die Registerbelegung bei der CPU 928. Sie unterscheidet sich von der Registerbelegung beim S- bzw. R-Prozessor!

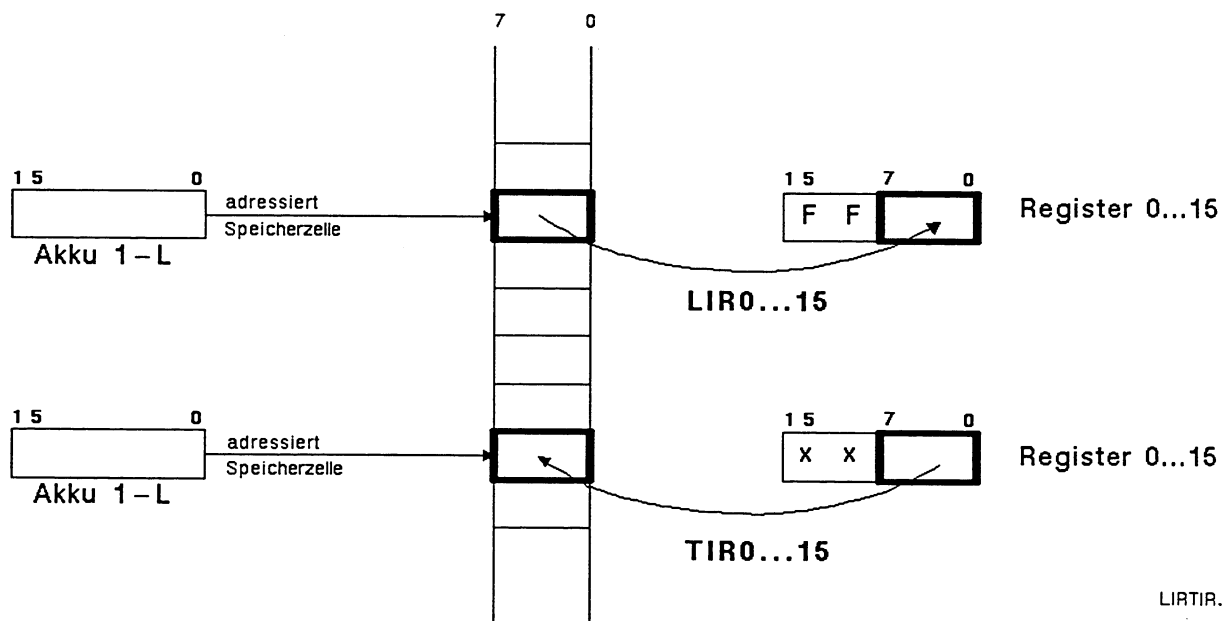
Register-Nummer	Registerbelegung
Register 0:	Akku 1-H (linkes Wort von Akku1, Bit 16 - 31)
Register 1:	Akku 1-L (rechtes Wort von Akku1, Bit 0 - 15)
Register 2:	Akku 2-H
Register 3:	Akku 2-L
Register 6:	DBA (Datenbaustein-Anfangsadresse)
Register 8:	DBL (Datenbaustein-Länge)
Register 9:	Akku 3-H
Register 10:	Akku 3-L
Register 11:	Akku 4-H
Register 12:	Akku 4-L
Register 15:	SAZ (STEP-Adreßzähler)

Die Register 4, 5, 7, 13 und 14 sind nicht vorhanden. LIR/TIR auf diesen Registernummern werden wie eine Nulloperation (NOP) behandelt.

## LIR und TIR auf 16-Bit-Speicherbereiche



## LIR und TIR auf 8-Bit-Speicherbereiche



LIR/TIR.

Wird mit LIR/TIR auf Speicherbereiche zugegriffen, die nur 8 Bit breit sind (für Speicheradressen  $\geq$  EE00H), so beachten Sie, daß

- bei TIR nur das Low-Byte des Registers übertragen wird (das High-Byte des Registers geht verloren), und
- bei LIR das High-Byte des Registers mit FFH beschrieben wird.

● **Register 0 bis 3 und 9 bis 12: Akku 1, 2, 3 und 4**

Die Akkumulatoren werden vom Prozessor bei der Programmbearbeitung als Zwischenspeicher verwendet. Mit den Befehlen TIR und LIR können Sie die Inhalte der Akkus in absolut adressierte Speicherzellen transferieren bzw. die Inhalte absolut adressierter Speicherzellen in die Akkus laden. Die Absolutadresse der Speicherzelle steht jeweils im Akku 1-L.

**Beispiel:**

Der Inhalt der Speicherzelle mit der Adresse A000 wird ins Merkerwort MW 100 geladen.

```
:L   KHA000      Adresse A000 der Speicherzelle in Akku 1 laden
:LIR 1           Inhalt der durch Akku 1 adressierten Speicherzelle
:           ins Register 1 = Akku 1 laden
:T   MW100       Inhalt der Adresse A000 im Merkerwort MW 100 ablegen
:BE
```

**Beispiel:**

Den Inhalt des Merkerwortes 200 wird in die Speicherzelle mit der Adresse A000 transferiert.

```
:L   MW200       Merkerwort MW 200 in den Akku 1 laden
:L   KHA000      Adresse A000, auf die transferiert werden soll, in
:           den Akku 1 laden (Merkerwort MW 200 nach Akku 2)
:TIR 3           Inhalt von Register 3 = Akku 2 in die durch Akku 1
:           adressierte Speicherzelle transferieren
:BE
```

● **Register 6: DBA (Datenbaustein-Anfangsadresse)**

Beim Aufschlagen eines Datenbausteins mit den Befehlen A DB und AX DX wird das Register 6 mit der Adresse des DWO im aufgeschlagenen Datenbaustein geladen. Diese Adresse ist in der Bausteinadreßliste im DB 0 enthalten.

Das DBA-Register wird vor jedem Aufruf des OBI oder FBO gleich '0' gesetzt.

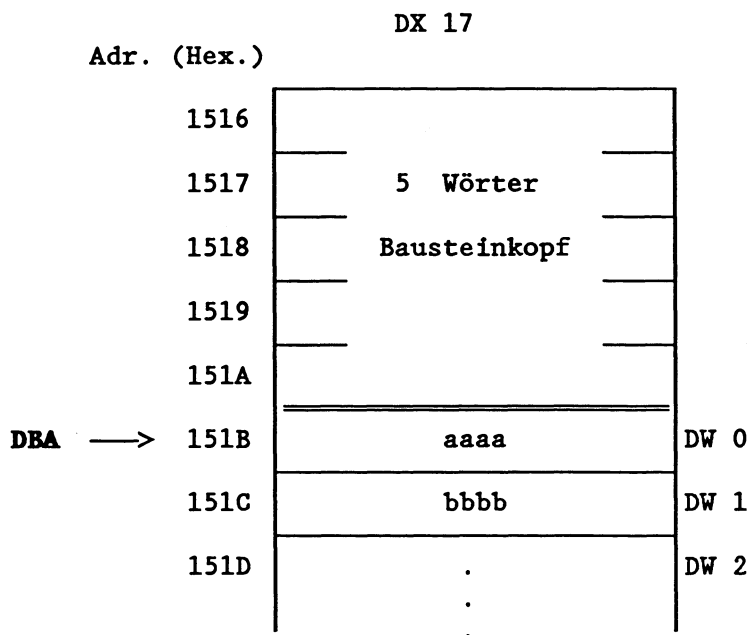
Das DBA-Register bleibt erhalten, wenn

- durch eine Sprunganweisung (SPA/SPB) die Programmbearbeitung in einem anderen Baustein fortgesetzt wird oder
- eine andere Programmbearbeitungsebene eingeschachtelt wird.

Es ändert sich, wenn

- ein anderer Datenbaustein aufgeschlagen wird oder
- ein Rücksprung in einen übergeordneten Baustein erfolgt nachdem im aufgerufenen Baustein ein neuer Datenbaustein aufgeschlagen wurde (siehe auch Kapitel 2.4.3, Gültigkeitsbereich von Datenbausteinen).

**Beispiel:**      AX   DX17



Bei Aufruf des DX 17 wird im DBA-Register die Adresse des Speicherwortes, in welchem das DW 0 hinterlegt ist, im Beispiel: DBA = 151B (Hexadez.) eingetragen.

Hinweis: Im USTACK ist die im DBA-Register eingetragene Adresse unter 'DB-ADR' angegeben.

Zugriffe auf Datenwörter erfolgen normalerweise mit den STEP5-Befehlen L/T DW, L/T DR, L/T DL, L/T DD, U/O/UN/ON/=/S/R Dx.y. Diese sind jedoch nur bis Datenwort DW 255 erlaubt. Durch Manipulieren des DBA-Registers können Sie mit diesen Befehlen auch auf Datenwörter > 255 zugreifen.

### Beispiel:

Durch Ändern von Register 6 wird das Datenwort DW 300 des Datenbausteins DB 100 geladen.

FB 7

NAME: LIR/TIR6

:L	BS34	Anfangsadresse der DB-Adressliste plus 100
:ADD	BF+100	ergibt Adresslisteneintrag des DB100
:LIR	1	Anfangsadresse des DB100 (DW0) nach Akku 1
:ADD	KF+200	Adresse des DW 200 im DB100 im Systemdaten-
:T	BS62	wort BS 62 ablegen
:L	BS20	Basisadresse Systemdaten laden
:ADD	KF+62	Adresse des BS 62 in Akku 1 laden
:LIR	6	DBA-Register mit dem Inhalt der Adresse des BS62
:		laden, d.h.,
:		der Datenbausteinanfang wird auf DW 200 gesetzt
:L	DW100	DW (200 + 100) = DW 300 laden
:T	MW100	DW 300 im Merkerwort MW100 ablegen
:BE		

### **Achtung:**

Wenn Sie wie oben das DBA-Register verstellen, wird das DBL-Register (siehe unten) nicht verändert. Damit ist eine Transferfehlerüberwachung nicht mehr gewährleistet!

Durch Anwendung des **Sonderfunktions-OBs 180** "Variabler Datenbausteinzugriff" können Sie das DBA-Register ebenfalls um eine vorgegebene Anzahl an Datenwörtern verschieben. Da der OB 180 gleichzeitig das DBL-Register verändert, werden Transferfehler weiterhin überwacht.

### **Beispiel:**

**FB7**

**NAME :OB180**

<b>:A</b>	<b>DB100</b>	DBA- und DBL-Register mit den Werten des
<b>:L</b>	<b>KF200</b>	DB 100 laden und mit Hilfe des OB180 das
<b>:SPA</b>	<b>OB180</b>	DBA-Register um 200 erhöhen und das DBL-
<b>:</b>		Register um 200 vermindern
<b>:SPB</b>	<b>=FEHL</b>	Fehlerausgang, falls der DB 100 weniger als
<b>:</b>		200 Datenwörter enthält
<b>:L</b>	<b>DW100</b>	DW 300 laden und
<b>:T</b>	<b>MW100</b>	im MW 100 ablegen
<b>:BEA</b>		
<b>FEHL :</b>		Programmteil zur Fehlerbehandlung
<b>:</b>		
<b>:BE</b>		



● **Register 8: DBL = Datenbaustein-Länge**

Zusätzlich zum DBA-Register wird bei jedem Aufruf eines Datenbausteins das DBL-Register geladen. Es enthält die Länge (in Wörtern) des aufgerufenen Datenbausteins ohne Baustein-Kopf.

Das DBL-Register wird vor jedem Aufruf des OBl oder FB0 gleich '0' gesetzt.

Das DBL-Register bleibt erhalten, wenn

- durch eine Sprunganweisung (SPA/SPB) die Programmbearbeitung in einem anderen Baustein fortgesetzt wird oder
- eine andere Programmbearbeitungsebene eingeschachtelt wird.

Es ändert sich, wenn

- ein anderer Datenbaustein aufgeschlagen wird oder
- ein Rücksprung in einen übergeordneten Baustein erfolgt, nachdem im aufgerufenen Baustein ein neuer Datenbaustein aufgeschlagen wurde (siehe auch Kapitel 2.4.3 "Gültigkeitsbereich von Datenbausteinen").

**Beispiel:**                    AX DX17

Adr. (Hex.)	DX 17	
1516		
1517	5 Wörter	
1518	Bausteinkopf	
1519		
151A		
<b>DBA</b> → 151B	aaaa	DW 0
151C	bbbb	DW 1
151D	cccc	DW 2
151E	dddd	DW 3
151F	eeee	DW 4
1520	ffff	DW 5
1521	gggg	DW 6
1522	hhhh	DW 7

}

> **DBL**

Bei Aufruf des DX 17 wird im DBL-Register die Anzahl der vorhandenen Datenwörter eingetragen, im Beispiel: DBL = 8 (DW 0 bis DW 7).

Hinweis: Im USTACK ist die im DBL-Register eingetragene Anzahl unter 'DBL-REG' angegeben.

● **Register 15: SAZ = Step-Adreßzähler**

Im Register 15 ist während der STEP5-Programmbearbeitung die Absolutadresse desjenigen Befehls im Programmspeicher eingetragen, der als nächster zu bearbeiten ist.

**Beispiel: Alle Datenwörter eines Datenbausteins werden mit einer Konstanten beschrieben.**

Das unten dargestellte Programm beschreibt alle Datenwörter des DB 50 mit der Konstanten KH A5A5. Nach Ändern der fettgedruckten STEP5-Befehle kann es auch zum Beschreiben anderer Datenbausteine (DB oder DX) mit beliebigen Werten benutzt werden. Nicht vorhandene Datenbausteine oder Datenbausteine mit null Datenwörtern werden erkannt und führen zum Sprung zur Marke NIVO.

Die Anfangsadresse (DBA) und Länge (DBL) des Datenbausteins wird über die Sonderfunktion OB 181 'Datenbaustein (DB/DX) testen' ermittelt.

Das Programm nutzt alle vier Akkumulatoren. Im Bild sehen Sie die Belegung der Akkumulatoren während des Programmablaufs bis zur Marke SCHL. Innerhalb der Schleife ändert sich die Akkumulatorbelegung nicht.

Der Akku 1 enthält zunächst die Adresse des letzten Datenwortes (DBA + DBL - 1) und wird mit jedem Schleifendurchlauf um eins vermindert.

Der Akku 2 enthält die Adresse des ersten Datenwortes (DBA). Die Schleife wird abgebrochen, sobald der Inhalt des Akku 1 kleiner als der Inhalt des Akku 2 ist.

Zum Beschreiben der Datenwörter wird der Befehl TIR 10 verwendet, der den Inhalt des Akku 3-L (die Konstante) unter der im Akku 1-L stehenden Adresse abspeichert.

:L	<b>KHA5A5</b>	Konstante, mit der alle Datenwörter
:		beschrieben werden sollen
:L	<b>KY1,50</b>	Typ und Nummer des Datenbausteins
:ENT		
:SPA	OB181	Sonderfunktions-OB 'Datenbausteine testen'
:SPB	=NIVO	Abbruch, falls DB50 nicht vorhanden
:TAK		
:ENT		
:+F		
:		Akku1 := Adresse des letzten Datenworts + 1
:		Akku2 := Adresse des ersten Datenworts
:		Akku3 := Konstante
:!=F		Abbruch, falls der DB50 null Datenwörter
:SPB	=NIVO	enthält
:		
SCHL	:ADD BF-1	Alle Datenwörter, beginnend mit dem letzten
:	:TIR 10	Datenwort, mit der im Akku 3-L enthaltenen
:	:<F	Konstanten beschreiben
:	:SPB =SCHL	
:		

```

:                                     Fortsetzung des Programms...
WEIT : .                             ...nachdem alle Datenwörter beschrieben
:                                     wurden
:BEA
NIVO : .                             ...falls der DB50 nicht vorhanden ist
:                                     oder null Datenwörter enthält.
:BE

```

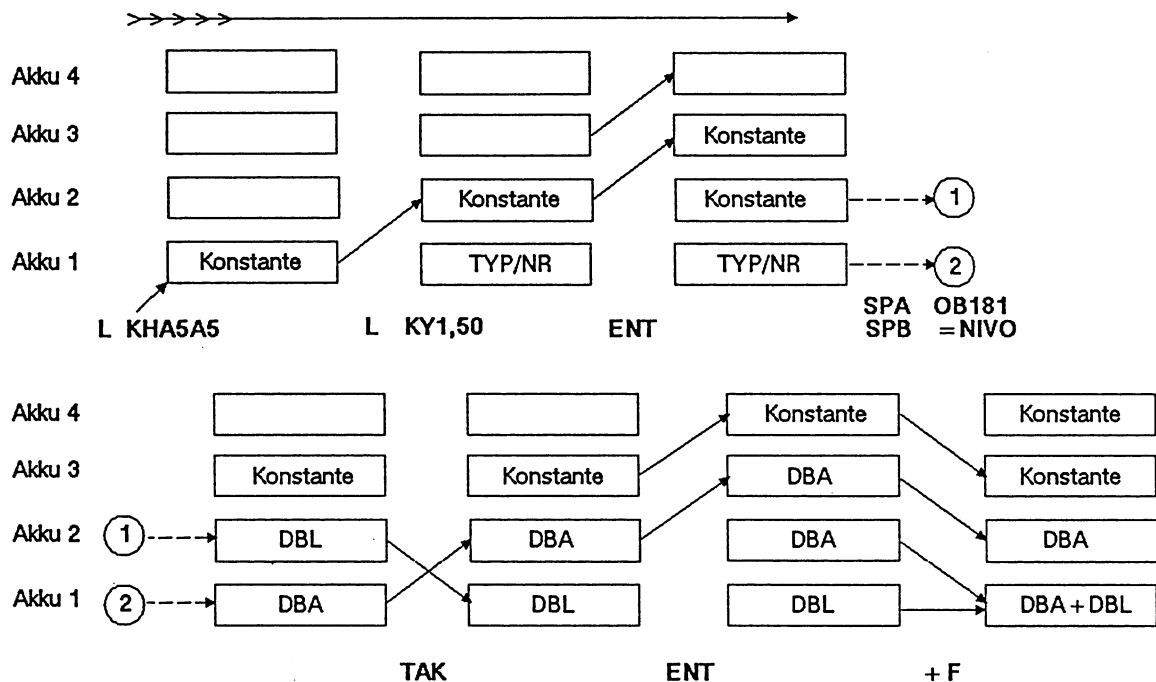


Abb. 9-2: Belegung der Akkumulatoren während des Programmablaufs

#### Hinweis:

Der Programmteil ab der Marke SCHL kann zum Beschreiben beliebiger Speicherbereiche (z.B.: Merker, Zeiten, Zähler) mit einer Konstanten genutzt werden.

#### Beispiel 9-3: Löschen aller Merkerbytes (MB0 bis MB255)

```

:L KB0      Konstante, mit der alle Merkerbytes beschrieben
:           werden sollen
:L BS14     Basisadresse des Merkerbereichs (= Adresse des
:           ersten Merkerbytes MB0)
:ENT
:L KF256    + Länge des Merkerbereichs
:ENT        = (Adresse des letzten Merkerbytes MB255) + 1
:+F
:
SCHL :ADD BF-1  Alle 256 Merkerbytes, beginnend mit dem Merker-
:TIR 10      byte MB255, mit der im Akku 3-LL enthaltenen
:<F          Konstanten beschreiben
:SPB =SCHL
:
:
:

```

## 9.2 Speicherblöcke transferieren

Mit den Systemoperationen TNB und TNW können Sie Speicherblöcke transferieren (max. 255 Byte mit TNB, max. 255 Wörter mit TNW).

Operation	Parameter	Beschreibung
TNB	0 bis 255	Transferiere Speicherblock (1 bis 255 Byte)
TNW	0 bis 255	Transferiere Speicherblock (1 bis 255 Wörter)

Mit den Befehlen TNB und TNW können Sie sowohl auf den lokalen Speicherbereich als auch auf den byteweise organisierten Teil des globalen Speicherbereichs (F000 bis F3FF, FC00 bis FFFF) zugreifen.

### TNB und TNW auf den Kachelbereich

Die Befehle TNB und TNW sind im Mehrprozessor-Automatisierungsgerät AG S5-135U nicht für den Zugriff auf den Kachelbereich (F400 - FBFF) geeignet. Verwenden Sie stattdessen die Befehle aus Kapitel 9.3.5 "Zugriff auf den Kachelspeicher" oder die Sonderfunktionen aus Kapitel 6.6 "Kachelzugriffe".

Der Parameter bei TNW/TNB gibt die Länge (Anzahl der Wörter/Anzahl der Bytes) des zu übertragenden Bereichs an. Die Endadresse des Quellbereichs muß vorher in den Akku 2, die Endadresse des Zielbereichs in den Akku 1 geladen werden. Es werden also jeweils die oberen (größten) Adressen von Quell- und Zielbereich angegeben. Der Transfer selbst erfolgt bei der CPU 928 "dekrementierend", d.h., er beginnt seine Übertragung mit der höchsten Adresse des Quellbereichs und beendet sie mit der niedrigsten.

Quellbereich und Zielbereich müssen vollständig in einem Speicherbereich liegen und dürfen sich nicht überschneiden. Dabei sind folgende Speicherbereiche durch Bereichsgrenzen unterschieden:

#### Adressen (hexadezimal)

- |    |               |  |
|----|---------------|--|
| 1. | 0000 bis 1FFF | Anwender-Modul (16 Bit) 8K-Wörter                    |
|    | 0000 bis 3FFF | Anwender-Modul (16 Bit) 16K-Wörter                   |
|    | 0000 bis 7FFF | Anwender-Modul (16 Bit) 32K-Wörter                   |
| 2. | 8000 bis DD7F | DB-RAM (16 Bit)                                      |
| 3. | DD80 bis EDFD | System-RAM (16 Bit: DB0, BA/BS, Zeiten, Zähler etc.) |
| 4. | EE00 bis EFFF | RAM (8 Bit: Merker, Prozeßabbild)                    |
| 5. | F000 bis FFFF | Peripherie (8 Bit)                                   |

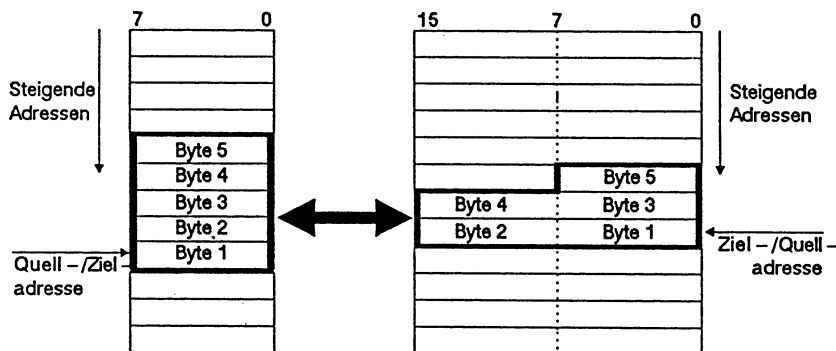
(Siehe Kapitel 8 "Speicherbelegung")

## Pseudobefehlsgrenzen bei TNB und TNW

Die Befehle TNB und TNW sind langlaufende STEP5-Befehle, die sogenannte 'Pseudobefehlsgrenzen' enthalten: D.h.: Die Datenübertragung erfolgt je nach Quell- und Zielbereich in Teilblöcken unterschiedlicher Größe. Wenn nun während der Übertragung eines Teilblocks ein Fehler (z.B. Zyklusfehler) oder eine Unterbrechung (z.B. durch Weck- oder Prozeßalarm) auftritt, so wird am Ende dieses Teilblocks an der Pseudobefehlsgrenze der entsprechende Organisationsbaustein eingeschachtelt. Voraussetzung für den Aufruf des Prozeßalarm-OBs oder eines Weckalarm-OBs an einer Pseudobefehlsgrenze ist, daß im DX 0 "Unterbrechbarkeit an Befehlsgrenzen" eingestellt ist.

Ausnahme: Treten während der Übertragung Quittungsverzögerungen und/oder Adressierfehler auf, so werden zuerst alle Teilblöcke übertragen und dann vor der Ausführung des nächsten Befehls einmalig der dafür vorgesehene Fehler-Organisationsbaustein aufgerufen (bei QVZ und ADF gleichzeitig nur der QVZ-OB). Als QVZ-Fehleradresse wird immer die niedrigste Adresse des Quellbereichs angegeben. Unabhängig davon können an den Pseudobefehlsgrenzen der OB 2, OB 10 bis OB 18 oder ein Fehler-Organisationsbaustein eingeschachtelt werden.

## TNB und TNW zwischen 8- und 16-Bit-Speicherbereichen

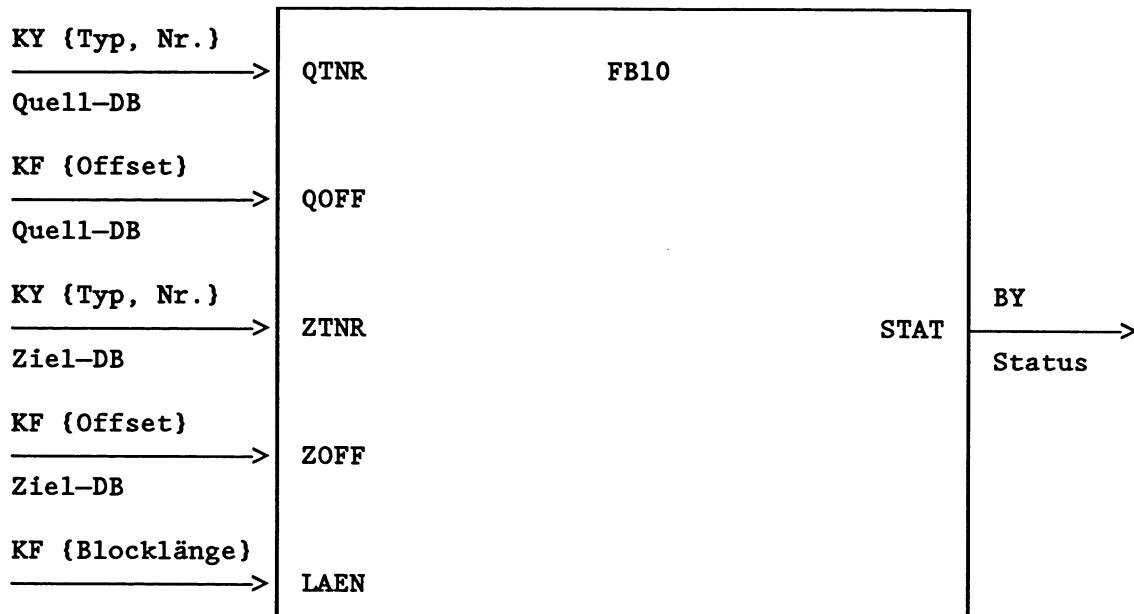


Übertragen der Bytes 1 bis 5:      L <Quelladresse>  
   L <Zieladresse>  
   TNB 5

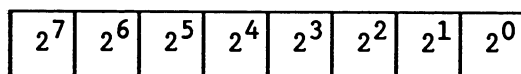
Übertragen der Bytes 1 bis 4:      L <Quelladresse>  
   L <Zieladresse>  
   TNW 2

### Beispiel:

Kopieren eines Blocks von max. 4095 Datenwörtern aus einem DB- oder DX-Datenbaustein in einen anderen DB- oder DX-Datenbaustein. Der Anfang des Blocks wird innerhalb des Quell- und Ziel-Datenbausteins durch je einen Offset-Wert zwischen 0 und 4095 festgelegt.



Vor dem Kopieren werden die Eingangsparameter geprüft. Im Fehlerfall wird im Ausgangsparameter STAT das Bit  $2^7 = 1$  gesetzt und in den Bits  $2^0$  bis  $2^2$  die Fehlerart angegeben:



↓  
0 = kein Fehler  
1 = Fehler

└───┘  
V  
Fehlerart:  
1 = Quell-DB = Ziel-DB  
2 = Offset oder Länge > 4095  
3 = Quell-DB nicht vorhanden oder unzulässig  
4 = Quell-DB zu kurz  
5 = Ziel-DB nicht vorhanden oder unzulässig  
6 = Ziel-DB im Nur-Lese-Speicher (EPROM-Modul)  
7 = Ziel-DB zu kurz

Der FB 10 gliedert sich in fünf Programmabschnitte mit folgenden Aufgaben:

#### 1. Eingangsparameter

- Prüfen, ob Quell- und Ziel-Datenbaustein nicht denselben Typ und dieselbe Nummer haben.
- Prüfen, ob die Eingangsparameter 'Quell-Offset', 'Ziel-Offset' und 'Blocklänge' kleiner 4096 sind.

## 2. Quell-Datenbaustein

- Prüfen, ob der Quell-Datenbaustein vorhanden und lang genug ist.
- Berechnen der absoluten Adresse des letzten Datenworts im Ziel-Block.

## 3. Ziel-Datenbaustein

- Prüfen, ob der Ziel-Datenbaustein vorhanden und lang genug ist und ob er im Schreib-Lese-Speicher (RAM-Modul) liegt.
- Berechnen der absoluten Adresse des letzten Datenworts im Ziel-Block.

## 4. Transfer

- Kopiervorgang durchführen mit Hilfe des TNW-Befehls. Blöcke mit mehr als 255 Wörtern werden in Teilblöcken zu je 128 Wörtern übertragen (Befehl TNW 128). Ein eventuell noch vorhandener Rest wird durch einen zusätzlichen TNW-Befehl übertragen.

## 5. Anzeige

- Versorgen des Ausgangsparameters 'Status' entsprechend dem Ergebnis der vorgenommenen Prüfungen.

### Belegte Speicherzellen:

MW 242	Endadresse des Datenziels
MW 244	Endadresse der Datenquelle
MW 246	Blocklänge
MW 248	Offset im Ziel-Datenbaustein
MW 250	Typ und Nummer des Ziel-Datenbausteins
MW 252	Offset im Quell-Datenbaustein
MW 254	Typ und Nummer des Quell-Datenbausteins
BS 60	Teil-Block-Zähler

### Programmierung des Funktionsbausteins FB 10

Hinweis: Soll ab Datenwort DW 0 kopiert werden, entfallen die kursiv gedruckten Programmteile. Es wird kein Offset-Wert angegeben.

#### FB10

##### NETZWERK 1

NAME	:DB-DB-TR	DATENBAUST.-DATENBAUST.-TRANSFER
BEZ	:QTNR	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY
BEZ	:QOFF	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
BEZ	:ZTNR	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KY
BEZ	:ZOFF	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
BEZ	:LAEN	E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF
BEZ	:STAT	E/A/D/B/T/Z: A BI/BY/W/D: BY

```

:
: LW =QTNR      BEGIN EINGANGS-PARAMETER
: T  MW254      TYP (DB/DX) UND NUMMER DES
: LW =ZTNR      QUELL-DATENBAUSTEINS
: T  MW250      TYP (DB/DX) UND NUMMER DES
: ! =F          ZIEL-DATENBAUSTEINS
: SPB =F001     QUELL-DB = ZIEL-DB ?
:              SPRUNG, FALLS JA
:
: LW =QOFF      OFFSET IM QUELL-
: T  MW252      DATENBAUSTEIN
: LW =ZOFF      OFFSET IM ZIEL-
: T  MW248      DATENBAUSTEIN
: OW
: LW =LAEN      LAENGE (ANZAHL DATENWORTE) DES
: T  MW246      ZU TRANSFERIERENDEN BLOCKES
: OW            (BLOCK-LAENGE)
: L  KHf000     QUELL-OFFSET, ZIEL-OFFSET ODER
: UW            LAENGE >= 4096 ?
: SPP =F002     SPRUNG, FALLS JA
:              END EINGANGS-PARAMETER
:
:
:              BEGIN QUELL-DATENBAUSTEIN
: L  MW254      TYP U. NUMMER DES QUELL-DATENBAUSTEINS
: SPA OB181     DATENBAUSTEIN TESTEN
: SPB =F003     SPRUNG, FALLS BST.-TEST NEGATIV
: TAK          A1: ANZAHL DW , A2: ADRESSE
: ENT          A3: ADRESSE
: L  MW252      OFFSET IM QUELL-DATENBAUSTEIN
: ENT          A3: ANZAHL DW, A4: ADRESSE
: L  MW246      BLOCK-LAENGE
: +F           OFFSET + BLOCK-LAENGE
: <F           ANZ. DW < OFFSET + BL.-LAENGE ?
: SPB =F004     SPRUNG, FALLS JA
: L  KB1        A2: OFFSET + BL.-L., A3: ADRESSE
: -F           OFFSET + BLOCK-LAENGE - 1
: +F           OFFSET + BL.-L. - 1 + ADRESSE
: T  MW244      END-ADRESSE DER DATEN-QUELLE
:              END QUELL-DATENBAUSTEIN
:
:
:              BEGIN ZIEL-DATENBAUSTEIN
: L  MW250      TYP U. NUMMER DES ZIEL-DATENBAUSTEINS
: SPA OB181     DATENBAUSTEIN TESTEN
: SPB =F005     SPRUNG, FALLS BST.-TEST NEGATIV
: SPM =F006     SPRUNG, FALLS BST. IM EPROM
: TAK          A1: ANZAHL DW, A2: ADRESSE
: ENT          A3: ADRESSE
: L  MW248      OFFSET IM ZIEL-DATENBAUSTEIN
: ENT          A3: ANZAHL DW, A4: ADRESSE
: L  MW246      BLOCK-LAENGE
: +F           OFFSET + BLOCK-LAENGE
: <F           ANZ. DW < OFFSET + BL.-LAENGE ?
: SPB =F007     SPRUNG, FALLS JA
: L  KB1        A2: OFFSET + BL.-L., A3: ADRESSE
: -F           OFFSET + BLOCK-LAENGE - 1
: +F           OFFSET + BL.-L. - 1 + ADRESSE
: T  MW242      END-ADRESSE DES DATEN-ZIELS
:              END ZIEL-DATENBAUSTEIN
:
:

```



:		BEGIN TRANSFER
:L	KB0	VERGLEICHSWERT
:L	MB246	BLOCK-LAENGE, HIGH-BYTE
:!=F		BLOCK-LAENGE >= 256 WORTE ?
:SLW	1	MULTIPL. MIT 2, ANZAHL TEIL-
:T	BS60	BLOECKE MIT JE 128 WORTEN
:L	MW244	END-ADRESSE DER DATEN-QUELLE
:L	MW242	END-ADRESSE DES DATEN-ZIELS
:SPB	=REST	SPRUNG, FALLS BLOCK-L. < 256 W.
SCHL	:TNW 128	TRANSFER EINES TEIL-BLOCKS
	:ADD KF-128	QUELL-END-ADRESSE UM LAENGE DES
	:TAK	TEIL-BLOCKS REDUZIEREN
	:ADD KF-128	ZIEL-END-ADRESSE UM LAENGE DES
	:TAK	TEIL-BLOCKS REDUZIEREN
	:SPA OB160	ZAEHL-SCHLEIFE
	:SPB =SCHL	SPRUNG, FALLS NICHT ALLE
	:	TEIL-BLOECKE TRANSFERIERT
REST	:B MW246	BLOCK-LAENGE, LOW-BYTE
	:TNW 0	REST-BLOCK TRANSFERIEREN
	:	END TRANSFER
	:	
	:	BEGIN ANZEIGE
	:L KB0	KENNUNG 00 (HEX.): KEIN FEHLER
ENDE	:T =STAT	AUSGANGS-PARAMETER STATUS/FEHLER
	:BEA	
F001	:L KB129	FEHLER-KENNUNG 81 (HEX.):
	:SPA =ENDE	QUELL-DB = ZIEL-DB
F002	:L KB130	FEHLER-KENNUNG 82 (HEX.):
	:SPA =ENDE	OFFSET ODER LAENGE >= 4096
F003	:L KB131	FEHLER-KENNUNG 83 (HEX.):
	:SPA =ENDE	QUELL-DB UNZULAESSIG
F004	:L KB132	FEHLER-KENNUNG 84 (HEX.):
	:SPA =ENDE	QUELL-DB ZU KURZ
F005	:L KB133	FEHLER-KENNUNG 85 (HEX.):
	:SPA =ENDE	ZIEL-DB UNZULAESSIG
F006	:L KB134	FEHLER-KENNUNG 86 (HEX.):
	:SPA =ENDE	ZIEL-DB IM NUR-LESE-SPEICHER
F007	:L KB135	FEHLER-KENNUNG 87 (HEX.):
	:SPA =ENDE	ZIEL-DB ZU KURZ
	:	END ANZEIGE
	:BE	

## 9.3 Operationen mit dem BR-Register

Das BR-Register (Basisadreßregister, 32 Bit) wird von den nachfolgend beschriebenen Lade-/Transferbefehlen (Kapitel 9.3.3 ff) zur Adressierung des Speichers verwendet. Zugriffen wird auf die Speicherzelle, deren absolute Adresse sich als Summe von BR-Registerinhalt und einer Konstanten errechnet:

Absolute Adresse = BR-Registerinhalt + Konstante

Das BR-Register bleibt erhalten, wenn

- durch eine Sprunganweisung (SPA/SPB) die Programmbearbeitung in einem anderen Baustein fortgesetzt wird oder
- eine andere Programmbearbeitungsebene eingeschachtelt wird.

Das BR-Register wird vor Aufruf einer Programmbearbeitungsebene gleich '0' gesetzt.

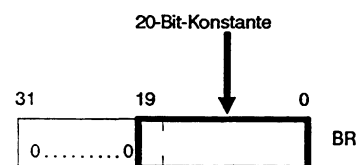
### 9.3.1 Laden des BR-Registers

Mit folgenden Befehlen läßt sich der Inhalt des BR-Registers neu laden bzw. modifizieren:

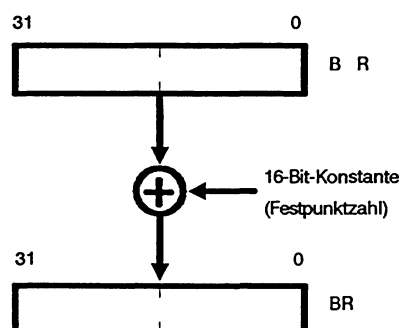
Operation	Parameter	Beschreibung
MBR	0 bis FFFFF	Lade das BR-Register mit einer 20-Bit-Konstanten <sup>1)</sup>
ABR	-32768 bis +32767	Addiere eine 16-Bit-Konstante zum Inhalt des BR-Registers

<sup>1)</sup> Die Bit  $2^{20}$  bis  $2^{31}$  des BR-Registers werden gleich '0' gesetzt.

MBR 0 bis FFFFF



ABR -32767 bis +32767



MBRABR.GEM

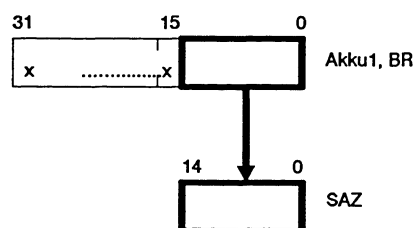
### 9.3.2 Verschieben von Registerinhalten

Um die Register insgesamt flexibler nutzen zu können, wurden neue Befehle eingeführt, mit denen die Inhalte einzelner Register verändert bzw. in andere Register transferiert werden können.

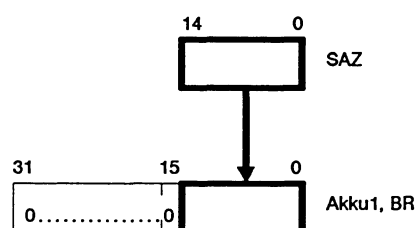
Operation	Parameter	Beschreibung
MAS	-	Transferiere den Inhalt des Akku 1 (Bit $2^0$ bis $2^{14}$ ) in das SAZ-Register (STEP-Adreßzähler)
MBS	-	Transferiere den Inhalt des BR-Registers (Bit $2^0$ bis $2^{14}$ , Basisadreßregister) in das SAZ-Register (STEP-Adreßzähler)
MSA	-	Transferiere den Inhalt des SAZ-Registers (STEP-Adreßzähler) in den Akku 1 <sup>1)</sup>
MSB	-	Transferiere den Inhalt des SAZ-Registers (STEP-Adreßzähler) in das BR-Register (Basisadreßregister) <sup>1)</sup>
MAB	-	Transferiere den Inhalt des Akku 1 (Bit $2^0$ bis $2^{31}$ ) in das BR-Register (Basisadreßregister)
MBA	-	Transferiere den Inhalt des BR-Registers (Basisadreßregister) in den Akku 1

<sup>1)</sup> Die Bits  $2^{15}$  bis  $2^{31}$  werden zu '0' gesetzt.

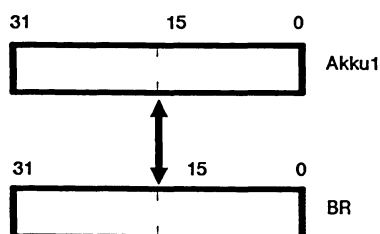
MAS, MBS



MSA, MSB



MAB, MBA



MASMB .GEM

### 9.3.3 Zugriffe auf den lokalen Speicher

Die folgenden Befehle ermöglichen den Zugriff auf den lokalen, *wortweise* organisierten Speicher über eine absolute Speicheradresse. Die Absolutadresse ist die Summe vom BR-Registerinhalt und der im Befehl enthaltenen 16-Bit-Konstanten (-32768 bis +32767).

Operation	Parameter	Beschreibung
LRW	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Wort in den Akku 1-L
LRD	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Doppelwort in den Akku 1
TRW	-32768 bis +32767	Transferiere den Inhalt des Akku 1-L in das durch BR-Register + Konstante adressierte Wort
TRD	-32768 bis +32767	Transferiere den Inhalt des Akku 1 in das durch BR-Register + Konstante adressierte Doppelwort

Die Absolutadresse muß zwischen 0 und EDFFH (bei LRW, TRW) bzw. 0 und EDFEH (bei LRD, TRD) liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

### 9.3.4 Zugriffe auf den globalen Speicher

Mit den folgenden Befehlen können Sie auf den globalen, *byte- oder wortweise* organisierten Speicher über eine absolute Speicheradresse zugreifen. Die Absolutadresse ist die Summe vom BR-Register-Inhalt und der im Befehl enthaltenen Konstanten (-32768 bis 32767).

#### Testen und Setzen einer Belegtzelle im Globalbereich

Der Zugriff einzelner Prozessoren auf gemeinsam genutzte Speicherbereiche kann über eine Belegtzelle gesteuert werden. Jedem gemeinsam genutzten Speicherbereich wird eine Belegtzelle zugeordnet, die von allen beteiligten Prozessoren vor jedem Zugriff getestet werden muß. Die Belegtzelle enthält entweder den Wert '0' oder die Steckplatzkennung des Prozessors, der gerade den Speicherbereich benutzt und ihn durch Beschreiben der Belegtzelle mit '0' wieder freigeben muß.

Der Befehl TSG unterstützt das Testen und Setzen einer Belegtzelle.

Operation	Parameter	Beschreibung
TSG	-32768 bis +32767	Teste und setze die durch BR-Register + Konstante adressierte Belegtzelle

Die Absolutadresse muß zwischen 0 und EFFFH liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

Als Belegtzelle wird das Low-Byte des durch BR-Register + Konstante adressierten Wortes verwendet. Falls der Inhalt des Low-Bytes '0' ist, trägt der TSG-Befehl die Steckplatzkennung (aus BS 29) in die Belegtzelle ein.

Das Testen (= Lesen) und das eventuelle Belegen (= Schreiben) bilden eine Programmeinheit, die nicht unterbrochen werden kann.

Das Testergebnis ist über die Anzeigen ANZ0 und ANZ1 auswertbar:

ANZ0	ANZ1	Bedeutung
0	0	Inhalt der Belegtzelle ist '0'; der Prozessor trägt seine Steckplatzkennung ein.
0	1	Die eigene Steckplatzkennung ist bereits in der Belegtzelle eingetragen.
1	0	Die Belegtzelle enthält eine fremde Steckplatzkennung.

#### **WICHTIG!**

**Der Befehl TSG muß von allen Prozessoren verwendet werden, die synchronisiert auf einen gemeinsamen globalen Speicherbereich zugreifen sollen.**

Beachten Sie auch die Erläuterungen zu den Befehlen SES und SEF (Semaphore setzen/freigeben, Kapitel 3.2.2) und zum Sonderfunktions-Organisationsbaustein OB 218 (Belegen einer Kachel, Kapitel 6.6.3).

## Lade- und Transferoperationen für den byteweise organisierten globalen Speicher

Operation	Parameter	Beschreibung
LB GB	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Byte in den Akku 1-LL
LB GW	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Wort in den Akku 1-L
LB GD	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Doppelwort in den Akku 1
TB GB	-32768 bis +32767	Transferiere den Inhalt des Akku 1-LL in das durch BR-Register + Konstante adressierte Byte
TB GW	-32768 bis +32767	Transferiere den Inhalt des Akku 1-L in das durch BR-Register + Konstante adressierte Wort
TB GD	-32768 bis +32767	Transferiere den Inhalt des Akku 1 in das durch BR-Register + Konstante adressierte Doppelwort

Die Absolutadresse muß liegen

- zwischen 0 und EFFFH (bei LB GB, TB GB),
- zwischen 0 und EFFEh (bei LB GW, TB GW),
- zwischen 0 und EFFCh (bei LB GD, TB GD).

Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

## Lade- und Transferoperationen für den wortweise organisierten globalen Speicher

Operation	Parameter	Beschreibung
LW GW	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Wort in den Akku 1-L
LW GD	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Doppelwort in den Akku 1
TW GW	-32768 bis +32767	Transferiere den Inhalt des Akku 1-L in das durch BR-Register + Konstante adressierte Wort
TW GD	-32768 bis +32767	Transferiere den Inhalt des Akku 1 in das durch BR-Register + Konstante adressierte Doppelwort

Die Absolutadresse muß zwischen 0 und EFFFH (bei LW GW, TW GW) bzw. 0 und EFFEh (bei LW GD, TW GD) liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

### 9.3.5 Zugriffe auf den Kachelspeicher

Der Globalbereich enthält zwischen den Adressen F400H bis FBFFH ein 'Fenster' zum Einblenden eines von max. 256 Speicherbereichen (= Kacheln). Eine Kachel belegt max. 2K Adressen und kann byteweise oder wortweise organisiert sein. Vor jedem Zugriff auf den Kachelbereich muß eine der 256 Kacheln durch Eintrag ihrer Kachelnummer in das Select-Register ausgewählt werden. Das Beschreiben des Select-Registers und der anschließende Zugriff auf den Kachelbereich ist ununterbrechbar.

Mit den folgenden Befehlen können Sie auf *byte- oder wortweise* organisierte Kacheln über eine absolute Speicheradresse zugreifen. Die Absolutadresse ist die Summe vom BR-Register-Inhalt und der im Befehl enthaltenen Konstanten (-32768 bis 32767).

Vor jedem Zugriff (Laden/Transferieren) auf den Kachelbereich muß eine der 256 Kacheln aufgeschlagen werden. Dazu übergeben Sie die Nummer der aufzuschlagenden Kachel im Akku 1-L; diese Nummer wird durch den Befehl ACR in das Kachel-Register eingetragen. Alle folgenden Kacheloperationen schreiben vor dem Kachelzugriff den Inhalt des Kachel-Registers in das Select-Register.

Das Kachel-Register bleibt erhalten, wenn

- durch eine Sprunganweisung (SPA/SPB) die Programmbearbeitung in einem anderen Baustein fortgesetzt wird oder
- eine andere Programmbearbeitungsebene eingeschachtelt wird.

Das Kachel-Register wird vor Aufruf einer Programmbearbeitungsebene gleich '0' gesetzt.

### Aufschlagen einer Kachel

Operation	Parameter	Beschreibung
ACR		Schlage diejenige Kachel auf, deren Nummer im Akku 1-L steht

zulässige Werte: 0 bis 255

Die Kachelnummer muß zwischen 0 und 255 liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

### Testen und Setzen einer Belegtzelle im Kachelbereich

Der Zugriff einzelner Prozessoren auf gemeinsam genutzte Speicherbereiche kann über eine Belegtzelle gesteuert werden. Jedem gemeinsam genutzten Speicherbereich wird eine Belegtzelle zugeordnet, die von allen beteiligten Prozessoren vor jedem Zugriff getestet werden muß. Die Belegtzelle enthält entweder den Wert '0' oder die Steckplatzkennung des Prozessors, der gerade den Speicherbereich benutzt und ihn durch Beschreiben der Belegtzelle mit '0' wieder freigeben muß.

Der Befehl TSC unterstützt das Testen und Setzen einer Belegtzelle.

Operation	Konstante	Beschreibung
TSC	-32768 bis +32767	Teste und setze die durch BR-Register + Konstante adressierte Belegtzelle aus der aufgeschlagenen Kachel

Die Absolutadresse muß zwischen F400H und FBFFH liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

Als Belegtzelle wird das Low-Byte des durch BR-Register + Konstante adressierten Wortes verwendet. Falls der Inhalt des Low-Bytes '0' ist, trägt der TSC-Befehl die Steckplatzkennung (aus BS 29) in die Belegtzelle ein.



Das Testen (= Lesen) und das eventuelle Belegen (= Schreiben) bilden eine Programmeinheit, die nicht unterbrochen werden kann.

Das Testergebnis ist über die Anzeigen ANZ0 und ANZ1 auswertbar:

ANZ0	ANZ1	Bedeutung
0	0	Inhalt der Belegtzelle ist '0'; der Prozessor trägt seine Steckplatzkennung ein.
0	1	Die eigene Steckplatzkennung ist bereits in der Belegtzelle eingetragen.
1	0	Die Belegtzelle enthält eine fremde Steckplatzkennung.

#### **WICHTIG!**

Der Befehl TSC muß von allen Prozessoren verwendet werden, die synchronisiert auf einen gemeinsamen globalen Speicherbereich zugreifen sollen.

Beachten Sie auch die Erläuterungen zu den Befehlen SES und SEF (Semaphore setzen/freigeben, Kapitel 3.2.2) und zum Sonderfunktions-Organisationsbaustein OB 218 (Belegen einer Kachel, Kapitel 6.6.3).

#### **Lade- und Transferoperationen für bytesteise organisierte Kacheln**

Operation	Parameter	Beschreibung
LB CB	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Byte aus der aufgeschlagenen Kachel in den Akku 1-LL
LB CW	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Wort aus der aufgeschlagenen Kachel in den Akku 1-L
LB CD	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Doppelwort aus der aufgeschlagenen Kachel in den Akku 1

Fortsetzung:

Operation	Parameter	Beschreibung
TB CB	-32768 bis +32767	Transferiere den Inhalt des Akku 1-LL in das durch BR-Register + Konstante adressierte Byte auf der aufgeschlagenen Kachel
TB CW	-32768 bis +32767	Transferiere den Inhalt des Akku 1-L in das durch BR-Register + Konstante adressierte Wort auf der aufgeschlagenen Kachel
TB CD	-32768 bis +32767	Transferiere den Inhalt des Akku 1 in das durch BR-Register + Konstante adressierte Doppelwort auf der aufgeschlagenen Kachel

Die Absolutadresse muß liegen

- zwischen F400H und FBFFH (bei LB CB, TB CB),
- zwischen F400H und FBFEH (bei LB CW, TB CW),
- zwischen F400H und FBFCH (bei LB CD, TB CD).

Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den OB 31 auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

## Lade- und Transferoperationen für wortweise organisierte Kacheln

Operation	Parameter	Beschreibung
LW CW	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Wort aus der aufgeschlagenen Kachel in den Akku 1-L
LW CD	-32768 bis +32767	Lade das durch BR-Register + Konstante adressierte Doppelwort aus der aufgeschlagenen Kachel in den Akku 1
TW CW	-32768 bis +32767	Transferiere den Inhalt des Akku 1-L in das durch BR-Register + Konstante adressierte Wort auf der aufgeschlagenen Kachel
TW CD	-32768 bis +32767	Transferiere den Inhalt des Akku 1 in das durch BR-Register + Konstante adressierte Doppelwort auf der aufgeschlagenen Kachel

Die Absolutadresse muß zwischen F400H und FBFFH (bei LW CW, TW CW) bzw. F400H und FBFEH (bei LW CD, TW CD) liegen. Ist dies nicht der Fall, erkennt der Prozessor einen Laufzeitfehler und ruft den **OB 31** auf. Im Akku 1 sind Fehlerkennungen hinterlegt, die den aufgetretenen Fehler näher erläutern (siehe Kapitel 5.6.2 "Sonstige Laufzeitfehler").

## **10 Mehrprozessorbetrieb**

### **10.1 Hinweise**

Für komplexe Automatisierungsaufgaben bietet Ihnen das AG 135U den großen Vorteil, daß Sie im Zentralgerät bis zu vier Prozessoren (sog. CPUs) gleichzeitig betreiben können. Jeder Prozessor bearbeitet dabei unabhängig von den anderen Prozessoren sein Programm. Folgende Prozessoren können Sie im Mehrprozessorbetrieb in unterschiedlichen Kombinationen betreiben:

- S-Prozessor, besonders geeignet zum Steuern (binäre Steuerungsaufgaben), Überwachen und Melden; belegt 1 Steckplatz.
- R-Prozessor, besonders geeignet zum Rechnen und Regeln (digitale Steuerungsaufgaben), zur Kommunikation, zum Überwachen und Melden; belegt 1 Steckplatz.
- M-Prozessor, vielseitig verwendbar, zur Programmierung in höheren Programmiersprachen; belegt 1 Steckplatz.
- CPU 928, universell einsetzbar, besonders geeignet zum Steuern, Rechnen und Regeln (binäre und digitale Steuerungsaufgaben); belegt zwei Steckplätze.

#### **Mehrprozessorbetrieb - wann und wie ?**

- Wenn Ihr Anwenderprogramm zu umfangreich für einen Prozessor ist und Speicherplatz knapp wird, so verteilen Sie Ihr Programm auf mehrere Prozessoren.
- Wenn ein bestimmter Teil Ihrer Anlage besonders schnell bearbeitet werden soll, so trennen Sie den betreffenden Programnteil aus dem Gesamtprogramm heraus und lassen diesen von einem eigenen 'schnellen' Prozessor bearbeiten.
- Wenn Ihre Anlage aus mehreren Teilen besteht, die gut voneinander abzugrenzen und damit relativ eigenständig zu steuern bzw. regeln sind, so lassen Sie Anlagenteil 1 von Prozessor 1, Anlagenteil 2 von Prozessor 2 usw. bearbeiten.

Beachten Sie zum Mehrprozessorbetrieb die Beschreibung "Mehrprozessorbetrieb im AG 135U" (C79000-B8500-C500-01). Sie befindet sich in Register 7 dieses Gerätehandbuches und erklärt das schrittweise Vorgehen bei der Inbetriebnahme Ihres Mehrprozessorgerätes, gibt Hinweise zur Bedienung, beschreibt einige typische Fehlerbilder und verweist auf mögliche Fehlerursachen.

**WICHTIG!**

Sie befinden sich im Mehrprozessorbetrieb, sobald Sie im Zentralgerät einen *Koordinator* (= *KOR*) gesteckt haben, unabhängig davon, ob Sie ihn zusammen mit nur einem oder mehreren Prozessoren betreiben!

Der Koordinator hat die Aufgabe, den Datenaustausch zwischen den einzelnen Prozessoren zu koordinieren. Dazu muß er wissen, welche Steckplätze er 'versorgen' soll, d.h., wievielen Prozessoren er reihum Zeitabschnitte zuteilen muß, in denen sie auf den S5-Bus zugreifen können.

**WICHTIG!**

Stellen Sie deshalb auf dem Koordinator die Anzahl der zu versorgenden Steckplätze ein (siehe Betriebsanleitung des Koordinators)! Beachten Sie bei dieser Einstellung, ob die verwendeten Prozessoren einfach-breit (z.B. R-Prozessor) oder doppelt-breit (z.B. CPU 928) sind.

Wenn Sie beispielsweise auf dem KOR die Anzahl '3' eingestellt haben, versorgt der KOR die ersten drei Steckplätze rechts vom Koordinator (Nr. 17, Nr. 19 und Nr. 27, siehe Betriebsanleitung der CPU 928).

Der vierte Steckplatz (Nr. 35) wird in diesem Fall vom KOR nicht versorgt. Ein eventuell dort gesteckter Prozessor kann nicht auf den S5-Bus zugreifen.

**Wichtig!**

**Koordinator und Prozessoren müssen lückenlos gesteckt werden!**

Sobald Sie sich im Mehrprozessorbetrieb befinden, müssen Sie außerdem für jeden beteiligten Prozessor den Datenbaustein DB 1 programmieren. Dieser enthält eine Liste der digitalen Ein- und Ausgänge sowie der Koppelmerkerein- und -ausgänge, die dem jeweiligen Prozessor zugeordnet sind (siehe Kapitel 10.3.1)!

## 10.2 Datenaustausch zwischen den Prozessoren

Für den zyklischen Austausch binärer Daten zwischen den Prozessoren oder zwischen Prozessor und Kommunikationsprozessoren stehen Ihnen die **'Koppelmerker'** (siehe Kapitel 10.2.1) zur Verfügung.

Beim Austausch größerer Datenmengen (z.B. ganzer Datenbausteine) zwischen R- und M-Prozessor und CPU 928 werden Sie unterstützt durch die **'Sonderfunktionen für die Mehrprozessorkommunikation'** OB 200 bis OB 205 (siehe Kapitel 10.2.2).

Zur Kommunikation mit Intelligenten Peripheriebaugruppen (IPs) und mit Kommunikationsprozessoren (CPs) stehen Ihnen die **'Handtierungsbausteine'** zur Verfügung (siehe Kapitel 6.9).

Wollen Sie längere Datenblöcke übertragen und sichergehen, daß die anderen Prozessoren diesen Transfer nicht unterbrechen, so können Sie dies softwaremäßig mit Hilfe der **'Semaphoren'** (siehe Kapitel 3.2.2) realisieren.

### 10.2.1 Koppelmerker

Für den zyklischen Austausch binärer Daten stehen Ihnen die 'Koppelmerker' zur Verfügung. Diese dienen in erster Linie zum byteweisen Übertragen von Informationen.

Dieser Datentransfer kann erfolgen zwischen

Prozessor(en)  $\longleftrightarrow$  Prozessor(en)

Prozessor(en)  $\longleftrightarrow$  Kommunikationsprozessor(en)

Das Systemprogramm überträgt die Koppelmerker einmal pro Zyklus. Bei einem Datentransfer zwischen Prozessoren werden die Koppelmerker physikalisch auf dem Koordinator zwischengespeichert.

Koppelmerker sind Merkerbytes, die transferiert werden. Sie werden für jeden Prozessor im Datenbaustein DB 1 als Ein- oder Ausgangskoppelmerker definiert.

Haben Sie z.B. das Merkerbyte 50 auf dem Prozessor 1 als Koppelmerker**ausgang** definiert, so wird dessen Signalzustand zyklisch über den Koordinator zu dem Prozessor übertragen, auf dem das Merkerbyte 50 als Koppelmerker**eingang** definiert ist.

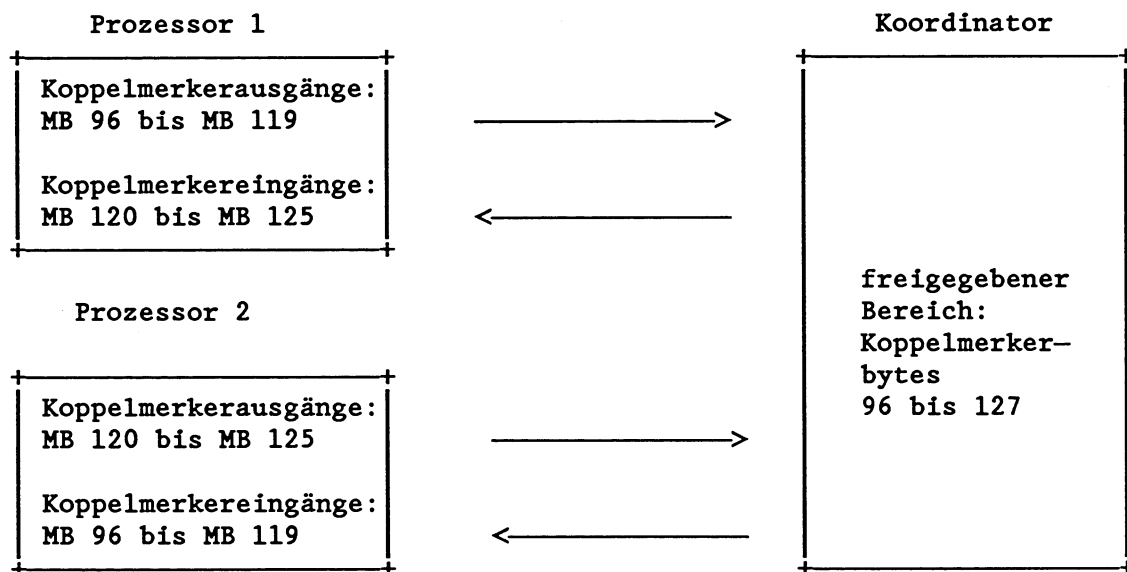
Der Speicherbereich für die Koppelmerker auf dem Koordinator und den Kommunikationsprozessoren umfaßt die Adressen **F200H bis F2FFH**). Auf einem Prozessor/Kommunikationsprozessor stehen Ihnen 256 Koppelmerkerbytes zur Verfügung.

## Datenaustausch zwischen Prozessoren

Der Datentransfer zwischen einzelnen Prozessoren (CPU 928, R-, S- und M-Prozessor) erfolgt über den Koordinator. Die Prozessoren lesen ihre im DB 1 als Einganskoppelmerker definierten Merkerbytes vom KOR bzw. schreiben ihre als Ausgangskoppelmerker definierten Merkerbytes zum KOR.

Auf dem Koordinator geben Sie die benötigte Anzahl der Koppelmerker frei: Die max. 256 Koppelmerker können Sie durch Brückeneinstellung in Bereiche von 32 Bytes einteilen (8 Bereiche). Beachten Sie hierzu die Betriebsanleitung des verwendeten Koordinators.

Beispiel:



### WICHTIG!

- Als Koppelmerker dürfen Sie nur diejenigen Merkerbytes angeben, die auf dem Koordinator freigegeben sind!
- Haben Sie auf einem oder mehreren Prozessoren ein bestimmtes Merkerbyte als Koppelmerkereingang definiert, so muß es auf einem anderen Prozessor als KoppelmerkerAusgang definiert sein. Und: Ein Merkerbyte darf nur auf einem Prozessor als KoppelmerkerAusgang gekennzeichnet sein; Sie können es jedoch auf z.B. drei weiteren Prozessoren als Koppelmerkereingang definieren!
- Die auf einem Prozessor nicht als Koppelmerker definierten Merkerbytes können wie "normale" Merker verwendet werden!
- Geben Sie nur die Anzahl an Koppelmerkern an, die Sie tatsächlich benötigen: Je kleiner die Anzahl der Koppelmerker, um so kürzer die Übertragungszeit!

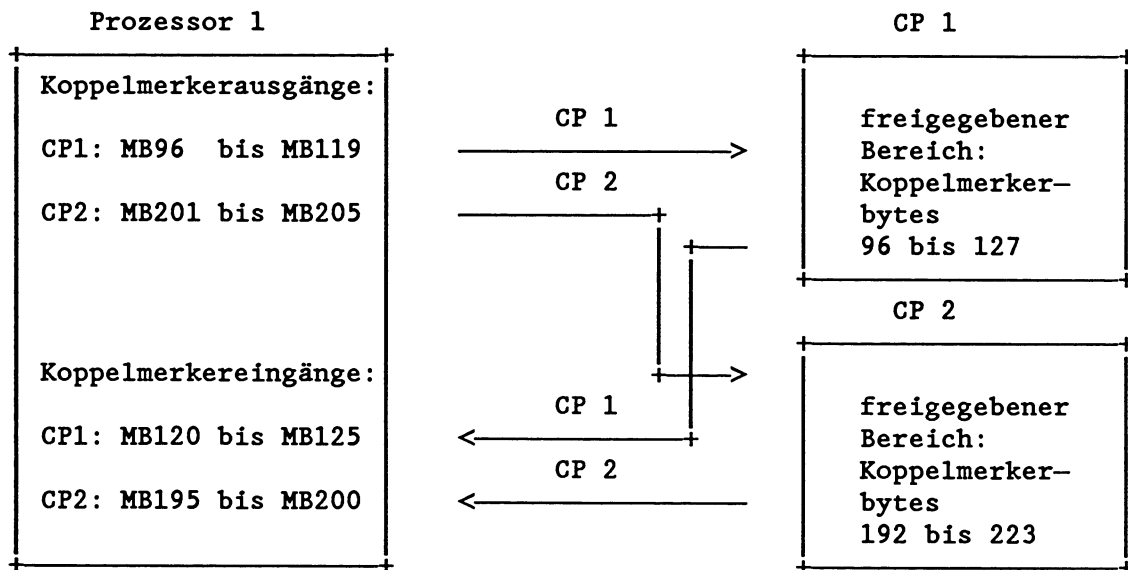


## Datenaustausch zwischen Prozessoren und Kommunikationsprozessoren

Sollen Daten zwischen *einem* Prozessor und *einem* Kommunikationsprozessor übertragen werden, müssen Sie die benötigte Anzahl der Koppelmerker auf dem Kommunikationsprozessor (CP) freigeben. Auch dort stehen Ihnen 256 Bytes zur Verfügung, die in Bereiche von 32 Bytes eingeteilt werden können.

Wenn von *einem* Prozessor Daten zu *mehreren* Kommunikationsprozessoren übertragen werden sollen, dürfen sich die auf den Kommunikationsprozessoren freigegebenen Bereiche nicht überschneiden, damit Adressen nicht doppelt belegt werden.

Beispiel:



Wenn Sie Koppelmerker *gleichzeitig* auf dem Koordinator und einem oder mehreren Kommunikationsprozessoren benutzen wollen, so müssen Sie ebenfalls eine Doppeladressierung vermeiden: Teilen Sie die Koppelmerker auf dem KOR und den CPs in Bereiche von je 32 Bytes ein; diejenigen Koppelmerkerbytes, die Sie auf dem Kommunikationsprozessor verwenden, blenden Sie auf dem Koordinator durch Entfernen von Brücken aus (siehe Betriebsanleitung des verwendeten Koordinators).

Auch hier gilt, daß ein bestimmtes Merkerbyte nur auf einem Prozessor als KoppelmerkerAusgang definiert werden darf. Hingegen kann ein bestimmtes Merkerbyte auf mehreren Prozessoren als Eingangskoppelmerker gekennzeichnet sein.

## **Übertragung der Koppelmerker im Mehrprozessorbetrieb**

Die im DB 1 angegebenen Koppelmerker werden dann übertragen, wenn der Prozessor vom Koordinator das Signal erhält, daß er auf den Peripheriebus zugreifen darf.

Wollen mehrere Prozessoren gleichzeitig auf den Bus zugreifen, so erteilt der Koordinator reihum jedem Prozessor das Busfreigabesignal. Dabei darf ein Prozessor jeweils nur ein Byte übertragen. Durch diese verzahnte Übertragung kann es vorkommen, daß zusammengehörige Koppelmerkerinformationen auseinandergerissen werden und anschließend mit falschen Werten gearbeitet wird.

Wenn Informationen übertragen werden sollen, die mehr als ein Byte umfassen, verwenden Sie den Sonderfunktions-Organisationsbaustein OB 224: Durch Aufruf des OB 224 erreichen Sie eine **blockweise** Übertragung aller im DB 1 angegebenen Koppelmerker. Solange ein Prozessor mit der Koppelmerkerübertragung beschäftigt ist, kann er von einem anderen Prozessor nicht unterbrochen werden. Da der nächste Prozessor mit seiner Übertragung warten muß, wird die zyklische Programmbearbeitung solange verzögert. Beachten Sie, daß bei Verwendung des OB 224 unter Umständen die Zykluszeit erheblich verlängert wird!  
(siehe Kapitel 6.8.6)

### **10.2.2 Mehrprozessorkommunikation**

Die Sonderfunktions-Organisationsbausteine OB 200 bis OB 205 ermöglichen im Mehrprozessorbetrieb den Austausch größerer Datenmengen (z.B. ganzer Datenbausteine) zwischen R- und M-Prozessor und CPU 928, wobei die Daten im Koordinator KOR C zwischengespeichert werden.

Der OB 200 richtet im Koordinator KOR C den Zwischenspeicher ein, in dem die zu übertragenden Datenblöcke vorübergehend zwischengespeichert werden müssen.

Der OB 202 übergibt einen Datenblock in den Zwischenspeicher des KOR C und gibt an, wieviele Datenblöcke noch gesendet werden können.

Der OB 203 ermittelt die Anzahl der freien Speicherblöcke im Zwischenspeicher des Koordinators KOR C.

OB 204 übernimmt einen Datenblock vom Zwischenspeicher des KOR C und zeigt an, wieviele Datenblöcke noch empfangen werden können.

OB 205 ermittelt die Anzahl belegter Speicherblöcke im Zwischenspeicher des KOR C.

Eine detaillierte Bedienungsanleitung zu diesen Sonderfunktions-Organisationsbausteinen mit dem Titel "SIMATIC S5 Automatisierungsgerät AG 135U, Mehrprozessorkommunikation" befindet sich in Register 8 dieses Gerätehandbuchs.

### **10.2.3 Zusammenhängende Datenblöcke "geschützt" übertragen**

Wollen Sie längere Datenblöcke übertragen und sichergehen, daß die anderen Prozessoren diesen Transfer nicht unterbrechen, so können Sie dies softwaremäßig mit Hilfe der 'Semaphoren' (siehe Kapitel 3.2.2) realisieren.

### 10.3 Peripheriezuteilung

Der Peripheriebereich jedes Prozessors umfaßt die Adressen **F000H bis FFFFH**. In diesem Bereich werden die Peripheriebaugruppen adressiert, hier liegen die Koppelmerker und der Kachelbereich. Auf diesen Peripheriebereich können alle Prozessoren lesend und schreibend zugreifen. Der Koordinator koordiniert die Zugriffe der einzelnen Prozessoren auf diesen Peripheriebereich.

#### 10.3.1 Datenbaustein DB 1

Im Mehrprozessorbetrieb müssen Sie für jeden Prozessor den Datenbaustein DB 1 programmieren. Dadurch legen Sie fest, mit welchen Ein- und Ausgängen, mit welchen Koppelmerkerein- und Koppelmerkerausgängen der jeweilige Prozessor arbeiten soll.

##### **WICHTIG!**

**Nur die im DB 1 definierten Ein- und Ausgänge werden bei der Aktualisierung des Prozeßabbilds berücksichtigt!**

##### **Eingabe/Änderung des DB 1**

1. On-line über das Programmiergerät im Stoppzustand des Prozessors, wenn der Prozessor mit einem Anwender-RAM bestückt ist.
2. Über die Programmierung des Anwender-EPROMs.

##### **WICHTIG!**

**Der eingegebene bzw. geänderte DB 1 wird nur über die Anlaufart "Neustart" vom Prozessor übernommen!**

**So programmieren Sie den DB 1 maskenunterstützt über Softkeys:**

**DB 1:**

DIGITALE EINGAENGE	,	0,	1,	2,	3,	7,	10,	,	,	,	,
	,	,	,	,	,	,	,	,	,	,	,
DIGITALE AUSGAENGE	,	0,	2,	4,	12,	,	,	,	,	,	,
	,	,	,	,	,	,	,	,	,	,	,
KOPPELMERKER-EINGAENGE	,	50,	51,	60,	,	,	,	,	,	,	,
	,	,	,	,	,	,	,	,	,	,	,
KOPPELMERKER-AUSGAENGE	,	70,	72,	100,	,	,	,	,	,	,	,
	,	,	,	,	,	,	,	,	,	,	,
ZEITENBLOCKLAENGE	,	128,									

(Beachten Sie, daß Sie einen mit "S5-DOS" per Maske eingegebenen DB 1 unter bestimmten Umständen nicht fehlerfrei mit der PG-Software "Studos" auslesen können.)

**So erstellen Sie den DB 1 manuell:**

- Die Datenwörter 0, 1 und 2 müssen die DB1-Anfangskennung enthalten. Sie sind deshalb fest zu belegen mit

DW 0: KH = 4D41  
DW 1: KH = 534B  
DW 2: KH = 3031

- Ab Datenwort 3 werden die einzelnen Operandenbereiche angegeben.

Für jeden Operandenbereich geben Sie eine bestimmte Kennung ein: Die möglichen Kennwörter sind:

Kennwort für digitale Eingänge	KH = DE00
Kennwort für digitale Ausgänge	KH = DA00
Kennwort für Eingangskoppelmerker	KH = CE00
Kennwort für Ausgangskoppelmerker	KH = CA00
Kennwort für Zeitenblocklänge	KH = BB00

Anschließend an das Kennwort listen Sie die Nummern der verwendeten Ein- und Ausgänge und die gewünschte Zeitenblocklänge im Festpunktformat auf.

- Als DB-1-Endekennung enthält das letzte Datenwort den Wert KH = EEEE.

#### **WICHTIG!**

- Die Reihenfolge der Einträge ist beliebig. Beachten Sie dabei, daß das Prozeßabbild der Ein- und Ausgänge in genau der Reihenfolge aktualisiert wird, in der sie im DB 1 eingetragen sind.
- Mehrfacheinträge gleicher Bytes, z.B. für Testzwecke, sind möglich. Auch hier beachten Sie bitte, daß das Prozeßabbild mehrfach eingetragener Bytes mehrfach aktualisiert wird.
- Nach dem letzten Eintrag im DB 1 müssen Sie als Endekennung KH = EEEE eingeben!

Ein Beispiel für das Erstellen des DB 1 finden Sie auf der folgenden Seite.

## Beispiel für den DB 1

0 :	KH = 4D41;	DW 0-2:
1 :	KH = 534B;	Anfangskennung
2 :	KH = 3031;	für DB 1
3 :	KH = DE00;	Kennwort für digitale Eingänge
4 :	KF = +00000;	Eingangsbyte 0
5 :	KF = +00001;	Eingangsbyte 1
6 :	KF = +00002;	Eingangsbyte 2
7 :	KF = +00003;	Eingangsbyte 3
8 :	KF = +00007;	.
9 :	KF = +00010;	.
10 :	KH = DA00;	Kennwort für digitale Ausgänge
11 :	KF = +00000;	Ausgangsbyte 0
12 :	KF = +00002;	Ausgangsbyte 2
13 :	KF = +00004;	.
14 :	KF = +00012;	.
15 :	KH = CE00;	Kennwort für Koppelmerkereingänge
16 :	KF = +00050;	Merkerbyte 50
17 :	KF = +00051;	.
18 :	KF = +00060;	.
19 :	KH = CA00;	Kennwort für Koppelmerkerausgänge
20 :	KF = +00070;	Merkerbyte 70
21 :	KF = +00072;	.
22 :	KF = +00100;	.
23 :	KH = BB00;	Kennwort für Zeitenblocklänge *
24 :	KF = +00128;	Timer 0 bis Timer 127
25 :	KH = EEEE;	Endekennung

\* Durch Eintrag einer Zeitenblocklänge im DB 1 können Sie angeben, wieviele Zeitzellen das Systemprogramm zyklisch aktualisieren soll. Dieses Systemverhalten sollte im DX 0 parametrisiert werden, siehe Kapitel "Erweiterter Datenbaustein DX 0".

Bei Neustart wird der DB 1 vom Systemprogramm übernommen. Es überprüft dabei, ob die im DB 1 angegebenen Ein- und Ausgänge bzw. Koppelmerker auf entsprechenden Baugruppen quittieren. Falls nicht, geht der Prozessor mit DB-1-Fehler in den Stoppzustand mit langsamem Blinken der STOP-Led. Ihr Anwenderprogramm wird nicht bearbeitet.

Sobald Sie einen DB 1 programmiert haben und dieser durch die Anlaufart "Neustart" vom Prozessor übernommen worden ist, gelten folgende Regeln:

- Zugriffe auf Peripheriebaugruppen über das Prozeßabbild sind nur für die im DB 1 angegebenen Ein- und Ausgänge zulässig (Befehle L/T EB, EW, ED, AB, AW, AD und Verknüpfungsoperationen mit Ein- und Ausgängen).
- Direktes Laden von Peripheriebytes unter Umgehung des Prozeßabbildes mit den Befehlen L PB/PY, PW, QB, QW sind für alle quittierenden Eingänge - unabhängig von einem Eintrag im DB 1 - möglich.
- Direkter Transfer (T PB/PY, PW, QB, QW) auf Bytes von 0 bis 127 ist nur für die im DB 1 angegebenen Ausgänge möglich, da beim Direkttransfer zusätzlich das Prozeßabbild beschrieben wird. Direkter Transfer auf Byteadressen > 127 ist unabhängig von einem Eintrag im DB 1 möglich.

## 10.4 Anlauf im Mehrprozessorbetrieb

So starten Sie den Koordinator im Mehrprozessorbetrieb:

- Die Betriebsartenschalter aller gesteckten Prozessoren stehen auf 'RUN'. Der Betriebsartenschalter des Koordinators steht auf 'STOP'.
- Betätigen Sie den Betriebsartenschalter am Koordinator von 'STOP' nach 'RUN'.

(Der Anlauf des Automatisierungsgerätes im Mehrprozessorbetrieb allein durch das Starten des Koordinators ist nur dann möglich, wenn der Stoppzustand auch tatsächlich vom Koordinator verursacht worden ist.)

oder:

- Die Betriebsartenschalter aller gesteckten Prozessoren und der des Koordinators stehen auf 'RUN'.
- Durch die PG-Funktion "AG-Start" starten Sie den Prozessor, der den Stoppzustand verursacht hat, in der erwünschten Anlaufart.

Die Anlaufart der einzelnen Prozessoren richtet sich nun danach, ob und wie sie zwischenzeitlich im Stoppzustand bedient worden sind. Es ist damit möglich, daß einzelne Prozessoren einen manuellen Wiederanlauf, andere einen Neustart durchführen.

Sind die Prozessoren in der Zwischenzeit *nicht* bedient worden, so führen sie einen manuellen Wiederanlauf (CPU 928 und R-Prozessor) oder einen manuellen Neustart ohne Rücksetzen (S-Prozessor) durch.

### WICHTIG!

Durch die unterschiedlichen Anlaufarten können, falls das Automatisierungsgerät vorher im Zyklus war, über die Koppelmerker falsche Signalzustände von einem Prozessor an einen anderen weitergegeben werden. Dies verhindern Sie durch entsprechende Programmierung der Anlauf-OBs 20, 21 und 22.

Bei Spannungsausfall und anschließender Spannungswiederkehr wird der Koordinator automatisch mitgestartet. Alle S-Prozessoren führen in diesem Fall einen automatischen Neustart mit Gedächtnis, alle CPUs 928 und R-Prozessoren einen automatischen Wiederanlauf bzw. einen automatischen Neustart durch, je nach Voreinstellung im DX 0.

Der Anlauf der einzelnen Prozessoren ist im Mehrprozessorbetrieb zeitlich **synchronisiert**, d.h., die einzelnen Prozessoren warten solange, bis alle anderen ihren Anlauf beendet haben, und beginnen dann gleichzeitig ihren zyklischen Betrieb. Bei der CPU 928 und beim R-Prozessor können Sie diese Anlaufsynchronisation durch Einstellung im DX 0 abwählen.

## 10.5 Testbetrieb

So lösen Sie den Testbetrieb aus:

- Die Funktion "Testbetrieb" muß am Koordinator freigegeben sein.
- Betätigen Sie den Betriebsartenschalter am KOR von 'STOP' auf 'TEST'. Die BASP-Led erlischt.
- An den Prozessoren, die in RUN gehen sollen, wählen Sie die Anlaufart.

### Besonderheiten im Testbetrieb

Im Testbetrieb können Sie die gesteckten Prozessoren einzeln in Betrieb nehmen oder sie beliebig miteinander kombinieren. Prozessoren, die sich im Stoppzustand befinden, können dabei nicht mehr das gesamte AG blockieren.

Der Anlauf der einzelnen Prozessoren ist im Testbetrieb *nicht* synchronisiert. Je nach Länge der Anlauf-OBs 20, 21 oder 22 beginnen die Prozessoren zu unterschiedlichen Zeitpunkten ihren zyklischen Betrieb.

Tritt bei einem Prozessor ein Fehler auf, so geht im Testbetrieb nur dieser in den Stoppzustand über. Die anderen Prozessoren werden durch den Fehler nicht beeinflusst.

### WICHTIG!

Im Testbetrieb wird die Ausgabe des Signals BASP bei allen Prozessoren unterdrückt. Bei Auftreten eines Fehlers werden die digitalen Peripherieausgänge *nicht* gesperrt (Ausnahmen siehe oben).

### WICHTIG!

Schalten Sie nach abgeschlossener Inbetriebnahme den Testbetrieb durch Einstellung auf dem Koordinator unbedingt inaktiv! So verhindern Sie eine Fehlbedienung, die unter Umständen zu gefährlichen Anlagenzuständen führen kann!



**Zusammenfassung: So nehmen Sie Ihr Mehrprozessorgerät in Betrieb**

- Anzahl der Prozessoren auf dem Koordinator einstellen.  
Koppelmerker auf dem Koordinator freigeben.
- Prozessoren lückenlos in das Zentralgerät stecken.
- Versorgungsspannung einschalten.
- Betriebsartenschalter des Koordinators in 'STOP'-Stellung bringen.
- Alle gesteckten Prozessoren urlöschen.
- Anwenderprogramme in die Prozessoren laden.
- Bei allen Prozessoren Neustart durchführen.
- Betriebsartenschalter des Koordinators in 'RUN'- oder 'TEST'-Stellung bringen.

## 11 Testhilfsmittel: On-line-Funktionen

Ein wichtiges Hilfsmittel zum Testen Ihres Anwenderprogramms sind die On-line-Funktionen. Zur Bedienung des Programmiergerätes und zur Anwendung dieser Funktionen finden Sie detaillierte Hinweise in Ihrem Programmiergeräte-Handbuch. Im vorliegenden Kapitel werden einige Besonderheiten von On-line-Funktionen bei der CPU 928 beschrieben.

Im Automatisierungsgerät werden die On-line-Funktionen an definierten Punkten ausgeführt. Hierbei gibt es Punkte im Systemprogramm (= Systemkontrollpunkte) und Punkte im Anwenderprogramm (= Anwenderkontrollpunkte).

Durch die Tätigkeiten an einem Kontrollpunkt wird dort die Programmbearbeitung zusätzlich bis maximal 5 ms belastet.

### Systemkontrollpunkte

Im Betriebszustand STOP existiert der Systemkontrollpunkt Stop, der regelmäßig aufgerufen wird.

Im Betriebszustand RUN wird der Systemkontrollpunkt Zyklus am Ende der Programmbearbeitungsebene ZYKLUS vor der Prozeßabbildaktualisierung aufgerufen.

Befindet sich der Prozessor im WARTEZUSTAND, wird dort regelmäßig der Systemkontrollpunkt Wartezustand aufgerufen.

Zusätzlich gibt es einen zeitbedingten Systemkontrollpunkt Time-out. Dieser wird nur dann aufgerufen, wenn keiner der anderen drei Systemkontrollpunkte innerhalb 250ms erreicht wurde. Dieser Systemkontrollpunkt kann also während der Programmbearbeitung eingeschachtelt werden. Dies kann z.B. bei einer Dauerschleife im Anwenderprogramm auftreten oder bei Zyklen, die länger als 250ms dauern.

Die Bearbeitung einer On-line-Funktion an einem Systemkontrollpunkt wird nach maximal 5ms unterbrochen und beim nächsten Systemkontrollpunkt fortgesetzt (siehe auch Tabelle 11.11).

### Anwenderkontrollpunkte

Bei den Testfunktionen 'STATUS' und 'BEARBEITUNGSKONTROLLE' werden Anwenderkontrollpunkte verwendet. Ein Anwenderkontrollpunkt wird aufgerufen, wenn ein Befehl ausgeführt ist, der vom PG markiert ist.

### Betriebszustand WARTEZUSTAND

Bisher sind Ihnen die Betriebszustände STOP, ANLAUF und RUN bekannt. Bei der On-line-Funktion 'BEARBEITUNGSKONTROLLE' nimmt der Prozessor einen weiteren Betriebszustand ein: den WARTEZUSTAND. Wenn der Prozessor sich im Wartezustand befindet, können noch weitere On-line-Funktionen aufgerufen werden.

## **Eigenschaften des Wartezustands**

- Im Wartezustand findet keine Anwenderprogrammbearbeitung statt.
- LEDs auf der Frontplatte:

RUN-LED:	aus
STOP-LED:	aus
BASP-LED:	an
- Alle Zeiten sind 'eingefroren', d.h., es laufen keine Timer (die Zeitzellen werden nicht verändert). Ebenso bleiben alle Systemzeiten stehen, wie für die Regelung und für die zeitgesteuerte Bearbeitung.

Nach Verlassen des Wartezustands (siehe Kapitel 10.3) laufen die Timer weiter.

- Unterbrechungsursachen wie z.B. PEU, BAU, MPSTP oder Stoppschalter werden im Wartezustand registriert, aber es erfolgt keine Reaktion darauf.

Wenn im Wartezustand Unterbrechungsursachen registriert worden sind, so werden die dazugehörigen Programmbearbeitungsebenen unmittelbar nach Verlassen des Wartezustands aufgerufen.

Tritt NAU auf, wird der Wartezustand verlassen und die Online-Funktion 'BEARBEITUNGSKONTROLLE' abgebrochen. Nach Netzein ist in den Steuerbits 'BARBEND' angekreuzt. Der Stoppzustand kann nur mit Neustart verlassen werden.

## 11.1 On-line-Funktion 'STATUS VARIABLEN'

Mit der On-line-Funktion 'STATUS VARIABLEN' können Sie sich die aktuellen Signalzustände bestimmter Operanden (Prozeßvariablen) ausgeben lassen. Die Funktion aktiviert Systemkontrollpunkte im Zyklus, im Stopp- und im Wartezustand. Wenn der Systemkontrollpunkt erreicht ist, wird der *zu diesem Zeitpunkt aktuelle* Signalzustand der gewünschten Prozeßvariablen ausgegeben. Sie können alle Prozeßvariablen angeben. Im Bereich des Prozeßabbaus wird kein ADF ausgelöst.

### Ablauf der Funktion während der Programmbearbeitung:

Läuft die Funktion im Betriebszustand ANLAUF oder RUN, wird die Programmbearbeitung solange fortgesetzt, bis der Systemkontrollpunkt 'Zyklus' erreicht ist. Dann werden die Signalzustände der Operanden am Zyklusende abgefragt und ausgegeben. Eingänge werden aus dem **Prozeßabbild** gelesen. Solange die Funktion nicht abgebrochen wird, werden bei laufender Programmbearbeitung die Signalzustände aktualisiert. Die Signalzustände werden dabei *nicht an jedem* Systemkontrollpunkt abgefragt.

Wird der Systemkontrollpunkt 'Zyklus' *nicht* erreicht, erfolgt *keine Ausgabe* der Signalzustände (z.B. bei einer Dauerschleife im Anwenderprogramm)!

### Ablauf der Funktion im Stoppzustand:

Wenn die Funktion 'STATUS VARIABLEN' im STOP läuft, werden die Signalzustände der Operanden ausgegeben, wie sie am Systemkontrollpunkt 'Stoppzustand' vorliegen. Wichtig ist dabei, daß die **Eingänge direkt** von der Peripheriebaugruppe abgefragt und ausgegeben werden. Dadurch läßt sich zum Beispiel testen, ob ein Peripherie-Eingangssignal tatsächlich zum Prozessor gelangt. Sie können auch im Mehrprozessorbetrieb *alle* Eingänge angeben unabhängig von der Zuteilung im DB 1. Die Ausgänge werden vom Prozeßabbild gelesen.

### Ablauf der Funktion im Wartezustand:

Die Funktion 'STATUS VARIABLEN' können Sie auch aufrufen, wenn der Prozessor sich mit der Funktion 'BEARBEITUNGSKONTROLLE' im Wartezustand befindet. Am Systemkontrollpunkt 'Wartezustand' werden die Signalzustände der Operanden abgefragt und ausgegeben. Wie im Stoppzustand werden dabei die Eingänge *direkt*, die Ausgänge aus dem **Prozeßabbild** gelesen.

Wenn der Prozessor von einem Betriebszustand in den anderen wechselt (z.B. RUN -> STOP -> MANUELLER WIEDERANLAUF), bleibt die Funktion weiterhin aufgerufen. Beendet wird 'STATUS VARIABLEN' durch Betätigen der Abbruchtaste am Programmiergerät.

Hinweis: Die Variablen werden *nicht in jedem folgenden Durchlauf des Zyklus* ausgegeben.

## 11.2 On-line-Funktion 'STATUS'

Mit Hilfe der On-line-Funktion 'STATUS' können Sie an beliebiger Stelle im Anwenderprogramm zusammenhängende Befehlsfolgen in einem Baustein testen.

Zu jedem ausgeführten Befehl im Baustein werden die aktuellen Werte der Operanden, die Akku-Inhalte, das VKE etc. am Programmiergerät ausgegeben. Auch die Parametrierung von Funktionsbausteinen kann auf diese Weise getestet werden: Angezeigt werden die aktuellen Werte der Aktualoperanden.

### Funktion aufrufen und Haltepunkt vorgeben

Wenn Sie die Funktion 'STATUS' am PG aufrufen und Bausteinart und Bausteinnummer (eventuell mit Schachtelreihenfolge und Suchbegriff) des zu testenden Bausteins eingeben, so geben Sie damit einen sog. Haltepunkt vor.

Bei Aufruf der Funktion während der Programmbearbeitung im ANLAUF oder im RUN wird die Programmbearbeitung solange fortgesetzt, bis der durch den vorgebenen Haltepunkt markierte Befehl in der richtigen Schachtelfolge erreicht ist. Danach werden die überwachten Befehle jeweils bis zur Befehlsgrenze ausgeführt und die Ergebnisse der Befehlsbearbeitung am PG ausgegeben.

Die Funktion 'STATUS' läßt sich auch im Stoppzustand aufrufen. Danach ist sowohl ein Neustart als auch ein manueller Wiederanlauf möglich. Der Prozessor bearbeitet daraufhin das Anwenderprogramm bis zum vorgegebenen Haltepunkt. Dann werden die Daten zu der gewünschten Befehlsfolge ausgegeben. Somit eignet sich die Funktion 'STATUS' zum Beispiel auch dazu, das Anwenderprogramm im Anlauf oder im ersten Zyklus zu testen.

Hinweis: Die Ergebnisse der Befehlsbearbeitung werden *nicht in jedem folgenden Durchlauf des Zyklus* ausgegeben.

### Einschachtelungen und Unterbrechungen

Eine durch einen vorgegebenen Haltepunkt markierte Befehlsfolge wird vollständig durchlaufen, auch wenn zwischendurch an einer Befehlsgrenze eine andere Programmbearbeitungsebene (z.B. ein Fehler-OB, ein Prozeß- oder ein Weckalarm) eingeschachtelt und abgearbeitet wird.

Führt in einer eingeschachtelten Programmbearbeitungsebene eine Unterbrechungsursache den Prozessor in Stopp, so werden im Stoppzustand die Daten bis zu demjenigen Befehl ausgegeben, der als letzter vor der Einschachtelung ausgeführt worden ist. Die Daten der restlichen Befehle werden mit '0' aufgefüllt (auch SAZ = 0).

Wenn der Prozessor von einem Betriebszustand in den anderen wechselt (z.B. RUN -> STOP -> MANUELLER WIEDERANLAUF), bleibt die Funktion weiterhin aufgerufen. Beendet wird 'STATUS' durch Betätigen der Abbruchtaste am Programmiergerät.

### 11.3 On-line-Funktion 'BEARBEITUNGSKONTROLLE'

Mit der On-line-Funktion 'BEARBEITUNGSKONTROLLE' können Sie an beliebiger Stelle im Anwenderprogramm einzelne Programmschritte testen. Dazu halten Sie die Programmbearbeitung an und lassen den Prozessor dann einen Befehl nach dem anderen bearbeiten. Zu jedem ausgeführten Befehl werden die aktuellen Signalzustände der Operanden, die Akku-Inhalte, das VKE etc. am Programmiergerät ausgegeben.

#### Funktion aufrufen und 1. Haltepunkt vorgeben

Zum Aufruf der Funktion 'BEARBEITUNGSKONTROLLE' geben Sie Bausteinart und Bausteinnummer (eventuell mit Schachtelreihenfolge) des zu testenden Bausteins an und markieren am PG den ersten Befehl, dessen Daten ausgegeben werden sollen. Damit geben Sie einen ersten Haltepunkt vor. In den Steuerbits wird 'BARB' angekreuzt. Die Befehlsausgabe wird gesperrt (BASP-LED = an).

Hinweis: Wenn Sie am Koordinator Testbetrieb einstellen, wird die Befehlsausgabe nicht gesperrt (BASP-Led = aus). Werden jetzt Befehle bearbeitet, die digitale Peripherie ändern, oder führt der Prozessor die Prozeßabbildaktualisierung durch, geben die Signalformer entsprechende Signale aus.

Wenn Sie den 1. Haltepunkt während der Programmbearbeitung im ANLAUF oder RUN vorgeben, setzt der Prozessor die Programmbearbeitung solange fort, bis der durch den vorgebenen Haltepunkt markierte Befehl erreicht ist. Der Befehl wird bis zur Befehlsgrenze ausgeführt. (Die Befehle BMW und BDW werden einschließlich substituiertem Befehl bearbeitet.)

Anschließend geht der Prozessor in den Wartezustand über. Dort werden die Daten des bearbeiteten Befehls ausgegeben.

#### Aufruf der Funktion im Stop:

Auch im Stoppzustand können Sie die Funktion 'BEARBEITUNGSKONTROLLE' aufrufen und einen ersten Haltepunkt vorgeben. Der Prozessor bleibt weiterhin im Stoppzustand. Sie können jetzt sowohl einen Neustart als auch einen manuellen Wiederanlauf durchführen. Der Prozessor führt die Programmbearbeitung bis zum markierten Befehl aus und verfährt dann wie oben.

## **Funktion fortführen und weiteren Haltepunkt vorgeben**

**Ausgangspunkt:** Der Prozessor befindet sich im Wartezustand.

Um die Funktion 'BEARBEITUNGSKONTROLLE' fortzuführen, haben Sie zwei Möglichkeiten.

### **1. Sie geben einen Folgehaltepunkt vor:**

Der vorgegebene Haltepunkt wird um einen Befehl verschoben. Der Prozessor verläßt den Wartezustand und setzt die Programmbearbeitung um diesen einen Befehl fort. Wenn der Befehl bis zur Befehlsgrenze bearbeitet ist, geht der Prozessor erneut in den Wartezustand über und gibt dort die Daten aus. Wird der Folgebefehl jedoch in einer eingeschachtelten Programmbearbeitungsebene erreicht, setzt der Prozessor die Programmbearbeitung fort. Der Folgehaltepunkt bleibt weiterhin vorgegeben.

**Wichtig:** Im Stoppzustand können Sie nur einen 1. Haltepunkt und keinen Folgehaltepunkt vorgeben!

### **2. Sie geben einen neuen Haltepunkt vor:**

Sie geben am PG einen beliebigen anderen Befehl im gleichen oder in einem anderen Baustein vor. Der Prozessor setzt die Programmbearbeitung fort, bis er den neuen Haltepunkt erreicht. Der Befehl wird bis zur Befehlsgrenze bearbeitet. Dann geht der Prozessor in den Wartezustand über und gibt dort die Daten aus.

Sie können den Prozessor mit Bearbeitungskontrolle auch um einen kompletten Zyklus weiterlaufen lassen (zyklusweise testen). Dafür setzen Sie im Wartezustand den Haltepunkt auf denselben Befehl, wie vorher. Allerdings darf sich der Befehl nicht in einer Programmschleife befinden. In diesem Fall wird die Schleife einmal durchlaufen; es erfolgt keine Programmbearbeitung über die Zyklusgrenze hinweg.

**Hinweise:** Im Wartezustand können Sie andere Funktionen wie 'AUSGABE BUCH', 'STATUS VARIABLEN' oder 'STEUERN VARIABLE' aufrufen. Sobald die Programmbearbeitung nach Verlassen des Wartezustands fortgesetzt wird, laufen die Timer und die Systemzeiten weiter, bis ein Haltepunkt erreicht wird.

### **Haltepunkt zurücknehmen:**

Ist ein vorgegebener Haltepunkt noch nicht erreicht, so haben Sie die Möglichkeit, diesen nachträglich zurückzunehmen, indem Sie am PG die Abbruchtaste betätigen. Der Prozessor geht daraufhin in den Wartezustand über. Danach können Sie einen neuen Haltepunkt vorgeben oder 'BEARBEITUNGSKONTROLLE ENDE' aufrufen.

## Funktion abbrechen

Durch Aufrufen von 'BEARBEITUNGSKONTROLLE ENDE' können Sie während der Programmbearbeitung, im Warte- und im Stoppzustand die Funktion abbrechen. Der Prozessor geht in Stopp (bzw. bleibt im Stopp). Die STOP-LED blinkt langsam. In den Steuerbits wird 'BARBEND' angekreuzt. Anschließend ist ein Neustart erforderlich. Tritt während der Funktion 'BEARBEITUNGSKONTROLLE' ein Schnittstellenfehler (Unterbrechung am PG-Kabel) oder NAU auf, wird die Funktion abgebrochen wie oben.

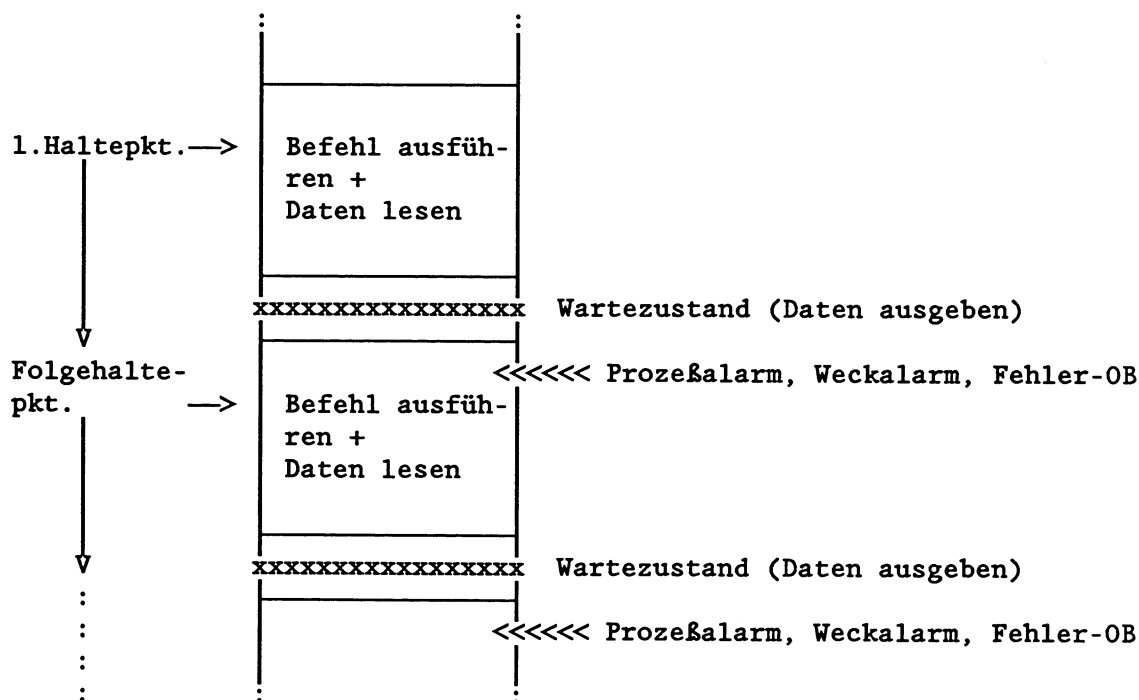
## Einschachtelungen

Bei aufgerufener 'BEARBEITUNGSKONTROLLE' können beim Übergang aus dem Wartezustand andere Programmbearbeitungsebenen eingeschachtelt werden.

Wenn der Befehl am Haltepunkt bearbeitet ist und an dieser Stelle eine andere Programmbearbeitungsebene aufgerufen ist (z.B. ein Fehler-OB, ein Prozeß- oder ein Weckalarm), so wird diese erst eingeschachtelt und vollständig abgearbeitet, wenn der Wartezustand wieder verlassen wird.

### WICHTIG!

Die Daten werden an der Befehlsgrenze gelesen und dort ausgegeben. Alle zugehörigen Einschachtelungen sind noch nicht bearbeitet.





Sind im Wartezustand Anforderungen wie PEU, MP-STP, Stoppschalter usw. aufgetreten, werden diese nur registriert. Sofort nach Verlassen des Wartezustands können diese wirksam werden: Eine Programmbearbeitungsebene wird eingeschachtelt oder eine Unterbrechung führt in den Stoppzustand.

Es gilt die Reihenfolge des Ereignisses. Gleichzeitige Anforderungen werden priorisiert.

**Hinweis:** Wenn der Prozessor im Wartezustand ist und eine Einschachtelung angefordert ist, haben Sie die Möglichkeit, einen Haltepunkt auf einen Befehl der Einschachtelung zu setzen. So können Sie z.B. bei einem Befehl, der einen QVZ auslöst, direkt danach den QVZ-Fehler-OB beobachten.

### **Unterbrechungen**

- Programmbearbeitung (Anlauf/Run) --> Stoppzustand

Treten während der Programmbearbeitung Unterbrechungsursachen auf (z.B. MP-STP, PEU, Stoppschalter, Fehler-OB nicht programmiert etc.) und der vorgegebene Haltepunkt ist noch nicht erreicht, geht der Prozessor sofort in den *Stoppzustand* über. Wenn nun ein Anlauf (Neustart oder Manueller Wiederanlauf) durchgeführt wird, bleibt die Funktion 'BEARBEITUNGSKONTROLLE' weiterhin aufgerufen, der Haltepunkt ist weiterhin vorgegeben.

- Befehlsbearbeitung am Haltepunkt (Anlauf/Run) --> Stoppzustand

Wenn während der Befehlsbearbeitung am Haltepunkt oder Folgehaltepunkt Stopp-Bedingungen auftreten (Stoppschalter, STEP5-Befehl 'STP'), geht der Prozessor unmittelbar nach der Befehlsbearbeitung in *Stopp* und übergibt dort die Daten.

Wird im Stoppzustand kein neuer Haltepunkt vorgegeben, geht der Prozessor nach einem Anlauf in den *Wartezustand* über. 'BEARBEITUNGSKONTROLLE' bleibt weiterhin aufgerufen.

- Wartezustand --> Stoppzustand

Unterbrechungsursachen, die im Wartezustand auftreten (z.B. MP-STP, PEU oder Stoppschalter), oder vom vorausgegangenen Befehl stammen (Fehler, der in *Stop* führt), werden zwar registriert, der Prozessor bleibt jedoch im *Wartezustand*. Erst wenn im Wartezustand ein neuer Haltepunkt vorgegeben wird und der Prozessor den Wartezustand verläßt, bewirken die aufgetretenen Unterbrechungsursachen einen Übergang in den *Stoppzustand*. Der vorgegebene Haltepunkt wird nicht erreicht.

Wird jetzt ein Anlauf (Neustart oder Manueller Wiederanlauf) durchgeführt, ist der Haltepunkt weiterhin vorgegeben.

### **WICHTIG!**

Wird im Wartezustand der Stoppschalter eingelegt, geht der Prozessor erst nach Verlassen des Wartezustands in den Stoppzustand.

### **WICHTIG!**

Führen Unterbrechungsursachen den Prozessor während der 'BEARBEITUNGSKONTROLLE' in den Stoppzustand, bleibt nach einem anschließenden Anlauf die Funktion 'BEARBEITUNGSKONTROLLE' (und ein evtl. vorgegebener Haltepunkt) weiterhin aktiv!

## 11.4 On-line-Funktion 'STEUERN'

Mit Hilfe der Funktion 'STEUERN' können Sie die *Ausgangsbytes* des Automatisierungsgerätes direkt unter Umgehung des Prozeßabbildes manuell auf einen gewünschten Signalzustand einstellen oder nicht quittierende Signalformer (digitale Peripherie 0 bis 127) zu erkennen (Meldung am PG). Sie haben die Möglichkeit, die von den Ausgängen versorgten Prozeßgeräte (Motor, Ventil) direkt zu überprüfen und zu steuern.

### WICHTIG!

**Die Funktion 'STEUERN' ist nur im Stoppzustand zulässig!**

### Funktion aufrufen

Bei Aufruf der Funktion im Stoppzustand wird die Befehlsausgabesperre aufgehoben (BASP = aus). Die gesamte digitale Peripherie (F000H bis F07FH) wird gelöscht, indem jede Adresse mit dem Wert '0' beschrieben wird. Während des Löschens der Peripherie ist die Funktion nicht unterbrechbar.

Die Peripherieausgänge werden byteweise gesteuert, direkt und ohne das Prozeßabbild der Ausgänge zu beeinflussen!

Im Mehrprozessorbetrieb können Sie alle Peripherieausgänge steuern (unabhängig von einer Peripheriezuteilung im DBI).

Wenn die Funktion aktiv ist (Meldung "Steuern fertig" am PG), können Sie einen Neustart oder einen manuellen Wiederanlauf durchführen. Nach einem erneuten Übergang in den Stoppzustand können Sie wieder steuern. Die Ausgangssignalformer werden in diesem Fall nicht gelöscht.

### Abbrechen der Funktion

Die Funktion wird beendet durch Drücken der Abbruchtaste am PG. Die Befehlsausgabesperre wird wieder ausgegeben (BASP = an).

## 11.5 On-line-Funktion 'STEUERN VARIABLEN'

Mit der On-line-Funktion 'STEUERN VARIABLEN' können Sie die Werte von Operanden (Prozeßvariablen) einmalig verändern. Dies ist in jedem Betriebszustand des Prozessors zulässig. Sie können alle Prozeßvariablen angeben. Im Bereich des Prozeßabbilds wird kein ADF ausgelöst.

Die Änderung wird an einem Systemkontrollpunkt wirksam. Beachten Sie, daß die gesteuerten Werte nachträglich überschrieben werden können (z.B. durch das Anwenderprogramm oder die Prozeßabbildaktualisierung)!

Hinweis: Das PG steuert Prozeßvariablen E, A, M byteweise und DW, T, Z wortweise.

### WICHTIG!

**Wenn Sie mehrere Operanden steuern, dann werden die geänderten Bytes (bei DW, T, Z die Wörter) nacheinander über mehrere Systemkontrollpunkte verteilt im Speicher geändert.**

## 11.6 On-line-Funktion 'Speicher KOMPRIMIEREN'

Mit dieser Funktion werden alle gültigen Bausteine des Anwenderprogramms bündig hintereinandergeschoben. Dies wird getrennt im RAM-Modul und im DB-RAM durchgeführt. Lücken, die beim Löschen oder Korrigieren von Bausteinen entstehen, verschwinden. Hierzu wird jeweils ein kompletter Baustein zum Anfang des Speicherbereiches hin geschoben. Dies kann am Systemkontrollpunkt 'Zyklus' und 'Stop' durchgeführt werden.

Die Funktion wird mit einer Fehlermeldung abgebrochen, wenn am Systemkontrollpunkt 'Stop' der BSTACK nicht leer ist. Dies ist der Fall, wenn während der Programmbearbeitung eine Unterbrechung in den Stoppzustand führt. In diesem Fall kann nur im Zyklus weiterkomprimiert werden.

Fällt während des Komprimierens das Netz aus, wird kein weiterer Baustein verschoben. Bei erneutem Aufruf von 'Speicher KOMPRIMIEREN' wird weitergeschoben.

### WICHTIG!

Die Funktion 'Speicher KOMPRIMIEREN' erkennt folgende Fehler im Bausteinspeicher:

- falsche Bausteinlänge
- verfälschtes Muster '7070' im Bausteinkopf
- ungültiger Bausteintyp  
(bei OBs ungültige Bausteinnummer).

Die Funktion wird abgebrochen. Am PG wird eine Meldung ausgegeben. Daraufhin muß aufgelöscht werden. Die Funktion kann erst nach dem Umlöschen erneut aufgerufen werden.

Hinweis: 'Speicher KOMPRIMIEREN' ist nicht zulässig, solange 'BEARBEITUNGSKONTROLLE' aktiv ist!

## 11.7 On-line-Funktionen 'START'/'STOP'

Die PG-Bedienung entspricht der manuellen Bedienung. Mit dem Aufruf der Funktion 'STOP' können Sie das Automatisierungsgerät in den Stoppzustand bringen. Bei demjenigen Prozessor, an dem das PG angeschlossen ist, sehen Sie folgendes Bild:

STOP-LED: an  
BASP-LED: an

In den Steuerbits ist 'PG-STP' angekreuzt. Bei Mehrprozessorbetrieb ist bei den anderen Prozessoren das Steuerbit 'MP-STP' gesetzt.

Sie können einen Prozessor mit Neustart oder Wiederanlauf bedienen. Im Einzelprozessorbetrieb verläßt der Prozessor den Stoppzustand. Im Mehrprozessorbetrieb wird zunächst die Anlaufart registriert (Steuerbit 'NEUST' oder 'M W A' ist gesetzt), der Prozessor bleibt aber im Stopp. Mit der nachfolgenden Bedienung 'System starten' können Sie das Automatisierungsgerät starten. Dies entspricht der Bedienung des Koordinators (Schalter auf RUN).

Eine weitere Möglichkeit bietet die On-line-Funktion 'START' im Mehrprozessorbetrieb: Sie können nacheinander bei allen Prozessoren die Anlaufart wählen und erst beim letzten Prozessor das AG starten.

### **11.8 On-line-Funktion 'AG URLÖSCHEN'**

Sie können einen Prozessor vom PG aus urlöschen (entspricht 'Alle Bausteine löschen'). Dabei wird das Urlöschen unbedingt durchgeführt (siehe Kapitel 4.2).

Befindet sich der Prozessor beim Aufruf von 'AG URLÖSCHEN' im Zustand ANLAUF oder RUN, wird zunächst ein Übergang in den Stoppzustand durchgeführt. Dabei wird - wenn geladen - der Organisationsbaustein OB 28 aufgerufen.

Hinweis: 'AG URLÖSCHEN' ist nicht zulässig, solange 'BEARBEITUNGSKONTROLLE' aktiv ist!

### **11.9 On-line-Funktion 'AUSGABE ADRESSE'**

Mit der Funktion 'AUSGABE ADRESSE' können Sie am PG den Inhalt von Speicher- und Peripherieadressen wortweise hexadezimal ausgeben. Sie können alle Adressen ansprechen. Im Bereich des Prozeßabbilds wird kein ADF ausgelöst, im Peripheriebereich entsteht kein QVZ.

Im byteadressierbaren Bereich (Merker, Prozeßabbild) wird das Highbyte als 'FF' dargestellt.

Im Peripheriebereich wird bei quittierenden Adressen das Highbyte als '00' ausgegeben. Quittiert eine Peripherieadresse nicht, wird das Highbyte als 'FF' angezeigt.

### **11.10 On-line-Funktion 'SPEICHERAUSBAU'**

Die Funktion 'SPEICHERAUSBAU' zeigt Ihnen am PG die höchste nutzbare Adresse des RAM-Moduls an (bei EPROM wird '0' angezeigt) und die letzte mit Bausteinen des Anwenderprogramms belegte Adresse des Speichermoduls.

### 11.11 Tabelle: Tätigkeiten an Kontrollpunkten

Tätigkeiten bei On-line-Funktionen		Systemkontrollpunkt				Anwender- kontrollpunkt
		Stop	Zyklus	Wartezustand	Timeout	
EingabeAdresse: Daten schreiben	1)	*	*	*	*	
Bausteineingabe: Baustein gültig erklären		*	*	*	*	
Baustein löschen		*	*	*	*	
Speicher komprimieren: Baustein verschieben	2)	* 3)	*			
Start/Stop		*	*	*	*	
Urlöschen		*	*		*	
STATUS VAR: Daten lesen und ausgeben		*	*	*		
STATUS: Daten lesen und ausgeben						*
Bearbeitungskontrolle: Haltepunktvorgabe		*	*	*	*	
Daten lesen						*
Daten ausgeben		*		*		
Steuern Signalformer	1)	*				
Steuern VAR	1)	*	*	*	*	

ONLINE.GEM

1) Tätigkeiten, die über mehrere Systemkontrollpunkte verteilt werden können

2) Pro Systemkontrollpunkt maximal ein Baustein

3) Nur wenn kein BSTACK-Eintrag

**Tabelle: Tätigkeiten, die an System- und Anwenderkontrollpunkten durchgeführt werden**

**ANHANG A: Technische Daten AG SS-135U**

	S-Prozessor	R-Prozessor	CPU 928
typ. Befehlsausführungszeiten für Bitbefehle mit M, E, A D Formaloperand	1,1 <sub>us</sub> 100 <sub>us</sub> 94 <sub>us</sub>	22 <sub>us</sub> 37 <sub>us</sub> 46 <sub>us</sub>	1 <sub>us</sub> 34 <sub>us</sub> 24 <sub>us</sub>
typ. Befehlsausführungszeiten für Wortbefehle - Ladeoperationen L MB (Byte) L MW (Wort) L MD (Doppelwort) - Fest- und Gleitpunktarithmetik	31 <sub>us</sub> 33 <sub>us</sub> 70 <sub>us</sub> <900 <sub>us</sub>	15 <sub>us</sub> 15 <sub>us</sub> 20 <sub>us</sub> <86 <sub>us</sub>	12 <sub>us</sub> 12 <sub>us</sub> 16 <sub>us</sub> <69 <sub>us</sub>
zyklische Programmbearbeitung (Einzelprozessorbetrieb) - Grundlast beim Aufruf OB1/FB0:	221/268 <sub>us</sub>	107/119 <sub>us</sub>	147/149 <sub>us</sub>
- Zuschlag für die Prozeßabbild- aktualisierung in Abhängigkeit von der Anzahl der E/A-Bytes (n) mit $0 < n \leq 256$	23 <sub>us</sub> + n x 1,63 <sub>us</sub>	33 <sub>us</sub> + n x 6 <sub>us</sub>	18 <sub>us</sub> + n x 1,58 <sub>us</sub>
- Zuschlag für die Koppelmerker- übertragung in Abhängigkeit von der Anzahl der Koppelmer- ker (n) mit $0 < n \leq 256$	25 <sub>us</sub> + n x 1,93 <sub>us</sub>	35 <sub>us</sub> + n x 6,5 <sub>us</sub>	19 <sub>us</sub> + n x 1,84 <sub>us</sub>
- Zuschlag für Zeitzellenbear- beitung in Abhängigkeit von der Zeitenblocklänge (ZBL) ZBL=0 ZBL#0 n = Anzahl der <u>laufenden</u> Zeitzellen (Raster: 10 ms) - alarmgesteuerte Programmbear- beitung Verlängerung der Zykluszeit durch Einschachtelung eines leeren OB 2 (ohne STEP5-Bef.) an einer Bausteingrenze  Reaktionszeit	alle 10 ms 20 <sub>us</sub> 50 <sub>us</sub> + ZBL x 9,3 <sub>us</sub> + n x 17,9 <sub>us</sub>  439 <sub>us</sub>  425 <sub>us</sub>	alle 2,5 ms 50 <sub>us</sub> 60 <sub>us</sub> + ZBL x 1,56 <sub>us</sub> + n x 1,24 <sub>us</sub>  367 <sub>us</sub>  300 <sub>us</sub>	alle 10 ms 5 <sub>us</sub> 200 <sub>us</sub> + n x 0,35 <sub>us</sub> (bei $0 < n \leq 128$ ) 400 <sub>us</sub> + n x 0,35 <sub>us</sub> (bei $128 < n \leq 256$ )  330 <sub>us</sub>  280 <sub>us</sub>

	S-Prozessor	R-Prozessor	CPU 928
- zeitgesteuerte Programmbearbeitung Verlängerung der Zykluszeit durch Einschachtelung eines leeren OB 13 (ohne STEP5-Bef.) an einer Befehlsgrenze	327,us	375,us	340,us für den ersten Weckalarm-OB 180,us für jeden weiteren, zum gleichen Zeitpunkt fälligen Weckalarm-OB
Zeittakt für den Aufruf des zeitgesteuerten Programms	100 ms	100 ms	10, 20, 50, 100, 200, 500 ms, 1, 2, 5 sec
Zykluszeitüberwachung Voreinstellung einstellbar zwischen triggerbar	100 ms - ja	150 ms 1..4000 ms ja	150 ms 1..6000 ms ja
Größe des Anwenderspeichers (in K Wörter)	≤ 32	≤ 32	≤ 32
Größe des Speichers für Datenbausteine (DB-RAM, in K Wörter)	ca. 3,7	ca. 11,1	ca. 23,3
Anzahl der Zeit- und Zählzellen	je 128	je 128	je 256
Anzahl der Merkerbytes	je 256	je 256	je 256

### Begriffserläuterungen:

**Grundlast:** Als Grundlast wird derjenige Teil der zyklischen Systemlaufzeit bezeichnet, der ohne Prozeßabbildaktualisierung und Koppelmerkerübertragung meßbar ist.

**Reaktionszeit :** Als Reaktionszeit wird die Zeit vom Aktivieren der Programmbearbeitungsebene PROZESSALARM bis zur Bearbeitung des ersten Befehls im OB 2 bezeichnet, vorausgesetzt, der OB 2 wird sofort nach Erkennen des Prozeßalarms aufgerufen. Muß hingegen auf die nächste Befehls- oder Bausteingrenze gewartet werden, verlängert sich die Reaktionszeit.

## ANHANG B: Übersicht über die Fehlerkennungen

### Systemdatum 3 und 4

SD3      SD4

#### Aufbau der Bausteinadreiblisten:

8001H	yyyyH	Falsche Bausteinlänge	yyyy = Adr. des Bausteins mit falscher Länge
8002H	yyyyH	Berechnete Endadresse des Bausteins im Speicher falsch	yyyy = Bausteinadresse
8003H	yyyyH	Ungültige Bausteinkennung	yyyy = Adr. des Bausteins mit falscher Kennung
8004H	yyyyH	Zu große Organisationsbausteinnummer (erlaubt: OB 1 bis 39)	yyyy = Adresse des Bausteins mit
8005H	yyyyH	Datenbausteinnummer 0 (erlaubt: DB1 bis 255)	yyyy = Adr. des Bausteins mit falscher Nummer falscher Nummer

#### Aufbau der Adreßlisten für die Prozeßabbild-Aktualisierung:

0410H	yyyyH	Unzulässige Kennung	yyyy = Falsche Kennung
0411H	yyyyH	Falsche Anzahl Adressen bei Adreßliste "Digitale Eingänge" (erlaubt: 128)	yyyy = Falsche Anzahl Adressen
0412H	yyyyH	Falsche Anzahl Adressen bei Adreßliste "Digitale Ausgänge" (erlaubt: 128)	yyyy = Falsche Anzahl Adressen
0413H	yyyyH	Falsche Anzahl Adressen bei Adreßliste "Koppelmerker-Eingang" (erlaubt: 256)	yyyy = Falsche Anzahl Adressen
0414H	yyyyH	Falsche Anzahl Adressen bei Adreßliste "Koppelmerker-Ausgang" (erlaubt: 256)	yyyy = Falsche Anzahl Adressen
0415H	yyyyH	Ungültige Anzahl Zeitzellen (erlaubt: 256)	yyyy = Falsche Anzahl Zeitzellen
0419H	yyyyH	Quittungsverzug bei digitalen Eingängen	yyyy = Adresse des nicht quittierten Eingangsbytes
041AH	yyyyH	Quittungsverzug bei digitalen Ausgängen	yyyy = Adresse des nicht quittierten Ausgangsbytes
041BH	yyyyH	Quittungsverzug bei Koppelmerker-Eingang	yyyy = Adresse des nicht quittierten Merkerbytes
041CH	yyyyH	Quittungsverzug bei Koppelmerker-Ausgang	yyyy = Adresse des nicht quittierten Merkerbytes

#### Auswertung des DB 2:

0421H	DByyH	Datenbaustein nicht geladen	yy = Nummer des nicht geladenen DBs
0422H	FByyH	Funktionsbaustein nicht geladen	yy = Nummer des nicht geladenen FBs
0423H	FByyH	Funktionsbaustein nicht erkannt	yy = Nummer des nicht erkannten FBs
0424H	FByyH	Funktionsbaustein mit falscher PG-Software geladen	yy = Nummer des Funktionsbausteins
0425H	DByyH	Falsche Regler-Datenbaustein-Länge	yy = Nummer des Datenbausteins
0426H	-	Speicherplatz im DB-RAM nicht ausreichend	

#### Auswertung des DX 0:

0431H	yyyyH	Unzulässige Kennung	yyyy = Falsche Kennung
0432H	yyyyH	Unzulässiger Parameter	yyyy = Falscher Parameter
0434H	yyyyH	Unzulässige Anzahl Zeitzellen (erlaubt: 256)	yyyy = Falsche Anzahl Zeitzellen
0435H	yyyyH	Unzulässige Zykluszeit (erlaubt: 1 ms bis 4 sec)	yyyy = Falsche Zeitgröße

### Akku 1 und Akku 2

Akku1      Akku2

#### Reglerbearbeitung:

0801H	DByyH	Abtastzeitfehler	yy = Nr des betreffenden DBs	Aufruf OB 34
0802H	DByyH	Regler-DB nicht geladen	yy = Nr des nicht geladenen DBs	Aufruf OB 34
0803H	FByyH	Regler-FB nicht geladen	yy = Nr des nicht geladenen FBs	Aufruf OB 34
0804H	FByyH	Regler-FB nicht erkannt	yy = Nr des FBs	Aufruf OB 34
0805H	FByyH	Regler-FB mit falscher PG-Software geladen	yy = Nr des FBs	Aufruf OB 34
0806H	DByyH	Falsche Regler-Datenbaustein-Länge	yy = Nr des DB	Aufruf OB 34
0880H	yyyyH	Quittungsverzug (QVZ) während der Reglerbearbeitung		Aufruf OB 34



# STEP5-Befehlscodefehler:

1801H	-	Substitutionsfehler beim BBS-Befehl	Aufruf OB 27
1802H	-	Substitutionsfehler beim BDW-/BMW-Befehl	Aufruf OB 27
1803H	-	Substitutionsfehler beim B=-/BI=-Befehl	Aufruf OB 27
1804H	-	Substitutionsfehler beim L=-/T=-Befehl	Aufruf OB 27
1805H	-	Substitutionsfehler beim U=-/UN=-/O=-/ON=-/=-/S=- und RB=-Befehl	Aufruf OB 27
1811H	-	Befehl mit unzulässigem Opcode	Aufruf OB 29
1812H	-	Befehl mit unzulässigem Opcode (H-Byte des 1. Befehlswortes = 68H)	Aufruf OB 29
1813H	-	Befehl mit unzulässigem Opcode (H-Byte des 1. Befehlswortes = 78H)	Aufruf OB 29
1814H	-	Befehl mit unzulässigem Opcode (H-Byte des 1. Befehlswortes = 70H)	Aufruf OB 29
1815H	-	Befehl mit unzulässigem Opcode (H-Byte des 1. Befehlswortes = 60H)	Aufruf OB 29
1821H	-	Unzulässiger Parameter bei A DB0, A DB1 und A DB2	Aufruf OB 30
182BH	-	Unzulässiger Parameter bei SPA(B) OB 0	Aufruf OB 30
182CH	-	Unzulässiger Parameter bei SPA(B) OB > 39: Sonderfkt. nicht vorhanden	Aufruf OB 30
182DH	-	Unzulässiger Parameter bei AX DX0	Aufruf OB 30
182EH	-	Unzulässiger Parameter bei LMW/TMW/LPW/TPW/LQW/TQW/LDD/TDD/BMW 255	Aufruf OB 30
182FH	-	Unzulässiger Parameter bei LEW/TEW/LAW/TAW 127	Aufruf OB 30
1830H	-	Unzulässiger Parameter bei LMD/TMD 253, 254, 255	Aufruf OB 30
1831H	-	Unzulässiger Parameter bei LED/TED/LAD/TAD 125, 126, 127	Aufruf OB 30
1832H	-	Unzulässiger Parameter bei RLD/RRD/SVD/SLD 33-255	Aufruf OB 30
1833H	-	Unzulässiger Parameter bei SLW/SRW/LIR/TIR 16-255	Aufruf OB 30
1834H	-	Unzulässiger Parameter bei SES/SEF 32-255	Aufruf OB 30
1835H	-	Unzulässiger Parameter bei U=/UN=O=/ON=/S=/RB=/=/RD=/FR=/SI=/SE=/SVZ=/ SSV=/SAR=/L=/LC=LW=/T= 0, 127-255	Aufruf OB 30
1836H	-	Unzulässiger Parameter bei B=/LD= 0, 126-255	Aufruf OB 30

# STEP5-Laufzeitfehler:

1A01H	-	Nicht geladener Datenbaustein bei ADB	Aufruf OB 19
1A02H	-	Nicht geladener Datenbaustein bei AXDX	Aufruf OB 19
1A03H	-	Nicht geladener Baustein bei SPA(B) FB, OB, PB und SB	Aufruf OB 19
1A04H	-	Nicht geladener Baustein bei BA(B) FX	Aufruf OB 19
1A05H	-	Nicht geladener Datenbaustein bei OB 254 bzw. 255	Aufruf OB 19
1A11H	-	Zugriff auf ein nicht definiertes Datenwort	Aufruf OB 32
1A12H	-	Transferfehler auf ein nicht definiertes Datenwort (T DR)	Aufruf OB 32
1A13H	-	Transferfehler auf ein nicht definiertes Datenwort (T DL)	Aufruf OB 32
1A14H	-	Transferfehler auf ein nicht definiertes Datenwort (T DW)	Aufruf OB 32
1A15H	-	Transferfehler auf ein nicht definiertes Datenwort (T DD)	Aufruf OB 32
1A21H	-	Fehler bei EDB, EXDX: DB existiert bereits	Aufruf OB 31
1A22H	-	Fehler bei EDB, EXDX: Unzulässige DB-Länge	Aufruf OB 31
1A23H	-	Fehler bei EDB, EXDX: Speicherplatz im RAM reicht nicht aus	Aufruf OB 31
1A25H	-	Fehler bei BI=: Unzulässiger Parameter im Akku 1	Aufruf OB 31
1A29H	-	Klammerstackunter- od. -überlauf nach U(, O(,)	Aufruf OB 31
1A2AH	-	Fehler bei ADB oder AXDX: Bausteinlänge im DB-Kopf zu klein (<5 Wörter)	Aufruf OB 31
1A2BH	-	Funktionsbaustein mit falscher PG-Software geladen	Aufruf OB 31
1A2CH	-	Fehler bei ACR: Kachelnummer in Akku 1-L > 255	Aufruf OB 31
1A31H	-	SF-Fehler bei OB 254 bzw. OB 255: DB bereits im RAM vorhanden	Aufruf OB 31
1A32H	-	SF-Fehler bei OB 254 bzw. OB 255: Neuer DB bereits vorhanden	Aufruf OB 31
1A33H	-	SF-Fehler bei OB 254 bzw. OB 255: Speicherplatz im RAM reicht nicht aus	Aufruf OB 31
1A3AH	-	SF-Fehler bei OB 221: Unzulässiger Wert für die neue Zykluszeit	Aufruf OB 31
1A3BH	-	SF-Fehler bei OB 223: Unterschiedliche Anlaufarten im Mehrprozessorbetrieb	Aufruf OB 31
1A41H	-	SF-Fehler bei OB 240, 241 oder 242: Unzulässige Schieberegister- oder DB-Nr.	Aufruf OB 31
1A42H	-	SF-Fehler bei OB 241: Schieberegister nicht initialisiert	Aufruf OB 31
1A43H	-	SF-Fehler bei OB 240: Speicherplatz im DB-RAM reicht nicht aus	Aufruf OB 31
1A44H	-	SF-Fehler bei OB 240: DW 0 des Datenbausteins hat nicht den Inhalt '0'	Aufruf OB 31
1A45H	-	SF-Fehler bei OB 240: Unzulässige Schieberegisterlänge im DW 1	Aufruf OB 31
1A46H	-	SF-Fehler bei OB 240: Unzulässige Zeigerposition oder Zeigeranzahl	Aufruf OB 31
1A47H	-	SF-Fehler bei OB 120: Unzulässige Werte im Akku 1 oder Akku 2-L	Aufruf OB 31
1A48H	-	SF-Fehler bei OB 122: Unzulässige Werte im Akku 1	Aufruf OB 31
1A49H	-	SF-Fehler bei OB 110: Unzulässige Werte im Akku 1	Aufruf OB 31
1A4AH	-	SF-Fehler bei OB 121: Unzulässige Werte im Akku 1 oder Akku 2-L	Aufruf OB 31
1A4BH	-	SF-Fehler bei OB 123: Unzulässige Werte im Akku 1	Aufruf OB 31

1A50H	-	Fehler bei LRW, TRW: Unzulässige Adresse	Aufruf 08 31
1A51H	-	Fehler bei LRD, TRD: Unzulässige Adresse	Aufruf 08 31
1A52H	-	Fehler bei TSG, LBGB, LWGW, TBGB, TWGW: Unzulässige Adresse	Aufruf 08 31
1A53H	-	Fehler bei LBGW, LWGD, TBGW, TWGD: Unzulässige Adresse	Aufruf 08 31
1A54H	-	Fehler bei LBGD, TBGD: Unzulässige Adresse	Aufruf 08 31
1A55H	-	Fehler bei TSC, LBCB, LWCD, TBCW, TWCD: Unzulässige Adresse	Aufruf 08 31
1A56H	-	Fehler bei LBCW, LWCD, TBCW, TWCD: Unzulässige Adresse	Aufruf 08 31
1A57H	-	Fehler bei LBCD, TBCD: Unzulässige Adresse	Aufruf 08 31
1A58H	-	Fehler bei TNW/TNB: Quellblock liegt nicht vollständig in einem Bereich	Aufruf 08 31
1A59H	-	Fehler bei TNW/TNB: Zielblock liegt nicht vollständig in einem Bereich	Aufruf 08 31
<b>Quittungsverzug:</b>			
1E23H	yyyyH	Quittungsverzug (QVZ) im Anwenderprogramm	yyyy = QVZ-Adresse
1E25H	yyyyH	Quittungsverzug beim Prozeßabbild der digitalen Ausgänge	
		yyyy = Adresse des nicht quitierten Ausgangsbytes	Aufruf 08 24
1E26H	yyyyH	Quittungsverzug beim Prozeßabbild der digitalen Eingänge	
		yyyy = Adresse des nicht quitierten Eingangsbytes	Aufruf 08 24
1E27H	yyyyH	Quittungsverzug beim Prozeßabbild der Koppelmerker-Ausgänge	
		yyyy = Adresse des nicht quitierten Merkerbytes	Aufruf 08 24
1E28H	yyyyH	Quittungsverzug beim Prozeßabbild der Koppelmerker-Eingänge	
		yyyy = Adresse des nicht quitierten Merkerbytes	Aufruf 08 24



## ANHANG C: Übersicht über den STEP5-Operationsvorrat

### Grundoperationen

Operation	Parameter
-----------	-----------

#### • Binäre Verknüpfungsoperationen:

U	E	0.0 bis 127.7
U	A	0.0 bis 127.7
U	M	0.0 bis 255.7
U	D	0.0 bis 255.15
U	T	0 bis 255
U	Z	0 bis 255
UN	E	0.0 bis 127.7
UN	A	0.0 bis 127.7
UN	M	0.0 bis 255.7
UN	D	0.0 bis 255.15
UN	T	0 bis 255
UN	Z	0 bis 255
O	E	0.0 bis 127.7
O	A	0.0 bis 127.7
O	M	0.0 bis 255.7
O	D	0.0 bis 255.15
O	T	0 bis 255
O	Z	0 bis 255
ON	E	0.0 bis 127.7
ON	A	0.0 bis 127.7
ON	M	0.0 bis 255.7
ON	D	0.0 bis 255.15
ON	T	0 bis 255
ON	Z	0 bis 255
)		
U(		
O(		
O		

#### o Vergleichsoperationen:

!=F	
<<F	
>F	
>=F	
<<F	
<=F	
!=D	
<<D	
>D	
>=D	
<<D	
<=D	
!=G	
<<G	
>G	
>=G	
<<G	
<=G	

Operation	Parameter
-----------	-----------

#### o Speicheroperationen:

S	E	0.0 bis 127.7
S	A	0.0 bis 127.7
S	M	0.0 bis 255.7
S	D	0.0 bis 255.5
R	E	0.0 bis 127.7
R	A	0.0 bis 127.7
R	M	0.0 bis 255.7
R	D	0.0 bis 255.15
=	E	0.0 bis 127.7
=	A	0.0 bis 127.7
=	M	0.0 bis 255.7
=	D	0.0 bis 255.15

#### o Ladeoperationen:

L	EB	0 bis 127
L	EW	0 bis 126
L	ED	0 bis 124
L	AB	0 bis 127
L	AW	0 bis 126
L	AD	0 bis 124
L	MB	0 bis 255
L	MW	0 bis 254
L	MD	0 bis 252
L	DL	0 bis 255
L	DR	0 bis 255
L	DW	0 bis 255
L	DD	0 bis 254
L	T	0 bis 255
L	Z	0 bis 255
L	PY	0 bis 127
		128 bis 255
L	PW	0 bis 126
		128 bis 254
L	QB	0 bis 255
L	QW	0 bis 254
LC	T	0 bis 255
LC	Z	0 bis 255
L	KB	0 bis 255
L	KC	2 alphanum. Zeichen
L	KM	Bitmuster (16 Bit)
L	KH	0 bis FFFF
L	KF	-32 768 bis +32 767
L	KY	0 bis 255 für jedes Byte
L	KT	0.0 bis 999.3
L	KZ	0 bis 999
L	KG	1)

1)  $\pm 0.1469368 \times 10^{-38}$  bis  
 $\pm 0.1701412 \times 10^{39}$

Operation	Parameter
-----------	-----------

o Zeit- und Zähloperationen:

SI	T	0	bis 255
SV	T	0	bis 255
SE	T	0	bis 255
SS	T	0	bis 255
SA	T	0	bis 255
R	T	0	bis 255
S	Z	0	bis 255
R	Z	0	bis 255
ZV	Z	0	bis 255
ZR	Z	0	bis 255

o Transferoperationen:

T	EB	0	bis 127
T	EW	0	bis 126
T	ED	0	bis 124
T	AB	0	bis 127
T	AW	0	bis 126
T	AD	0	bis 124
T	MB	0	bis 255
T	MW	0	bis 254
T	MD	0	bis 252
T	DR	0	bis 255
T	DL	0	bis 255
T	DW	0	bis 255
T	DD	0	bis 254
T	PY	0	bis 127
		128	bis 255
T	PW	0	bis 126
		128	bis 254
T	QB	0	bis 255
T	QW	0	bis 254

o Bausteinaufrufe:

SPA	PB	0	bis 255
SPA	FB	0	bis 255
BA	FX	0	bis 255
SPA	SB	0	bis 255
SPA	OB	1	bis 39
SPA	OB	1)	40 bis 255
SPB	PB	0	bis 255
SPB	FB	0	bis 255
BAB	FX	0	bis 255
SPB	SB	0	bis 255
SPB	OB	1	bis 39
SPB	OB	1)	40 bis 255
A	DB	3	bis 255
AX	DX	1	bis 255
BE			
BEB			
BEA			

- 1) Aufruf einer Sonderfunktion  
2) Systemoperation

Operation	Parameter
-----------	-----------

o Arithmetische Operationen:

+F	
-F	
xF	
:F	
+G	
-G	
xG	
:G	

o Sonstige Operationen:

NOP	0	
NOP	1	
STP		
BLD	0	bis 255

**Ergänzende Operationen**

Operation	Parameter
-----------	-----------

o Verknüpfungsoperationen, wortweise:

UW	
OW	
XOW	

o Zeit- und Zähloperationen:

FR	T	0	bis 255
FR	Z	0	bis 255
FR	=		Formaloperand
SI	=		Formaloperand
SE	=		Formaloperand
SVZ	=		Formaloperand
SSV	=		Formaloperand
SAR	=		Formaloperand
RD	=		Formaloperand

o Ladeoperationen:

L	=		Formaloperand
LC	=		Formaloperand
LW	=		Formaloperand
LD	=		Formaloperand
L	BS	0	bis 255
L	BT	0	bis 255
L	BA	0	bis 255
L	BB	0	bis 255
LIR	2)	0	bis 15

Operation	Parameter
-----------	-----------

o Ladeoperationen  
(Fortsetzung):

LRW	-32768 ... +32767
LRD	-32768 ... +32767
LB GB	-32768 ... +32767
LB GW	-32768 ... +32767
LB GD	-32768 ... +32767
LW GW	-32768 ... +32767
LW GD	-32768 ... +32767
LB CB	-32768 ... +32767
LB CW	-32768 ... +32767
LB CD	-32768 ... +32767
LW CW	-32768 ... +32767
LW CD	-32768 ... +32767

o Transferoperationen:

T	=	= Formaloperand
T	BA	0 bis 255
T	BB	0 bis 255
T	BS <sup>2)</sup>	0 bis 255
T	BT	0 bis 255
TIR	<sup>2)</sup>	0 bis 15
TNB	<sup>2)</sup>	0 bis 255
TNW	<sup>2)</sup>	0 bis 255
TRW		-32768 ... +32767
TRD		-32768 ... +32767
TSG		-32768 ... +32767
TB GB		-32768 ... +32767
TB GW		-32768 ... +32767
TB GD		-32768 ... +32767
TW GW		-32768 ... +32767
TW GD		-32768 ... +32767
TSC		-32768 ... +32767
TB CB		-32768 ... +32767
TB CW		-32768 ... +32767
TB CD		-32768 ... +32767
TW CW		-32768 ... +32767
TW CD		-32768 ... +32767

o Verknüpfungsoperationen,  
binär:

U	=	Formaloperand
UN	=	Formaloperand
O	=	Formaloperand
ON	=	Formaloperand

o Umwandlungsfunktionen:

KEW	
KZW	
KZD	
DEF	
DUF	

Operation	Parameter
-----------	-----------

o Umwandlungsfunktionen:  
(Fortsetzung):

DED	
DUD	
FDG	
GFD	

o Schiebefunktionen:

SLW	0 bis 15
SRW	0 bis 15
SLD	0 bis 32
SVD	0 bis 32
RLD	0 bis 32
RRD	0 bis 32
SVW	0 bis 15

o Sprungfunktionen:

SPA	=	Symboladresse
SPB	=	Symboladresse
SPZ	=	Symboladresse
SPN	=	Symboladresse
SPP	=	Symboladresse
SPM	=	Symboladresse
SPO	=	Symboladresse
SPS	=	Symboladresse
SPR		-32768 ... +32767

o BR-Register-Operationen:

MBR	0 ... FFFFF
ABR	-32768 ... +32767
MAS	
MAB	
MSA	
MSB	
MBA	
MBS	

o Setzoperationen:

S	=	Formaloperand
RB	=	Formaloperand
=	=	Formaloperand

o Sonstige Funktionen:

ACR	
AF	
AS	
ENT	
D	0 bis 255
I	0 bis 255
B	= Formaloperand

Operation	Parameter
-----------	-----------

o Sonstige Funktionen:  
(Fortsetzung):

B DW	0 bis 255
B MW	0 bis 255
BI <sup>2)</sup>	
B BS	0 bis 255
TAK	
BLD	0 bis 255
E DB	0 bis 255
EX DX	0 bis 255
SES	0 bis 31
SEF	0 bis 31

o Arithmetische Operation:

ADD BF	-128 bis +127
ADD KF	-32 768 bis +32 767
ADD DF <sup>2)</sup>	-2147483648 bis +2147483647
+D <sup>2)</sup>	
-D <sup>2)</sup>	

<sup>2)</sup> Systemoperationen

## ANHANG D: STEP5-Befehlsübersicht (alphabetisch)

Die mit \* gekennzeichneten Befehle gehören zu den Ergänzenden Operationen und sind nur in Funktionsbausteinen (FB/FX) gültig!

STEP5-Befehl	Befehlsgruppe	STEP5-Befehl	Befehlsgruppe
!=D	Vergleichsoperation	BE	Bausteinende
!=F	"	BEA	"
!=G	"	BEB	"
)	Verknüpf.oper.binär	BI	* Systemoperation!
+D	* Systemoperation!	BLD	sonst. Operation
+F	arithmet. Operation	B MW	* "
+G	"	D	* dekrementieren
-D	* Systemoperation!	DED	* Umwandlungsoperation
-F	arithmet. Operation	DEF	* "
-G	"	DUD	* "
:F	"	DUF	* "
:G	"	E DB	DB erzeugen
xF	"	ENT	* sonst. Operation
xG	"	EX DX	DX erzeugen
<=D	Vergleichsoperation	FDG	* Umwandlungsoperation
<=F	"	FR =	* Zeit/Zähloperation
<=G	"	FR T	* Zeitoperation
<D	"	FR Z	* Zähloperation
<F	"	GFD	* Umwandlungsoperation
<G	"	I	* inkrementieren
=	* Setzoperation	KEW	* Umwandlungsoperation
=A	Speicheroperation	KZD	* "
=D	"	KZW	* "
=E	"	L AB	Ladeoperation
=M	"	L AW	"
><D	Vergleichsoperation	L BA	* "
><F	"	L BB	* "
><G	"	L BS	* "
>=D	"	L BT	* "
>=F	"	LB CB	* Systemoperation!
>=G	"	LB CD	* "
>D	"	LB CW	* "
>F	"	LB GB	* "
>G	"	LB GD	* "
ABR	* Systemoperation!	LB GW	* "
ACR	* Systemoperation!	LC =	* Ladeoperation
A DB	Bausteinaufruf	LC T	"
ADD BF	* arithm. Operation	LC Z	"
ADD DF	* Systemoperation!	LD =	* "
ADD KF	* arithm. Operation	L DD	"
AF	* sonst. Operation	L DL	"
AS	* "	L DR	"
AX DX	Bausteinaufruf	L DW	"
BA FX	"	L EB	"
BAB FX	"	L ED	"
B =	* sonst. Operation	L EW	"
B BS	* Systemoperation!	LIR	* Systemoperation!
B DW	* sonst. Operation	L KB	Ladeoperation



STEP5-Befehl	Befehlsgruppe
L KF	"
L KG	"
L KH	"
L KM	"
L KT	"
L KY	"
L KZ	"
L MB	"
L MD	"
L MW	"
L PB/PY	"
L PW	"
L QB	"
L QW	"
LRD *	Systemoperation!
LRW *	Systemoperation!
L T	Ladeoperation
LW = *	"
LW CD *	Systemoperation!
LW CW *	"
LW GD *	"
LW GW *	"
L Z	Ladeoperation
L = *	"
MAB *	Systemoperation!
MAS *	"
MBA *	"
MBR *	"
MBS *	"
MSA *	"
MSB *	"
NOP 0	Nulloperation
NOP 1	"
O	Verknüpf.oper.binär
O(	"
O = *	"
O A	"
O D	"
O E	"
O M	"
ON = *	"
ON A	"
ON D	"
ON E	"
ON M	"
ON T	"
ON Z	"
O T	"
OW *	Verknüpf.oper.digit.
O Z	Verknüpf.oper.binär
R A	Speicheroperation
RB = *	Setzoperation
R D	Speicheroperation
RD = *	Zeit/Zähloperation
R E	Speicheroperation

STEP5-Befehl	Befehlsgruppe
RLD *	Schiebefunktion
R M	Speicheroperation
RRD *	Schiebefunktion
RT	Zeitoperation
RZ	Zähloperation
S = *	Setzoperation
S A	Speicheroperation
SAR= *	Zeit/Zähloperation
SA T	Zeitoperation
S D	Speicheroperation
S E	"
SEF *	sonst. Operation
SES *	"
SE = *	Zeitoperation
SE T	"
SI = *	"
SI T	"
SLD *	Schiebefunktion
SLW *	"
S M	Speicheroperation
SPA= *	Sprungoperation
SPA FB	Baustein aufruf
SPA OB	Baust.aufr.+ Sprung.
SPA PB	Baustein aufruf
SPA SB	"
SPB = *	Sprungoperation
SPB FB	Baustein aufruf
SPB OB	Baust.aufr.+ Sprung.
SPB PB	Baustein aufruf
SPB SB	"
SPM = *	Sprungoperation
SPN = *	"
SPO = *	"
SPP = *	"
SPR *	Systemoperation!
SPS = *	Sprungoperation
SPZ = *	"
SRW *	Schiebefunktion
SS T	Zeitoperation
SSV= *	Zeit/Zähloperation
STP	Stoppbefehl
SVD *	Schiebefunktion
SV T	Zeitoperation
SVW *	Schiebefunktion
SVZ= *	Zeit/Zähloperation
S Z	Zähloperation
T = *	Transferoperation
T AB	"
T AD	"
TAK *	sonst.Operation
T AW	Transferoperation
T BA *	"
T BB *	"
TB CB *	Systemoperation!
TB CD *	"

STEP5-Befehl	Befehlsgruppe
TB CW *	"
TB GB *	"
TB GD *	"
TB GW *	"
T BS *	Transferoperation
T BT *	"
T DD	"
T DL	"
T DR	"
T DW	"
T ED	"
T EW	"
TIR *	Systemoperation!
T MB	Transferoperation
T MD	"
T MW	"
TNB *	Systemoperation!
TNW *	"
T PB/PY	Transferoperation
T PW	"
T QB	"
T QW	"
TRD *	Systemoperation!
TRW *	"
TSC *	"
TSG *	"
TW CD *	"
TW CW *	"
TW GD *	"
TW GW *	"
U(	Verknüpf.oper.binär
U = *	"
U A	"
U D	"
U E	"
U M	"
UN = *	"
UN A	"
UN D	"
UN E	"
UN M	"
UN T	"
UN Z	"
U T	Verknüpf.oper.binär
UW *	Verknüpf.oper.digit.
U Z	Verknüpf.oper.binär
XOW *	Verknüpf.oper.digit.
Z RZ	Zähloperation
Z VZ	"

## **ANHANG E: STEP5-Befehle der CPU 928, sortiert nach dem Befehlscode**

### **Erläuterungen:**

#### **- Spalte 'Befehlscode':**

Der Befehlscode besteht aus maximal drei Wörtern (max. 48 Bit), die entweder als Hexadezimalzahl oder - bei wenigen Befehlen - als Bitmuster dargestellt sind. Jedes Bit läßt sich einem der folgenden Bereiche zuordnen:

'Opcode' (bestimmt die Art des Befehls)  
'Parameter' (bestimmt, womit der Befehl arbeitet)  
'irrelevant' (wird nicht ausdekodiert)

Die Bitpositionen, die den Parameter enthalten, sind mit den Buchstaben 'p' (1. Parameter, z.B. Byteadresse) und 'q' (2.Parameter, z.B. Bitadresse) gekennzeichnet.

Bitpositionen, die nicht ausdekodiert werden, sind mit dem Buchstaben 'x' gekennzeichnet.

#### **- Spalte 'Parameterbereich':**

Enthält den erlaubten Wertebereich der im Befehlscode mit den Buchstaben 'p' bzw. 'q' gekennzeichneten Bitpositionen.  
Alle Angaben sind dezimal.

#### **- Spalte 'STEP5':**

Enthält die zur Programmierung in AWL vorgesehene Kurzbezeichnung (STEP5-Mnemonik) .

#### **- Spalte 'Bemerkung':**

Änderungen gegenüber R- Prozessor:

P = Parameterbereich um 128 Zähler/Zeiten vergrößert  
N = neuer Befehl

Befehlscode	Parameterbereich	STEP5	Bemerkung
Wort			
---1 ---2 ---3			
00xx		NOP 0	
0100		KEW	
02pp	0-255	LT	P
03pp	0-255	TNB	
04pp	0-255	FRT	P
0500		BEB	
06pp	1-126	FR=	
07pp	1-126	U=	
0800		AS	
0880		AF	
0900		KZW	
0App	0-255	LMB	
0Bpp	0-255	TMB	
0Cpp	0-255	LCT	P
0Dpp	-128,+127	SPO=	
0Epp	1-126	LC=	
0Fpp	1-126	O=	
10xx		BLD	
11pp	0-255	I	
12pp	0-254	LMW	
13pp	0-254	TMW	
14pp	0-255	SAT	P
15pp	-128,+127	SPP=	
16pp	1-126	SAR=	
17pp	1-126	S=	
18pp	0-255	BBS	
19pp	0-255	D	
1App	0-252	LMD	
1Bpp	0-252	TMD	
1Cpp	0-255	SVT	P
1Dpp	0-255	SPBFB	
1Epp	1-126	SVZ=	P
1Fpp	1-126	=	
20pp	3-255	ADB	
2120		>F	
2140		<F	
2160		><F	
2180		!=F	
21A0		>=F	
21C0		<=F	
22pp	0-255	LDL	
23pp	0-255	TDL	
24pp	0-255	SET	P
25pp	-128,+127	SPM=	
26pp	1-126	SE=	
27pp	1-126	UN=	
28pp	0-255	LKB	
29pp	0-32	SLD	
2App	0-255	LDR	
2Bpp	0-255	TDR	
2Cpp	0-255	SST	P
2Dpp	-128,+127	SPA=	
2Epp	1-126	SSV=	
2Fpp	1-126	ON=	

Befehlscode	Parameterbereich	STEP5	Bemerkung
<i>Wort</i>			
---1 ---2 ---3			
3001 pppp	0-65535	LKZ	
3002 pppp	"	LKT	
3004 pppp	"	LKF	
3010 pppp	"	LKC	
3020 pppp	"	LKY	
3040 pppp	"	LKH	
3080 pppp	"	LKM	
3120		>G	
3140		<G	
3160		><G	
3180		!=G	
31A0		>=G	
31C0		<=G	
32pp	0-255	LDW	
33pp	0-255	TDW	
34pp	0-255	SIT	P
35pp	-128,+127	SPN=	
36pp	1-126	SI=	
37pp	1-126	RB=	
3800 pppp pppp	0-4294967295	LKG	
3920		>D	
3940		<D	
3960		><D	
3980		!=D	
39A0		>=D	
39C0		<=D	
3App	0-254	LDD	
3Bpp	0-254	TDD	
3Cpp	0-255	RT	P
3Dpp	0-255	SPAFB	
3Epp	1-126	RD=	
3Fpp	1-126	LW=	
40pp	0-15	LIR	
4100		UW	
42pp	0-255	LZ	P
43pp	0-255	TNW	
44pp	0-255	FRZ	P
4500	-128,+127	SPZ=	
46pp	1-126	L=	
47pp	0-255	LBB	N
48pp	0-15	TIR	
4900		OW	
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
0100 1010 0ppp pppp	0-127	LEB	
0100 1010 1ppp pppp	0-127	LAB	
0100 1011 0ppp pppp	0-127	TEB	
0100 1011 1ppp pppp	0-127	TAB	
<i>Wort</i>			
---1 ---2 ---3			
4Cpp	0-255	LCZ	P
4Dpp	1-255	SPBOB	
4Epp	0-254	BMW	
4Fpp	0-255	LBT	N

Befehlscode	Parameterbereich	STEP5	Bemerkung
<i>Wort</i>			
---1 ---2 ---3			
50pp	-128,+127	ADDBF	
5100		XOW	
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
0101 0010 0ppp pppp	0-126	LEW	
0101 0010 1ppp pppp	0-126	LAW	
0101 0011 0ppp pppp	0-126	TEW	
0101 0011 1ppp pppp	0-126	TAW	
<i>Wort</i>			
---1 ---2 ---3			
54pp	0-255	ZRZ	
55pp	0-255	SPBPB	
56pp	1-125	LD=	
57pp	0-254	LQW	
5800 pppp	-32768,+32767	ADDKF	
5900		-F	
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
0101 1010 0ppp pppp	0-124	LED	
0101 1010 1ppp pppp	0-124	LAD	
0101 1011 0ppp pppp	0-124	TED	
0101 1011 1ppp pppp	0-124	TAD	
<i>Wort</i>			
---1 ---2 ---3			
5Cpp	0-255	SZ	P
5Dpp	0-255	SPBSB	
5Fpp	0-255	LQB	
6000		:F	
6003		:G	
6004		xF	
6005 pppp pppp	-2147483648, +2147483647	ADDDF	N
6007		xG	
6008		ENT	
6009		-D	N
600B		-G	
600C xxpp	-128,+127	SPS=	
600D		+D	N
600F		+G	
61pp	0-15	SLW	
62pp	0-255	LBS	
63pp	0-255	TBS	
64pp	0-32	RLD	
6500		BE	
6501		BEA	
66pp	1-126	T=	
67pp	0-255	TBB	N
6800 pppp	-32768,+32767	LRW	N
68p1	0-15	SVW	
6802		GFD	
6803 pppp	-32768,+32767	TRW	N

Befehlscode	Parameterbereich	STEP5	Bemerkung
Wort			
---1 ---2 ---3			
6804 pppp	-32768,+32767	LRD	N
6805 pppp	-32768,+32767	TRD	N
6806		FDG	
6807		KZD	
6808		DUF	
680A		DUD	
680C		DEF	
680E		DED	
6819		MAS	N
6829		MAB	N
6849		MSA	N
6869		MSB	N
6889		MBA	N
6899		MBS	N
69pp	0-15	SRW	
6App	0-255	LBA	
6Bpp	0-255	TBA	
6Cpp	0-255	ZVZ	P
6Dpp	1-255	SPAOB	
6Epp	0-255	BDW	
6Fpp	0-255	TBT	N
7002		TAK	
7003		STP	
700B pppp	-32768,+32767	SPR	
71pp	0-32	SVD	
72pp	0-255	LPY	
73pp	0-255	TPY	
74pp	0-32	RRD	
75pp	0-255	SPAPB	
76pp	1-125	B=	
77pp	0-254	TQW	
7801 xxpp	0-255	BAFX	
7802 xxpp	0-255	BABFX	
7803 xxpp	1-255	AXDX	
7804 xxpp	0-255	EXDX	
7805 xxpp	0-255	EDB	
7806 xxpp	0-31	SES	
7807 xxpp	0-31	SEF	
78p9 pppp	-32768,+32767	MBR	N
780A pppp	-32768,+32767	ABR	N
780D pppp	-32768,+32767	LBCB	N
780E pppp	-32768,+32767	LBGB	N
781D pppp	-32768,+32767	LBCW	N
781E pppp	-32768,+32767	LBGW	N
782D pppp	-32768,+32767	LBCD	N
782E pppp	-32768,+32767	LBGD	N
783D		ACR	N
783F 0qpp	0.0-255.15	UD	
783F 1qpp	0.0-255.15	OD	
783F 2qpp	0.0-255.15	UND	
783F 3qpp	0.0-255.15	OND	
783F 4qpp	0.0-255.15	SD	
783F 5qpp	0.0-255.15	RD	
783F 6qpp	0.0-255.15	=D	

Befehlscode	Parameterbereich	STEP5	Bemerkung
<i>Wort</i>			
---1 ---2 ---3			
785D pppp	-32768,+32767	LWCW	N
785E pppp	-32768,+32767	LWGW	N
786D pppp	-32768,+32767	LWCD	N
786E pppp	-32768,+32767	LWGD	N
788D pppp	-32768,+32767	TBCB	N
788E pppp	-32768,+32767	TBGB	N
789D pppp	-32768,+32767	TBCW	N
789E pppp	-32768,+32767	TBGW	N
78AD pppp	-32768,+32767	TBCD	N
78AE pppp	-32768,+32767	TBGD	N
78CD pppp	-32768,+32767	TSC	N
78CE pppp	-32768,+32767	TSG	N
78DD pppp	-32768,+32767	TWCW	N
78DE pppp	-32768,+32767	TWGW	N
78ED pppp	-32768,+32767	TWCD	N
78EE pppp	-32768,+32767	TWGD	N
7900		+F	
7App	0-254	LPW	
7Bpp	0-254	TPW	
7Cpp	0-255	RZ	P
7Dpp	0-255	SPASB	
7E00		BI	
7Fpp	0-255	TQB	
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
1000 0qqq pppp pppp	0.0-255.7	UM	
1000 1qqq pppp pppp	0.0-255.7	OM	
1001 0qqq pppp pppp	0.0-255.7	SM	
1001 1qqq pppp pppp	0.0-255.7	=M	
1010 0qqq pppp pppp	0.0-255.7	UNM	
1010 1qqq pppp pppp	0.0-255.7	ONM	
1011 0qqq pppp pppp	0.0-255.7	RM	
<i>Wort</i>			
---1 ---2 ---3			
B8pp	0-255	UZ	P
B9pp	0-255	OZ	P
BAxx		U(	
BBxx		O(	
BCpp	0-255	UNZ	P
BDpp	0-255	ONZ	P
BFxx		)	
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
1100 0qqq 0ppp pppp	0.0-127.7	UE	
1100 0qqq 1ppp pppp	0.0-127.7	UA	
1100 1qqq 0ppp pppp	0.0-127.7	OE	
1100 1qqq 1ppp pppp	0.0-127.7	OA	
1101 0qqq 0ppp pppp	0.0-127.7	SE	
1101 0qqq 1ppp pppp	0.0-127.7	SA	
1101 1qqq 0ppp pppp	0.0-127.7	=E	
1101 1qqq 1ppp pppp	0.0-127.7	=A	



Befehlscode	Parameterbereich	STEP5	Bemerkung
<i>Bitmuster (Wort1)</i>			
5432 1098 7654 3210			
1110 0qqq 0ppp pppp	0.0-127.7	UNE	
1110 0qqq 1ppp pppp	0.0-127.7	UNA	
1110 1qqq 0ppp pppp	0.0-127.7	ONE	
1110 1qqq 1ppp pppp	0.0-127.7	ONA	
1111 0qqq 0ppp pppp	0.0-127.7	RE	
1111 0qqq 1ppp pppp	0.0-127.7	RA	
<i>Wort</i>			
---1 ---2 ---3			
F8pp	0-255	UT	P
F9pp	0-255	OT	P
FApp	-128,+127	SPB=	
FBxx		0	
FCpp	0-255	UNT	P
FDpp	0-255	ONT	P
FFxx		NOP 1	

## **ANHANG F: STEP5-Befehle, die in der CPU 928 nicht enthalten sind**

Beachten Sie bitte, daß die folgenden STEP5-Befehle der S5-150U in der CPU 928 nicht ablauffähig sind:

<b>BAS</b>	Befehlsausgabe sperren
<b>BAF</b>	Befehlsausgabe freigeben
<b>P</b> E, A, M, Z, T, D, BA, BB, BS, BT	Prüfe Bit auf '1'
<b>PN</b> E, A, M, Z, T, D, BA, BB, BS, BT	Prüfe Bit auf '0'
<b>SU</b> E, A, M, Z, T, D, BA, BB, BS, BT	Setze Bit unbedingt
<b>RU</b> E, A, M, Z, T, D, BA, BB, BS, BT	Rücksetze Bit unbedingt
<b>LIM</b>	Lade Interrupt-Maske
<b>SIM</b>	Setze Interrupt-Maske
<b>UBE</b>	Unterbrechungsbaustein-Ende
<b>STW</b>	Stoppbefehl der Weckalarm- bearbeitung
<b>AFS</b>	Adressierfehler-Interrupt sperren
<b>AFF</b>	Adressierfehler-Interrupt freigeben
<b>AAF</b>	Anforderungsalarmbearbei- tung freigeben
<b>AAS</b>	Anforderungsalarmbearbei- tung sperren

## **ANHANG G: Übersicht über die Kennungen der Programmbearbeitungsebenen**

Die Kennungen entsprechen den im USTACK unter 'EBENE' eingetragenen Kennungen (hexadezimal).

0002H - Neustart

0004H - Zyklus

0006H - Weckalarm 5sec  
0008H - Weckalarm 2sec  
000AH - Weckalarm 1sec  
000CH - Weckalarm 500ms  
000EH - Weckalarm 200ms  
0010H - Weckalarm 100ms  
0012H - Weckalarm 50ms  
0014H - Weckalarm 20ms  
0016H - Weckalarm 10ms

0018H - nicht belegt  
001AH - nicht belegt

001CH - Regelung

001EH - nicht belegt  
0020H - nicht belegt  
0022H - nicht belegt

0024H - Prozeßalarm

0026H - nicht belegt  
0028H - nicht belegt  
002AH - nicht belegt

002CH - Abbruch

002EH - nicht belegt

0030H - Weckfehler  
0032H - Reglerfehler  
0034H - Zyklusfehler

0036H - nicht belegt

0038H - Befehlscodefehler  
003AH - Laufzeitfehler  
003CH - Adressierfehler  
003EH - Quittungsverzug

0040H - nicht belegt  
0042H - nicht belegt

0044H - Manueller Wiederanlauf

0046H - Automatischer Wiederanlauf

## Angang H: Beispiel zur Auswertung des Unterbrechungsstacks

Dieses (stark vereinfachte) Beispiel verdeutlicht Ihnen eine mögliche Vorgehensweise bei der Auswertung des USTACK.

Beachten Sie bitte dazu auch das Kapitel 5.3 "Steuerbits und Unterbrechungsstack"!

Ausgangspunkt: Der Prozessor hat die zyklische Programmbearbeitung abgebrochen und ist in den Stoppzustand übergegangen.

Um die Ursache herauszufinden, wählen Sie am Programmiergerät die ONLINE-Funktion "Ausgabe USTACK" an.

Als erstes werden am PG die Steuerbits ausgegeben:

>>STP<< X	STP-6	FE-STP	BARBEND	PG-STP	STP-SCH	STP-BEF X	MP-STP
>>ANL<<	ANL-6	NEUST	M W A	A W A	ANL-2	NEU-ZUL X	MWA-ZUL X
<<RUN<<	RUN-6	EINPROZ X	BARB	OB1GEL X	FB0GEL	OBPROZA	OBWECKA
32KWRAM	16KWRA X	8KWRAM	EPROM	KM-AUS	KM-EIN	DIG-EIN X	DIG-AUS X
URGELOE	URL-IA	STP-VER	ANL-ABB	UA-PG	UA-SYS	UA-PRFE	UA-SCH
DX0-FE	FE-22	MOD-FE	RAM-FE	DB0-FE	DB1-FE	DB2-FE	KOR-FE
N A U	P E U	B A U	STUE-FE	Z Y K	Q V Z	A D F	WECK-FE
B C F	FE-6	FE-5	FE-4	FE-3	L Z F X	REG-FE	DOPP-FE

In den Steuerbits ist der aktuelle Betriebszustand des Prozessors vermerkt (>>STP<<), und es sind bestimmte Eigenschaften des Prozessors markiert (OB1 geladen, Einprozessorbetrieb, 16KW-Anwenderspeicher usw.). In der obersten Zeile ist als Ursache für den Stoppzustand 'STP-BEF' angekreuzt. Da wir in unserem Fall einen STP-Befehl im STEP-5-Anwenderprogramm ausschließen können, kommt nur noch ein Stoppbefehl vom Systemprogramm aufgrund eines nicht geladenen Fehler-OBs in Frage. In der untersten Zeile ist die Kennung 'LZF' markiert. Möglicherweise hat das Systemprogramm bei Auftreten eines Laufzeitfehlers festgestellt, daß der zugehörige Fehler-Organisationsbaustein nicht programmiert ist. Da es jedoch verschiedene Laufzeitfehler gibt, und wir nicht wissen, welcher davon aufgetreten ist, sind die Informationen aus den Steuerbits noch nicht ausreichend.

Wir lassen uns den USTACK ausgeben:

## UNTERBRECHUNGS-STACK

TIEFE: 01

BEF-REG: 0000 SAZ: 0000 DB-ADR: 0000 BA-ADR: 0000

BST-STP: 0001 -NR.: DB-NR: -NR.:  
REL-SAZ: DBL-REG: 0000

EBENE: 003A UAMK: 0120 UALW: 0000

AKKU1: 0000 1A01 AKKU2: 0000 0000 AKKU3: 0000 0000 AKKU4: 0000 0000

ERGEBNISANZEIGE: ANZ1 ANZ0 OVFL OVFLS ODER STATUS VKE  $\overline{\text{ERAB}}$

STOERUNGSURSACHE: NAU PEU BAU MPSTP ZYK QVZ ADF STP  
X

BCF S-6 LZF REG STUEB STUEU WECK DOPP

Der USTACK mit der Tiefe 01 repräsentiert diejenige Programmbearbeitungsebene, die als letzte vor dem Übergang in den Stoppzustand aktiviert war. An der Kennung '3A'H (hinter EBENE) sehen wir, daß dies der USTACK der Programmbearbeitungsebene LAUFZEITFEHLER ist. Im Akku 1 ist die Fehlerkennung '1A01'H hinterlegt. Damit wissen wir, daß es zu einem Laufzeitfehler kam aufgrund eines nicht geladenen Datenbausteins beim Befehl 'A DB'. Da der zugehörige Fehler-OB 19 in unserem Anwenderprogramm nicht vorhanden ist, hat das Systemprogramm die Programmbearbeitung abgebrochen (STP). Im Unterbrechungsanzeigen-Maskenwort UAMK sind die Unterbrechungsursachen mitgeführt: Die Kennung '0120'H entspricht dem Bitmuster '0000 0001 0010 0000'. Bit 2<sup>5</sup> (LZF) und Bit 2<sup>8</sup> (STP) sind gesetzt.

Jetzt gilt es herauszufinden, in welchem Baustein und durch welchen Befehl der Laufzeitfehler verursacht worden ist.

Durch Weiterschalten erscheint am PG der USTACK mit der Tiefe 02:

TIEFE: 02

BEF-REG: 2006      SAZ: 0037    DB-ADR: 0000    BA-ADR: 0000

BST-STP: 0001      OB-NR.: 1    DB-NR: -NR.:

REL-SAZ: 0004    DBL-REG: 0000

EBENE: 0004      UAMK: 0020    UALW: 0000

AKKU1: 0000 0001    AKKU2: 0000 0000    AKKU3: 0000 0000    AKKU4: 0000 0000

ERGEBNISANZEIGE:    ANZ1 ANZ0 OVFL OVFLS ODER STATUS VKE  $\overline{\text{ERAB}}$

STOERUNGSURSACHE:    NAU    PEU    BAU    MPSTP    ZYK    QVZ    ADF    STP

BCF    S-6    LZF    REG    STUEB    STUEU    WECK    DOPP

X

An der Kennung '04'H (hinter EBENE) sehen wir, daß dies der USTACK der unterbrochenen Programmbearbeitungsebene ZYKLUS ist. Der STEP-Adreßzähler (SAZ) zeigt auf die Adresse '37'H. Auf dieser Absolutadresse ist der fehlerverursachende Befehl im Anwenderspeicher hinterlegt. Die Unterbrechung ist im Organisationsbaustein OB 1 aufgetreten. Innerhalb des OB 1 liegt der fehlerverursachende Befehl auf der Relativadresse '04'H (REL-SAZ). Dieser Befehl führte, wie wir bereits festgestellt haben zu einem Laufzeitfehler (siehe UAMK, Bit 2<sup>5</sup> und STOERUNGSURSACHE).

Wir lassen uns jetzt am Programmiergerät über die ONLINE-Funktion "SUCHLAUF" den fehlerhaften Befehl ausgeben. Wir geben dazu den betreffenden Baustein (OB 1) und die Relativadresse des Befehls ein.

F 1	F 2	F 3	F 4	F 5	F 6	F 7	F 8
SYMB.ANZ.	BIB.NR.	SUCHLAUF					

AUSGABE GERAET:    AG BAUST: OB 1      SUCHLAUF: 4H

Am PG sehen wir nach erfolgtem Suchlauf den Befehl 'A DB 6', der für die Unterbrechung verantwortlich ist, da ein Datenbaustein mit der Nummer 6 im Anwenderspeicher nicht vorhanden ist.

OB 1

NETZWERK 1      0000

0004      :A    DB    6

0005      :

0006      :

0007      :

0008      :BE

## **Stichwortverzeichnis**

### **1**

16-Bit-Festpunktzahlen, 2-5

### **3**

32-Bit-Festpunktzahlen, 2-5, 2-6

## **A**

Abtastzeit, 4-23

Abtastzeitfehler, 5-39

Adreßraumaufteilung

CPU 928, 8-2

System-RAM, 8-3

Adressierfehler, 5-34

AG URLÖSCHEN, 11-11

Akku 1 und 2

auswerten, 5-4

Akkumulatoren, 2-4, 3-10

löschen, 6-7

rollen, 6-8

Aktualoperanden, 2-23, 2-24, 2-26, 3-40

ANLAUF

auslösen, 4-9

Fehler, 5-21, 5-26

Unterbrechungen, 4-14

Anlaufarten

vergleichen, 6-50

Anlaufsynchronisation, 10-12

Anweisungsliste (AWL), 2-2

Anwenderkontrollpunkte, 11-1

Anwenderprogramm, 1-4, 2-3

Anwenderschnittstelle, 3-9

Anwenderspeicher, 8-7

ANZ1 und ANZ0, 3-12

Arithmetische Operationen, 3-22, 3-41

Ausgabe ADRESSE, 11-11

Ausgabe BSTACK, 5-4

Ausgabe USTACK, 5-2

Ausgangsbytes

steuern, 11-9

Ausgangskoppelmerker, 3-8, 10-5, 10-10

Automatischer Wiederanlauf, 4-13

## **B**

BA-Bereich, 8-12

BASP (Befehlsausgabesperre), 5-36, 10-13

Baustein, 2-3, 2-9

Anfangsadresse, 3-4

Aufruffehler, 5-30

Bausteinadreßliste (DB0), 3-4

Bausteinanfangsadressen (DB0), 8-8

Bausteinarten, 2-9

Bausteinaufrufe, 3-23

bedingt, 2-14, 2-26

Fehler, 5-30

unbedingt, 2-14, 2-26

Bausteine

korrigieren, 2-12

löschen, 2-12

Bausteingrenzen, 4-3

- Bausteinkopf, 2-10, 8-7
- Bausteinnummern, 2-11
- Bausteinparameter, 2-23, 2-24
- Bausteinrumpf, 2-11
- Bausteinstack (BSTACK), 3-5
  - lesen, 6-12
  - auswerten, 5-4
- Bausteinvorkopf, 2-11
- BB-Bereich, 8-12
- BCD-codierte Zahl, 2-8
- BEARBEITUNGSKONTROLLE, 11-5
- Bearbeitungsoperationen, 3-50
- Befehlscode, E-1
- Befehlscodefehler, 5-27
- Befehlsgrenzen, 4-3
- Bestückungsmöglichkeiten, 1-1
- Betriebszustände
  - ANLAUF, 4-9
  - RUN, 4-16
  - STOP, 4-6
  - Übersicht, 4-2
  - WARTEZUSTAND, 11-1
- Bit-Anzeigen, 3-11
  - Zugriff auf, 6-5
- Blocktransfer, 6-22, 6-24
- BR-Register, 9-19
  - laden, 9-19
- BS 3 und 4
  - auswerten, 5-3
- BS-Bereich, 8-12
- BT-Bereich, 8-12
- Byte (Darstellung), 6-34

## C

- Code-Bausteine, 1-4, 2-9
- CPU-Kennung, 8-26

## D

- Daten
  - technische, 11-1
- Datenbaustein DB 0, 2-35
- Datenbaustein DB 1, 2-35, 10-2, 10-9
- Datenbaustein DB 2, 2-36
- Datenbaustein DX 0, 2-36, 7-1
- Datenbausteine, 1-4, 2-9, 2-10, 2-31
  - ändern, 2-31
  - Aufbau, 2-31
  - aufschlagen, 2-33
  - duplizieren, 6-30
  - erzeugen, 3-52
  - Gültigkeitsbereich, 2-34
  - programmieren, 2-32
  - testen, 6-20
  - übertragen, 6-29
  - verschieben, 6-29
  - Zugriff auf, 2-33, 6-16
- Datenformate, 2-32
- Datentransfer, 10-4, 10-5, 10-6
  - geschützter, 10-8
- DB1-Fehler, 10-11
- DB-RAM, 6-57



DB0-Fehler, 5-22  
DB1-Fehler, 5-22  
DB2-Fehler, 5-24  
DBA-Register, 9-7  
DBL-Register, 9-10  
Dekrementieren, 3-48  
Dezimalzahlen, 2-5  
Digitale Ausgänge, 10-10  
Digitale Eingänge, 10-10  
Doppeladressierung, 10-6  
Doppelwort (Darstellung), 6-34  
Doppelwortwandlung, 3-46  
Dualzahlen, 2-5  
DX0-Aufbau, 7-2  
DX0-Fehler, 5-25

## **E**

Eingangskoppelmerker, 3-8, 10-5, 10-10  
EPROM-Modul, 2-3, 3-6  
ERAB (Erstabfrage), 3-11, 3-13  
Ergänzende Operationen, 2-1, 3-37  
Ergebnisanzeigen, 3-11  
Erweiterungsgeräte, 3-17  
Exponent, 2-6  
Exponentialzahl, 2-6

## **F**

Fehler-LEDs, 5-2  
Fehler-OBs  
    Unterbrechungen, 5-20  
Fehlerbehandlung  
    über Organisationsbausteine, 5-18  
Fehlerdiagnose, 5-2  
Fehlerkennungen  
    Übersicht, B-1  
Fehlerschachtelung, 5-12  
Fehlerursachen  
    suchen, 5-6  
Festpunkt-Gleitpunkt-Wandlung, 3-47  
Festpunktwandlung, 3-46  
Formaloperanden, 2-22  
Funktionen  
    neue, 1-7  
Funktionsbaustein FB 0, 2-30  
Funktionsbausteine, 2-10, 2-19  
    ändern, 2-25  
    Aufbau, 2-20  
    aufrufen, 2-26  
    programmieren, 2-22  
    parametrieren, 2-26  
Funktionsplan (FUP), 2-2

## **G**

Gesamtzykluszeit, 3-7  
Gleitpunktzahlen, 2-6  
    eingeben, 2-7  
GRAPH 5, 2-2  
Grundoperationen, 2-1

## **H**

Haltepunkt, 11-4, 11-5  
Hantierungsbausteine, 6-56

## **I**

Initialisierung, 5-21  
  Fehler, 5-21  
Inkrementieren, 3-48  
Interruptleitung, 4-24

## **K**

Kacheln, 6-32  
Kachelspeicher  
  -zugriffe, 9-24  
Kommunikationsprozessor (CP), 10-6  
Komplementbildung, 3-46  
Kontaktplan (KOP), 2-2  
Koordinator, 10-2, 10-5  
  starten, 10-12  
Koppelmerker, 3-7, 10-4  
  übertragen, 6-50, 10-7

## **L**

Ladeoperationen, 3-14, 3-39  
Laufzeit, 3-7  
Laufzeitfehler  
  sonstige, 5-32  
Linkspunktzahl, 6-74, 6-78

## **M**

Mantisse, 2-6  
Manueller Wiederanlauf, 4-12  
Mehrprozessorbetrieb, 10-1, 10-9  
  Anlauf, 10-12  
Mehrprozessorkommunikation, 10-8  
Merkerbytes  
  rückschreiben, 6-24  
  übertragen, 6-22

## **N**

Netzspannungsausfall (NAU), 4-10, 4-13, 4-14, 10-12  
  im Anlauf, 4-14  
Netzspannungswiederkehr, 4-10, 4-13, 10-12  
Neustart, 4-12  
  auslösen, 4-9  
Nulloperationen, 3-23

## **O**

On-line-Funktionen, 11-1  
Operandenteil, 2-4  
Operationscodefehler, 5-28  
Operationsteil, 2-4  
OR (Oder), 3-11  
Organisationsbausteine, 2-9  
  Aufbau, 2-13  
  aufrufen, 2-14  
  programmieren, 2-13  
  Sonderfunktionen, 2-18  
  spezielle, 2-16  
OS (Overflow speichernd), 3-12  
OV (Overflow), 3-12

## P

- P-Peripherie, 3-17
- P-Regler, 6-76
- Parameterfehler, 5-29
- PD-Regler, 6-76
- Peripherie, 3-17, 8-5
  - adressen, 6-33, 8-4
  - zugriffe, 8-6
- Peripheriebaugruppen, 10-9, 10-11
- Peripheriebus, 10-7
- PI-Regler, 6-76
- PID-Algorithmus, 6-67, 6-72
- PID-Regler, 6-65
  - Abkürzungen, 6-77
  - Übergabe-DB, 6-69
- Programmbausteine, 2-9
  - Aufbau, 2-13
  - aufrufen, 2-14
  - programmieren, 2-13
- Programmbearbeitung
  - alarmgesteuerte, 4-24
  - beeinflussen, 3-9
  - Regler-, 4-23
  - zeitgesteuerte, 4-18
  - zyklische, 3-6, 4-17
- Programmbearbeitungsebenen
  - Kennungen, G-1
  - Merkmale, 4-3
  - Priorität, 4-3
  - Übersicht, 4-2
- Programmierbeispiele, 3-24
- Programmorganisation, 3-1
- Prozeßabbild, 3-17, 5-34, 5-35, 8-6
- Prozeßabbild der Eingänge, 3-7
- Prozeßabbild der Ausgänge, 3-7
- Prozeßalarme
  - flankengetriggert, 4-25
  - freigeben, 3-52, 4-26
  - pegelgetriggert, 4-25
  - sperrern, 3-52, 4-26, 6-44
  - Unterbrechungen, 4-24
  - verzögern, 6-44
- Pseudobefehlsgrenzen, 6-1, 9-14

## Q

- Q-Peripherie, 3-17
- Quittungsverzug, 5-35

## R

- RAM-Modul, 3-5
- Reaktionszeit, 4-27
- Rechenoperationen, 3-22, 3-41
- Register, 1-6
  - zugriffe, 9-5
  - verschieben, 9-20
- Registerbelegung, 9-5
- Regler-Alarm
  - Unterbrechungen, 4-23
- Reglerausgänge, 6-73
- Reglereingänge, 6-73
- Reglerfehler, 5-37

Reglerkenngrößen, 6-76  
Reglerstruktur R64, 4-23  
Restzyklus, 4-12, 4-14, 4-16  
Rücksprungadresse, 3-4  
RUN

    Fehler, 5-26

## S

SAZ (Step-Adreßzähler), 9-11  
Schachtelungstiefe, 1-7, 3-4  
Schiebeoperationen, 3-44  
Schieberegister, 6-57  
    bearbeiten, 6-63  
    initialisieren, 6-60  
    löschen, 6-64  
Schleifenzähler, 6-10  
Schmiermerker, 4-27  
Schrittbausteine, 2-10  
    Aufbau, 2-13  
    aufrufen, 2-14  
    programmieren, 2-13  
Semaphoren  
    Anwendungsbeispiel, 3-56  
    freigeben, 3-53  
    setzen, 3-53  
Sonderfunktionen, 6-1  
    Fehler, 6-1  
    Übersicht, 6-3  
Speicher KOMPRIMIEREN, 11-10  
SPEICHERAUSBAU, 11-11  
Speicherbereiche, 1-5  
Speicherblöcke  
    transferieren, 9-13  
Speicheroperationen, 3-14, 3-37  
Speicherorganisation, 9-1  
Speicherzugriffe  
    absolute, 9-1  
Sprungoperationen, 3-43  
STA (Status), 3-11  
Standardfunktionsbausteine, 2-29  
STATUS VARIABLEN, 11-3  
Steckplätze, 10-2  
Steckplatzkennung, 8-25  
STEP5-Gesamtoperationsvorrat, C-1  
STEP5-Operationen, 2-4  
    binäre, 3-10  
    digitale, 3-10  
    organisatorische, 3-10  
Steuerbits  
    Abkürzungen, 5-8  
    auswerten, 5-7  
STEUERN, 11-9  
STEUERN VARIABLEN, 11-9  
Stop-Anweisung, 3-23  
STOP-LED  
    Dauerlicht, 4-6  
    langsames Blinken, 4-7  
    schnelles Blinken, 4-7  
Stoppzustand, 4-6  
    Ausgänge, 4-8  
Substitutionsfehler, 5-28

- Systemdatenbelegung, 8-13
- Systemdatenwort BS 3 + 4
  - auswerten, 5-3
- Systemkontrollpunkte, 11-1
- Systemoperationen, 2-1, 3-37
- Systemprogramm, 1-4, 2-3
  - Quersumme lesen, 6-52

## T

- Technische Daten 135U, A-1
- Testbetrieb, 10-13
  - auslösen, 10-13
  - Besonderheiten, 10-13
- Transferfehler, 5-31
- Transferoperationen, 3-14, 3-39

## U

- Umwandlungsoperationen, 3-46
- Unterbrechungsanzeigen-Löschwort (UALW), 8-18
- Unterbrechungsanzeigen-Sammelwort (UAMW), 8-19, 8-20
- Unterbrechungsstelle, 5-7
- Unterbrechungsursachen, 5-21, 5-26, 5-27
- Urlöschen
  - anfordern, 4-7
  - durchführen, 4-8
- USTACK
  - Abkürzungen, 5-12
  - auswerten, 5-12
  - auswerten (Beispiel), 5-16, H-1

## V

- Vergleichsoperationen, 3-14
- Verknüpfungen
  - binäre, 3-13, 3-37
  - digitale, 3-42
- VKE (Verknüpfungsergebnis), 2-14, 2-15, 3-11, 3-13
- Vorzeichen, 2-6, 2-8
  - erweitern, 6-43

## W

- Wartezustand
  - Eigenschaften, 11-2
- Weckalarme
  - Priorisierung, 4-19
  - sperrern, 6-44
  - Unterbrechungen, 4-20
  - verzögern, 6-43
- Weckfehler, 4-20, 4-21, 5-37
- Wiederanlauf
  - auslösen, 4-10
- Wort (Darstellung), 6-34
- Wort-Anzeigen, 3-11, 3-12
  - Zugriff auf, 6-5

## **Z**

Zahlendarstellungen, 2-5  
Zähloperationen, 3-18, 3-38  
Zählschleifen, 6-10  
Zählwert, 3-19  
Zeitenblocklänge, 10-10, 10-11  
Zeitoperationen, 3-18, 3-38  
Zeitraster, 3-19  
Zeitwert, 3-19  
Zwischenmerker, 4-27  
Zyklus, 4-17  
    Unterbrechungen, 4-18  
Zykluszeit, 1-3, 3-7, 5-36  
    einstellen, 6-49  
    neu starten, 6-49  
Zykluszeitfehler, 5-36

## **Verzeichnis der Abbildungen, Beispiele und Übersichten**

### Kap. 1

Abb.: Typische Anlagenstruktur AG S5-135U.....1-2

### Kap. 2

Abb.: Darstellungsarten der Programmiersprache STEP5.....2-2

Abb.: Ablage der Bausteine in den Programmspeicher.....2-11

Abb.: Aufbau eines Organisations-, Programm- und Schrittbausteins.2-13

Abb.: Bausteinaufrufe, die die Bearbeitung eines Programmbausteins  
freigeben.....2-14

Übersicht: Sonderfunktions-Organisationsbausteine in der CPU 928..2-18

Abb.: Aufbau eines Funktionsbausteins (FB/FX).....2-20

Beispiel: Programmierung eines Funktionsbausteins.....2-25

Abb.: Aufruf und Parametrierung eines Funktionsbausteins.....2-27

Abb.: Aufbau eines Datenbausteins.....2-32

Abb.: Aufschlagen von Datenbausteinen und Zugriff auf Datenwörter.2-34

Abb.: Gültigkeitsbereich eines aufgerufenen Datenbausteins.....2-35

### Kap. 3

Beispiel: Organisation des Anwenderprogramms nach Programmstruktur.3-2

Beispiel: Organisation des Anwenderprogramms nach Anlagenstruktur..3-3

Beispiel: Bausteinschachtelungstiefe und Bausteinstack (BSTACK)....3-5

Abb.: Ablage der Bausteine in den Programmspeicher.....3-6

Abb.: Zyklische Programmbearbeitung.....3-8

### Kap. 4

Übersicht: Betriebszustände und Programmbearbeitungsebenen.....4-2

Abb.: Unterbrechungsgesteuerte Programmbearbeitung an  
Bausteingrenzen.....4-27

### Kap. 5

Abb.: Beispiel für den USTACK-Aufbau.....5-16

### Kap. 6

Abb.: Prinzipskizze des Schieberegisters mit 3 Zeigern und  
12 Speicherzellen.....6-57

Abb.: Prinzipskizze des Schieberegisters mit 3 Zeigern und  
12 Speicherzellen vor dem ersten Takt.....6-58

Abb.: Prinzipskizze des Schieberegisters mit 3 Zeigern und  
12 Speicherzellen nach dem ersten Takt.....6-59

Abb.: Aufbau des Datenbausteins zur Initialisierung eines  
Schieberegisters.....6-60

Abb.: Blockschaltbild des PID-Reglers.....6-65

### Kap. 7

Abb.: Aufbau des DX 0.....7-2

### Kap. 8

Abb.: Adreßraumaufteilung in der CPU 928.....8-2

Abb.: Adreßraumaufteilung System-RAM (16 Bits).....8-3

Abb.: Adreßraumaufteilung Peripherie (8 Bits).....8-4

Abb.: Adreßraumaufteilung für Peripherie/Programmierung.....8-5

### Kap. 9

Abb.: Globaler und lokaler Speicher.....9-2

Abb.: Belegung der Akkumulatoren während des Programmablaufs.....9-12

Beispiel: Löschen aller Merkerbytes (MB0 bis MB255) .....9-12

# SIEMENS



# SIEMENS

## SIMATIC S5

Mehrprozessor-Kommunikation

AG S5-135U, CPU 922 (R-Prozessor),

CPU 928 und CPU 928B

AG S5-155U, CPU 946/947

---

Bedienungsanleitung

C79000-B8500-C468-05

---

	Seite
<b>1</b>	<b>Einführung ..... 3</b>
<b>1.1</b>	<b>Konfiguration..... 4</b>
<b>1.2</b>	<b>Prinzip ..... 4</b>
<b>1.3</b>	<b>Sender-/Empfänger-Identifikation ..... 5</b>
<b>1.4</b>	<b>Zwischenspeicherung der Daten ..... 6</b>
<b>1.5</b>	<b>System-Anlauf ..... 9</b>
<b>1.6</b>	<b>Aufruf und Schachtelung der Sonderfunktions- Organisationsbausteine OB 200 und OB 202 bis OB 205 ..... 10</b>
<b>1.7</b>	<b>Parallelverarbeitung im Mehrprozessor-Automatisierungsgerät ..... 11</b>
<b>1.8</b>	<b>Belegte Bereiche ..... 11</b>
<b>1.9</b>	<b>Laufzeit..... 12</b>
<b>2</b>	<b>Parametrierung..... 14</b>
<b>2.1</b>	<b>Auswerten der Ausgangsparameter ..... 14</b>
2.1.1	Ergebnisanzeigen ..... 15
2.1.2	Anzeigenbyte: Initialisierungskonflikt/Fehler/Warnung ..... 16
<b>3</b>	<b>Funktion INITIALISIEREN (OB 200) ..... 21</b>
<b>3.1</b>	<b>Eingangsparameter..... 23</b>
3.1.1	Betriebsart (Automatisch/Manuell) ..... 23
3.1.2	Anzahl CPUs..... 24
3.1.3	Baustein-Kennung und -Nummer / Anfangsadresse der Zuordnungsliste..... 24
<b>3.2</b>	<b>Ausgangsparameter ..... 26</b>
3.2.1	Anzeigenbyte Initialisierungskonflikt ..... 26
3.2.2	Gesamt-Kapazität ..... 28
<b>4</b>	<b>Funktion SENDEN (OB 202) ..... 29</b>
<b>4.1</b>	<b>Eingangsparameter..... 29</b>
4.1.1	Empfangs-CPU ..... 29
4.1.2	Baustein-Kennung und -Nummer / Block-Nummer..... 29
<b>4.2</b>	<b>Ausgangsparameter ..... 31</b>
4.2.1	Anzeigenbyte Fehler/Warnung..... 31
4.2.2	Sende-Kapazität..... 33

<b>5</b>	<b>Funktion SENDE-TEST (OB 203)</b>	<b>34</b>
<b>5.1</b>	<b>Eingangsparameter</b>	<b>34</b>
5.1.1	Empfangs-CPU	34
<b>5.2</b>	<b>Ausgangsparameter</b>	<b>34</b>
5.2.1	Anzeigenbyte	34
5.2.2	Sende-Kapazität	35
<b>6</b>	<b>Funktion EMPFANGEN (OB 204)</b>	<b>36</b>
<b>6.1</b>	<b>Eingangsparameter</b>	<b>36</b>
6.1.1	Sende-CPU	36
<b>6.2</b>	<b>Ausgangsparameter</b>	<b>37</b>
6.2.1	Anzeigenbyte	37
6.2.2	Empfangs-Kapazität	38
6.2.3	Baustein-Kennung und -Nummer	38
6.2.4	Adresse des ersten/letzten empfangenen Datenwortes	39
<b>7</b>	<b>Funktion EMPFANGS-TEST (OB 205)</b>	<b>40</b>
<b>7.1</b>	<b>Eingangsparameter</b>	<b>40</b>
7.1.1	Sende-CPU	40
<b>7.2</b>	<b>Ausgangsparameter</b>	<b>40</b>
7.2.1	Anzeigenbyte	40
7.2.2	Empfangs-Kapazität	41
<b>8</b>	<b>Anwendungen</b>	<b>42</b>
<b>8.1</b>	<b>Aufruf der Sonderfunktions-OB über Funktionsbausteine</b>	<b>42</b>
8.1.1	Vorbesetzen der Verbindung (FB 200)	43
8.1.2	Senden eines Datenblocks (FB 202)	45
8.1.3	Sendemöglichkeit testen (FB 203)	47
8.1.4	Empfangen eines Datenblocks (FB 204)	48
8.1.5	Empfangsmöglichkeit testen (FB 205)	50
<b>8.2</b>	<b>Übertragen von Datenbausteinen</b>	<b>51</b>
8.2.1	Funktionsbeschreibung	51
8.2.2	Übertragen eines Datenbausteins (FB 110)	51
8.2.3	Anwendungsbeispiel (für AG S5-135U)	54
<b>8.3</b>	<b>Erweiterung des Koppelmerkerbereichs</b>	<b>56</b>
8.3.1	Problemstellung	56
8.3.2	Lösung	57
8.3.3	Datenstruktur	57
8.3.4	Programmstruktur	60
8.3.5	Senden von Datenwortbereichen (FB 100)	62
8.3.6	Empfangen von Datenwortbereichen (FB 101)	65
8.3.7	Anwendungsbeispiel (für AG S5-135U)	68

# 1 Einführung

Die Mehrprozessor-Automatisierungsgeräte S5-135U und S5-155U können Sie jeweils mit bis zu vier CPUs bestücken. Für den Datenaustausch zwischen den CPUs stehen folgende Hilfsmittel ("Werkzeuge") zur Verfügung, die Sie sowohl einzeln als auch miteinander kombiniert einsetzen können:

- M-Merker werden übertragen, indem sie in **einer** CPU als Ausgangskoppelmerker und in einer (oder mehreren) anderen CPU(s) als Eingangskoppelmerker definiert werden.
- Für die Übertragung von Datenbausteinen, genauer: von Datenblöcken mit einer Größe von max. 64 byte ( = 32 Datenwörtern), stehen Ihnen - in die CPU integriert - folgende Sonderfunktionen zur Verfügung:

<b>INITIALISIEREN (OB 200):</b>	Vorbesetzen
<b>SENDEN (OB 202):</b>	Senden eines Datenblocks
<b>SENDE-TEST (OB 203):</b>	Sendemöglichkeit testen
<b>EMPFANGEN (OB 204):</b>	Empfangen eines Datenblocks
<b>EMPFANGS-TEST (OB 205):</b>	Empfangsmöglichkeit testen

Für die Anwendung dieser Funktionen genügen Grundkenntnisse über die Programmiersprache STEP 5 und über die Arbeitsweise von SIMATIC S5-Automatisierungsgeräten. Diese Grundkenntnisse werden in den im Literaturverzeichnis aufgeführten Schriften vermittelt.

Während die Koppelmerker "automatisch" vom Systemprogramm aktualisiert werden, sind die Funktionen **INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN UND EMPFANGS-TEST** von Ihnen mit den Befehlen **SPA OB** bzw. **SPB OB** als Sonderfunktions-Organisationsbausteine aufzurufen.

## 1.1 Konfiguration

### AG S5-135U oder AG S5-155U

Diese AGs enthalten den S5-Bus und im Mehrprozessorbetrieb die folgenden Komponenten:

- **1 Koordinator 923 C (KOR C)**

Diese Baugruppe enthält vier sogenannte Kacheln. Das sind Speicherbereiche mit einer Größe von jeweils 1024 bytes. Sie belegen alle den gleichen Adressraum F400H bis F7FFH. Die Auswahl (Adressierung) der "aktuellen" Kachel erfolgt durch das sogenannte Select- (bzw. Ident-, oder auch Kacheladreß-)Register (ähnlich Chip-Select). Für die vier Kacheln des Koordinator 923C sind die Select-Nummern 252, 253, 254 und 255 **fest vergeben**; sie dienen der hier vorgestellten Mehrprozessor-Kommunikation.

- **2 bis 4 CPUs**

Für AG S5-135U: CPUs 922 (R-Prozessoren), CPUs 928 , CPUs 928B oder CPUs 920 (M-Prozessoren)

Für AG S5-155U: CPUs 946/947, CPUs 922 (R-Prozessoren), CPUs 928 , CPUs 928B oder CPUs 920 (M-Prozessoren).

Diese CPUs können in beliebiger Kombination miteinander Daten austauschen; die gleichzeitige Verwendung von "Hantierungsbausteinen" (sie benutzen ebenfalls die Kacheladressierung) ist uneingeschränkt möglich.

Im gleichen Rahmen zusätzlich gesteckte CPUs 921 (S- Prozessoren) können Sie an der hier beschriebenen Mehrprozessorkommunikation **nicht** beteiligen. Die S-Prozessor-Hantierungsbausteine dürfen Sie nicht aufrufen , solange die R- und M-Prozessoren, die CPUs 928 , CPUs 928B und CPUs 946/947 ihre Hantierungsbausteine oder die Mehrprozessor-Kommunikation bearbeiten. Eine Kommunikation zwischen CPUs kann aber in jedem Fall mittels Koppelmerker stattfinden.

## 1.2 Prinzip

Um Daten zu übertragen, müssen Sie auf der Sende-CPU die Funktion SENDEN aktivieren und auf der Empfangs-CPU die Funktion EMPFANGEN.

Dabei werden aufeinanderfolgende Datenwörter eines DB- oder DX-Datenbausteins, die sich in der Sende-CPU befinden, über den Koordinator 923C zur Empfangs-CPU transportiert und dort im DB- bzw. DX-Datenbaustein mit derselben Nummer und unter derselben Datenwort-Adresse abgelegt; d.h. es handelt sich um ein "1:1"-Kopieren.

**Beispiel:**

	<b>Sende-Daten in der Sende-CPU</b>	<b>Empfangs-Daten in der Empfangs-CPU</b>
<b>Daten-Baustein</b>	DB 17	DB 17
<b>Datenwort-Adresse</b>	DW 32 bis DW 63	DW 32 bis DW 63

Die Datenmenge, welche mittels der Funktionen SENDEN bzw. EMPFANGEN übertragen werden kann, beträgt normalerweise 32 Wörter.

Falls die Baustein-Länge (ohne Kopf) kein Vielfaches von 32 Wörtern beträgt, so werden beim letzten Block ausnahmsweise weniger als 32 Wörter übertragen.

Der Datenbaustein in der Empfangs-CPU kann länger oder kürzer sein als der Sende-Datenbaustein. Entscheidend ist, daß die von der Funktion SENDEN übertragenen Datenwörter im Empfangs-Datenbaustein existieren; andernfalls erkennt die Funktion EMPFANGEN einen Fehler.

### 1.3 Sender-/Empfänger-Identifikation

Die CPUs werden derart durchnummeriert, daß die am weitesten links gesteckte CPU die Nummer 1 erhält, die nach rechts folgenden CPUs erhalten eine jeweils um eins erhöhte Nummer.

**Beispiel:**

AG S5-135U/155U:

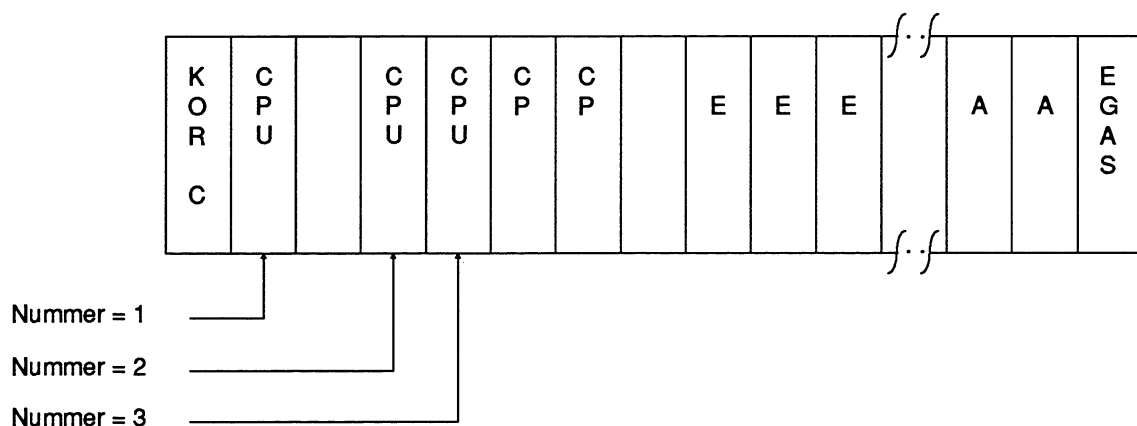


Bild 1 Sender-/Empfänger-Identifikation

## 1.4 Zwischenspeicherung der Daten

Die Zykluszeit einer CPU wird bestimmt von der Menge der zu bearbeitenden Aufgaben und den spezifischen Leistungsmerkmalen der CPU. In die Zykluszeit gehen u. a. ein:

- Der Umfang der einzelnen Programmteile,
- Die Häufigkeit, in der die Programmteile durchlaufen (mehrfache Aufrufe, Schleifen),
- Die Anzahl der projizierten Regler (bei CPU 922, CPU 928, CPU 928B).

Außerdem **variiert** die Zykluszeit einer CPU abhängig von bedingten Baustein-Aufrufen (z.B. SPB PB xy), dem Auftreten von Alarmen (z.B. alarmgesteuerte Bearbeitung über OB 2) und dgl. mehr. D.h.: Im Mehrprozessorbetrieb ist die zyklische Programmbearbeitung in jeder einzelnen CPU **asynchron** zu den zyklischen Programmbearbeitungen der restlichen CPUs.

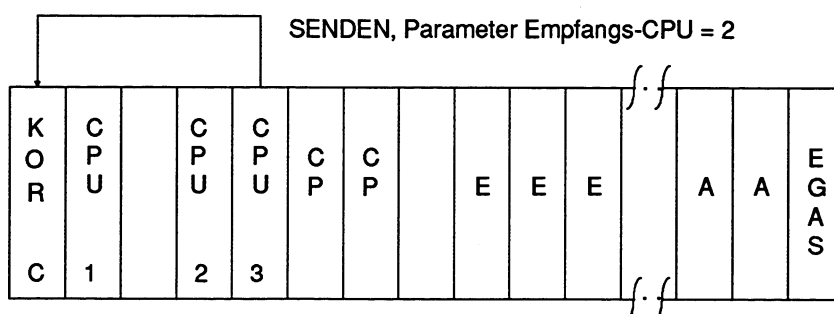
Im Gegensatz zur zyklischen Programmbearbeitung wird die zeitgesteuerte Bearbeitung nach einem Taktsignal periodisch bearbeitet, beispielsweise alle 100 ms (OB 13). Bei diesem Beispiel kann das Taktsignal einer CPU gegenüber einer anderen CPU einen zeitlichen Versatz von bis zu 100 ms aufweisen.

Aufgrund dieser asynchronen Abläufe werden die zu übertragenden Daten im Koordinator 923C zwischengespeichert.

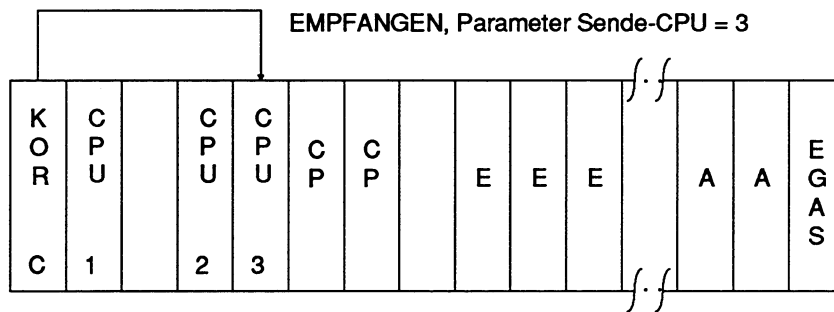
Die Nummer der "eigenen" CPU sowie die Nummer eines Empfängers (Funktion SENDEN) bzw. die Nummer eines Senders (Funktion EMPFANGEN) legen Quelle und Ziel fest.

### Beispiel: Datenübertragung von CPU 3 nach CPU 2

1. Schritt:



## 2. Schritt:



- 1. Schritt: Dem Zwischenspeicher liegt das FIFO-Prinzip zugrunde (First in - first out, Warteschlangen-Prinzip). Somit ist die Empfangsreihenfolge gleich der Sendereihenfolge. Dies gilt für jede einzelne Verbindungsstrecke (gekennzeichnet durch Sende- und Empfangs-CPU) und ist von den anderen Verbindungen unabhängig.
- 2. Schritt: Der Zwischenspeicher ist batteriegepuffert; damit ist der "Automatische Wiederanlauf nach Netzausfall" uneingeschränkt möglich. Durch einen Netzausfall während einer "laufenden" Datenübertragung gehen im Automatisierungsgerät keine Daten verloren.

Die Speicherkapazität des Koordinators 923C beträgt 48 Blöcke mit je 32 Wörtern. Die Funktion INITIALISIEREN teilt diese Speicherblöcke den einzelnen Verbindungsstrecken zu.

Jeder **Speicherblock** (die Länge beträgt immer 32 Wörter) nimmt genau einen **Datenblock** auf (mit einer Länge zwischen einem Datenwort und 32 Datenwörtern). Ein **Datenblock** wird von einem SENDEN-Baustein in einen **Speicherblock** eingetragen und von einem EMPFANGEN-Baustein wieder ausgetragen.

Die Anzahl der für eine Verbindungsstrecke vergebenen Speicherblöcke steht in direktem Zusammenhang mit den Parametern Sende-Kapazität (Funktion SENDEN, SENDE-TEST) und Empfangs-Kapazität (Funktion EMPFANGEN, EMPFANGS-TEST).

Die Sende-Kapazität gibt an, wieviele der für eine Verbindungsstrecke reservierten Speicherblöcke zu einem bestimmten Zeitpunkt frei sind.


Die Empfangs-Kapazität gibt an, wieviele der für eine Verbindungsstrecke reservierten Speicherblöcke zu einem bestimmten Zeitpunkt belegt sind.

Die Summe aus Sende- und Empfangs-Kapazität ist zu jedem Zeitpunkt gleich der Anzahl der für eine Verbindungsstrecke vergebenen Speicherblöcke.



**Beispiel:**

Möglicher Ablauf der Datenübertragung nachdem der Verbindungsstrecke "Von CPU 3 nach CPU 2" mit Hilfe der Funktion INITIALISIEREN sieben Speicherblöcke zugewiesen worden sind.

Sender: CPU 3		Empfänger: CPU 2		
Ablauf	Sende-Kapazität (freie Speicherblöcke)	Zeit	Empfangs-Kapazität (belegte Speicherblöcke)	Ablauf
Senden eines Datenblocks (A)	7		0	Initialisieren
Senden von vier Datenblöcken (B,C,D,E)	6		1	
	2		5	Empfangen von zwei Datenblöcken (A, B)
Senden von vier Datenblöcken (F,G,H,I)	4		3	
	0		7	Empfangen von fünf Datenblöcken (C,D,E,F,G)
Senden von zwei Datenblöcken (K,L)	5		2	Empfangen von zwei Datenblöcken (H,I)
	5		2	

**BEACHTEN SIE**



Senden/Empfangen von n Datenblöcken bedeutet, daß die entsprechende Funktion n-mal nacheinander aufgerufen wird.

Um eine einfachere Darstellung zu erreichen, wird in diesem Beispiel zunächst entweder gesendet oder empfangen.  
Das gleichzeitige Senden (CPU 3) und Empfangen (CPU 2) ist jedoch möglich und sinnvoll (siehe Kapitel "Parallelverarbeitung im Mehrprozessor-Automatisierungsgerät"). Im Beispiel werden während des Sendens der Datenblöcke K und L die Datenblöcke H und I empfangen.

Das Beispiel verdeutlicht die Warteschlangen-Organisation des Zwischenspeichers: Die zuerst gesendeten Datenblöcke (A,B,C...) werden zuerst empfangen (A,B,C...).

### Zusammenfassung:

Das Zwischenspeichern im Koordinator KOR 923C hat die Aufgabe, die asynchronen Abläufe von Sende- und Empfangs-CPU und deren unterschiedliche Arbeitsgeschwindigkeiten auszugleichen.

Da die Kapazität des Zwischenspeichers begrenzt ist, sollte der Empfänger "häufig" und "regelmäßig" überprüfen, ob Daten gespeichert sind (Funktion EMPFANGS-TEST, Empfangs-Kapazität > 0) bzw. versuchen, gespeicherte Daten abzuholen (Funktion EMPFANGEN). Zweckmäßigerweise ist die Funktion EMPFANGEN solange wiederholt aufzurufen, bis die Empfangs-Kapazität Null wird. Ein solcher Ablauf bewirkt, daß die gesendeten Daten nicht lange zwischengespeichert bleiben, sondern dem Empfänger aktuell zur Verfügung stehen. Zusätzlich werden damit Speicherblöcke frei (die Sende-Kapazität steigt); ein Blockieren des Senders (d.h. die Sende-Kapazität ist "erschöpft", ist Null) wird verhindert.

Während die Empfangs-Kapazität Null den Idealzustand repräsentiert (alle gesendeten Daten vom Empfänger abgeholt), deutet die Sende-Kapazität Null auf Projektierungsfehler hin:

- die Funktion SENDEN wird zu häufig aufgerufen,
- die Funktion EMPFANGEN wird zu selten aufgerufen,
- der Verbindungsstrecke sind zu wenig Speicherblöcke zugewiesen. Die Kapazität des Zwischenspeichers reicht nicht aus, ein vorübergehendes Ungleichgewicht zwischen Sende- und Empfangs-Häufigkeit zu kompensieren.

## 1.5 System-Anlauf

Die Mehrprozessor-Kommunikation erfordert, daß bei allen beteiligten CPUs der STOP-RUN-Übergang ( = ANLAUF) **gleich** abläuft, d.h. entweder einheitlich NEUSTART oder einheitlich WIEDERANLAUF.

Durch entsprechende

- **Bedienung** (Frontschalter, Programmiergerät),
- **Parametrierung** (DX 0) und/oder
- **Programmierung** (mittels des Sonderfunktions-Organisationsbausteins OB 223 "Stopp bei uneinheitlicher Anlaufart im Mehrprozessorbetrieb"),

muß die **einheitliche** Anlaufart zumindest bei den an der Kommunikation beteiligten CPUs sichergestellt werden (vergl. Kapitel 10).

## NEUSTART:

Im Organisationsbaustein OB 20 (NEUSTART) ist von einer CPU mittels der Funktion INITIALISIEREN der Zwischenspeicher (im KOR 923C) einzurichten. Hierbei werden eventuell noch vorhandene Daten zerstört.

Anschließend, also noch im ANLAUF, können Sie in den einzelnen CPUs die Funktionen SENDEN, SENDE-TEST, EMPFANGEN, EMPFANGS-TEST aufrufen. Durch geeignete Programmierung müssen Sie gewährleisten, daß dies erst geschieht, nachdem die Initialisierung des Zwischenspeichers im Koordinator korrekt durchgeführt wurde.

Nach Beendigung des ANLAUFs, also im RUN, wird das Anwenderprogramm von Anfang an bearbeitet, d.h. der erste Befehl des OB 1 bzw. des FB 0.

## WIEDERANLAUF:

In den Organisationsbausteinen OB 21 (MANUELLER WIEDERANLAUF) und OB 22 (AUTOMATISCHER WIEDERANLAUF) dürfen Sie die Funktion INITIALISIEREN **nicht** benutzen. Der Aufruf der Funktionen SENDEN, SENDE-TEST, EMPFANGEN, EMPFANGS-TEST kann zu Schwierigkeiten führen; hierzu sind die Hinweise im nachfolgenden Kapitel zu beachten.

Nach Beendigung des WIEDERANLAUFs, also im RUN, wird das Anwenderprogramm nicht von Anfang an bearbeitet, **sondern** an der unterbrochenen Stelle fortgesetzt. Die Unterbrechungsstelle kann sich z.B. innerhalb der Funktion SENDEN befinden.

## 1.6 Aufruf und Schachtelung der Sonderfunktions- Organisationsbausteine OB 200 und OB 202 bis OB 205

Als einfachste Vorgehensweise ergibt sich folgendes:

- Aufruf der Funktion INITIALISIEREN nur im Neustart- Organisationsbaustein OB 20;
- Aufruf der Funktion SENDEN, SENDE-TEST, EMPFANGEN, EMPFANGS-TEST entweder **nur** innerhalb der zyklischen Programmbearbeitung **oder** **nur** innerhalb der zeitgesteuerten Programmbearbeitung.

### BEACHTEN SIE:



Abhängig von der Parametrierung des DX 0 ("Unterbrechung an Befehlsgrenzen" bei CPU 928B , CPU 928 und CPU 920 bzw. "155U-Mode" bei der CPU 946/947) und der Art der Programmbearbeitung (WIEDERANLAUF, Unterbrechungsbehandlung, z.B. OB 26 bei Zykluszeitfehler) ist es möglich, daß eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN und EMPFANGS-TEST unterbrochen wird. Falls eine an der Unterbrechungsstelle eingeschachtelte Anwenderschnittstelle (z.B. der OB 13 bei Unterbrechbarkeit an Befehlsgrenzen oder der OB 22 nach einem Netzausfall) ebenfalls eine der Funktionen SENDEN, SENDE-TEST, EMPFANGEN und EMPFANGS-TEST enthält, so erkennen diese einen unzulässigen Aufruf (Doppelaufruf) und signalisieren Ihnen dies durch eine Fehler-Anzeige (Fehler-Nr. 67, Kapitel 2.1.2).

## 1.7 Parallelverarbeitung im Mehrprozessor-Automatisierungsgerät

Wenn Sie das Vorbesetzen des Zwischenspeichers abgeschlossen haben (Funktion INITIALISIEREN), so können Sie die Funktionen SENDEN, SENDE-TEST, EMPFANGEN, EMPFANGS-TEST in beliebiger Kombination und Parametrierung in allen CPUs gleichzeitig und parallel bearbeiten lassen.

Betrachtet man eine einzelne Verbindungsstrecke (von CPU 'SE' nach CPU 'EM'), so ist die gleichzeitige Bearbeitung der Funktion SENDEN (CPU 'SE') und der Funktion EMPFANGEN (CPU 'EM') möglich: Während die CPU 'SE' weitere Datenblöcke zum Koordinator sendet, kann die CPU 'EM' bereits zwischengespeicherte Datenblöcke vom Koordinator empfangen (abholen).

## 1.8 Belegte Bereiche

Die Sonderfunktions-Organisationsbausteine OB 200 und OB 202 bis OB 205 benötigen keinen Arbeitsbereich (z.B. für die Zwischenspeicherung von Variablen) und rufen keine Datenbausteine auf. Sie greifen selbstverständlich auf Bereiche zu, die Parameter beinhalten, wobei nur die als Ausgangsparameter gekennzeichneten verändert werden. Außerdem beeinflussen sie die Ergebnisanzeigen (ANZ 1, VKE usw., siehe Kapitel 2.1).

- CPU 922, CPU 928, CPU 928B: Die Inhalte AKKU 1 bis AKKU 4 sowie die Registerinhalte werden von den Sonderfunktions-OB der Mehrprozessorkommunikation nicht verändert.
- CPU 946/947: Alle Registerinhalte sowie AKKU 1, 2 und 3 bleiben gleich, verändert wird lediglich der AKKU 4.

## 1.9 Laufzeit

Sonderfunktions-OB		Laufzeit			
Baustein-Name	Baustein-Funktion	R-Prozessor	CPU 928	CPU 946/947	CPU 928B
OB 200 / Initialisie- ren	Vorbesetzen	230 ms	130 ms	128 ms	130 ms
OB 202 / Senden	Senden eines Datenblocks (32 Daten- worte)	806 $\mu$ s (294 $\mu$ s Grundlast + 16 $\mu$ s/Wort); 118 $\mu$ s bei Warnung	666 $\mu$ s (250 $\mu$ s Grundlast + 13 $\mu$ s/Wort); 115 $\mu$ s bei Warnung	762 $\mu$ s (426 $\mu$ s Grundlast + 21 $\mu$ s/Doppel- wort); 243 $\mu$ s bei Warnung	696 $\mu$ s (280 $\mu$ s Grundlast + 13 $\mu$ s/Wort); 145 $\mu$ s bei Warnung
OB 203 / Sendetest	Sendemög- lichkeit testen	72 $\mu$ s	50 $\mu$ s	207 $\mu$ s	80 $\mu$ s
OB 204 / Empfangen	Empfangen eines Datenblocks (32 Daten- worte)	825 $\mu$ s (281 $\mu$ s Grundlast + 17 $\mu$ s/Wort); 115 $\mu$ s bei Warnung	660 $\mu$ s (244 $\mu$ s Grundlast + 13 $\mu$ s/Wort); 98 $\mu$ s bei Warnung	772 $\mu$ s (421 $\mu$ s Grundlast + 22 $\mu$ s/Doppel- wort); 243 $\mu$ s bei Warnung	690 $\mu$ s (274 $\mu$ s Grundlast + 13 $\mu$ s/Wort); 128 $\mu$ s bei Warnung
OB 205 / Empfangs- Test	Empfangs- möglichkeit testen	70 $\mu$ s	48 $\mu$ s	223 $\mu$ s	78 $\mu$ s

"Laufzeit" ist die Bearbeitungszeit der Sonderfunktions-Organisationsbausteine; die Zeit(-dauer) zwischen Aufruf eines Bausteins und seinem Abschluß kann erheblich größer werden, falls er von höherpriorigen Tätigkeiten unterbrochen wird (z.B. Zeitzellenaktualisierung, Reglerbearbeitung usw.).

Die oben genannten Laufzeiten ergeben sich unter der Voraussetzung, daß von vier gesteckten CPUs nur diejenige CPU auf den S5-Bus zugreift, deren Laufzeiten gemessen werden. Falls weitere CPUs den Bus intensiv nutzen, steigt insbesondere beim Senden/Empfangen die Laufzeit an.

Ein wichtiges Leistungsmerkmal einer Verbindungsstrecke (von CPU 'SE' nach CPU 'EM') ist die gesamte Datenübertragungszeit. Sie setzt sich aus den folgenden drei Komponenten zusammen:

- Sende-Dauer (siehe Laufzeit)
- Dauer der Zwischenspeicherung (im Koordinator KOR 923C)
- Empfangs-Dauer (siehe Laufzeit)

**Wie lange die zu übertragenden Daten 'unterwegs' sind, wird also wesentlich von der Dauer der Zwischenspeicherung und damit von der Struktur des Anwenderprogramms bestimmt (vergl. Kapitel "Zwischenspeicherung der Daten").**

## 2 Parametrierung

Die "eigentlichen" Parameter befinden sich in einem max. 10 byte großen Datenfeld im M-Merkerbereich. Die Nummer des ersten Merker-Bytes im Datenfeld (= Zeiger auf Datenfeld) muß in den AKKU-1-L geladen werden. Zulässige Werte dafür sind 0 bis 246.

Das Datenfeld untergliedert sich in einen Bereich für **Eingangsparameter**, einen Bereich für **Ausgangsparameter**.

- **Eingangsparameter:**

Die Eingangsparameter werden vollständig oder teilweise von den Funktionen gelesen und ausgewertet, schreibende Zugriffe finden nicht statt.

- **Ausgangsparameter:**

Die Ausgangsparameter werden vollständig oder teilweise von den Funktionen beschrieben, lesende Zugriffe finden nicht statt.

### BEACHTEN SIE:



Sie können für **alle** Kommunikationsfunktionen **einen Merkerbereich mit 10 Merker-Bytes** vorsehen. Die einzelnen Funktionen benötigen jedoch eine unterschiedliche Anzahl von Bytes. Diese sind bei den Einzelfunktionen (Kapitel 3 bis 7) aufgeführt.

### Beispiel: Datenfeld mit Parametern der Funktion EMPFANGEN (OB 204)

MB x + 0:	Sende-CPU	Eingangsparameter
MB x + 1:	—	nicht belegt
MB x + 2:	Anzeigenbyte	Ausgangsparameter
MB x + 3:	Empfangs-Kapazität	Ausgangsparameter
MB x + 4:	Baustein-Kennung	Ausgangsparameter
MB x + 5:	Baustein-Nummer	Ausgangsparameter
MB x + 6:	Adresse des ersten	Ausgangsparameter
MB x + 7:	empfangenen Datenwortes	Ausgangsparameter
MB x + 8:	Adresse des letzten	Ausgangsparameter
MB x + 9:	empfangenen Datenwortes	Ausgangsparameter

Dieses Beispiel verdeutlicht, daß die Nummer des ersten M-Merkerbytes im Datenfeld nicht größer als (MB) 246 sein kann und darf, da andernfalls das bis zu 10 byte große Parameterfeld die Grenzen des Merkerbereichs überschreiten würde (MB 255).

### 2.1 Auswerten der Ausgangsparameter

Ausgangsparameter sind Daten, die die Funktion dem Anwenderprogramm zur Auswertung bereitstellt. Sie geben u.a. Hinweise, ob eine Funktion überhaupt bearbeitet werden konnte; falls nein, so zeigen sie den Grund für den Funktionsabbruch an.

## 2.1.1 Ergebnisanzeigen

Die Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN, EMPFANGS-TEST beeinflussen die Ergebnisanzeigen (siehe Programmieranleitungen der jeweiligen CPUs, Allgemeine Hinweise zu den STEP-5-Operationen):

- die OV- und OS-Bits (Wortanzeigen) werden immer gelöscht,
- die OR-, STA-, ERAB-Bits (Bitanzeigen) werden immer gelöscht,
- VKE, ANZ 1 und ANZ 0 geben Auskunft, ob eine Funktion korrekt und vollständig abgearbeitet worden ist.

VKE = 0: Funktion korrekt und vollständig abgearbeitet

VKE = 1: Funktion abgebrochen; möglicherweise hat der Zeiger auf das Datenfeld im Merkerbereich einen unzulässigen Wert, d.h., das Low-Word des Akkus enthält einen Wert größer 246. **In den nachfolgenden Kapiteln wird vorausgesetzt, daß der Zeiger auf das Datenfeld einen korrekten Wert enthält.** Dann ist im ersten Byte der Ausgangsparameter die Abbruchursache in detaillierter Form hinterlegt.

ANZ 1 = 1: Zusätzliche Anzeige einer Warnung (Warnungs-Nr. 1 oder 2)

ANZ 0 = 1: Zusätzliche Anzeige eines Fehlers (Fehler-Nr. 1-9)

Fall	Anzeigen			Auswertung mit Operation
	VKE	ANZ 1	ANZ 0	
Funktion vollständig und korrekt abgearbeitet	0	0	0	SPB =
Funktion abgebrochen, Zeiger auf Datenfeld unzulässig	1	0	0	SPB =
Funktion abgebrochen aufgrund eines Initialisierungskonfliktes	1	0	0	SPB =
Funktion abgebrochen aufgrund eines Fehlers	1	0	1	SPB = und SPM =
Funktion abgebrochen aufgrund einer Warnung	1	1	0	SPB = und SPP =



## 2.1.2 Anzeigenbyte: Initialisierungskonflikt/Fehler/Warnung

Das erste Byte im Feld der Ausgangsparameter (Anzeigenbyte) zeigt ebenfalls an, ob eine Funktion korrekt und vollständig abgearbeitet worden ist. Der Grund eines Funktionsabbruchs wird detaillierter dargestellt als in den Ergebnisanzeigen.

Unter der oben getroffenen Vereinbarung, daß zumindest der Zeiger auf das Datenfeld einen korrekten Wert enthält, ist dieses Byte **immer** relevant.

Ist die Funktion korrekt und vollständig abgearbeitet worden, so sind alle Bits gelöscht (= 0), und alle weiteren Ausgangsparameter sind ebenfalls relevant.

Ist die Funktion mit einer Warnung abgebrochen worden (Bit  $2^7 = 1$ ), so ist nur noch die Anzeige der Sende-/Empfangs- Kapazität relevant, weitere Ausgangsparameter (falls vorhanden) sind unverändert.

Ist die Funktion mit einem Fehler (Bit  $2^6 = 1$ ) oder einem Initialisierungskonflikt (Bit  $2^5 = 1$ ) abgebrochen worden, so sind alle weiteren Ausgangsparameter unverändert.

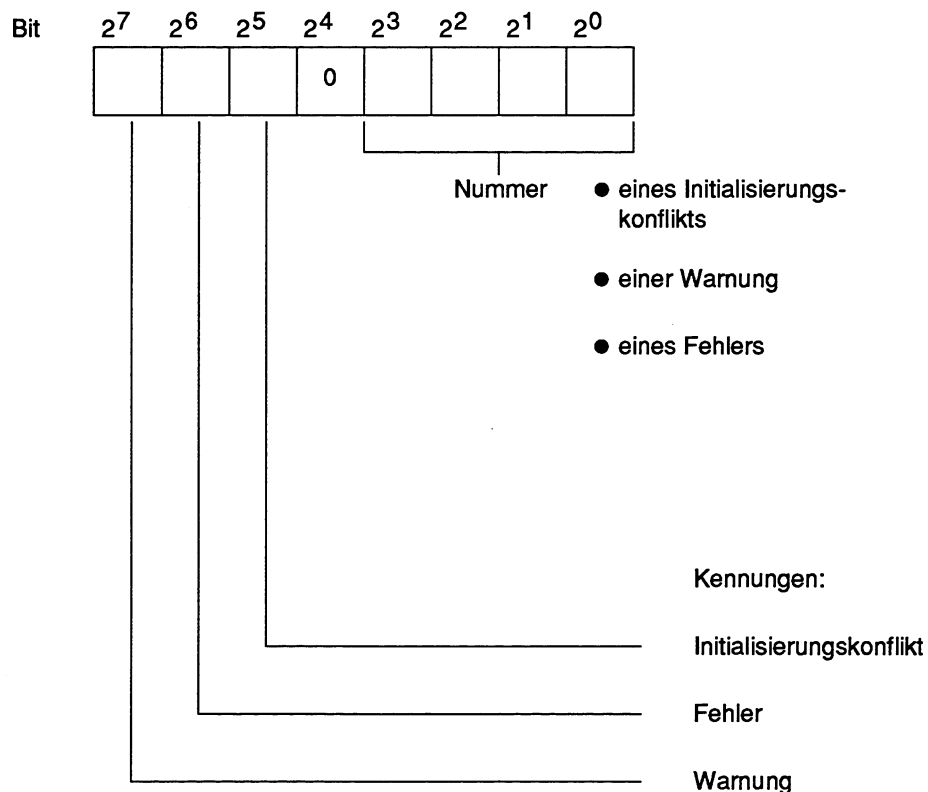


Bild 2 Codierung des ersten Bytes

## Auswertung

Die Kennungen in den Bitpositionen  $2^5$  bis  $2^7$  zeigen u.a. an, welche Bedeutung die Nummer in den Bitpositionen  $2^0$  bis  $2^3$  hat.

Neben dieser bitweisen Auswertung ist es auch möglich, das gesamte Anzeigenbyte als vorzeichenlose Festpunktzahl zu interpretieren. Bei einer **byteweisen Interpretation** des Anzeigenbytes ergeben sich Nummerngruppen mit folgender Bedeutung:

Nummern-Gruppe	Bedeutung
0	Funktion korrekt und vollständig abgearbeitet
33 bis 42	Funktion abgebrochen aufgrund eines Initialisierungskonfliktes
65 bis 73	Funktion abgebrochen aufgrund eines Fehlers
129 bis 130	Funktion abgebrochen aufgrund einer Warnung

Die Werte der nachfolgenden Nummern geben **auch** an, in **welcher Reihenfolge** Fehler bzw. Initialisierungskonflikte von den Funktionen erkannt und angezeigt werden.

### Beispiel:

Die Funktion SENDEN zeigt Fehler an und wird nicht durchgeführt. Wenn Sie nun Programm- und/oder Parameteränderungen durchführen und die Funktion SENDEN erneut einen Fehler mit höherer Nummer als vorher anzeigt, so können Sie daraus schließen, daß Sie einen von mehreren Fehlern beseitigt haben.

## Initialisierungskonflikt

Ein Initialisierungskonflikt kann nur bei der Funktion INITIALISIEREN auftreten. Er erfordert eine Änderung in der Programmierung/Parametrierung.

### Initialisierungskonflikt-Nummern (byteweise Auswertung des Anzeigenbytes):

- (33) Die für die Mehrprozessorkommunikation benötigten Kacheln (Nr. 252 bis Nr. 255) sind nicht bzw. nicht vollständig vorhanden.
- (34) Die für die Mehrprozessorkommunikation benötigten Kacheln (Nr. 252 bis 255) sind defekt.
- (35) Der Parameter "Automatisch/Manuell" ist unzulässig. Unterscheiden Sie folgende Fälle:

Die Kennung "Automatisch/Manuell" ist kleiner 1.

Die Kennung "Automatisch/Manuell" ist größer 2.

- **(36)** Der Parameter "Anzahl CPUs" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Anzahl der CPUs ist kleiner 2.
  - Die Anzahl der CPUs ist größer 4.
- **(37)** Der Parameter "Baustein-Kennung" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Baustein-Kennung ist kleiner 1.
  - Die Baustein-Kennung ist größer 2.
- **(38)** Der Parameter "Baustein-Nummer" ist unzulässig, da es sich um einen Datenbaustein mit spezieller Bedeutung handelt. Unterscheiden Sie folgende Fälle:
  - Falls Baustein-Kennung = 1 : DB 0, DB 1, DB 2
  - Falls Baustein-Kennung = 2 : DX 0, DX 1, DX 2
- **(39)** Der Parameter "Baustein-Nummer" ist fehlerhaft, da der parametrierte Datenbaustein nicht existiert.
- **(40)** Der Parameter "Anfangsadresse der Zuordnungsliste" ist zu groß bzw. der Datenbaustein zu kurz.
- **(41)** Die Zuordnungsliste im Datenbaustein ist nicht korrekt aufgebaut.
- **(42)** Die Summe der vergebenen Speicherblöcke ist größer als 48.

## Fehler

Ein aufgetretener Fehler erfordert eine Änderung in der Programmierung/Parametrierung.

### Fehler-Nummern (bytestweise Auswertung des Anzeigenbytes):

- **(65)** Der Parameter "Empfangs-CPU" (SENDEN, SENDE-TEST) ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Nummer der "Empfangs-CPU" ist größer 4.
  - Die Nummer der "Empfangs-CPU" ist kleiner 1.
  - Die Nummer der "Empfangs-CPU" ist gleich der "eigenen" Nummer.
- **(66)** Der Parameter "Sende-CPU" (EMPFANGEN, EMPFANGS-TEST) ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Nummer der "Sende-CPU" ist größer 4.
  - Die Nummer der "Sende-CPU" ist kleiner 1.
  - Die Nummer der "Sende-CPU" ist gleich der "eigenen" Nummer.
- **(67)** Der Aufruf des Sonderfunktions-Organisationsbausteins ist fehlerhaft (SENDEN, EMPFANGEN, SENDE-TEST, EMPFANGS-TEST). Unterscheiden Sie folgende Fälle:
  - a) Folgefehler, da die Funktion INITIALISIEREN nicht aufgerufen oder mit Initialisierungskonflikt beendet wurde.
  - b) Doppelaufruf: Der Aufruf dieser Funktion SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST ist unzulässig, da in dieser CPU bereits in einer untergeordneten Bearbeitungsebene (z.B. zyklische Programmbearbeitung) eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGSTEST aufgerufen wurde (siehe Kap. "Aufruf und Schachtelung der Sonderfunktions-Organisationsbausteine").
  - c) Die "eigene" CPU-Nummer ist fehlerhaft (Systemdaten zerstört); nach NETZ EIN / NETZ AUS wird die CPU-Nr. vom Systemprogramm erneut erzeugt.
- **(68)** Die Verwaltungsdaten (Warteschlangenverwaltung) der angewählten Verbindungsstrecken sind fehlerhaft; den Zwischenspeicher im Koordinator 923C richten Sie mit Hilfe der Funktion INITIALISIEREN neu ein (SENDEN, EMPFANGEN, SENDE-TEST, EMPFANGS-TEST).

- (69) Der Parameter "Baustein-Kennung" (SENDEN) bzw. die vom Sender überlieferte Baustein-Kennung (EMPFANGEN) ist unzulässig. Unterscheiden Sie folgende Fälle:

Die Baustein-Kennung ist kleiner 1.

Die Baustein-Kennung ist größer 2.

- (70) Der Parameter "Baustein-Nummer" (SENDEN) bzw. die vom Sender überlieferte Baustein-Nummer (EMPFANGEN) ist unzulässig, da es sich um einen Datenbaustein mit spezieller Bedeutung handelt. Unterscheiden Sie folgende Fälle:

Falls Baustein-Kennung = 1 : DB 0, DB 1, DB 2

Falls Baustein-Kennung = 2 : DX 0, DX 1, DX 2

- (71) Der Parameter "Baustein-Nummer" (SENDEN) bzw. die vom Sender überlieferte Baustein-Nummer (EMPFANGEN) ist fehlerhaft. Der parametrisierte Datenbaustein existiert nicht.
- (72) Der Parameter "Block-Nummer" (SENDEN) ist fehlerhaft. Der Datenbaustein ist zu kurz bzw. die Block-Nummer zu groß.
- (73) Der Datenbaustein ist zu klein, um den vom Sender gelieferten Datenblock aufzunehmen (EMPFANGEN).

### Warnung

Die Funktion konnte nicht durchgeführt werden; der Funktionsaufruf ist zu wiederholen, z.B. im nächsten Zyklus.

### Warnungs-Nummern (bytwweise Auswertung des Anzeigenbytes):

- (129) Die Funktion SENDEN kann keine Daten übergeben, da die Sende-Kapazität bereits beim Funktionsaufruf gleich Null war.
- (130) Die Funktion EMPFANGEN kann keine Daten übernehmen, da die Empfangs-Kapazität bereits beim Funktionsaufruf gleich Null war.

### 3 Funktion INITIALISIEREN (OB 200)

#### Aufrufparameter:

##### 1. Datenfeld

Vor Aufruf des OB 200 müssen Sie im Datenfeld die Eingangsparameter bereitstellen.

Der OB 200 benötigt im Datenfeld 8 M-Merkerbytes für Ein- und Ausgangsparameter:

MB x + 0:	Betriebsart (Automatisch/ Manuell )	Eingangsparameter
MB x + 1:	Anzahl CPUs	Eingangsparameter
MB x + 2:	Baustein-Kennung	Eingangsparameter
MB x + 3:	Baustein-Nummer	Eingangsparameter
MB x + 4:	Anfangsadresse der	Eingangsparameter
MB x + 5:	Zuordnungsliste	Eingangsparameter
MB x + 6:	Anzeigenbyte	Ausgangsparameter
MB x + 7:	Gesamt-Kapazität	Ausgangsparameter

##### 2. AKKU-1-L:

Nr. des 1. Merkerbytes 'x' im Datenfeld,  
zulässige Werte:

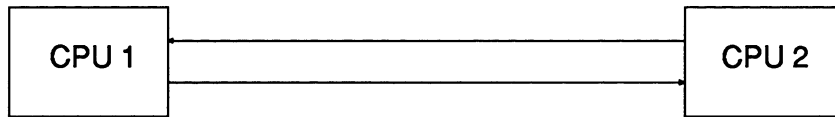
AKKU-1-LH: 0  
AKKU-1-LL: 0 bis 246

Um von einer CPU an eine andere CPU Daten übertragen zu können, müssen diese vorübergehend zwischengespeichert werden. Die Funktion INITIALISIEREN richtet zu diesem Zweck einen Zwischenspeicher im Koordinator KOR 923C ein.

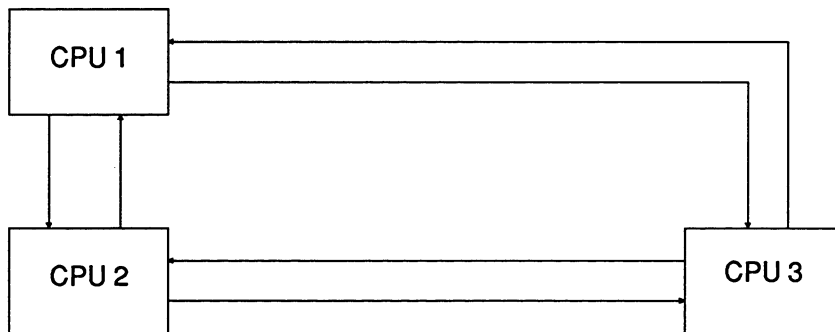
Die Speicherkapazität wird in Blöcken (mit einer Größe von 32 Wörtern) gemessen bzw. ausgedrückt.

Jeder Speicherblock (die Länge beträgt immer 32 Wörter) nimmt genau einen Datenblock auf (mit einer Länge zwischen einem Datenwort und 32 Datenwörtern). Ein Datenblock wird von einem SENDEN-Baustein in einen Speicherblock eingetragen und von einem EMPFANGEN-Baustein wieder ausgetragen.

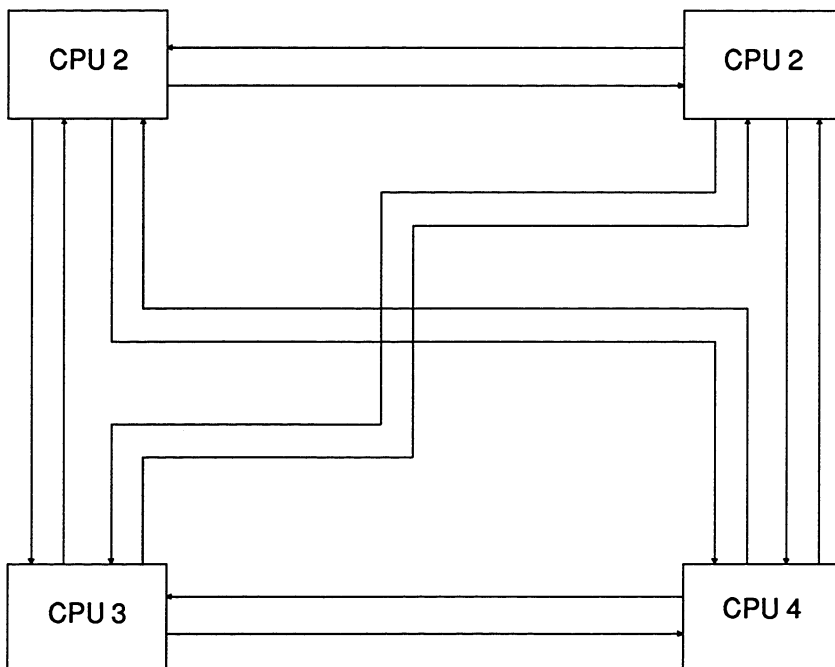
Bei zwei gesteckten CPUs ergeben sich zwei Verbindungsstrecken (Transferrichtungen, "Kanäle"):



Bei drei gesteckten CPUs ergeben sich sechs Verbindungsstrecken:



Bei vier gesteckten CPUs ergeben sich zwölf Verbindungsstrecken:



Mit der Funktion INITIALISIEREN wird festgelegt, wie die insgesamt **48** zur Verfügung stehenden Speicherblöcke den maximal zwölf Verbindungsstrecken zugeordnet werden.  
D.h.: Jeder möglichen Verbindungsstrecke, gekennzeichnet durch die Parameter "Sende-CPU" und "Empfangs-CPU", steht eine bestimmte Speicherkapazität zur Verfügung.

#### BEACHTEN SIE:



Bevor auf den CPUs die Funktionen SENDEN / EMPFANGEN / SENDE-TEST / EMPFANGS-TEST aufgerufen werden dürfen, muß zuerst auf einer CPU die Funktion INITIALISIEREN aufgerufen und vollständig und fehlerfrei abgearbeitet worden sein.

Falls die Funktion INITIALISIEREN mehrfach nacheinander aufgerufen wird, so gilt die zuletzt parametrisierte Zuordnung. Während die Funktion INITIALISIEREN von einer CPU bearbeitet wird, dürfen keine weiteren Funktionen, also auch nicht die Funktion INITIALISIEREN, auf anderen CPUs aufgerufen werden.

### 3.1 Eingangsparameter

#### 3.1.1 Betriebsart (Automatisch/Manuell)

Betriebsart = 1 : Automatisch

Betriebsart = 2 : Manuell

Betriebsart = 0 oder 3 bis 255 : Unzulässig, führt zu einem Initialisierungskonflikt

#### Betriebsart "Automatisch"

Wird die Betriebsart "Automatisch" gewählt, so werden die zur Verfügung stehenden Speicherblöcke **gleichmäßig** entsprechend der Anzahl der CPUs aufgeteilt:

Anzahl der CPUs	Anzahl der Verbindungsstrecken	Speicherblöcke pro Verbindungsstrecke
2	2	24
3	6	8
4	12	4
0; 1; 5 bis 255	Unzulässig, führt zu einem Initialisierungs-Konflikt	



### **Betriebsart "Manuell"**

Wenn Sie die Betriebsart "Manuell" wählen, so müssen Sie in einem Datenbaustein eine Zuordnungsliste bereitstellen, in der nach einem festgelegten Schema die 48 zur Verfügung stehenden Speicherblöcke (oder weniger) den maximal 12 Verbindungsstrecken zugeordnet werden. Diese Funktion ist insbesondere sinnvoll, falls nicht alle CPUs in gleichem Umfang Daten miteinander austauschen. Beispielsweise können Sie CPUs 921 (S-Prozessoren) an der hier beschriebenen Mehrprozessorkommunikation nicht beteiligen; den betroffenen Verbindungsstrecken brauchen und sollten Sie keine Speicherblöcke zuweisen. Die Parameter

- Baustein-Kennung,
- Baustein-Nummer und die
- Anfangsadresse der Zuordnungsliste

legen fest, wo die Zuordnungsliste hinterlegt ist. Diese drei Parameter sind also nur in der Betriebsart "Manuell" relevant.

### **3.1.2 Anzahl CPUs**

Dieser Parameter ist nur relevant, wenn die Betriebsart "Automatisch" gewählt wurde; (siehe 3.1.1).

### **3.1.3 Baustein-Kennung und -Nummer / Anfangsadresse der Zuordnungsliste**

Diese Parameter sind nur relevant, wenn die Betriebsart "Manuell" gewählt wurde.

#### **Baustein-Kennung und -Nummer:**

Kennung = 1: DB-Datenbaustein  
Kennung = 2: DX-Datenbaustein  
Kennung = 0 oder 3 bis 255 : Unzulässig, führt zu einem Initialisierungskonflikt

Als Baustein-Nummer geben Sie die Nummer des Datenbausteins DB oder DX an, in dem die Zuordnungsliste liegt.

#### **Anfangsadresse der Zuordnungsliste:**

Sie ergibt zusammen mit der Baustein-Kennung und -Nummer den Bereich (genauer: die Anfangsadresse des Bereichs), in dem die Zuordnungsliste hinterlegt ist.

Die Zuordnungsliste enthält weitere Eingangsparameter der Funktion INITIALISIEREN, d.h., es erfolgen nur lesende Zugriffe (keine Veränderung des Inhalts). Sie hat folgenden Aufbau:

**Zuordnungsliste:**

Datenwort	:	Format	Wert	:	Bedeutung
DW n+ 0	:	KC	S1	:	Sender = CPU 1
DW n+ 1	:	KY	2, <b>a</b>	:	Empfänger = CPU 2
DW n+ 2	:	KY	3, <b>b</b>	:	Empfänger = CPU 3
DW n+ 3	:	KY	4, <b>c</b>	:	Empfänger = CPU 4
DW n+ 4	:	KC	S2	:	Sender = CPU 2
DW n+ 5	:	KY	1, <b>d</b>	:	Empfänger = CPU 1
DW n+ 6	:	KY	3, <b>e</b>	:	Empfänger = CPU 3
DW n+ 7	:	KY	4, <b>f</b>	:	Empfänger = CPU 4
DW n+ 8	:	KC	S3	:	Sender = CPU 3
DW n+ 9	:	KY	1, <b>g</b>	:	Empfänger = CPU 1
DW n+ 10	:	KY	2, <b>h</b>	:	Empfänger = CPU 2
DW n+ 11	:	KY	4, <b>i</b>	:	Empfänger = CPU 4
DW n+ 12	:	KC	S4	:	Sender = CPU 4
DW n+ 13	:	KY	1, <b>k</b>	:	Empfänger = CPU 1
DW n+ 14	:	KY	2, <b>l</b>	:	Empfänger = CPU 2
DW n+ 15	:	KY	3, <b>m</b>	:	Empfänger = CPU 3

**BEACHTEN SIE**

Dieser Aufbau muß eingehalten werden, auch wenn weniger als vier CPUs gesteckt sind.

Anstelle der Kleinbuchstaben a bis m (hier fettgedruckt) sind Zahlen zwischen 0 und 48 einzusetzen; **ihre Summe darf 48 nicht überschreiten.**

Ein entsprechendes Beispiel finden Sie auf der nächsten Seite.

### Beispiel:

Sie haben drei CPUs gesteckt, von CPU 2 sind sehr viele Daten an die beiden anderen zu übertragen. Diese wiederum senden nur wenige Daten zurück an CPU 2 als Rückmeldung in einem logischen Quittungsverkehr. Zwischen CPU 1 und CPU 3 ist kein Datenaustausch nötig.

### Zuordnungsliste:

Datenwort	:	Format	Wert	:	Bedeutung
DW n+ 0	:	KC	S1	:	Sender = CPU 1
DW n+ 1	:	KY	2, 2	:	Empfänger = CPU 2
DW n+ 2	:	KY	3, 0	:	Empfänger = CPU 3
DW n+ 3	:	KY	4, 0	:	Empfänger = CPU 4
DW n+ 4	:	KC	S2	:	Sender = CPU 2
DW n+ 5	:	KY	1, 22	:	Empfänger = CPU 1
DW n+ 6	:	KY	3, 22	:	Empfänger = CPU 3
DW n+ 7	:	KY	4, 0	:	Empfänger = CPU 4
DW n+ 8	:	KC	S3	:	Sender = CPU 3
DW n+ 9	:	KY	1, 0	:	Empfänger = CPU 1
DW n+ 10	:	KY	2, 2	:	Empfänger = CPU 2
DW n+ 11	:	KY	4, 0	:	Empfänger = CPU 4
DW n+ 12	:	KC	S4	:	Sender = CPU 4
DW n+ 13	:	KY	1, 0	:	Empfänger = CPU 1
DW n+ 14	:	KY	2, 0	:	Empfänger = CPU 2
DW n+ 15	:	KY	3, 0	:	Empfänger = CPU 3

## 3.2 Ausgangsparameter

### 3.2.1 Anzeigenbyte Initialisierungskonflikt

Dieses Byte informiert Sie, ob die Funktion INITIALISIEREN korrekt und vollständig abgearbeitet worden ist.

#### Initialisierungskonflikt:

Die aufgeführten Initialisierungskonflikte werden entsprechend der aufsteigenden Reihenfolge ihrer Nummern von der Funktion erkannt und angezeigt.

Ein aufgetretener Initialisierungskonflikt erfordert eine Änderung in der Programmierung / Parametrierung.

**Initialisierungskonflikt-Nummern (bytestweise Auswertung des Anzeigenbytes):**

- (33) Die für die Mehrprozessorkommunikation benötigten Kacheln (Nr. 252 bis Nr. 255) sind nicht bzw. nicht vollständig vorhanden.
- (34) Die für die Mehrprozessorkommunikation benötigten Kacheln (Nr. 252 bis 255) sind defekt.
- (35) Der Parameter "Automatisch / Manuell" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Kennung "Automatisch/Manuell" ist kleiner 1.
  - Die Kennung "Automatisch/Manuell" ist größer 2.
- (36) Der Parameter "Anzahl CPUs" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Anzahl der CPUs ist kleiner 2.
  - Die Anzahl der CPUs ist größer 4.
- (37) Der Parameter "Baustein-Kennung" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Baustein-Kennung ist kleiner 1.
  - Die Baustein-Kennung ist größer 2.
- (38) Der Parameter "Baustein-Nummer" ist unzulässig, da es sich um einen Datenbaustein mit spezieller Bedeutung handelt. Unterscheiden Sie folgende Fälle:
  - Falls Baustein-Kennung = 1 : DB 0, DB 1, DB 2
  - Falls Baustein-Kennung = 2 : DX 0, DX 1, DX 2
- (39) Der Parameter "Baustein-Nummer" ist fehlerhaft, da der parametrisierte Datenbaustein nicht existiert.
- (40) Der Parameter "Anfangsadresse der Zuordnungsliste" ist zu groß bzw. der Datenbaustein zu kurz.
- (41) Die Zuordnungsliste im Datenbaustein ist nicht korrekt aufgebaut.
- (42) Die Summe der vergebenen Speicherblöcke ist größer als 48.

**Fehler:**

Eine Anzeige der Nummerngruppe "Fehler" kann bei der Funktion INITIALISIEREN nicht auftreten.

**Warnung:**

Eine Anzeige der Nummerngruppe "Warnung" kann bei der Funktion INITIALISIEREN nicht auftreten.

### **3.2.2 Gesamt-Kapazität**

Dieser Parameter gibt an, wieviel der zur Verfügung stehenden 48 Speicherblöcke den Verbindungsstrecken zugeordnet sind.

In der Betriebsart "Automatisch" wird dieser Parameter auf jeden Fall den Wert 48 beinhalten. Bei der Betriebsart "Manuell" kann der Wert kleiner sein als 48. Dies bedeutet, daß vorhandene Speicherkapazität nicht genutzt wird.

## 4 Funktion SENDEN (OB 202)

### Aufrufparameter

#### 1. Datenfeld:

Vor Aufruf des OB 202 müssen Sie im Datenfeld die Eingangsparameter bereitstellen.

Der OB 202 benötigt im Datenfeld 6 M-Merkerbytes für Ein- und Ausgangsparameter:

MB x + 0:	Empfangs-CPU	Eingangsparameter
MB x + 1:	Baustein-Kennung	Eingangsparameter
MB x + 2:	Baustein-Nummer	Eingangsparameter
MB x + 3:	Block-Nummer	Eingangsparameter
MB x + 4:	Anzeigenbyte	Ausgangsparameter
MB x + 5:	Sende-Kapazität	Ausgangsparameter

#### 2. AKKU-1-L:

Nr. des 1. Merkerbytes 'x' im Datenfeld,  
zulässige Werte:

AKKU-1-LH: 0  
AKKU-1-LL: 0 bis 246

Die Funktion SENDEN übergibt einen Datenblock in den Zwischenspeicher des Koordinators KOR 923C. Zusätzlich zeigt sie an, wieviele Datenblöcke noch gesendet und zwischengespeichert werden können.

### 4.1 Eingangsparameter

#### 4.1.1 Empfangs-CPU

Die zu sendenden Daten sind für die Empfangs-CPU bestimmt; der zulässige Wert liegt zwischen 1 und 4, muß jedoch ungleich der "eigenen" Nummer sein.

#### 4.1.2 Baustein-Kennung und -Nummer / Block-Nummer

##### Baustein-Kennung:

Kennung = 1:	DB-Datenbaustein
Kennung = 2:	DX-Datenbaustein
Kennung = 0 oder 3 bis 255:	Unzulässig, führt zu einer Fehleranzeige

**Baustein-Nummer:**

Die Baustein-Nummer ergibt zusammen mit der Baustein-Kennung (s.o.) und der Block-Nummer (s.u.) den Bereich, dem die Sende-Daten entnommen werden (und in dem sie auf der Empfangs-CPU abgelegt werden).

Hierbei ist zu beachten, daß bestimmten Datenbausteinen spezielle Bedeutungen zugeordnet sind, beispielsweise dem DB 0, DB 1 oder dem DX 0 (siehe Programmieranleitungen der jeweiligen CPUs). Diese Datenbausteine dürfen deshalb für die hier beschriebene Datenübertragung nicht verwendet werden!

Eine Verwendung dieser Bausteinnummer führt zum Abbruch der Funktion mit Fehleranzeige.

**Block-Nummer:**

Die Block-Nummer kennzeichnet den zu sendenden Datenbereich.

Block- Nummer	Datenbereich	
	erstes Datenwort	letztes Datenwort
0	DW 0	DW 31
1	DW 32	DW 63
2	DW 64	DW 95
3	DW 96	DW 127
4	DW 128	DW 159
5	DW 160	DW 191
6	DW 192	DW 223
7	DW 224	DW 255
8	DW 256	DW 287
9	DW 288	DW 319
:	:	:
:	:	:

Folgende Fälle sind zu unterscheiden:

- Ist der Datenbaustein ausreichend lang, so ergibt sich ein 32 Wörter großer Bereich gemäß der oben angeführten Tabelle.
- Liegt das Datenbaustein-Ende innerhalb des parametrisierten Blockes, so wird ein Bereich mit einer Länge zwischen einem und 31 Wörtern übertragen.
- Ist die ermittelte erste Datenwort-Adresse bereits außerhalb der Datenbaustein-Länge, so wird von der Funktion SENDEN ein Fehler erkannt und angezeigt.

Beispiel:

Datenbaustein mit der Länge 80 Wörter: DW 0 bis DW 74, 5 Wörter sind Bausteinkopf

Block- Nummer	Datenbereich		Länge
	erstes Datenwort	letztes Datenwort	
0	DW 0	DW 31	32 Wörter
1	DW 32	DW 63	32 Wörter
2	DW 64	DW 74	11 Wörter
3 und größer			

4.2 Ausgangsparameter

4.2.1 Anzeigenbyte Fehler/Warnung

Dieses Byte informiert Sie, ob die Funktion SENDEN korrekt und vollständig abgearbeitet worden ist.

Fehler:

Ein aufgetretener Fehler erfordert eine Änderung in der Programmierung/Parameterierung.

Fehler-Nummer (bytestweise Auswertung des Anzeigenbytes):

- (65) Der Parameter "Empfangs-CPU" ist unzulässig. Unterscheiden Sie folgende Fälle:

Die Nummer der Empfangs-CPU ist größer 4.

Die Nummer der Empfangs-CPU ist kleiner 1.

Die Nummer der Empfangs-CPU ist identisch mit der "eigenen" Nummer.



- **(67)** Der Aufruf des Sonderfunktions-Organisationsbausteins ist fehlerhaft. Unterscheiden Sie folgende Fälle:
  - a) Folgefehler, da die Funktion INITIALISIEREN nicht aufgerufen oder mit Initialisierungskonflikt beendet wurde.
  - b) Doppelaufruf: Der Aufruf dieser Funktion SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST ist unzulässig, da in dieser CPU bereits in einer untergeordneten Bearbeitungsebene (z.B. zyklische Programmbearbeitung) eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST aufgerufen wurde (siehe Kap. "Aufruf und Schachtelung der Sonderfunktions-Organisationsbausteine").
  - c) Die "eigene" CPU-Nummer ist fehlerhaft (Systemdaten zerstört); nach NETZ EIN/NETZ AUS wird die CPU-Nr. vom Systemprogramm erneut erzeugt.
- **(68)** Die Verwaltungsdaten (Warteschlangenverwaltung) der angewählten Verbindungsstrecken sind fehlerhaft; der Zwischenspeicher im Koordinator 923C ist mit Hilfe der Funktion INITIALISIEREN neu einzurichten.
- **(69)** Der Parameter "Baustein-Kennung" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Baustein-Kennung ist kleiner 1.
  - Die Baustein-Kennung ist größer 2.
- **(70)** Der Parameter "Baustein-Nummer" ist unzulässig, da es sich um einen Datenbaustein mit spezieller Bedeutung handelt. Unterscheiden Sie folgende Fälle:
  - Falls Baustein-Kennung = 1 : DB 0, DB 1, DB 2
  - Falls Baustein-Kennung = 2 : DX 0, DX 1, DX 2
- **(71)** Der Parameter "Baustein-Nummer" ist fehlerhaft. Der parametrisierte Datenbaustein existiert nicht.
- **(72)** Der Parameter "Block-Nummer" ist fehlerhaft. Der Datenbaustein ist zu kurz bzw. die Block-Nummer zu groß.

**Warnung:**

Die Funktion konnte nicht durchgeführt werden; der Funktionsaufruf ist zu wiederholen, z.B. im nächsten Zyklus.

**Warnungs-Nummer (byteweise Auswertung des Anzeigenbytes):**

- (129) Die Funktion SENDEN kann keine Daten übergeben, da die Sende-Kapazität (s.u.) bereits beim Funktionsaufruf gleich null war.

**Initialisierungskonflikt:**

Eine Anzeige der Nummerngruppe "Initialisierungskonflikt" kann bei der Funktion SENDEN nicht auftreten.

### **4.2.2 Sende-Kapazität**

Der Parameter "Sende-Kapazität" zeigt an, wieviel Datenblöcke noch gesendet und zwischengespeichert werden können.

## 5 Funktion SENDE-TEST (OB 203)

### Aufrufparameter

#### 1. Datenfeld

Vor Aufruf des OB 203 müssen Sie im Datenfeld die Eingangsparameter bereitstellen.

Der OB 203 benötigt im Datenfeld 4 M-Merkerbytes für Ein- und Ausgangsparameter:

MB x + 0:	Empfangs-CPU	Eingangsparameter
MB x + 1:	—	nicht belegt
MB x + 2:	Anzeigenbyte	Ausgangsparameter
MB x + 3:	Sende-Kapazität	Ausgangsparameter

#### 2. AKKU-1-L:

Nr. des 1. Merkerbytes 'x' im Datenfeld,  
zulässige Werte:

AKKU-1-LH: 0  
AKKU-1-LL: 0 bis 246

Die Funktion SENDE-TEST ermittelt die Anzahl der freien Speicherblöcke im Zwischenspeicher des Koordinators KOR 923C.

Entsprechend dieser Anzahl m kann die Funktion SENDEN m-mal aufgerufen werden um m Datenblöcke zu übergeben.

### 5.1 Eingangsparameter

#### 5.1.1 Empfangs-CPU

Die Nummer der "eigenen" CPU sowie die Nummer der Empfangs- CPU kennzeichnen diejenige Verbindungsstrecke, für die die Sende-Kapazität (s.u.) ermittelt wird.

### 5.2 Ausgangsparameter

#### 5.2.1 Anzeigenbyte

Dieses Byte informiert Sie, ob die Funktion SENDE-TEST korrekt und vollständig abgearbeitet worden ist.

#### Fehler:

Ein aufgetretener Fehler erfordert eine Änderung in der Programmierung/Parametrierung.

**Fehler-Nummer (bytwweise Auswertung des Anzeigenbytes):**

- (65) Der Parameter "Empfangs-CPU" ist unzulässig. Unterscheiden Sie folgende Fälle:

Die Nummer der Empfangs-CPU ist größer 4.

Die Nummer der Empfangs-CPU ist kleiner 1.

Die Nummer der Empfangs-CPU ist identisch mit der "eigenen" Nummer.

- (67) Der Aufruf des Sonderfunktions-Organisationsbausteins ist fehlerhaft. Unterscheiden Sie folgende Fälle:

Folgefehler, da die Funktion INITIALISIEREN nicht aufgerufen oder mit Initialisierungskonflikt beendet wurde.

Doppelaufruf: Der Aufruf dieser Funktion SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST ist unzulässig, da in dieser CPU bereits in einer untergeordneten Bearbeitungsebene (z.B. zyklische Programmbearbeitung) eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST aufgerufen wurde (siehe Kap. "Aufruf und Schachtelung der Sonderfunktions-Organisationsbausteine").

Die "eigene" CPU-Nummer ist fehlerhaft (Systemdaten zerstört); nach NETZ EIN/NETZ AUS wird die CPU-Nr. vom Systemprogramm erneut erzeugt.

- (68) Die Verwaltungsdaten (Warteschlangenverwaltung) der angewählten Verbindungsstrecken sind fehlerhaft; der Zwischenspeicher im Koordinator C ist mit Hilfe der Funktion INITIALISIEREN neu einzurichten.

**Warnung:**

Eine Anzeige der Nummerngruppe "Warnung" kann bei der Funktion SENDE-TEST nicht auftreten.

**Initialisierungskonflikt:**

Eine Anzeige der Nummerngruppe "Initialisierungskonflikt" kann bei der Funktion SENDE-TEST nicht auftreten.

## 5.2.2 Sende-Kapazität

Der Parameter "Sende-Kapazität" zeigt an, wieviel Datenblöcke gesendet und zwischengespeichert werden können.

## 6 Funktion EMPFANGEN (OB 204)

### Aufrufparameter

#### 1. Datenfeld

Vor Aufruf des OB 204 müssen Sie im Datenfeld die Eingangsparameter bereitstellen.  
Der OB 204 benötigt im Datenfeld 10 M-Merkerbytes für Ein- und Ausgangsparameter:

MB x + 0:	Sende-CPU	Eingangsparameter
MB x + 1:	—	nicht belegt
MB x + 2:	Anzeigenbyte	Ausgangsparameter
MB x + 3:	Empfangs-Kapazität	Ausgangsparameter
MB x + 4:	Baustein-Kennung	Ausgangsparameter
MB x + 5:	Baustein-Nummer	Ausgangsparameter
MB x + 6:	Adresse des ersten	Ausgangsparameter
MB x + 7:	empfangenen Datenwortes	Ausgangsparameter
MB x + 8:	Adresse des letzten	Ausgangsparameter
MB x + 9:	empfangenen Datenwortes	Ausgangsparameter

#### 2. AKKU-1-L

Nr. des 1. Merkerbytes 'x' im Datenfeld,  
zulässige Werte:                      AKKU-1-LH: 0  
   AKKU-1-LL: 0 bis 246

Die Funktion EMPFANGEN übernimmt einen Datenblock vom Zwischenspeicher des Koordinators KOR 923C. Zusätzlich zeigt sie an, wieviele Datenblöcke noch zwischengespeichert sind und empfangen werden können.

Die Funktion EMPFANGEN sollte in einer Schleife so oft aufgerufen werden, bis alle zwischengespeicherten Datenblöcke übernommen sind.

### 6.1 Eingangsparameter

#### 6.1.1 Sende-CPU

Der Empfangs-Baustein empfängt Daten, die die Sende-CPU geliefert hat; der zulässige Wert liegt zwischen 1 und 4, muß jedoch ungleich der "eigenen" Nummer sein.

## 6.2 Ausgangsparameter

### 6.2.1 Anzeigenbyte

Dieses Byte informiert Sie, ob die Funktion EMPFANGEN korrekt und vollständig abgearbeitet worden ist.

#### Fehler:

Ein aufgetretener Fehler erfordert eine Änderung in der Programmierung/Parametrierung.

#### Fehler-Nummer (bitweise Auswertung des Anzeigenbytes):

- (66) Der Parameter "Sende-CPU" ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Nummer der Sende-CPU ist größer 4.
  - Die Nummer der Sende-CPU ist kleiner 1.
  - Die Nummer der Sende-CPU ist gleich der "eigenen" Nummer.
- (67) Der Aufruf des Sonderfunktions-Organisationsbausteins ist fehlerhaft. Unterscheiden Sie folgende Fälle:
  - Folgefehler, da die Funktion INITIALISIEREN nicht aufgerufen oder mit Initialisierungskonflikt beendet wurde.
  - Doppelaufruf: Der Aufruf dieser Funktion SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST ist unzulässig, da in dieser CPU bereits in einer untergeordneten Bearbeitungsebene (z.B. zyklische Programmbearbeitung) eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST aufgerufen wurde (siehe Kap. "Aufruf und Schachtelung der Sonderfunktions-Organisationsbausteine").
  - Die "eigene" CPU-Nummer ist fehlerhaft (Systemdaten zerstört); nach NETZ EIN/NETZ AUS wird die CPU-Nr. vom Systemprogramm erneut erzeugt.
- (68) Die Verwaltungsdaten (Warteschlangenverwaltung) der angewählten Verbindungsstrecken sind fehlerhaft; der Zwischenspeicher im Koordinator C ist mit Hilfe der Funktion INITIALISIEREN neu einzurichten.
- (69) Die vom Sender überlieferte Baustein-Kennung ist unzulässig. Unterscheiden Sie folgende Fälle:
  - Die Baustein-Kennung ist kleiner 1.
  - Die Baustein-Kennung ist größer 2.

- (70) Die vom Sender überlieferte Baustein-Nummer ist unzulässig, da es sich um einen Datenbaustein mit spezieller Bedeutung handelt. Unterscheiden Sie folgende Fälle:

Falls Baustein-Kennung = 1 : DB 0, DB 1, DB 2

Falls Baustein-Kennung = 2 : DX 0, DX 1, DX 2

- (71) Die vom Sender überlieferte Baustein-Nummer ist fehlerhaft. Der parametrisierte Datenbaustein existiert nicht.
- (73) Der Datenbaustein ist zu klein, um den vom Sender gelieferten Datenblock aufzunehmen.

**Warnung:**

Die Funktion konnte nicht durchgeführt werden; der Funktionsaufruf ist zu wiederholen, z.B. im nächsten Zyklus.

**Warnungs-Nummer (bitweise Auswertung des Anzeigenbytes):**

- (130) Die Funktion EMPFANGEN kann keine Daten übernehmen, da die Empfangs-Kapazität bereits beim Funktionsaufruf gleich Null war.

**Initialisierungskonflikt:**

Eine Anzeige der Nummerngruppe "Initialisierungskonflikt" kann bei der Funktion EMPFANGEN nicht auftreten.

## **6.2.2 Empfangs-Kapazität**

Der Parameter "Empfangs-Kapazität" zeigt an, wieviel Datenblöcke noch zwischengespeichert sind und empfangen werden können.

## **6.2.3 Baustein-Kennung und -Nummer**

**Baustein-Kennung:**

Kennung = 1: DB-Datenbaustein  
Kennung = 2: DX-Datenbaustein

### **Baustein-Nummer**

Die Baustein-Nummer ergibt zusammen mit der Baustein-Kennung (s.o.) und den Adressen des ersten und letzten Datenworts (s.u.) den Bereich, in dem die Empfangs-Daten von der Funktion EMPFANGEN abgelegt worden sind (und aus dem sie in der Sende-CPU von der Funktion SENDEN entnommen worden sind).

Hierbei ist zu beachten, daß sich die Empfangs-Datenbausteine in einem Schreib-/Lese-Speicher (RAM) befinden sollten; die Verwendung von Nur-Lese-Speichern (EPROM) ist höchstens bei Sende-Datenbausteinen sinnvoll.

### **6.2.4 Adresse des ersten/letzten empfangenen Datenwortes**

Die Differenz zwischen den Adressen des ersten und des letzten übertragenen Datenwortes beträgt maximal 31, da pro Funktionsaufruf maximal 32 Wörter übertragen werden.



## 7 Funktion EMPFANGS-TEST (OB 205)

### Aufrufparameter:

#### 1. Datenfeld

Vor Aufruf des OB 205 müssen Sie im Datenfeld die Eingangsparameter bereitstellen.

Der OB 205 benötigt im Datenfeld 4 M-Merkerbytes für Ein- und Ausgangsparameter:

MB x + 0:	Sende-CPU	Eingangsparameter
MB x + 1:	—	nicht belegt
MB x + 2:	Anzeigenbyte	Ausgangsparameter
MB x + 3:	Empfangs-Kapazität	Ausgangsparameter

#### 2. AKKU-1-L:

Nr. des 1. Merkerbytes 'x' im Datenfeld,  
zulässige Werte:

AKKU-1-LH: 0  
AKKU-1-LL: 0 bis 246

Die Funktion EMPFANGS-TEST ermittelt die Anzahl belegter Speicherblöcke im Zwischenspeicher des Koordinators KOR 923C. Entsprechend dieser Anzahl m kann die Funktion EMPFANGEN m-mal aufgerufen werden um m Datenblöcke zu übernehmen.

### 7.1 Eingangsparameter

#### 7.1.1 Sende-CPU

Die Nummer der "eigenen" CPU sowie die Nummer der Sende-CPU kennzeichnen diejenige Verbindungsstrecke, für die die Empfangs-Kapazität (s.u.) ermittelt wird.

### 7.2 Ausgangsparameter

#### 7.2.1 Anzeigenbyte

Dieses Byte informiert Sie, ob die Funktion EMPFANGS-TEST korrekt und vollständig abgearbeitet worden ist.

### Fehler:

Ein aufgetretener Fehler erfordert eine Änderung in der Programmierung/Parametrierung.

**Fehler-Nummer (byteweise Auswertung des Anzeigenbytes):**

- **(66)** Der Parameter "Sende-CPU" ist unzulässig. Unterscheiden Sie folgende Fälle:

Die Nummer der Sende-CPU ist größer 4.

Die Nummer der Sende-CPU ist kleiner 1.

Die Nummer der Sende-CPU ist identisch mit der "eigenen" Nummer.

- **(67)** Der Aufruf des Sonderfunktions-Organisationsbausteins ist fehlerhaft. Unterscheiden Sie folgende Fälle:

Folgefehler, da die Funktion INITIALISIEREN nicht aufgerufen oder mit Initialisierungskonflikt beendet wurde.

Doppelaufruf: Der Aufruf dieser Funktion SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST ist unzulässig, da in dieser CPU bereits in einer untergeordneten Bearbeitungsebene (z.B. zyklische Programmbearbeitung) eine der Funktionen INITIALISIEREN, SENDEN, SENDE-TEST, EMPFANGEN oder EMPFANGS-TEST aufgerufen wurde (siehe Kap. "Aufruf und Schachtelung der Sonderfunktions-Organisationsbausteine").

Die "eigene" CPU-Nummer ist fehlerhaft (Systemdaten zerstört); nach NETZ EIN/NETZ AUS wird die CPU-Nr. vom Systemprogramm erneut erzeugt.

- **(68)** Die Verwaltungsdaten (Warteschlangenverwaltung) der angewählten Verbindungsstrecken sind fehlerhaft; der Zwischenspeicher im Koordinator C ist mit Hilfe der Funktion INITIALISIEREN neu einzurichten.

**Warnung:**

Eine Anzeige der Nummerngruppe "Warnung" kann bei der Funktion EMPFANGS-TEST nicht auftreten.

**Initialisierungskonflikt:**

Eine Anzeige der Nummerngruppe "Initialisierung" kann bei der Funktion EMPFANGS-TEST nicht auftreten.

## **7.2.2 Empfangs-Kapazität**

Der Parameter "Empfangs-Kapazität" zeigt an, wieviel Datenblöcke zwischengespeichert sind und empfangen werden können.

## 8 Anwendungen

Bei Verwendung eines der nachfolgend aufgeführten Funktionsbausteine und von Alarmen (z.B. OB 2) ist darauf zu achten, daß am Anfang einer Unterbrechungsbehandlung die Schmiermerker gerettet und am Ende wieder zurückgeschrieben werden.



### BEACHTEN SIE:

Dies gilt auch bei der Einstellung "Unterbrechung an Bausteingrenzen", da der Aufruf der Sonderfunktions-Organisationsbausteine eine Bausteingrenze darstellt.

### 8.1 Aufruf der Sonderfunktions-OB über Funktionsbausteine

Die nachfolgend vorgestellten fünf Funktionsbausteine (FB 200 und FB 202 bis FB 205) enthalten den Aufruf des jeweiligen Sonderfunktions-Organisationsbausteins zur Mehrprozessorkommunikation (OB 200 und OB 202 bis OB 205).

Die Nummern der Funktionsbausteine sind frei gewählt und können geändert werden. Die Parameter der Sonderfunktions-OB werden bei Aufruf der Funktionsbausteine als Aktualparameter übergeben. Der direkte Aufruf der Sonderfunktions-Organisationsbausteine ist zwar lauffähiger, aber wegen der fehlenden Formalparameter schwieriger lesbar.

FB-Nr.:	FB-Name:	Funktion
FB 200	INITIAL	Vorbesetzen
FB 202	SENDEN	Senden eines Datenblockes
FB 203	SEND-TST	Sendemöglichkeit testen
FB 204	EMPFANG	Empfangen eines Datenblockes
FB 205	EMPF-TST	Empfangsmöglichkeit testen

Der Merkerbereich von MB 246 bis maximal MB 255 wird von den Funktionsbausteinen als Parameterfeld für die Sonderfunktions-Organisationsbausteine benutzt.

Die genaue Bedeutung der Ein- und Ausgangsparameter ist der Beschreibung des verwendeten Sonderfunktions-Organisationsbausteins zu entnehmen.



### BEACHTEN SIE:

Es handelt sich bei den folgenden Anwendungsbeispielen um fertige Applikationen, die Sie durch Abschreiben programmieren können.

8.1.1 Vorsetzen der Verbindung (FB 200)

Parameter-	Bedeutung			Art	Typ	Parameter-
Name						feld
AUMA	Automatik/Manuell	E			BY	MB 246
ANZC	Anzahl CPUs	E			BY	MB 247
TNZU	Typ (H-Byte) und Nummer (L-Byte) des Datenaustauschs, der die Zuordnungsliste enthält	E			W	MW 248
ANFZ	Anfangsadresse der Zuordnungsliste	E			W	MW 250
INIK	Initialisierungs-Konflikt	A			BY	MB 252
GKAP	Gesamt-Kapazität	A			BY	MB 253

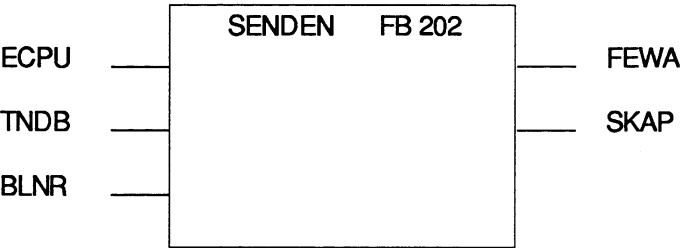


FB 200				LAE= 45	ABS
NETZWERK 1					
NAME:INITIAL					
BEZ :AUMA	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY	
BEZ :ANZC	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY	
BEZ :TNZU	E/A/D/B/T/Z:	E	BI/BY/W/D:	W	
BEZ :ANFZ	E/A/D/B/T/Z:	E	BI/BY/W/D:	W	
BEZ :INIK	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY	
BEZ :GKAP	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY	

0017	:L	= AUMA	AUTOMATIK/MANUELL
0018	:T	MB 246	
0019	:L	= ANZC	ANZAHL CPUs
001A	:T	MB 247	
001B	:L	= TNZU	DB-TYP, DB-NR.
001C	:T	MW 248	
001D	:L	= ANFZ	ANFANGSADRESSE DER ZUORDNUNGSLISTE
001E	:T	MW 250	
001 F	:		
0020	:L	KB 246	SF-OB:
0021	:SPA	OB 200	INITIALISIEREN
0022	:		
0023	:L	MB 252	INITIALISIERUNGS-KONFLIKT
0024	:T	= INIK	
0025	:L	MB 253	GESAMT-KAPAZITÄT
0026	:T	= GKAP	
0027	:BE		

8.1.2 Senden eines Datenblocks (FB 202)

Parameter-Name	Bedeutung	Art	Typ	Parameter-Feld
ECPU	Empfangs-CPU	E	BY	MB 246
TNDB	Typ (H-Byte) und Nummer (L-Byte) des Quell-Datenbausteins	E	W	MW 247
BLNR	Block-Nummer	E	BY	MB 249
FEWA	Fehler/Warnung	A	BY	MB 250
SKAP	Sende-Kapazität	A	BY	MB 251

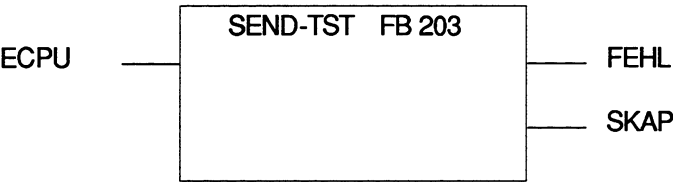


FB 202  
NETZWERK 1  
NAME:SENDEN  
BEZ :ECPU E/A/D/B/T/Z: E BI/BY/W/D: BY  
BEZ :TNDB E/A/D/B/T/Z: E BI/BY/W/D: W  
BEZ :BLNR E/A/D/B/T/Z: E BI/BY/W/D: BY  
BEZ :FEWA E/A/D/B/T/Z: A BI/BY/W/D: BY  
BEZ :SKAP E/A/D/B/T/Z: A BI/BY/W/D: BY

0014	:L	= ECPU	EMPFANGS-CPU
0015	:T	MB 246	
0016	:L	= TNDB	DB-TYP, DB-NR.
0017	:T	MW 247	
0018	:L	= BLNR	BLOCK-NUMMER
0019	:T	MB 249	
001A	:		
001B	:L	KB 246	SF-OB:
001C	:SPA	OB 202	SENDEN EINES DATENBLOCKES
001D	:		
001E	:L	MB 250	FEHLERWARNUNG
001F	:T	= FEWA	
0020	:L	MB 251	SENDE-KAPAZITÄT
0021	:T	= SKAP	
0022	:BE		

8.1.3 Sendemöglichkeit testen (FB 203)

Parameter-Name	Bedeutung	Art	Typ	Parameter-Feld
ECPU	Empfangs-CPU	E	BY	MB 246
FEHL	Fehler	A	BY	MB 248
SKAP	Sende-Kapazität	A	BY	MB 249



FB 203

NETZWERK 1

NAME:SEND-TST

BEZ :ECPU E/A/D/B/T/Z: E BI/BY/W/D: BY

BEZ :FEHL E/A/D/B/T/Z: A BI/BY/W/D: BY

BEZ :SKAP E/A/D/B/T/Z: A BI/BY/W/D: BY

LAE= 30

ABS

000E :L = ECPU EMPFANGS-CPU

000F :T MB 246

0010 :

0011 :L KB 246

0012 :SPA OB 203 SF-OB:SENDEMÖGLICHKEIT TESTEN

0013 :

0014 :L MB 248 FEHLER

0015 :T = FEHL

0016 :L MB 249 SENDE-KAPAZITÄT

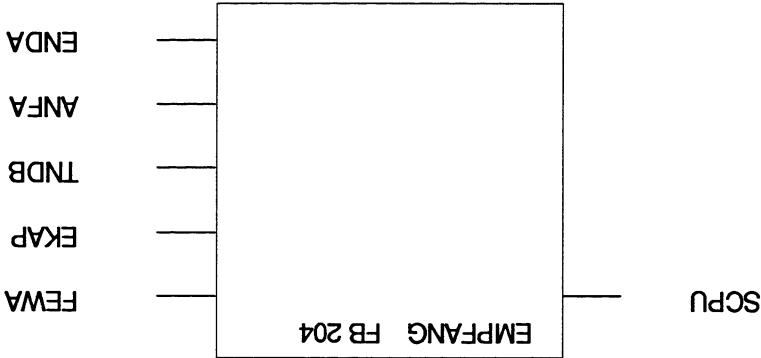
0017 :T = SKAP

0018 :BE



8.1.4 Empfangen eines Datenblocks (FB 204)

Parameter-Name	Bedeutung			Art	Typ	Parameter-Feld
SCPU	Sende-CPU			E	BY	MB 246
FEWA	Fehler/Warnung			A	BY	MB 248
EKAP	Empfangs-Kapazität			A	BY	MB 249
TNDB	Typ (H-Byte) und Nummer (L-Byte) des Ziel-Datenbausteins			A	W	MW 250
ANFA	Adresse des ersten empfangenen Datenwortes (Anfangsadresse)			A	W	MW 252
ENDA	Adresse des letzten empfangenen Datenwortes (Endadresse)			A	W	MW 254



FB 204

LAE= 45    ABS

NETZWERK 1

NAME:EMPFANG

BEZ	:SCPU	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY
BEZ	:FEWA	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY
BEZ	:EKAP	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY
BEZ	:TNDB	E/A/D/B/T/Z:	A	BI/BY/W/D:	W
BEZ	:ANFA	E/A/D/B/T/Z:	A	BI/BY/W/D:	W
BEZ	:ENDA	E/A/D/B/T/Z:	A	BI/BY/W/D:	W

0017	:L	= SCPU
0018	:T	MB 246
0019	:	
001A	:L	KB 246
001B	:SPA	OB 204

SENDE-CPU

SF-OB:  
EMPFANGEN EINES DATEN-  
BLOCKES

001C	:	
001D	:L	MB 248
001E	:T	= FEWA
001F	:L	MB 249
0020	:T	= EKAP
0021	:L	MW 250
0022	:T	= TNDB
0023	:L	MW 252
0024	:T	= ANFA
0025	:L	MW 254
0026	:T	= ENDA
0027	:BE	

FEHLERWARNUNG

EMPFANGS-KAPAZITÄT

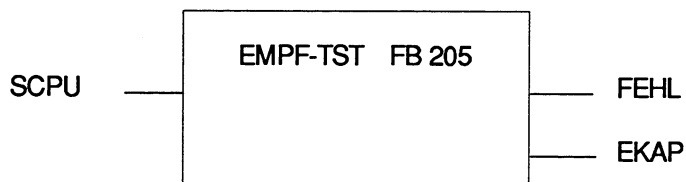
DB-TYP, DB-NR.

ANFANGSADRESSE

ENDADRESSE

### 8.1.5 Empfangsmöglichkeit testen (FB 205)

Parameter-Name	Bedeutung	Art	Typ	Parameter-Feld
SCPU	Sende-CPU	E	BY	MB 246
FEHL	Fehler	A	BY	MB 248
EKAP	Empfangs-Kapazität	A	BY	MB 249



FB 205  
 NETZWERK 1  
 NAME:EMPF-TST  
 BEZ :SCPU E/A/D/B/T/Z: E BI/BY/W/D: BY  
 BEZ :FEHL E/A/D/B/T/Z: A BI/BY/W/D: BY  
 BEZ :EKAP E/A/D/B/T/Z: A BI/BY/W/D: BY

000E :L = SCPU SENDE-CPU  
 000F :T MB 246  
 0010 :  
 0011 :L KB 246  
 0012 :SPA OB 205 SF-OB:  
 0013 : EMPFANGSMÖGLICHKEIT TESTEN  
 0014 :L MB 248 FEHLER  
 0015 :T = FEHL  
 0016 :L MB 249 EMPFANGS-KAPAZITÄT  
 0017 :T = EKAP  
 0018 :BE

## 8.2 Übertragen von Datenbausteinen

### 8.2.1 Funktionsbeschreibung

Der Funktionsbaustein UEBT-DAT (FB 110) überträgt eine parametrierbare Anzahl von Datenblöcken aus einem Datenbaustein einer CPU in den Datenbaustein gleichen Typs und gleicher Nummer einer anderen CPU. (Beschreibung der Parameterliste, Funktionsblock und STEP5-Programm siehe folgende Seite.)

Die FB-Nummer ist zufällig gewählt und kann geändert werden.

### 8.2.2 Übertragen eines Datenbausteins (FB 110)

Der zu übertragende Datenbereich wird über die Eingangsparameter ERSB ( = Nummer des ersten zu übertragenden Datenblocks) und ANZB ( = Anzahl der zu übertragenden Datenblöcke) festgelegt. Ein Datenblock besteht normalerweise aus 32 Datenwörtern. Je nach Datenbaustein-Länge werden beim letzten Datenblock gegebenenfalls weniger als 32 Datenwörter übertragen.

Die Übertragung wird mit einer positiven Flanke am Starteingang STAR angestoßen. Ist anschließend der Ausgangsparameter REST gleich Null, so konnte der Funktionsbaustein UEBT-DAT alle Datenblöcke (gemäß Parameter ANZB) senden.

Enthält der Ausgangsparameter REST jedoch einen Wert größer als Null, sind - z.B. im nächsten Zyklus - Folgeaufrufe erforderlich. In diesem Fall darf der gesamte Parametersatz (d.h. die Werte aller Parameter) vom Anwender(-Programm) erst dann verändert werden, wenn der Ausgangsparameter REST den Wert Null hat, d. h. wenn die Datenübertragung abgeschlossen ist.

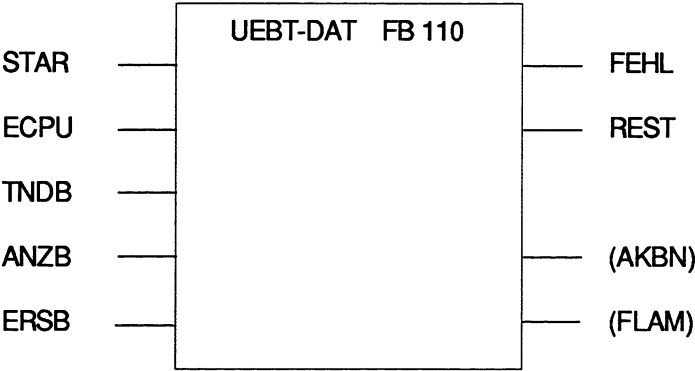
Der mehrfache Aufruf des Funktionsbausteins UEBT-DAT mit jeweils verschiedenen Parametersätzen ist möglich. Hierbei werden verschiedene Datenbereiche gleichzeitig ("ineinander verzahnt") übertragen. Zusätzlich können die Sonderfunktions-Organisationsbausteine zur Mehrprozessorkommunikation OB 202 bis OB 205 "direkt" eingesetzt werden. Von dieser Möglichkeit wird im Anwendungsbeispiel Gebrauch gemacht.

Falls innerhalb des Funktionsbausteins UEBT-DAT die Funktion SENDEN (OB 202) nicht korrekt abgearbeitet werden konnte, wird die jeweilige Fehlernummer im Ausgangsparameter FEHL übergeben, das VKE = '1' und der Ausgangsparameter REST = '0' gesetzt.

Der Funktionsbaustein UEBT-DAT verwendet die Merkerbytes MB 246 bis MB 251 als Schmiermerker. Alle anderen Variablen, deren Wert solange von Bedeutung ist, bis nach mehrfachem Aufruf des Funktionsbausteins UEBT-DAT der Ausgangsparameter REST = '0' ist, erhalten über den Mechanismus der Formal-/Aktual-Parameter Speicherplätze zugewiesen. Dieses Verfahren ist erforderlich, damit verschiedene Datenbausteine gleichzeitig übertragen werden können.

Parameter-Name	Bedeutung	Art	Typ
STAR	Übertragung des Datenbausteins nach positiver Flanke <b>starten</b> .	E	BI
ECPU	<b>E</b> mpfangs- <b>C</b> PU	E	BY
TNDB	<b>T</b> yp (H-Byte) und <b>N</b> ummer (L-Byte) des zu übertragenden <b>D</b> aten <b>a</b> usteins.	E	W
ANZB	<b>A</b> nzahl der zu übertragenden Daten <b>bl</b> öcke.	E	BY
ERSB	Nummer des <b>e</b> rsten zu übertragenden Daten <b>bl</b> ocks.	E	BY
FEHL	<b>F</b> ehler	A	BY
REST	Anzahl der noch zu übertragenden Datenblöcke.	A	BY
AKBN <sup>1)</sup>	<b>A</b> ktuelle <b>B</b> lock-Nummer	A	BY
FLAM <sup>1)</sup>	<b>F</b> lanken <b>m</b> erker	A	BI

<sup>1)</sup> Interne Zwischenmerker, nicht zur Auswertung vorgesehen.



FB 110

LAE= 89    ABS

NETZWERK 1

NAME:UEBT-DAT

BEZ	:STAR	E/A/D/B/T/Z:	E	BI/BY/W/D:	BI
BEZ	:ECPU	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY
BEZ	:TNDB	E/A/D/B/T/Z:	E	BI/BY/W/D:	W
BEZ	:ANZB	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY
BEZ	:ERSB	E/A/D/B/T/Z:	E	BI/BY/W/D:	BY
BEZ	:FEHL	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY
BEZ	:REST	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY
BEZ	:AKBN	E/A/D/B/T/Z:	A	BI/BY/W/D:	BY
BEZ	:FLAM	E/A/D/B/T/Z:	A	BI/BY/W/D:	BI

0020	:L	= ECPU	PARAMETERFELD FÜR SF-OB 202
0021	:T	MB 246	VORBESETZEN
0022	:L	= TNDB	
0023	:T	MW 247	
0024	:		
0025	:L	= REST	ZUERST EVENTUELL NOCH VOR-
0026	:L	KB 0	HANDENE BLÖCKE AUSSENDEN
0027	:>< F		
0028	:SPB	= UEBT	
0029	:		
002A	:UN	= STAR	POSITIVE FLANKE AM START-
002B	:RB	= FLAM	EINGANG ?
002C	:ON	= STAR	
002D	:O	= FLAM	
002E	:SPB	= GUT	
002F	:S	= FLAM	
0030	:		
0031	:L	= ANZB	DIE GLOBALEN MERKER NACH
0032	:T	= REST	EINER POSITIVEN FLANKE AM
0033	:L	= ERSB	EINGANG START INITIALISIEREN
0034	:T	= AKBN	
0035	:		
0036	:L	= REST	SOLANGE REST > < 0 IST,
0038 SCHL	:L	KF+ 0	WEITERHIN VERSUCHEN, DATEN-
0039	:!= F		BLÖCKE AUSZUSENDEN
003A	:SPB	= GUT	
003B UEBT	:L	= AKBN	
003C	:T	MB 249	
003D	:L	KB 246	SF-OB:
003E	:SPA	OB 202	SENDEN EINES DATENBLOCKES
003F	:L	MB 250	
0040	:SPM	= FEHL	ABBRUCH BEI FEHLER
0041	:SPP	= GUT	ABBRUCH, FALLS SENDE-KAP. = 0
0042	:L	= AKBN	BLOCK-NUMMER
0043	:I	1	INKREMENTIEREN
0044	:T	= AKBN	
0045	:L	= REST	ANZAHL DER VERBLEIBENDEN
0046	:D	1	BLÖCKE DEKREMENTIEREN

0047	:T	= REST	
0048	:SPA	= SCHL	
0049	:		
004A GUT	:U	M 0.0	REGULÄRES PROGRAMMENDE:
004B	:UN	M 0.0	
004C	:L	KB 0	VKE = 0, FEHL = 0
004D	:T	= FEHL	
004E	:BEA		
004F	:		
0050 FEHL	:T	= FEHL	PROGRAMMENDE BEI FEHLER:
0051	:L	KB 0	
0052	:T	= REST	VKE = 1, FEHL ENTHÄLT FEHLERNUMMER
0053	:BE		

### 8.2.3 Anwendungsbeispiel (für AG S5-135U)

Die CPU 1 soll im zyklischen Anwenderprogramm die Datenbausteine DB 3 (Datenblöcke 2 bis 5) und DB 4 (Datenblöcke 1 bis 3) an die CPU 2 senden. In der CPU 2 soll ebenfalls im zyklischen Anwenderprogramm die Funktion EMPFANGEN (OB 204) aufgerufen werden.

Folgende Bausteine müssen in die einzelnen CPUs geladen werden:

Funktion	CPU 1	CPU 2
Neustart-Baustein	OB 20	—
Zyklus-Baustein <sup>1)</sup>	FB 0	FB 0
Sende-DB	DB 3; DB 4	—
Empfangs-DB	—	DB 3; DB 4

<sup>1)</sup> In CPU 946/947 ist als Zyklus-Baustein nur der OB 1 zulässig.

Der OB 20 ruft die Funktion INITIALISIEREN (OB 200) auf und reserviert für die Verbindungsstrecke von CPU 1 nach CPU 2 einige Speicherblöcke.

Das zyklische Anwenderprogramm im Funktionsbaustein FB 0 der CPU 1 enthält zweimal den Aufruf des Funktionsbausteins UEBT-DAT mit jeweils unterschiedlichen Parametersätzen. Nach einer positiven Flanke am Eingang E 2.0 beginnt die Übertragung des ersten Datenbausteins DB 3. Eine positive Flanke am Eingang E 2.1 startet die Übertragung des zweiten Datenbausteins DB 4.

FB 0  
NETZWERK 1  
NAME:DEMO

LAE= 66    ABS

```

0005      :L   KB 2
0006      :T   MB 0
0007      :L   KY 1,3
0009      :T   MW 1
000A      :L   KB 4
000B      :T   MB 3
000C      :L   KB 2
000D      :T   MB 4
000E      :
000F      :SPA FB 110
0010 NAME :UEBT-DAT
0011 STAR :    E 2.0
0012 ECPU :    MB 0
0013 TNDB :    MW 1
0014 ANZB :    MB 3
0015 ERSB :    MB 4
0016 FEHL :    MB 5
0017 REST :    MB 6
0018 AKBN :    MB 7
0019 FLAM :    M 8.0
001A      :
001B      :
001C      :SPB = HALT
001D      :
001E      :L   KB 2
001F      :T   MB 10
0020      :L   KY 1,4
0022      :T   MW 11
0023      :L   KB 3
0024      :T   MB 13
0025      :L   KB 1
0026      :T   MB 14
0027      :
0028      :SPA FB 110
0029 NAME :UEBT-DAT
002A STAR :    E 2.1
002B ECPU :    MB 10
002C TNDB :    MW 11
002D ANZB :    MB 13
002E ERSB :    MB 14
002F FEHL :    MB 5
0030 REST :    MB 16
0031 AKBN :    MB 17
0032 FLAM :    M 8.1
0033      :
0034      :
0035      :SPB = HALT
0036      :BEA

```

AN DIE CPU 2 ..

.. AUS DEM DATENBAUSTEIN DB 3

.. VIER DATENBLÖCKE

.. AB DATENBLOCK 2 SENDEN

ABBRUCH NACH FEHLER

AN DIE CPU 2 ..

.. AUS DEM DATENBAUSTEIN DB 4

.. DREI DATENBLÖCKE

.. AB DATENBLOCK 1 SENDEN

ABBRUCH NACH FEHLER



0037 :  
0038 HALT :

Hier erfolgt die Fehlerbehandlung (z.B. Stop, Meldungsabgabe auf Drucker, ...)

0039 :BE

In der CPU 2 überträgt die vom FB 0 aufgerufene Funktion EMPFANGEN (OB 204) jeden ausgesendeten Datenblock in den zugehörigen Datenbaustein. Der vollständige Empfang eines Datenbausteins kann sich über mehrere Zyklen verteilen.

FB 0  
NETZWERK 1  
NAME:EMPF-DAT

LAE= 26 ABS

0005	:L	KB 1	DATEN VON CPU 1 EMPFANGEN
0006	:T	MB 246	
0007	:		
0008	SCHL	:L KB 246	SF-OB:
0009	:	<b>SPA OB 204</b>	EMPFANGEN
000A	:	SPM = FEHL	ABBRUCH BEI FEHLER
000B	:L	MB 249	DIE FUNKTION EMPFANGEN
000C	:L	KB 0	WIRD SOLANGE AUFGERUFEN,
000D	:	>< F	BIS DER ZWISCHENSPEICHER
000E	:	SPB = SCHL	KEINE WEITEREN DATENBLÖCKE
000F	:		MEHR ENTHÄLT, D. H. DIE
0010	:	BEA	EMPFANGSKAPAZITÄT = 0 IST.-
0011	FEHL	:	

Hier erfolgt die Fehlerbehandlung (z.B. Stop, Meldungsabgabe auf Drucker, ...)

0012 :BE

## 8.3 Erweiterung des Koppelmerkerbereichs

### 8.3.1 Problemstellung

In den Mehrprozessor-Automatisierungsgeräten AG S5-135U und AG S5-155U kann jedes der 256 Merkerbytes einer CPU durch Eintrag in den Datenbaustein DB 1 zum Eingangs- oder Ausgangs-Koppelmerker werden. Dadurch verringert sich jedoch die Anzahl der "normal" verwendbaren Merkerbytes. Weiterhin sind zur Übertragung eines Datensatzes (mehrere Bytes) zusätzliche Maßnahmen (Semaphor-Variable oder DX-0-Parametrierung "Koppelmerker im Block übertragen") notwendig, um zu verhindern, daß der Empfänger einen nur teilweise übertragenen Datensatz auswertet.

### 8.3.2 Lösung

Aufeinanderfolgende Datenwörter eines DB- oder DX- Datenbausteins, jeweils ab DW 0, werden als "Koppel-Datenwörter" definiert. Jede Verbindungsstrecke erhält "ihren" Datenbaustein und ist von den anderen Verbindungsstrecken völlig **unabhängig**.

Zu Beginn des Zyklus-Bausteins (CPU 946/947: OB 1, CPU 92x: OB 1 oder FB 0) werden mit Hilfe der Sonderfunktions-Organisationsbausteine zur Mehrprozessor-Kommunikation die Koppel-Datenwörter empfangen. Es folgt das "reguläre" zyklische Programm, welches die empfangenen Daten auswertet und Sende-Daten erzeugt. Sie werden am Zyklusende wiederum mit Hilfe der Sonderfunktions-Organisationsbausteine zur Mehrprozessorkommunikation gesendet. So können sie von den anderen CPUs bei deren Zyklusbeginn empfangen werden.

Für jede der max. 12 möglichen Verbindungsstrecken und unabhängig von den anderen Verbindungsstrecken gilt:

- Die Sende-CPU wird nur aktiv, falls die Empfangs-CPU die "alten" Daten vollständig dem Zwischenspeicher KOR 923C entnommen hat.
- Die Empfangs-CPU wird nur aktiv, falls die Sende-CPU die "neuen" Daten vollständig im Zwischenspeicher KOR 923C abgelegt hat.

Somit steht der Empfangs-CPU entweder ein kompletter neuer Datensatz zur Verfügung oder der alte Datensatz bleibt unverändert: **Keine Mischung von "alten" und "neuen" Daten!**

### 8.3.3 Datenstruktur

Welche Datenwörter (nachfolgend Datenwortbereich genannt) von welcher CPU zu welcher CPU zu übertragen sind, ist in der Verbindungsliste (siehe Tabelle auf der nächsten Seite) beschrieben. Sie befindet sich in einem zusätzlichen Datenbaustein, welcher in allen beteiligten CPUs vorhanden sein muß.

Die Datenwortbereiche beginnen immer ab Datenwort DW 0, ihre Länge wird in Blöcken angegeben. Hierbei ist zu beachten:

- Ein kompletter Block besteht aus 32 Datenwörtern.
- Ist der letzte Block eines Sende-Datenbausteins "angeschnitten", d.h. umfaßt er zwischen einem und 31 Datenwörtern, so werden entsprechend weniger Datenwörter übertragen.
- Ist ein Sende-Datenbaustein länger als die in der Verbindungsliste angegebene Blockanzahl, so können die überzähligen Datenwörter in der entsprechenden CPU verwendet werden.
- Ist ein Empfangs-Datenbaustein länger als der empfangene Datenwortbereich, so können die überzähligen Datenwörter in der entsprechenden CPU verwendet werden.

Aufbau der Verbindungsliste

	TEILLISTE 1			TEILLISTE 2		
Verbindungs- strecke		DB-Typ	DB- Nummer			Block anzahl
von CPU 1 nach ...	DW 0	S 1		DW 16	S 1	
... CPU 2	DW 1	...	...	DW 17	2	...
... CPU 3	DW 2	...	...	DW 18	3	...
... CPU 4	DW 3	...	...	DW 19	4	...
von CPU 2 nach ...	DW 4	S 2		DW 20	S 2	
... CPU 1	DW 5	...	...	DW 21	1	...
... CPU 3	DW 6	b	c	DW 22	3	a
... CPU 4	DW 7	...	...	DW 23	4	...
von CPU 3 nach ...	DW 8	S 3		DW 24	S 3	
... CPU 1	DW 9	...	...	DW 25	1	...
... CPU 2	DW 10	...	...	DW 26	2	...
... CPU 4	DW 11	...	...	DW 27	4	...
von CPU 4 nach ...	DW 12	S 4		DW 28	S 4	
... CPU 1	DW 13	...	...	DW 29	1	...
... CPU 2	DW 14	...	...	DW 30	2	...
... CPU 3	DW 15	...	...	DW 31	3	...
		2 <sup>15</sup>	2 <sup>0</sup> 2 <sup>15</sup>	2 <sup>0</sup>		

Die Verbindungsliste besteht aus zwei ähnlich strukturierten Teillisten zu je 16 Datenwörtern. Ausgehend von jeder der vier Sende-CPU's (S1, S2, S3, S4) sind zur Beschreibung jeder Verbindungsstrecke 3 Einträge vorgesehen:

- **Blockanzahl**

Die Anzahl der Blöcke legt die Größe (= Anzahl der Datenwörter) des zu übertragenden Datenwortbereiches fest. (Nicht vorhandene bzw. nicht genutzte Verbindungsstrecken werden mit Blockanzahl = 0 gekennzeichnet, ebenso bei DB-Typ und DB-Nummer.)

- **DB-Typ**

Typ des Datenbausteins, der den auszusendenden Datenwortbereich enthält.

- **DB-Nummer**

Nummer des Datenbausteins, der den auszusendenden Datenwortbereich enthält.

Diese Einträge können in der obigen Darstellung zeilenweise gelesen und ausgefüllt werden. Um beispielsweise die ersten zwei Datenblöcke aus dem Datenbaustein DB 10 von CPU 2 (S2) an die CPU 3 zu übertragen, ist folgender Eintrag notwendig:

CPU 2 (S 2) sendet ..



Die Teilliste 2 ist identisch mit der für die Funktion INITIALISIEREN (OB 200) benötigten Zuordnungsliste (Betriebsart "Manuell"). Innerhalb des Datenbausteins muß die Teilliste 1 die Datenwörter 0 bis 15 und die Teilliste 2 die Datenwörter 16 bis 31 belegen. Die mit Fettschrift hervorgehobenen Einträge dürfen nicht abgeändert werden.

### 8.3.4 Programmstruktur

Auf dem Koordinator werden von einer CPU beim Anlauf durch Aufruf der Funktion INITIALISIEREN (OB 200) genau so viele Speicherblöcke pro Verbindungsstrecke reserviert, wie Datenblöcke auf dieser Strecke zu übertragen sind.

Zum Aussenden und Empfangen der Datenwortbereiche werden auf jeder CPU zwei Funktionsbausteine verwendet:

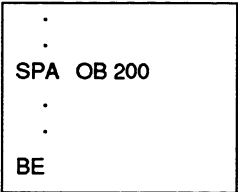
FB-Nr.	Name	Funktion
FB 100	SEND-DAT	Senden von Datenwortbereichen an die restlichen CPUs
FB 101	EMPF-DAT	Empfangen von Datenwortbereichen von den restlichen CPUs

Die FB-Nummern sind zufällig gewählt und können geändert werden.

Die Funktionsbausteine SEND-DAT und EMPF-DAT entnehmen der Verbindungsliste, welche Datenwortbereiche aus welchen Datenbausteinen auszusenden oder zu empfangen sind. Es wird immer der **gesamte** Datenwortbereich ausgesendet oder empfangen. Falls dies, wegen unzureichender Sende- oder Empfangs-Kapazität, noch nicht möglich ist, wird auf das Senden bzw. Empfangen verzichtet.

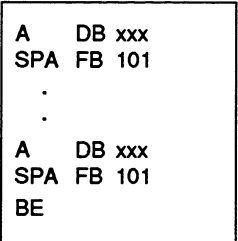
Neustart-OB zum Vorbesetzen  
des Zwischenspeichers im  
Koordinator 923C

OB 200 <sup>1)</sup>



Zyklisches Anwenderprogramm,  
das um den Aufruf der Funktions-  
bausteine EMPF-DAT und SEND-DAT  
erweitert ist.

OB 1 (FB 0)



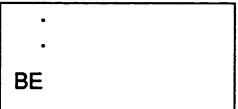
Funktionsbaustein: SEND-DAT  
Senden von Datenwortbereichen

FB 100



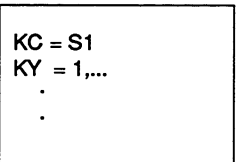
Funktionsbaustein: EMPF-DAT  
Empfangen von Datenwortbereichen

FB 101



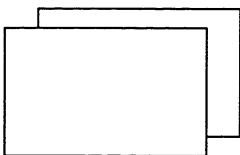
Datenbaustein, der die Ver-  
bindungsline enthält

DB XXX



Maximal drei Eingangs- und drei  
Ausgangsbausteine

DB yyy  
oder/und  
DX zzz



<sup>1)</sup> Der OB 200 darf  
nur in einem Pro-  
zessor aufgerufen  
werden

Bild 3 Übersicht über die in jeder CPU benötigten Bausteine

**BEACHTEN SIE:**

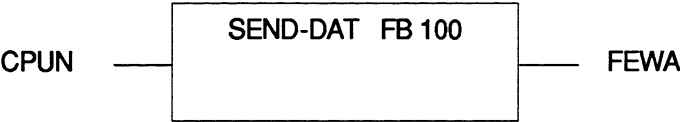


Die Funktionsbausteine SEND-DAT und EMPF-DAT beinhalten die Sonderfunktions-Organisationsbausteine zur Mehrprozessorkommunikation OB 202 bis OB 205. Der zusätzliche Aufruf dieser Organisationsbausteine außerhalb von SEND-DAT / EMPF-DAT ist nicht zulässig!

8.3.5 Senden von Datenwortbereichen (FB 100)

Vor Aufruf des FB 100 muß der Datenbaustein aufgeschlagen werden, der die Verbindungsliste enthält. Der Funktionsbaustein SEND-DAT benötigt zum Auswerten der in der Verbindungsliste enthaltenen Informationen die Nummer der CPU, auf der er aufgerufen wird. Falls innerhalb des Funktionsbausteins die Funktion SENDEN (OB 202) nicht korrekt abgearbeitet werden konnte, wird die jeweilige Fehler- oder Warnungs-Nummer im Ausgangsparameter FEWA übergeben und das VKE = 1 gesetzt. Zusätzlich enthält FEWA bei unzulässigem Eingangsparameter CPUN (CPU-Nummer) den Wert 16 (Bit  $2^4 = 1$ ). Der Funktionsbaustein SEND-DAT verwendet die Merkerbytes MB 239 bis MB 251 als Schmiermerker.

Parameter-Name	Bedeutung	Art	Typ
CPUN	Nummer der CPU, auf der der FB 100 aufgerufen wird. Zulässig sind die Nummern 1 bis 4.	D	KF
FEWA	Fehler/Warnung (siehe Funktion SENDEN (OB 202))	A	BY



FB 100  
 NETZWERK 1 0000  
 NAME:SEND-DAT  
 BEZ :CPUN E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF  
 BEZ :FEWA E/A/D/B/T/Z: A BI/BY/W/D: BY

000B	:LW = CPUN	CPUN = CPUN - 1
000C	:L KB 1	FEHLER FALLS:
000D	:-F	
000E	:SPM = FEWA	CPU-NR. < 1
000F	:L KB 3	
0010	:>F	
0011	:SPB = FEWA	CPU-NR > 4
0012	:TAK	
0013	:	
0014	:SLW 2	CPUN = CPUN * 4
0015	:T MB 245	BASISADRESSE
0016	:	
0017	:L KB 1	
0018	:T MB 244	VERBINDUNGSZÄHLER
0019	:	
001A SCHL	:L MB 245	BASISADRESSE
001B	:L MB 244	+ ZÄHLER
001C	:+F	
001D	:T MW 240	
001E	:ADD BF+ 16	+ OFFSET
001F	:T MW 242	
0020	:	
0021	:B MW 242	
0022	:L DR 0	ANZAHL DER RESERVIERTEN
0023	:T MB 239	BLÖCKE = 0 ?
0024	:L KB 0	
0025	:!= F	
0026	:SPB = LEER	
0027	:	
0028	:B MW 242	
0029	:L DL 0	NR. DER EMPFANGS-CPU
002A	:T MB 246	
002B	:L KB 246	SF-OB:
002C	:SPA OB 203	SENDEMÖGLICHKEIT TESTEN
002D	:L MB 248	ABBRUCH BEI FEHLER
002E	:SPB = OBFE	
002F	:	
0030	:L MB 249	SENDE-KAPAZITÄT > < ANZAHL
0031	:L MB 239	DER RESERVIERTEN BLÖCKE ?
0032	:>< F	
0033	:SPB = LEER	
0034	:	
0035	:L KB 0	BLOCKZÄHLER
0036	:T MB 249	
0037	:	
0038	:B MW 240	
0039	:L DW 0	TYP UND NUMMER DES
003A	:T MW 247	QUELL-DB



003B :  
 003C UEBT :L KB 246  
 003D :SPA OB 202  
 003E :L MB 250  
 003F :SPB = OBFE  
 0040 :  
 0041 :L MB 249  
 0042 :I 1  
 0043 :T MB 249  
 0044 :L MB 239  
 0045 : <F  
 0046 :SPB = UEBT  
 0047 :  
 0048 LEER :L MB 244  
 0049 :I 1  
 004A :T MB 244  
 004B :L KB 4  
 004C : <F  
 004D :SPM = SCHL  
 004E :L KB 0  
 004F :T = FEWA  
 0050 :BEA  
 0051 :  
 0052 FEWA :L KB 16  
 0053 OBFE :T = FEWA  
 0054 :BE

SF-OB:  
 SENDEN EINES DATENBLOCKS  
 ABBRUCH BEI FEHLERWARNUNG

BLOCK-NR. = BLOCK-NR. + 1

ALLE BLÖCKE ÜBERTRAGEN ?

VERBINDUNGSZÄHLER  
 INKREMENTIEREN

ALLE DREI VERBINDUNGS-  
 STRECKEN BEARBEITET ?

REGULÄRES PROGRAMMENDE:  
 VKE = 0, FEWA = 0

PROGRAMMENDE BEI FEHLER:  
 VKE = 1, FEWA ENTHÄLT  
 FEHLER-/WARNUNGS-NUMMER

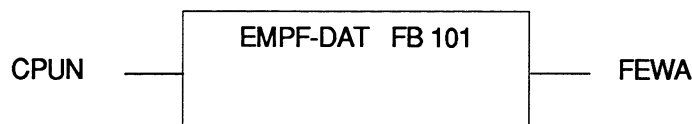
### 8.3.6 Empfangen von Datenwortbereichen (FB 101)

Vor Aufruf des FB 101 muß der Datenbaustein aufgeschlagen werden, der die Verbindungsliste enthält. Der Funktionsbaustein EMPF-DAT benötigt zum Auswerten der in der Verbindungsliste enthaltenen Informationen die Nummer der CPU, in der er aufgerufen wird.

Falls innerhalb des Funktionsbausteins die Funktion EMPFANGEN (OB 204) nicht korrekt abgearbeitet werden konnte, wird die jeweilige Fehler- oder Warnungs-Nummer im Ausgangsparameter FEWA übergeben und das VKE = 1 gesetzt. Zusätzlich enthält FEWA bei unzulässigem Eingangsparameter CPUN den Wert 16 (Bit  $2^4 = 1$ ).

Der Funktionsbaustein EMPF-DAT verwendet die Merkerbytes MB 242 bis MB 255 als Schmiermerker.

Parameter-Name	Bedeutung	Art	Typ
CPUN	Nummer der CPU, auf der der FB 101 aufgerufen wird. Zulässig sind die Nummern 1 bis 4.	D	KF
FEWA	Fehler/Warnung (siehe Funktion EMPFANGEN (OB 204))	A	BY



FB 101

LAE= 88

NETZWERK 1 0000

NAME:EMPF-DAT

BEZ :CPUN E/A/D/B/T/Z: D KM/KH/KY/KC/KF/KT/KZ/KG: KF  
BEZ :FEWA E/A/D/B/T/Z: A BI/BY/W/D: BY

000B :LW = CPUN

FEHLER FALLS:

000C :L KB 1

000D :< F

CPU-NR< 1

000E :SPB = FEWA

000F :LW = CPUN

0010 :L KB 4

0011 :>F

CPU-NR> 4

0012 :SPB = FEWA

0013 :

VERBINDUNGSZÄHLER

0014 :L KB 1

0015 :T MB 242

0016 :

0017 :L KB 16

0018 :T MW 244

ZEIGER AUF DIE TEILLISTE 2

0019 :

001A SUCH :L MW 244

TEILLISTE 2 SOLANGE DURCH-  
SUCHEN, BIS DER NÄCHSTE  
EINTRAG FÜR DIE EMPFANGS-  
CPU MIT DER NUMMER 'CPUN'  
GEFUNDEN IST.

001B :I 1

001C :T MW 244

001D :B MW 244

001E :L DL 0

001F :LW = CPUN

0020 :>< F

0021 :SPB = SUCH

0022 :

0023 :B MW 244

ANZAHL DER RESERVIERTEN  
SPEICHERBLÖCKE = 0 ?

0024 :L DR 0

0025 :T MB 243

0026 :L KB 0

0027 :!= F

0028 :SPB = LEER

0029 :

NUMMER DER SENDE-CPU  
AUS DEM ZEIGER AUF DIE  
TEILLISTE 2 BESTIMMEN

002A :L MW 244

002B :L KM 00000000 00001100

002D :UW

002E :SRW 2

002F :I 1

0030 :T MB 246

0031 :

SF-OB:  
EMPFANGSMÖGLICHKEIT  
TESTEN  
ABBRUCH BEI FEHLER

0032 :L KB 246

0033 :SPA OB 205

0034 :L MB 248

0035 :SPB = OBFE

0036 :

EMPFANGS-KAPAZITÄT = AN-  
ZAHL DER RESERVIERTEN  
SPEICHERBLÖCKE ?

0037 :L MB 249

0038 :L MB 243

0039 :>< F

003A :SPB = LEER

003B :

003C EMPF :L KB 246  
 003D :SPA OB 204  
 003E :L MB 248  
 003F :SPM = OBFE  
 0040 :  
 0041 :L MB 249  
 0042 :L KB 0  
 0043 : >< F  
 0044 :SPB = EMPF  
 0045 :  
 0046 LEER :L MB 242  
 0047 :I 1  
 0048 :T MB 242  
 0049 :L KB 4  
 004A : <F  
 004B :SPM = SUCH  
 004C :L KB 0  
 004D :T = FEWA  
 004E :BEA  
 004F :  
 0050 FEWA :L KB 16  
 0051 OBFE :T = FEWA  
 0052 :BE

SF-OB:  
 EMPFANGEN EINES DATEN-  
 BLOCKES  
 ABBRUCH BEI FEHLER ,  
 WARNUNG  
 BEI EMPFANGSKAPAZITÄT = 0  
 NÄCHSTE VERBINDUNGS-  
 STRECKE BEARBEITEN

VERBINDUNGSZÄHLER  
 INKREMENTIEREN

ALLE VERBINDUNGSSTRECKEN  
 BEARBEITET ?

REGULÄRES PROGRAMMENDE:  
 VKE = 0, FEWA = 0

PROGRAMMENDE BEI FEHLER:  
 VKE = 1, FEWA ENTHÄLT  
 FEHLER-/WARNUNGS-NUMMER

### 8.3.7 Anwendungsbeispiel (für AG S5-135U)

Zwischen drei CPUs sollen Daten ausgetauscht werden:

- **Von CPU 1 nach CPU 2:** Datenbaustein DB 3, DW 0 bis DW 127 ( = 4 Blöcke)
- **Von CPU 1 nach CPU 3:** Datenbaustein DX 4, DW 0 bis DW 63 ( = 2 Blöcke)
- **Von CPU 2 nach CPU 1 und CPU 3:** Datenbaustein DB 5, DW 0 bis DW 95 ( = 3 Blöcke)

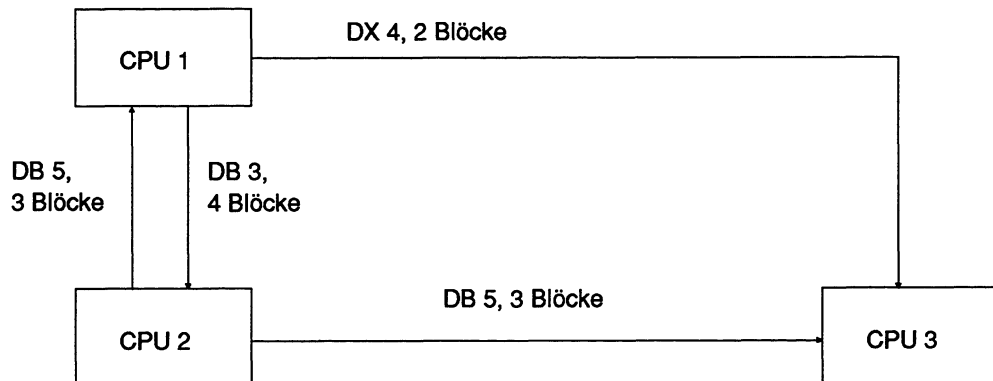


Bild 4 Datenaustausch zwischen 3 CPUs

Auf allen drei CPUs soll der Funktionsbaustein FB 0 die Schnittstelle zum zyklischen Anwenderprogramm bilden. Die CPU 1 soll bei NEUSTART die Funktion INITIALISIEREN (OB 200) aufrufen. Die Verbindungsliste soll im Datenbaustein DB 100 stehen.

Folgende Bausteine müssen in die einzelnen CPUs geladen werden:

Funktion	CPU 1	CPU 2	CPU 3
Anlauf-OB	OB 20	—	—
Anwenderprogramm	FB 0	FB 0	FB 0
FB: SEND-DAT	FB 100	FB 100	FB 100
FB: EMPF-DAT	FB 101	FB 101	FB 101
Verbindungsliste	DB 100	DB 100	DB 100
Eingangs-DB	DB 5	DB 3	DB 5; DX 4
Ausgangs-DB	DB 3; DX 4	DB 5	—

Zunächst wird die Verbindungsliste, deren Aufbau unter "Datenstruktur" beschrieben ist, erstellt und in den DB 100 eingetragen:

DB100

LAE= 37    ABS  
BLATT   1

—— Teilliste 1 ——

0 : KC    = S1  
1 : KY    = 001,003;  
2 : KY    = 002,004;  
3 : KY    = 000,000;  
4 : KC    = S2  
5 : KY    = 001,005;  
6 : KY    = 001,005;  
7 : KY    = 000,000;  
8 : KC    = S3  
9 : KY    = 000,000;  
10 : KY    = 000,000;  
11 : KY    = 000,000;  
12 : KC    = S4  
13 : KY    = 000,000;  
14 : KY    = 000,000;  
15 : KY    = 000,000;

Von CPU 1 zu ..  
.. CPU 2 den DB 3 senden.  
.. CPU 3 den DX 4 senden.

Von CPU 2 zu ..  
.. CPU 1 den DB 5 senden.  
.. CPU 3 den DB 5 senden.

—— Teilliste 2 ——

16 : KC    = S1  
17 : KY    = 002,004;  
18 : KY    = 003,002;  
19 : KY    = 004,000;  
20 : KC    = S2  
21 : KY    = 001,003;  
22 : KY    = 003,003;  
23 : KY    = 004,000;  
24 : KC    = S3  
25 : KY    = 001,000;  
26 : KY    = 002,000;  
27 : KY    = 004,000;  
28 : KC    = S4  
29 : KY    = 001,000;  
30 : KY    = 002,000;  
31 : KY    = 003,000;

Von CPU 1 zu ..  
.. CPU 2 vier Datenblöcke senden.  
.. CPU 3 zwei Datenblöcke senden.

Von CPU 2 zu ..  
.. CPU 1 drei Datenblöcke senden  
.. CPU 3 drei Datenblöcke senden.

Die Datenwörter DW 16 bis DW 31 enthalten die für die Funktion manuelles INITIALISIEREN (OB 200) notwendige Zuordnungsliste. Der OB 200 wird vom nachfolgend abgedruckten OB 20 der CPU 1 im Anlauf aufgerufen.

OB 20  
NETZWERK 1

LAE= 23    ABS

0000	:L	KB 2	MANUELLES INITIALISIEREN DER
0001	:T	MB 246	KACHELN
0002	:		
0003	:L	KY 1,100	IM DB 100 IST DIE ZUORDNUNGS-
0005	:T	MW 248	LISTE AB DEM DATENWORT 16
0006	:L	KF+ 16	EINGETRAGEN
0008	:T	MW 250	
0009	:		
000A	:L	KB 246	SF-OB:
000B	:SPA	OB 200	INITIALISIEREN
000C	:		
000D	:UN	M 252.5	BAUSTEINENDE, WENN KEIN
000E	:BEB		INITIALISIERUNGSKONFLIKT
000F	:		

Hier wird die Fehlerbehandlung im Falle eines Initialisierungskonfliktes eingefügt (z.B. Stop, Meldung auf Drucker ausgeben, oder ...)

0010        :BE

Auf jeder CPU wird das Anwenderprogramm um den Aufruf der Funktionsbausteine EMPF-DAT und SEND-DAT erweitert. Der abgedruckte Funktionsbaustein FB 0 ist für die CPU 1 bestimmt. Für den Ablauf auf den anderen CPUs muß lediglich der Eingangsparameter CPUN (CPU-Nummer) angepaßt werden.

FB0  
NETZWERK 1  
NAME:PROG-1

LAE= 31    ABS

0005        :A    DB100  
0006        :SPA FB101

VERBINDUNGSLISTE    DB 100  
EMPFANGEN DER EINGANGS-  
DATENBAUSTEINE

0007 NAME :EMPF-DAT  
0008 CPUN :    KF+ 1  
0009 FEWA :    MB0  
000A        :SPB = FEWA  
000B        :  
000C        :

ABBRUCH BEI FEHLERWARNUNG

Hier wird das zyklische Anwenderprogramm eingefügt, das Daten aus den Eingangsdatenbausteinen liest und Daten in die Ausgangsdatenbausteine einträgt.

000D        :  
000E        :  
000F        :  
0010        :A    DB100  
0011        :SPA FB100

VERBINDUNGSLISTE    DB 100  
SENDEN DER AUSGANGS-  
DATENBAUSTEINE

0012 NAME :SEND-DAT  
0013 CPUN :    KF+ 1  
0014 FEWA :    MB0  
0015        :SPB = FEWA  
0016        :BEA  
0017        :  
0018 FEWA :  
0019        :BE

ABBRUCH BEI FEHLERWARNUNG

NACH FEHLERWARNUNG  
FEHLERBEHANDLUNG DURCH-  
FÜHREN

Hier wird die Fehlerbehandlung eingefügt.  
(z.B. Stopp, Fehlermeldung auf Drucker oder Monitor ausgeben, oder ...)

#### BEACHTEN SIE:



Dieses Beispiel (Koppelmerkererweiterung mit Hilfe der Funktionsbausteine SEND-DAT und EMPF-DAT) kann nur dann korrekt ablaufen, wenn in keiner der CPUs zusätzlich die Sonderfunktions-Organisationsbausteine zur Mehrprozessorkommunikation OB 202 bis OB 205 aufgerufen werden.