SICOMP/System 6·000

ZE 03
Zentraleinheit
Minicomputer SICOMP M®

Beschreibung

Ausgabe März 1987

Der Text dieser Druckschrift wurde mit dem Siemens-Bürosystem
EMS 5600 Office erfaßt und gestaltet. Die Erstellung der Grafiken erfolgte
mit dem Siemens-Bürosystem 5800 Office

Inhalt

1 Anwendungsbereich

Die Zentraleinheit ZE 03 ist ein hochintegrierter, leistungsfähiger, freiprogrammierbarer 16-bit-Digitalrechner der Siemens Modellreihe SICOMP R M. Sie weist die folgenden herausragenden Eigenschaften auf:

- umfangreiche Matrix-Befehlsliste
- reaktionsschnelle Unterbrechungsstruktur und
- leistungsfähiges Ein-/Ausgabe-System.

Die Zentraleinheit steht im Mittelpunkt des Hardware-Teils eines Rechnersystems. Ein breites Spektrum peripherer Einheiten erweitert die Zentraleinheit zu einem leistungsfähigen Minicomputersystem (Bild 1.1).

Einsatzgebiete der ZE 03 sind vor allem Automatisierungsaufgaben des mittleren bis größeren Leistungsbereichs in der Büroorganisation und in technischen Prozessen. Darüber hinaus können komplexe Automatisierungsaufgaben im Rechnerverbund mit anderen Rechnern der Modellreihe SICOMP, der Siemens Systeme 300/6000 oder mit Rechnern fremder Systeme gelöst werden. Die ZE 03 ist Bestandteil des Systems SICOMP 70.

Eigenschaften der ZE 03:

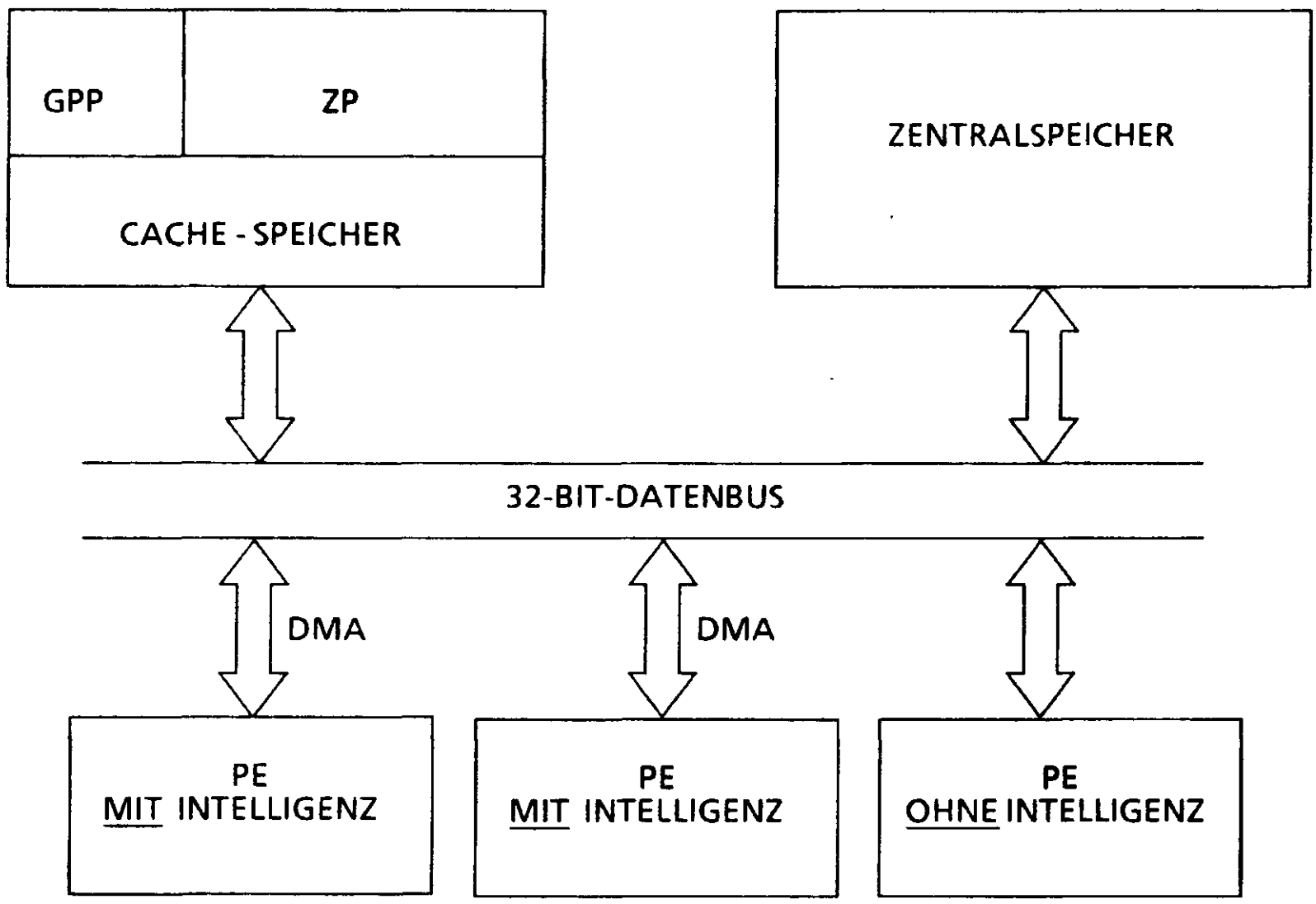
- 16-bit-Registermaschine mit den Vorteilen der einfachen Register-Register-Operationen, dem automatischen Zwischenspeichern von Verknüpfungsergebnissen und dem leistungsfähigen Adressierverfahren über Register
- Mikroprogrammierter Zentralprozessor (ZP), aufgebaut mit schnellen Bit-slice-Prozessoren (4-bit-slices, S-TTL)
- Matrix-Befehlsliste mit insgesamt 338 Befehlen, die zum Teil in einem separaten Gleitpunktprozessor implementiert sind
- Schnelles hardware-gesteuertes Unterbrechungssystem für die Echtzeit-Verarbeitung von internen und externen Ereignissen
- Leistungsfähiges Ein-/Ausgabe-System durch DMA-Fähigkeit an jedem EA-Steckplatz und einer möglichen Transferbreite von 32 bit
- Durch Verwendung von 256 K*bit-Speicherchips hochintegrierte Halbleiter- speichermodule mit einer maximalen Kapazität von 1,2 oder 4 M*byte je Modul.Fehlerkorrektur ist Standard
- Kompakter Aufbau der Zentraleinheit einschließlich Zentralspeicher (ZSP) und EA-Anschlußstellen in einem dreizeiligen Baugruppenträger, ausbaubar mit einem Erweiterungsbaugruppenträger (Aufbau teilweise in ES 902)

Die Leistung der Zentraleinheit kann durch optionelle Funktionseinheiten gesteigert werden:

- Schneller Cache-Speicher: Die Befehlsbearbeitung wird durch wesentlich geringere Zugriffs- und Zykluszeiten beschleunigt. Der Zentralprozessor greift wenn möglich auf den Cache-Speicher zu und nicht auf den Hauptspeicher. Dadurch wird der Systembus entlastet und der Zentralprozessor und peri
  phere Einheiten können parallel arbeiten; die Systemleistung steigt.

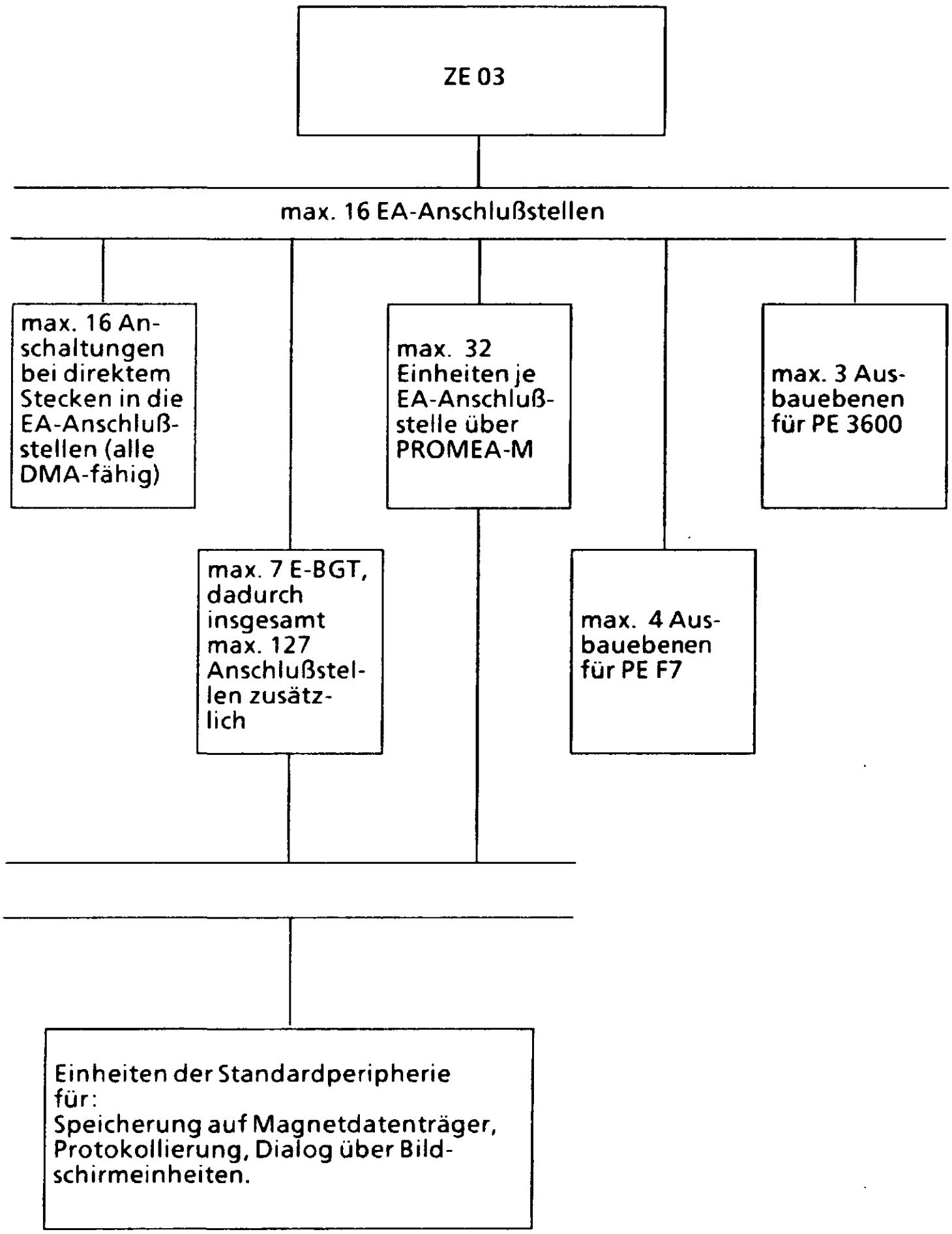
K* = 1024
M* = 1048576

- Gleitpunktprozessor: Er übernimmt im wesentlichen die Abarbeitung der Gleitpunktbefehle und doppelt langen Festpunktbefehle. Es stehen zwei Varianten zur Verfügung. Ein ein Standard-Gleitpunktprozessor und ein "High-Performance-Gleitpunktprozessor".



GPP = Gleitpunktprozessor
ZP = Zentralprozessor
PE = Periphere Einheit
DMA = direct memory access
(direkter Speicherzugriff)

Bild 1.1 Struktur der ZE 03



E-BGT = Erweiterungsbaugruppenträger
PE = Prozeßeinheit

Bild 1.2 EA-Anschlußmöglichkeiten im System SICOMP 70

2 Aufbau

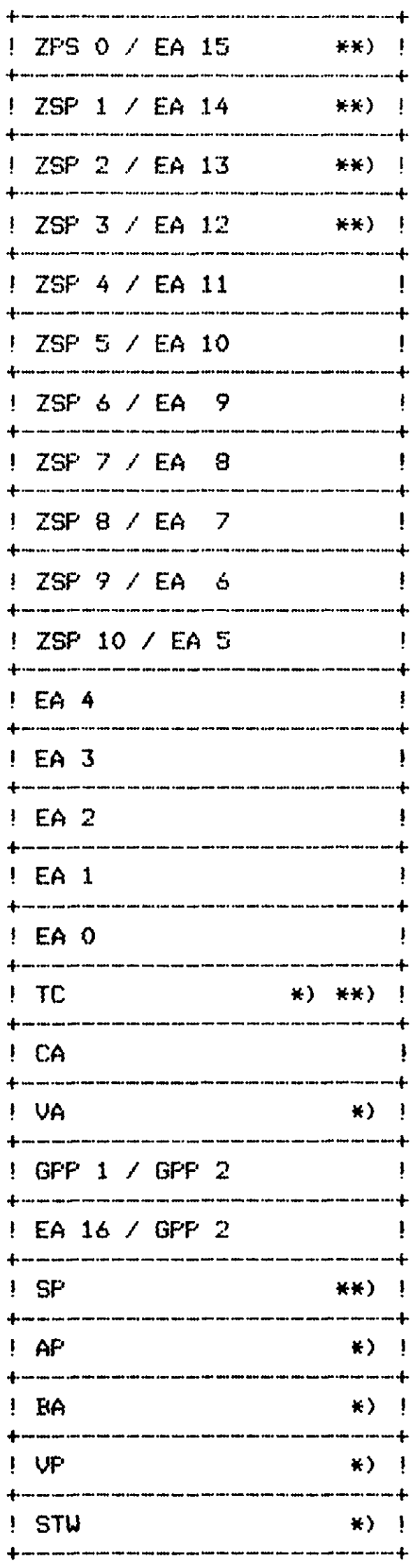
Unter dem Begriff Zentraleinheit werden alle technischen Einrichtungen einer Datenverarbeitungsanlage zusammengefaßt, die zum Verarbeiten von Informationen notwendig sind. Es handelt sich um folgende Funktionseinheiten:

- zentrale Verarbeitungseinheit mit
  Zentralprozessor,
  Unterbrechungssystem,
  Ein-/Ausgabe-System;
- Zentralspeicher mit
  Speichersteuerung,
  Speichermodulen;
- virtuelle Konsole.

2.1 Baugruppenträger der Zentraleinheit

Die Zentraleinheit ZE 03 ist in einem dreizeiligen Baugruppenträger mit den Abmessungen (B x H x T) 444 mm x 250 mm x 488 mm untergebracht.
Die Elektronik befindet sich auf Flachbaugruppen (4-Lagen-Multi-Layer-Leiterplatten) mit den Abmessungen 366,7 mm x 160 mm (3-fach Europaformat).

Der Funktionsumfang der zentralen Verarbeitungseinheit ist auf sechs Flachbaugruppen implementiert.
Diese Flachbaugruppen stecken zusammen mit jenen des Zentralspeichers, dem Cache-Speicher (schneller Pufferspeicher) und dem Gleitpunktprozessor in den Führungsleisten des dreizeiligen Baugruppenträgers mit 26 Einbauplätzen. Davon sind fünf Steckplätze ausschließlich für EA-Verkehr und elf Steckplätze wahlweise für EA-Verkehr oder als Speichersteckplätze verwendbar. Von diesen genannten Steckplätzen haben die EA-Steckplätze und sieben der elf wahlweise verwendbaren Steckplätze eine Breite eines Standard-Einbauplatzes (SEP; 15,24 mm), die restlichen sind 1 1/3 SEP breit.
Die Flachbaugruppen sind mit drei 96-poligen Messerleisten (ES 902, Reihe 1, Raster 2,54 mm) als Basisstecker bestückt, die in die Federleisten der Verdrahtungsplatte (4-Lagen-Multi-Layer) des Baugruppenträgers eingreifen. Diese Federleisten sind in die Verdrahtungsplatte eingepreßt.
Auf der Frontseite der Flachbaugruppen (den Basissteckern gegenüberliegende Seite) befinden sich je nach Funktion der Baugruppe ein oder mehrere Subminiatur D-Stecker (Frontstecker) zur Weitergabe von Informationen an Flachbaugruppen in anderen Baugruppenträgern bzw. an periphere Geräte.
Die Flachbaugruppen werden mit Steck- und Ziehhebeln, die sich beim Einschnappen selbst verriegeln, in der Führung des Baugruppenträgers bewegt.



Abkürzungen:

STW...Steuerwerk
VP....Verarbeitungsprozessor
BA....Befehlsaufbereitung
AP....Adreßprozessor
VA....Virtuelle Adressierung
TC....timing and control
(Speichersteuerung)
SP....Serviceprozessor
GPP1..Gleitpunktprozessor
Standardvariante
GPP2..Gleitpunktprozessor
Hochleistungsvariante
CA....Cache-Speicher
EA....Ein-/Ausgabe-Schnittstelle
ZSP...Zentralspeicher-Schnittstelle

*) Flachbaugruppe des Zentralprozessors

**) Steckplatzbreite 20,3 mm (1 1/3 SEP)
restliche Steckplätze 15,2 mm (1 SEP)

Bild 2.1 Rahmen der Zentraleinheit

2.2 Stromversorgung

Die Stromversorgung hat einen hohen Wirkungsgrad bei kleinem Volumen. Sie ist modular erweiterbar und in einem Baugruppenträger mit den Abmessungen (B x H x T) 444 mm x 250 mm x 488 mm untergebracht. Voll bestückt versorgt sie den maximalen Ausbau des Grundcontainers SICOMP 70:

Grundausbau	Modular erweiterungsfähig mit
Modul:	Modul:
2 x 5V/65A mit Logik	5V/65A
24V/20A mit Nachregler V/20A	Nachregler 12V/2A
- 12V/20A	24V/20A
Puffermodul (für 4M*byte Zentralspeicher)	Puffermodul (wird ab einem Zentralspeicher-Ausbau von mehr als 4 M*byte benötigt)

Der Wirkungsgrad der einzelnen Stromversorgungsmodule, die jeweils primär getaktet arbeiten, liegt bei rund 75 %.

Die Stromversorgungen werden einphasig an das 220-V-Wechselspannungsnetz angeschlossen.
Alle Spannungen sind bezugspotentialfrei. Sie erhalten ihr Bezugspotential mit Anschluß des gemeinsamen Pols an 0V. Die Verteilung der Betriebsspannungen wird über Kabel vorgenommen. Module mit gleicher Ausgangsspannung werden in Parallelschaltung betrieben.
Alle Ausgangsspannungen der Stromversorgungsmodule sind mit einem über-strom- und überspannungsschutz ausgerüstet. Die Stromversorgungsmodule sind leerlauffähig. Im Fehlerfall wird ein Schutzschalter ausgelöst. Nach Beheben des Fehlers kann die Stromversorgung mit Einlegen des Schutzschalters wieder eingeschaltet werden.

Die Stromversorgung erzeugt ferner die für die Zentraleinheit wichtigen Signale bei Spannungsausfall und Spannungswiederkehr. Netzeinbrüche bis 8 ms (bei max. Belastung der Stromversorgung) haben keinen Einfluß auf den ordnungsgemäßen Betrieb der Zentraleinheit. Bei längeren Netzeinbrüchen wird die Zentraleinheit über das Signal PF (power failure; Netzspannungsausfall) vom Spannungsausfall informiert. Zum Retten von Registern stehen dann noch ca. 2 ms zur Verfügung. Um ein sicheres, definiertes Beenden eines Peripherspeichertransfers zu ermöglichen, wird auch ein lastunabhängiges Netzspannungsausfallsignal (APF; arising power failure) zur Verfügung gestellt; daraus wird das Signal HLD (hold) abgeleitet. Bei Netzspannungswiederkehr geht die Stromversorgung ohne Operatoreingriff selbstständig wieder in Betrieb.

Zur Zentralspeicher-Datenerhaltung bei Netzspannungsausfall (größer als 10 ms) dient das Puffermodul, das eine Datenerhaltung im Zentralspeicher für mindestens 20 Minuten sicherstellt. Es besteht im wesentlichen aus einem primärgetakteten Laderegler und einem Akku. Der Akku ist ein wartungsfreier, gasungsarmer 24-V-Bleiakkumulator.
Der Überwachungsteil der Stromversorgung liefert bei einer Klemmenspannung des Akkus kleiner als 19 V das Signal BF (battery failure, Batterieausfall). Daraus bestimmt die Zentraleinheit die Wiederanlaufstrategie bei Spannungswiederkehr.

2.3 Container, Überwachung

Im System SICOMP 70 ist die ZE 03 im Container eingebaut. Der Einschub mit der ZE 03 ist, wie die anderen Einschübe des Systems (Peripherspeicherlaufwerke, Stromversorgung) auf Teleskopschienen ausfahrbar, um die Zugänglichkeit bei Wartung/Service zu gewährleisten.

Die Wärme wird bei allen Einschüben durch ein geräuscharmes Druckbelüftungssystem abgeführt, welches Bestandteil des Containers ist. Bei Ausfall des Belüftungssystems sorgt eine Überwachungslogik dafür, daß keine unzulässige Erwärmung und damit Beschädigung der Systemkomponenten auftritt.
Die Überwachungslogik ermöglicht außerdem das gemeinsame Ein/Ausschalten von Mehrcontainersystemen über den Schlüsselschalter des Zentraleinheitscontainers (Grundcontainer).

Die Abmessungen des Grundcontainers betragen (B x H x T) 555 mm x 1150 mm x 780 mm.

2.4 Busanpassung und Erweiterungsbaugruppenträger

In großen Systemausbauten, bei denen die EA-Steckplätze des Zentraleinheitsbaugruppenträgers nicht ausreichen, ermöglichen Busanpassungsbaugruppen eine Steckplatzvervielfachung durch Auslagerung des Systembusses in Erweiterungsrahmen.
Dabei erhält adreßmäßig der Zentraleinheitsbaugruppenträger die Nr. 0, die Erweiterungsbaugruppenträger können von 1 - 7 numeriert werden.

Der Anschluß eines Erweiterungsbaugruppenträgers geschieht durch Stecken einer Busanpassungsbaugruppe EA 03 - A in den Grundrahmen und einer weiteren, die mit der ersten verbunden ist, in den Erweiterungsrahmen (Bild 2.2).

Der Erweiterungsrahmen besitzt insgesamt 20 EA-Steckplätze und eine eigene Stromversorgung. Er ist zum Einbau in den SICOMP-Erweiterungscontainer vorgesehen.
Die Auslagerung des Systembusses mit Hilfe von Busanpassungen bietet dem Anwender alle Funktionen der EA-Schnittstelle über eine Entfernung von max. 30 m zwischen den Baugruppenträgern. Dies ermöglicht auch eine dezentrale Aufstellung des Erweiterungscontainers.

Erweiterungsbaugruppenträger können sowohl mehrfach an den Grundrahmen angeschlossen als auch (durch Stecken weiterer Busanpassungen) untereinander gekettet werden (s. Bild 2.3). Alle auf diese Weise möglichen Konfigurationen sind vollkommen transparent für die Systemsoftware und erhalten die Funktionalität der Schnittstellen.

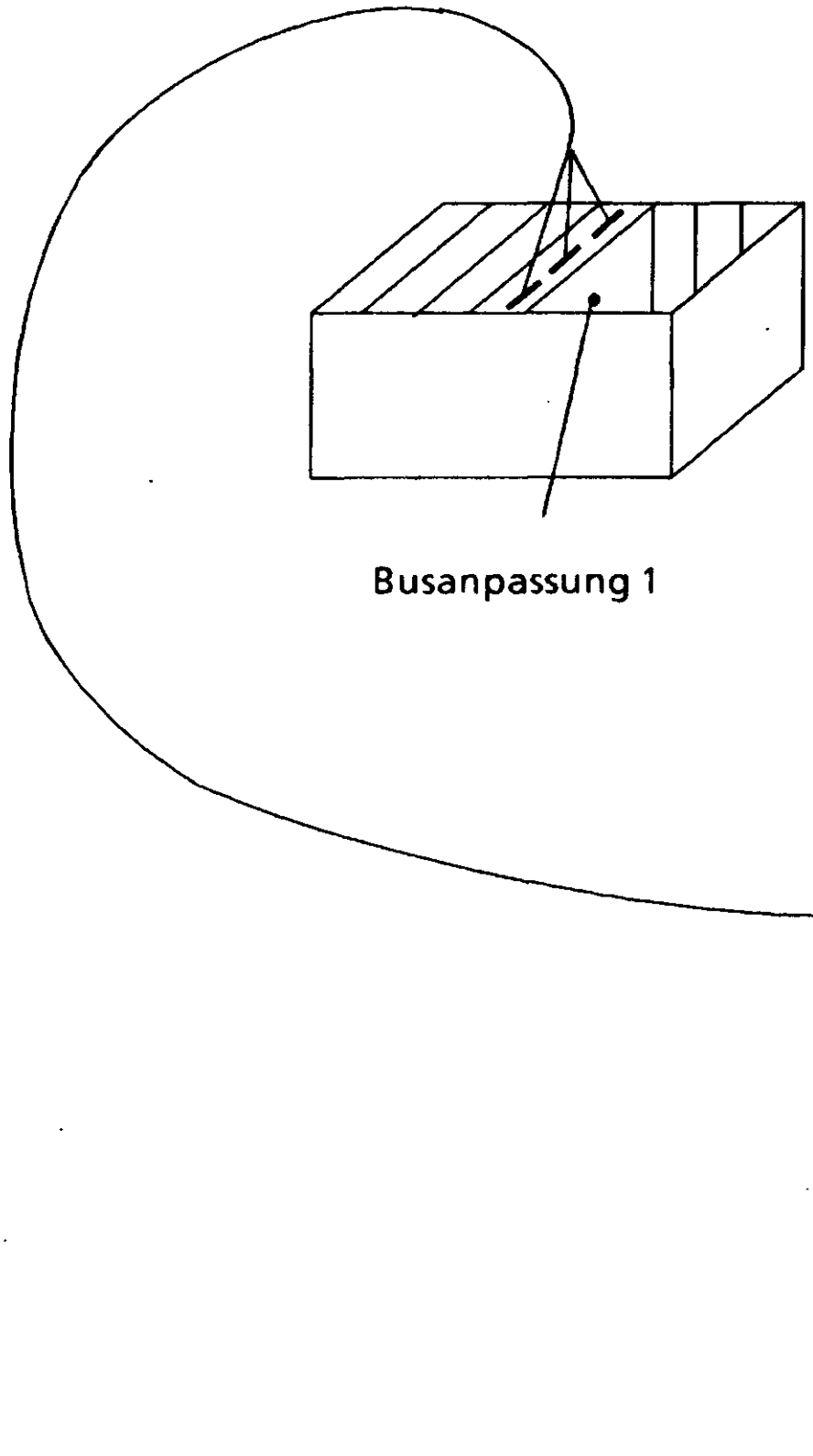


Bild 2.2 Anschluß eines Erweiterungsbaugruppenträgers an den Zentraleinheitsbaugruppenträger über Busanpassung

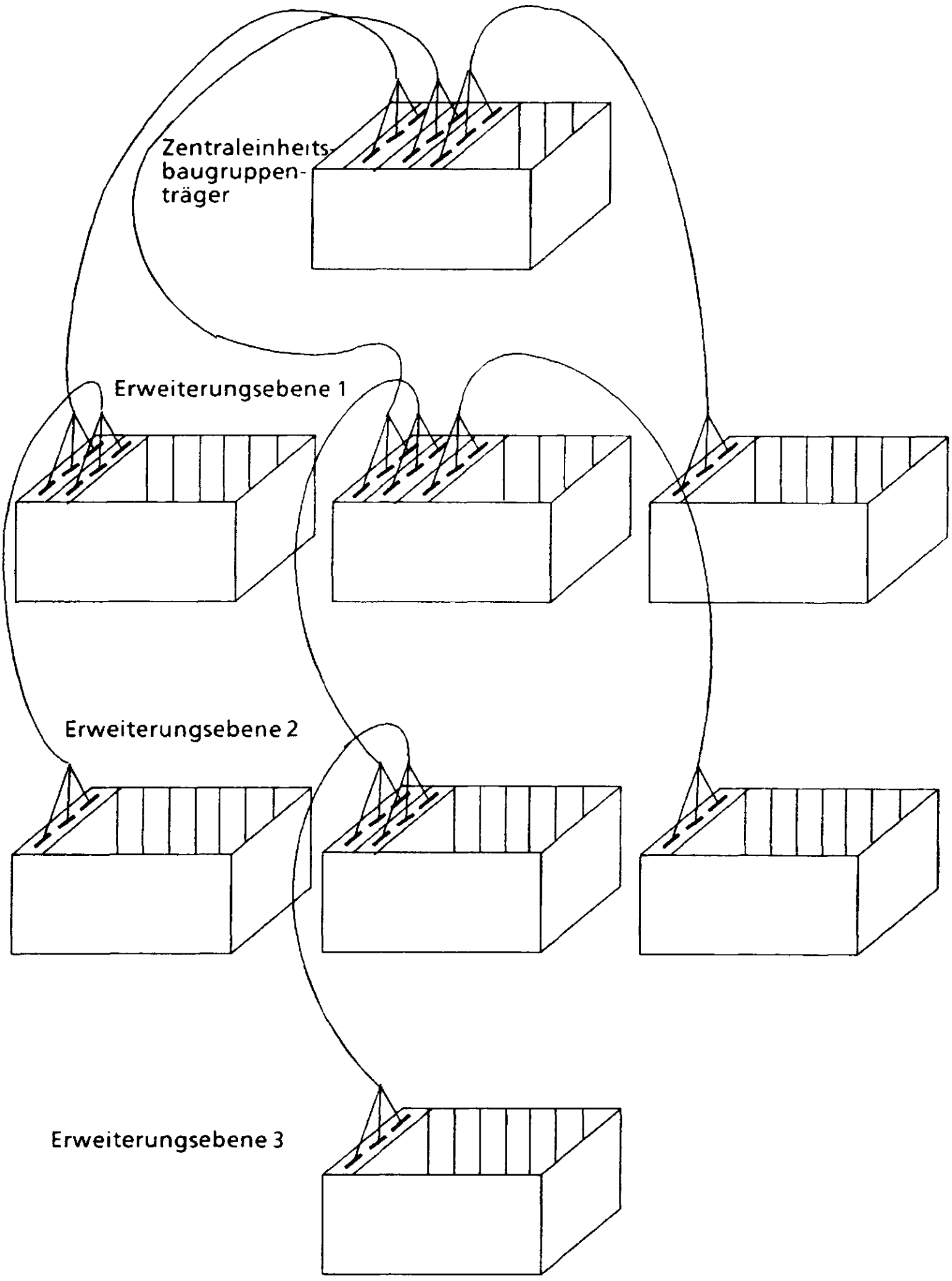


Bild 2.3 Reihung und Kettung von Erweiterungsbaugruppenträgern

Erweiterungsbaugruppenträger besitzen eine eigene Stromversorgung. Jede Busanpassung meldet der System-Software die Verfügbarkeit des zugehörigen Erweiterungsrahmens durch einen spezifischen Interrupt bei

- Einschalten bzw. Spannungswiederkehr,
- Spannungsausfall.

Der Spannungsausfalls-Interrupt wird von den Signalen HLD/TER bzw. RSP der Stromversorgung ausgelöst und aktiviert ein hochpriores Reaktionsprogramm in der Zentraleinheit. Da die Busanpassung den Interruptvektor aus der Adresse des Baugruppenträgers ableitet, können durch Untersuchen der Gerätelisten in der System-Software die an den Rahmen angeschlossenen, vom Spannungsausfall betroffenen Geräte ermittelt und nach Spannungswiederkehr neu parametriert werden.
Diese Vorkehrungen gewährleisten, daß ein partieller Spannungsausfall, wie er z. B. bei dezentraler Aufstellung der Erweiterungscontainer auftreten kann, nicht zur Blockierung des Systems führt.

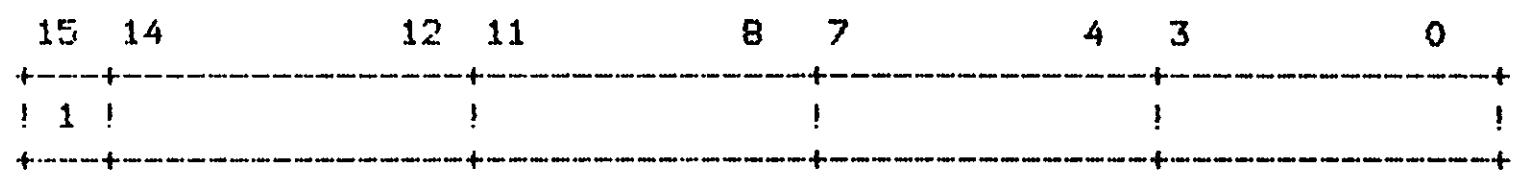


Bild 2.4 Aufteilung des 16-bit-EA-Adreßraums für Standardperipherie

Sollen in einen Erweiterungsrahmen mehr als die mit Bit 11 - 8 ansprechbaren 16 Geräteanschaltungen gesteckt werden, so läßt sich die Adreßrangierung an der Busanpassung umschalten. Bei halbierter Anzahl der adressierbaren Baugruppenträger (Bit 14 bis 12) kann dann als Zusatzinformation die Steckplatzposition (linke oder rechte Rahmenhälfte) angegeben werden.
Zur Identifikation des Rahmens werden Bit 13 und 14 verwendet (0 = Grund-BGT; 1,2,3 = Erweiterungs-BGT), Bit 12 bezeichnet linke oder rechte Hälfte (0 = links; 1 = rechts). Die Kombination 001 darf nicht verwendet werden.
Bei entsprechender Aufteilung der zur Verfügung stehenden Geräteadressen ist es außerdem möglich, mehreren Erweiterungsbaugruppenträgern die gleiche Rahmennummer zuzuteilen (Einstellung über Schalter am Rahmen).
Die Busanpassung besitzt einen Steuereingang, über den sie passiv geschaltet werden kann. Dadurch besteht die Möglichkeit, den jeweiligen Erweiterungsrahmen bzw. die angeschlossenen Geräteanschaltungen zwischen verschiedenen Zentraleinheiten umzuschalten (Bild 2.5). Eine neben die Busanpassung steckbare Bus-Umschalterbaugruppe (je nach Anwendererfordernissen auszulegen) kann die Koordinierung vornehmen und durch Erzeugung der Bussignale $\overline{HLD}$, $\overline{TER}$ und $\overline{RSP}$ einen definierten Abschluß laufender Gerätetransfers vor dem Umschalten veranlassen.

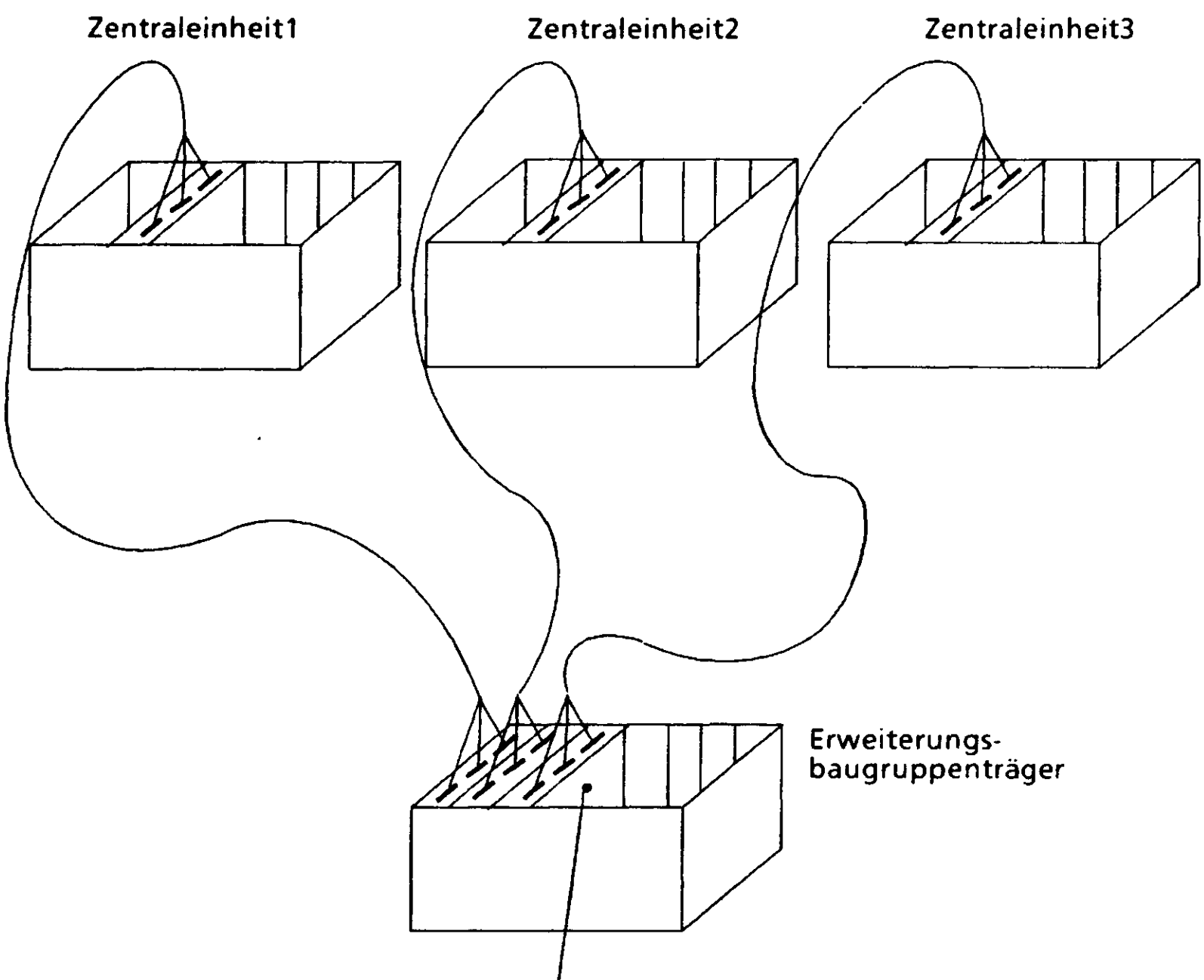


Bild 2.5 Umschaltungsmöglichkeiten des Erweiterungsbaugruppenträgers

3 Arbeitsweise

Die Beschreibung der Arbeitsweise ist nach funktionellen Komponenten der Zentraleinheit aufgeteilt. Diese sind:

- zentrale Verarbeitungseinheit (ZVE)
- Zentralspeicher (ZSP)
- EA-System
- virtuelle Konsole (VK).

In der zentralen Verarbeitungseinheit werden aufgrund der im Zentralspeicher abgelegten Befehle die vorgegebenen Operanden verknüpft. Außerdem reagiert sie auf interne und externe Unterbrechungsanforderungen über die EA-Schnittstellen können periphere Geräte angeschlossen werden, die über direkten Speicherzugriff DMA (direct memory access) oder mit Hilfe des Zentralprozessors Daten zum und vom Zentralspeicher transferieren können. Die virtuelle Konsole ist ein komfortables, leistungsstarkes Test- und Wartungshilfsmittel für das Testen von hardware-naher Software.

3.1 Zentrale Verarbeitungseinheit

Die zentrale Verarbeitungseinheit (ZVE) besteht aus:

- Zentralprozessor (ZP)
- Gleitpunktprozessor (GPP), optionell
- Cache-Speicher (CA), optionell

Der Zentralprozessor ist in einer Zweiprozessorstruktur ausgeführt, bestehend aus einem Verarbeitungsprozessor (VP) und einem Adreßprozessor (AP). Ersterer führt im wesentlichen die arithmetische Befehlsbearbeitung sowie Operandenadressierung durch, letzterer die Befehlsadressierung und organisatorische Routinen. Die Leistungsfähigkeit der zentralen Verarbeitungseinheit kann durch Verwendung eines Gleitpunktprozessors erhöht werden, in dem die Arithmetik der 32-bit- und 64-bit-Gleitpunktzahlen und die Multiplikation und Division der 32-bit-Festpunktzahlen hardwaremäßig implementiert ist.
Eine weitere Funktionseinheit, die zur wesentlichen Leistungssteigerung beiträgt, ist der schnelle Pufferspeicher (Cache-Speicher). Die Befehlsausführungszeiten verkürzen sich erheblich bei Zugriff des Zentralprozessor auf den Cache-Speicher anstelle des Zentralspeichers, da jener wesentlich kürzere Zugriffszeiten aufweist. Außerdem erspart der Cache-Speicher Zentralspeicher-Zugriffe, entlastet damit den Systembus und erhöht die Parallelarbeit, indem der Zentralprozessor die im Cache-Speicher hinterlegten Befehle ausführt und parallel dazu periphere Geräte auf den Zentralspeicher zugreifen können.

3.1.1 Zentralprozessor

Der Kern des Zentralprozessors wird von dem Adreßprozessor und dem Verarbeitungsprozessor gebildet. Die 16-bit-breiten Befehle und Daten gelangen zunächst auf die Befehlsaufbereitung (BA).
Auf dieser ist ein zweifacher Befehls-look-ahead implementiert. Dieser besteht im Kern aus drei Befehlsregistern (BFR1, BFR2, BFR3), welche

- zum Einlesen der Befehle aus dem Zentralspeicher bzw. Cache-Speicher (BFR 3),
- zum Dekodieren der Befehle (BFR 2) und
- zum Festhalten des Befehlscodes während der Befehlsabarbeitung durch das Mikroprogramm dienen (BFR 1).

Somit können während der Ausführung eines Befehls bereits der nächste und übernächste Befehl teilweise parallel mitbearbeitet werden. Durch diese parallele Abarbeitung aufeinanderfolgender Befehle können in der Bearbeitungszeit die Zeitanteile für Lesen und Dekodieren des Befehls eingespart werden. Die Befehlsaufbereitung steuert diese Vorgänge und stellt dem Verarbeitungsprozessor und Adreßprozessor die befehlsspezifische Mikroprogramm-Einstiegsadresse zur Verfügung (Dekodieren); auftretende Unterbrechungsereignisse werden ebenfalls in eine entsprechende Mikroprogramm-Adresse umgesetzt.

Die eigentliche Abarbeitung der Befehle erfolgt durch das Mikroprogramm von Verarbeitungsprozessor und Adreßprozessor. Die Mikroprogramme haben einen Umfang von 1024 x 90 bit (Verarbeitungsprozessor) bzw. 1024 x 110 bit (Adreßprozessor).

Durch Aufteilung des Prozessors in zwei Teilprozessoren mit speziellen Aufgaben kann eine effektivere Befehlsbearbeitung erreicht werden. Neben der Hauptaufgabe, der Befehlsadressierung und organisatorischen Routinen, übernimmt der Adreßprozessor auch Aufgaben des Verarbeitungsprozessors wie befehlsspezifische Kontrollen des Datenbereichs oder arithmetische Unterstützung bei komplexen Befehlen. Dem Verarbeitungsprozessor ist als Hauptaufgabe die arithmetische Befehlsbearbeitung sowie Operandenadressierung zugeordnet. Bei organisatorischen Routinen ist es auch Aufgabe des Verarbeitungsprozessors, den Adreßprozessor zu unterstützen. Ver- und Entsorgung des optionellen Gleitpunktprozessors sind ebenfalls Aufgaben des Verarbeitungsprozessors.

Die Zusammenarbeit zwischen den gleichwertigen Prozessoren (Verarbeitungs-, Adreß- und Befehlserweiterungsprozessor,im besonderen Gleitpunktprozessor) wird durch eine spezielle Kommunikation untereinander gewährleistet. Damit der eine Prozessor (z. B. Verarbeitungsprozessor) auch auf Ereignisse des anderen Prozessors (z. B. Adreßprozessor) reagieren kann, muß in diesem Fall der Adreßprozessor eine Kommunikationsanforderung stellen. Gleichzeitig liefert der Kommunikationsanforderungssteller (in diesem Fall Adreßprozessor) über einen internen Bus eine Mikroprogramm-Anweisung für den Verarbeitungsprozessor. Die Befehlsaufbereitung setzt bei angenommener Kommunikation diese Anwendung in eine Mikroprogramm-Verzweigungsadresse um.

Da manche Mikroprogramm-Routinen mehrfach vorkommen, ist es sinnvoll, diese als Unterprogramme ablaufen zu lassen. Dadurch kann es vorkommen,

daß aus diesen Unterprogrammen heraus Kommunikationsanforderungen gestellt werden, welche im konkreten Fall nicht nötig wären. Dies würde in manchen Fällen zu Fehlläufen führen; in diesen Fällen hat der jeweilige "Kommunikations-Empfänger" die Möglichkeit, die Anforderung zu verweigern.

Für den Anwender bietet der Zentralprozessor die Verwendung von 16 Hardware-Registern, den sogenannten Standardregistern, die er bei seinen Rechenoperationen ansprechen kann. Sie sind als Zwischenspeicher für Daten und Adressen verwendbar und können beim Ausführen der Befehle inkrementiert und dekrementiert werden. Dadurch lassen sich leistungsfähige Adressenrechnungen und Datenfeldbearbeitungen ohne zusätzlichen Befehlsaufwand ausführen.

Jedes Programm verfügt über einen unabhängigen Standardregistersatz, da der aktuelle Registerinhalt bei Programmunterbrechungen in ein programmspezifisches Depot im Zentralspeicher (Parametertafel) gerettet und - wenn nötig - auch von dort geladen wird.

Neben den 16 Hardware-Standardregistern verfügt die zentrale Verarbeitungseinheit über weitere 6 Spezialregister.

Die Spezialregister sind adressierbare Hardware-Register. Sie enthalten Steuerinformationen, die die Ausführung der Befehle beeinflussen, geben Auskunft über den Zustand des Zentralprozessors und beeinflussen den Zustandswechsel, d. h. den Wechsel zwischen den einzelnen Prioritätsebenen.

Der Inhalt der Spezialregister kann hardware-gesteuert verändert und, da den Programmen die Einrichtungen der zentralen Verarbeitungseinheit zugänglich sein müssen, mit Spezialregisterbefehlen gelesen oder eingeschrieben werden.

Ein Teil der Spezialregister ist programmspezifisch; diese werden bei einem Programmwechsel durch die Rett-Laderoutine in die Parametertafel geschrieben und bei Bedarf von dort gelesen (s. Kapitel 3.1.6).

Es gibt folgende Spezialregister:

a) Programmzustandsregister (PZR)

Das Programmzustandsregister enthält Informationen zum Steuern des Befehls- und Programmablaufs. Der Inhalt des Programmzustandsregisters ist als Programmzustandswort (PZW) in der Parametertafel (PT) enthalten und wird bei jedem Zustandswechsel und bei jedem Programmwechsel in der Parametertafel des gerade aktuellen Programms hinterlegt. Anschließend wird das Programmzustandsregister des neuen Programms geladen. Während eines neuen Programmablaufs kann der Inhalt des Programmzustandsworts über Spezialregisterbefehle gelesen und verändert werden. Über das Spezialregister 0 wird das gesamte Programmzustandswort, über das Spezialregister 1 nur das linke Byte angesprochen. Zum Vermeiden von Programmfehlläufen sollte der Anwender das linke PZW-Byte nur über das Spezialregister 1 verändern.

Die Bedeutung der PZR-Bits ist in Bild 3.1 dargestellt; sie ist in Kapitel 3.1.6 genau beschrieben.

PZR-Bit	ist gesetzt bei:
0	Ergebnisanzeigen (Sprungbedingung)
1	Ergebnisanzeigen (Sprungbedingung)
2	Ergebnisübertrag (übertragsspeicher)
3	Prozeßwechselsperre
4	Maske für Betragsrechnungsüberlauf (ohne Division)
5	Maske für Festpunktüberlauf (ohne Division)
6	---
7	Schreibschutz *
8	Programmtestmodus
9	---
10	nicht privilegierter Modus *
11	---
12	Simulationsmodus
13	Maske für Programmlaufzeitzähler (1 = aktiv)
14	Virtuelle Adressierung der Befehle (R1)
15	Virtuelle Adressierung der Daten (R0, R2 ... R15)

Bild 3.1 Programmzustandsregister PZR

* Der nicht privilegierte Modus ist gleichzeitig mit der Schreibschutzfunktion verknüpft (unabhängig von PZR-Bit 7).
  Nur im privilegierten Modus kann über das Bit 7 im PZR der Schreibschutz gesteuert werden.

b) Unterbrechungsregister (UR)

Im Unterbrechungsregister spiegelt sich die Ebenenstruktur der ZE 03 wieder. Jeder der 16 Prioritätsebenen ist ein Bit im Unterbrechungsregister zugeordnet. Dieses Bit kann durch Aktivieren eines Prozesses gesetzt werden. Alle Vorgänge zur Verarbeitung von Informationen in der Zentraleinheit (z. B. Anwenderprogramme) werden als Prozesse bezeichnet.
Solange das Bit einer bestimmten Ebene im Unterbrechungsregister gesetzt ist, befindet sich noch mindestens ein Prozeß ablauffähig in der Prioritätswarteschlange. Wird der letzte Prozeß einer Ebene deaktiviert, so wird auch das entsprechende Bit im Unterbrechungsregister gelöscht. Das Unterbrechungsregister ist durch den Spezialregisterbefehl "LES" lesbar.

c) Zentralprozessor-Zustandsregister (ZZR)

In diesem Register ist der aktuelle Prioritätszustand des Zentralprozessors in binär codierter Form gespeichert. Das Register hat 4 bit. Es kann per Spezialregisterbefehl nur gelesen werden (rechtsbündig).

d) Programmlaufzeitzähler (PLZ)

Der Programmlaufzeitzähler hat die Aufgabe, die Laufzeit von Programmen zu überwachen, um Endlosschleifen zu verhindern. Er hat eine Breite von 16 bit. Sein Inhalt wird nach jeweils 500 Mikroprogrammschritten um eins erhöht, d. h. nach ca. 100 µsec. Bei einem Überlauf wird eine Programmunterbrechung eingeleitet und Bit 13 im Unterbrechungsanzeigenwort (Zelle 2 der Parametertafel) gesetzt. In die Zelle 7 der Parametertafel (s. Kapitel 3.1.6) wird Null eingetragen.

Der Programmlaufzeitzähler wird mit PZR-Bit 13 = "1" freigegeben. Er läuft nur bei der Bearbeitung von Zentralprozessor-Befehlen.

e) Tafelzeigeregister 2 (TZR 2)

Das Tafelzeigeregister 2, mit einer Breite von 16 bit, dient der Adressierung von Daten. Zusammen mit dem Adressierungsweichenregister und dem Tafelzeigeregister 1 (TZR 1) können zwei unabhängige Adreßräume angesprochen werden. Das Tafelzeigeregister 2 wird bei jedem Zustandswechsel in bzw. aus der Zelle 6 der programmspezifischen Parametertafel (s. Kapitel 3.1.6) gerettet und geladen.

Der Mechanismus der Adreßübersetzung ist im Kapitel 3.1.8 beschrieben.

Ob bei einer Adreßübersetzung das Tafelzeigeregister 1 oder Tafelzeigeregister 2 benutzt wird, entscheidet das Adressierungsweichenregister.

f) Adressierungsweichenregister (AWR)

Wenn die Möglichkeit besteht, daß ein Programm gleichzeitig zu zwei 128K* byte großen virtuellen Adreßräumen zugreifen kann, ist eine Einrichtung notwendig, die entscheidet, wann auf welchen Adreßraum zugegriffen werden soll, d. h. mit welcher Übersetzungstafel gearbeitet werden soll.

Diese Aufgabe übernimmt das Adressierungsweichenregister, das für jedes Standardregister, welches zur Adressierung herangezogen wird, angibt, ob die Adreßübersetzung über Tafelzeigeregister 1 oder Tafelzeigeregister 2 durchzuführen ist. Die Bit-Nummer des Adressierungsweichenregisters entspricht der Nummer des Standardregisters. Bei gelöschtem Bit erfolgt eine Adreßübersetzung mittels Tafelzeigeregister 1 aus der Übersetzungstafel 1. Bei gesetztem Bit erfolgt eine Adreßübersetzung mittels Tafelzeigeregister 2 aus der Übersetzungstafel 2 (sofern - abhängig von PZR-Bit 15 - überhaupt virtuell adressiert wird).

Das Adressierungsweichenregister wird bei jedem Zustandswechsel in eine andere Ebene in bzw. aus der Zelle 5 der programmspezifischen Parametertafel gerettet und geladen. Die nachfolgende Tabelle 3.1 zeigt eine Zusammenstellung aller implementierten Spezialregister, ihrer Adressen und die anwendbaren Befehle.

Adresse (F1-Feld)	Name	Breite (Bit)	Anwendbare Befehle LAS	BSS	BLS	LES	Proz.Bef.
0	PZR	16	X	X	X	X	
1	PZR*	8	X	X	X	X	
3	UR	16				X	X
5	ZZR	4				X	
8	TZR2	16	X	X	X	X	
11	AWR	16	X	X	X	X	
12	PLZ	16	X	X	X	X	

PZR.....Programmzustandsregister
PZR*.....Programmzustandsregister (Maskenteil)
UR.....Unterbrechungsregister
ZZR.....Zentralprozessor-Zustandsregister
TZR2.....Tafelzeigerregister 2
AWR.....Adressierungsweichenregister
PLZ.....Programmlaufzeitzähler
Proz. Bef..Prozeßsteuerbefehle

Tabelle 3.1 : Spezialregister

Normal- und Simulationsmodus

Der Zentralprozessor hat zwei Betriebsmodi:

- den Normalmodus und
- den Simulationsmodus.

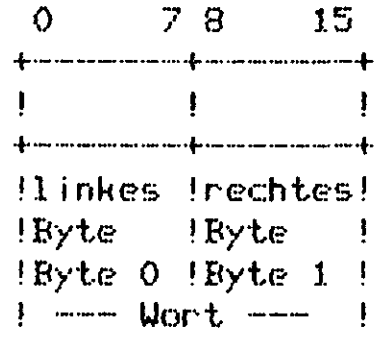
"Normale" Programme (Anwenderprogramme) und große Teile des Betriebssystems laufen im Normalmodus ab. Hier stehen ihnen alle Eigenschaften der Maschine zur Verfügung (virtuelle Adressierung, Schutzfunktionen). Wenn diese Anwenderprogramme allerdings Befehle enthalten, die im Zentralprozessor (oder/und Erweiterungsprozessor) nicht realisiert sind, schaltet die Zentraleinheit den Prozeß in den Simulationsmodus um.

Beginnend mit der Startadresse, die im Standardregister R1 der Simulationsregister in der Parametertafel steht, sollte das System Simulationsroutinen bereithalten, die nun im Simulationsmodus durchlaufen werden. Diese Simulationsroutinen können generiert (Betriebssystemkomponente) oder vom Anwender selbst erstellt werden (Sonderroutinen). Der Wechsel vom Simulations- in den Normalmodus erfolgt mit dem Befehl SPS (Springe aus dem Simulationsmodus). Die Fortsetzadresse des Normalmodus muß vor dem Wechsel vom Programm im Simulationsmodus errechnet und in das Standardregister R1/Normalmodus der Parametertafel eingetragen werden.

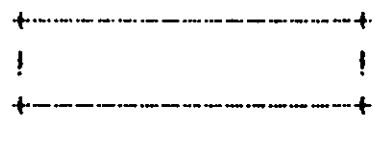
3.1.2 Informationsdarstellung

Im System 300/6000, Modellreihe SICOMP werden folgende Datenformate verwendet (Bitnumerierung nach SIBI, s. Anhang):

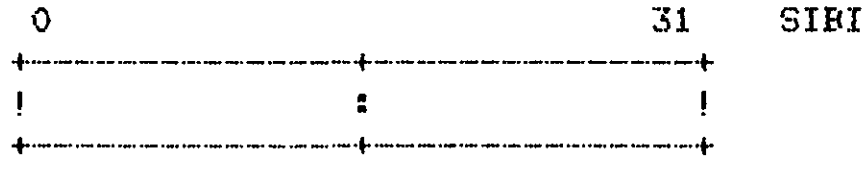
Das 8-bit-Byte im Wortraster



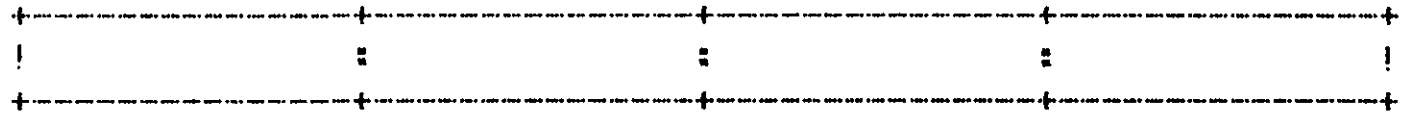
Das 16-bit-Wort



Das 32-bit-Doppelwort



Das 64-bit-Vierfachwort



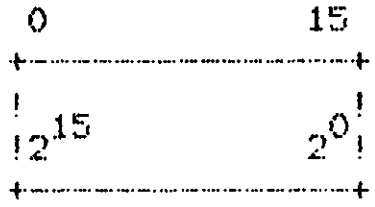
Die 32-bit-Doppelwörter können in einem Registerpaar oder in zwei aufeinanderfolgenden Zentralspeicherzellen abgelegt werden. Die 64-bit-Vierfachwörter werden in einem Registerquartett oder in vier aufeinanderfolgenden Speicherzellen gespeichert.
Diese Datenformate können mit folgenden Datentypen belegt werden:

- **Bitmuster** mit beliebigen Kombinationen von 0 und 1 innerhalb eines 16-bit-Worts zur Darstellung binärer Prozeßsignale.

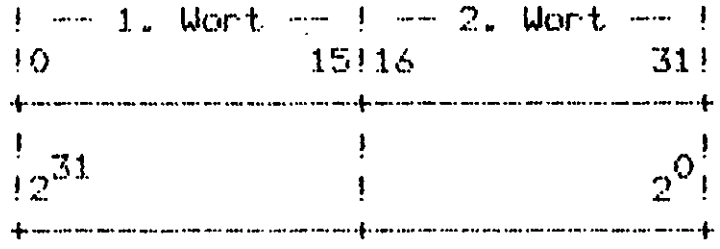
- Viele periphere Einheiten benutzen als Informationseinheit das **Byte** zur Darstellung alphanumerischer Zeichen, wobei der 7-bit-Code nach ISO 646/DIN 66 003 - internationale Referenzversion - verwendet wird.

- **Vorzeichenlose ganze Dualzahlen** (Betragszahlen) in den Formaten Wort und Doppelwort.

Wort — Zahlenbereich: $0 \leq Z \leq +(2^{16}-1)$

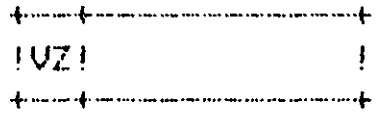


Doppelwort — Zahlenbereich: $0 \leq Z \leq +(2^{32}-1)$

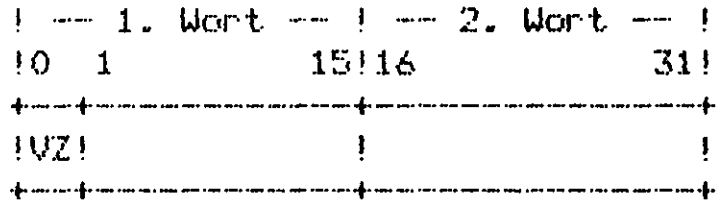


- **Vorzeichenbehaftete ganze Dualzahlen** (Festpunktzahlen) in den Formaten Wort, Doppelwort und Vierfachwort.

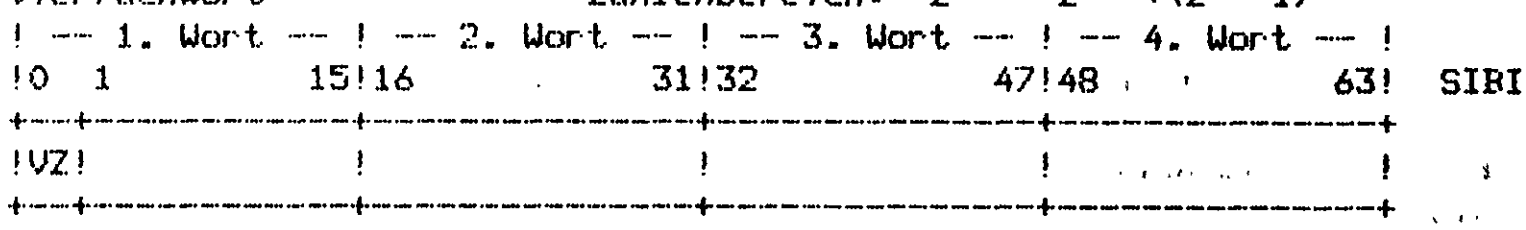
Wort — Zahlenbereich: $-2^{15} \leq Z \leq +(2^{15}-1)$



Doppelwort — Zahlenbereich: $-2^{31} \leq Z \leq +(2^{31}-1)$



Vierfachwort — Zahlenbereich: $-2^{63} \leq Z \leq +(2^{63}-1)$



VZ = Vorzeichen

Darstellung der vorzeichenbehafteten Dualzahlen im Wortformat:

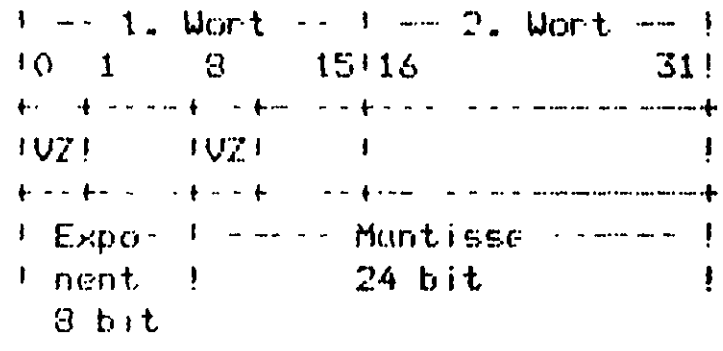
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=	-32768
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	=	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	=	+32767

Das Vorzeichen steht im Bit 0. Die Wertigkeit der Zahl beginnt bei Bit 1 mit 2exp14 und setzt sich fort mit 2exp13, 2exp12 usw. bis 2exp0 bei Bit 15. Eine negative Zahl wird durch ihr 2er-Komplement dargestellt. Ähnliches gilt bei den Formaten Doppelwort und Vierfachwort; das niederwertigste Bit (2exp0) steht ebenfalls am weitesten rechts (Bit 31 bzw. Bit 63).

Gleitpunktzahlen in den Formaten Doppelwort und Vierfachwort. Der Exponent ist eine 8-bit-lange Festpunktzahl im Bereich $-128 \leq EXP \leq +128$. Das Komma der Mantisse ist vor Bit 9 zu denken.

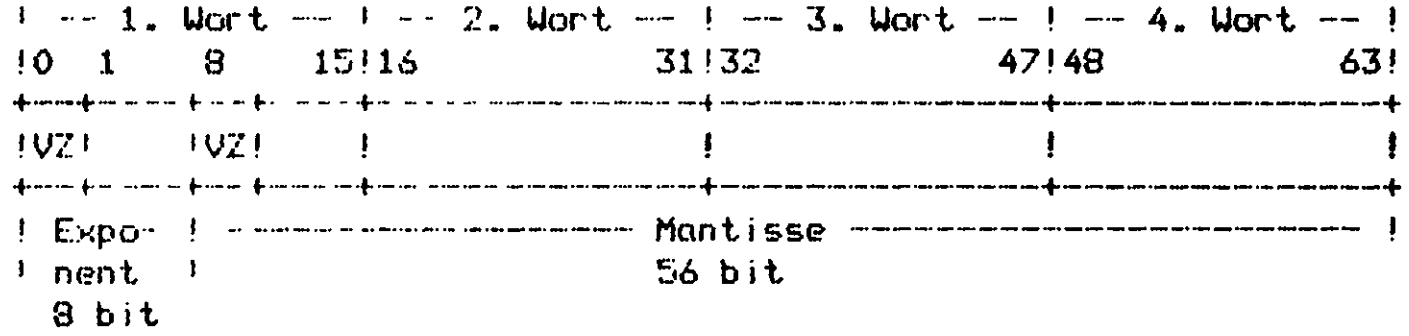
Zahlenbereich etwa: $-10^{38} \leq Z \leq +10^{38}$

Doppelwort



VZ = Vorzeichen

Vierfachwort



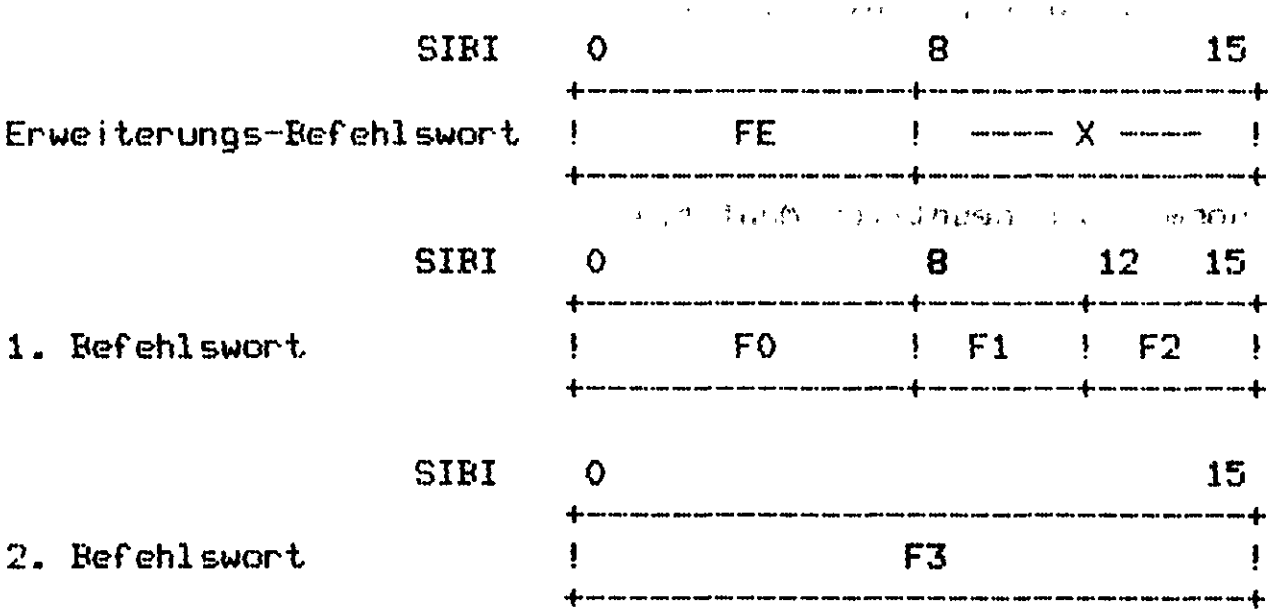
Das Vorzeichen des Exponenten steht in Bit 0; das Vorzeichen der Mantisse in Bit 8. Die Wertigkeit der Mantisse beginnt bei Bit 9 mit 2exp-1 und setzt sich fort mit 2exp-2, 2exp-3 usw. bis 2exp-23 (bzw. 2exp-55). Ein negativer Exponent und eine negative Mantisse werden durch das 2er-Komplement dargestellt (im nachstehenden Beispiel ist der bei der Berechnung negativer Zahlen notwendige Korrekturwert nochmals explizit angegeben).
Bei einer normierten positiven Mantisse ist Bit 9 als führendes Bit mit 1, bei einer normierten negativen Mantisse mit 0 vorbesetzt.

Exponent	Mantisse, normiert	
0 0 0 0 0 0 1 1	0 1 1 0 0 0 0 ... 0	= (2exp-1 + 2exp-2) x 2exp3 = 0,6 x 10exp1
0 0 0 0 0 1 0 0	0 1 1 0 1 1 0 ... 0	= (2exp-1 + 2exp-2 + 2exp-4 + 2exp-5) x 2exp4 = 0,135 x 10exp2
1 1 1 1 1 1 0 0	0 1 1 0 0 0 0 ... 0	= (2exp-1 + 2exp-2) x 2exp-(3+1) = 0,46875 x 10exp-1
0 0 0 0 0 0 1 1	1 0 1 0 0 0 0 ... 0	= -(2exp-1 + 2exp-3 + 2exp-4 + ... + 2exp-n + 2exp-n) x 2exp3 = -(2exp-1 + 2exp-2) x 2exp3 = -0,6 x 10exp1
0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 ... 0	= -(2exp-1 + 2exp-2 + ... + 2exp-n + 2exp-n) x 2exp0 = (-2exp0) x 2exp0) = -0,1 x 10exp1

3.1.3 Befehle

Der Befehlsvorrat umfaßt im Grundausbau 286 Befehle, die durch den Gleitpunktprozessor um 52 Befehle erweitert werden können (Summe 338 Befehle). Die verwendete Befehlsliste ist eine Matrix-Befehlsliste, die dadurch entsteht, daß eine bestimmte Zahl von Grundoperationen in mehrere Befehls-Varianten aufgespalten wird. Diese Varianten können ihrerseits in verschiedenen Formatmodifikationen ablaufen (z.B. Register-Register-Verkehr oder Register-Arbeitsspeicher-Verkehr). Zur Kennzeichnung der Operation stehen zunächst 8 bit zur Verfügung, womit 256 Befehle codiert werden können.
Zur Vergrößerung der Anzahl der möglichen Befehlscodierungen über 256 hinaus werden durch eine Art Substitutionsbefehl weitere Vielfache von 256 Befehlskodierungen erreicht. D.h. die Stelle, die sonst den Befehlscode enthält (in diesem Fall FE), verweist auf die eigentliche Stelle, die den Code enthält (F0). Durch Verwendung verschiedener Codes im FE-Feld verschafft man sich weitere Möglichkeiten für Befehlscodes.

Das Befehlswort ist in Felder (FE, F0, F1, F2 und F3) aufgeteilt.



Im FO-Feld steht der Operationsteil. Er ergibt sich aus der entsprechend dem Befehlsformat modifizierten Grundoperation. Das Format gibt an, wie die F1-, F2-, F3-Felder zu interpretieren sind.
Die Formate werden durch die Formatkennzeichen dargestellt; es sind dies: C, R, A und X. Sie haben folgende Bedeutung (s. auch Bild 3.2):

Feld	Kennzeichen	Bedeutung
F0	-	Operationsteil
F1	C	Sprungbedingung (siehe Sprungbefehl) bzw. Bitnummer (siehe Bit-Test-Befehle)
	R	Nummer des Standardregisters, das den Operanden enthält.
	A	Nummer des Standardregisters, das die Adresse der Arbeitsspeicherzelle bzw. die EA-Adresse enthält, in der der Operand steht.
	AI	Wie A, jedoch wird die Adresse nach dem Lesen um 1 erhöht.
	DA	Wie A, jedoch wird die Adresse vor dem Lesen um 1 erniedrigt.
F2	C	4-bit-Operand
	R A AI DA	wie beim F1-Feld
F3	RX	Inhalt des F3-Felds wird zum Inhalt des vom F2-Feld adressierten Standardregisters addiert. Summationsergebnis = Operand (außer USK s Sprungbefehle)
	AX	Inhalt des F3-Felds wird zum Inhalt des vom F2-Feld adressierten Standardregisters addiert. Summationsergebnis bezeichnet die Zentralspeicher-Adresse des 2. Operanden. In beiden Formaten wird zum F3-Feld Null addiert, wenn im F2-Feld das Standardregister 0 angesprochen wird.

Tabelle 3.2 Formatkennzeichen

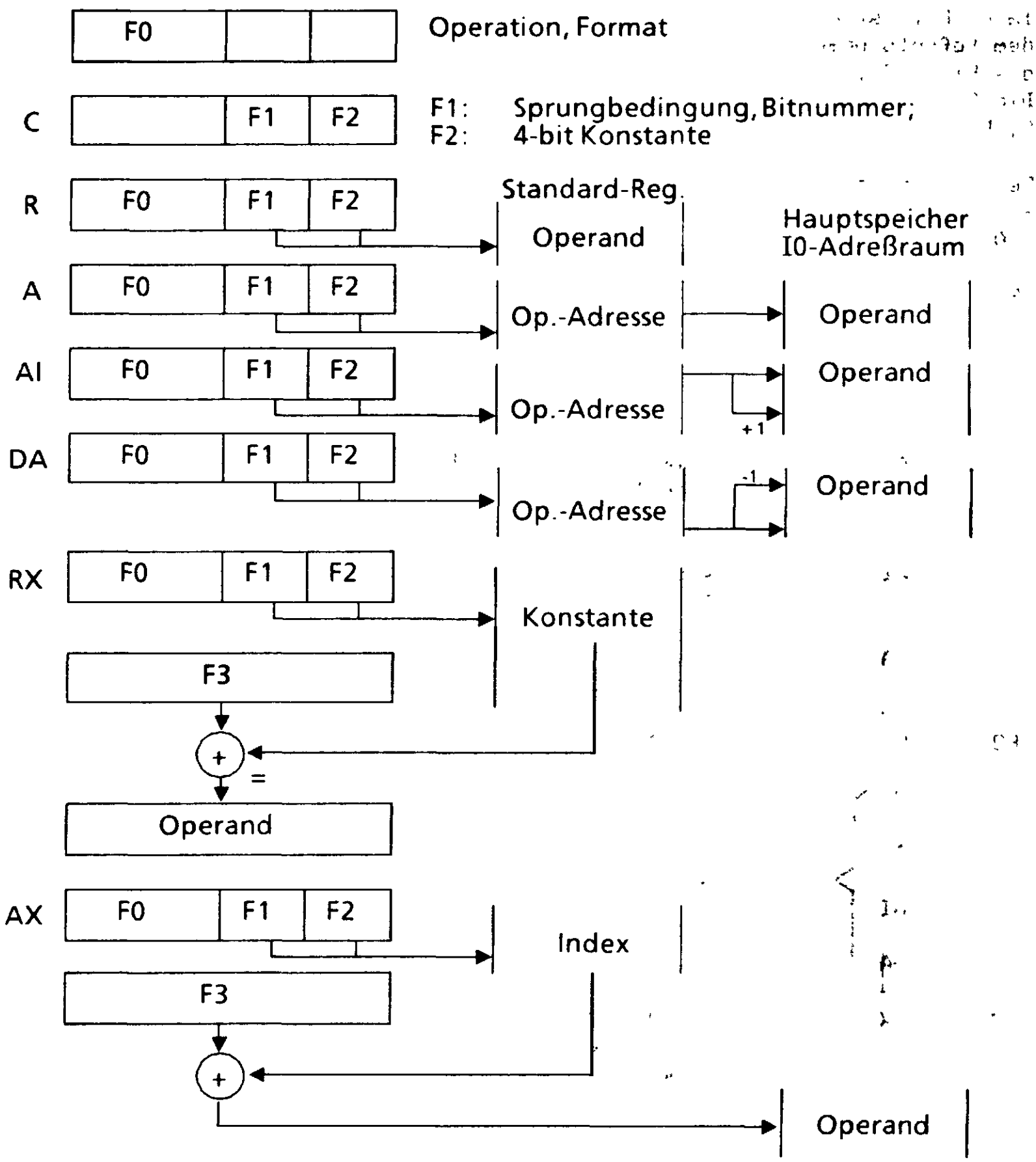


Bild 3.2: Befehlsformate

Tabelle 3.3 Bedeutung der Felder F1, F2, F3 bezüglich der Grundbefehle:

Befehle	Bedeutung F1	Bedeutung F2/F3
Ladebefehle	Ziel der Information	Quelle der Information
Speicherbefehl	Quelle der Information	Ziel der Information
Arithmetische und Boolsche Befehle	1. Operand Ergebnis	2. Operand
Vergleichsbefehle	1. Operand	2. Operand
Bit-Testbefehle	Bit-Nummer	Operand
Sprungbefehle (ohne USK)	Bedingung bzw. Rücksprung-Adresse	Zieladresse
Sprungbefehl USK	Zieladresse	Rücksprungadresse
Schiebebefehle	Operand	Schiebezahl
Spezialregister-Befehle	Spezialregister-Nummer	Quelle Bitmuster Ziel
EA-Befehle	IO-Adresse	Ziel oder Quelle der Information
Feldsuchbefehle	Muster	Anfangsadresse
Fädelbefehle (ohne AHD)	Element, vor dem in die Fädelliste eingefügt werden soll	ein- bzw. auszufügendes Element
Fädelbefehl AHD	auszufügendes Element	Element, nach dem aus der Fädelliste ausgefügt werden soll
Bytefeldbefehle TBF, VBF	Anfangsadresse	Anfangsadresse

Die einzelnen Befehle werden bearbeitet, indem festgelegte Mikroprogramm-Befehle in entsprechender Abfolge durchlaufen werden. Ein solcher Mikrobefehl wird in einem Maschinenzyklus ausgeführt, der, abhängig vom Mikrobefehl selbst, 150 ns bis 400 ns dauert. Auf Mikroprogramm-Ebene ist ein Pipeline-Register eingerichtet, das den jeweils in Ausführung befindlichen Mikrobefehl enthält. Dadurch kann parallel dazu während des gleichen Zeitintervalls bereits der nächste Mikrobefehl im Mikroprogrammspeicher adressiert werden. Er steht am Ende dieses Intervalls bereit, in das Pipeline-Register übernommen zu werden. Es entfällt somit in der nötigen Bearbeitungszeit jener Anteil, der für die Adressierung des Mikroprogramm-Speichers nötig wäre, da dieser Vorgang überlappend zur Bearbeitung des vorigen Befehls durchgeführt wird.

Die Zentraleinheit enthält eine Erweiterungsprozessorschnittstelle zum Anschluß eines Zusatzprozessors (z. B. Gleitpunktprozessor). Sie ermöglicht es, anhand einer Mikrobefehlsliste Kommandos zur Datenkommunika-tion an den Zentralprozessor zu geben. Damit können Daten direkt aus dem Zentralprozessor gelesen bzw. in den Zentralprozessor geschrieben werden. Dieser direkte Datenzugriff erlaubt es, aktuelle Programmdaten (Standard-, Spezialregister) ohne die sonst notwendige Programmunterbrechung (Zustandswechsel) auszulesen.

Der universelle Charakter dieser Erweiterungsprozessorschnittstelle bietet die Möglichkeit des Anschlusses eines weitgehend beliebigen Erweiterungsprozessors. Erscheint ein Befehl, der vom Zentralprozessor nicht interpretiert werden kann (NNN), so versucht dieser in den Simulationsmodus zu wechseln. Dies kann durch einen Erweiterungsprozessor (EP) verhindert werden, der die Bearbeitung dieses NNN übernimmt. Die Ver- und Entsorgung des Erweiterungsprozessors mit den relevanten Daten wird über die Erweiterungsprozessorschnittstelle abgewickelt.

Service-Mittel können ebenfalls die Erweiterungsprozessorschnittstelle benutzen, um aktuelle programmspezifische Daten auszulesen. Hierfür ist es nicht notwendig, das laufende Programm zu unterbrechen, da direkt auf die Hardware-Register zugegriffen werden kann.

Sofern kein Erweiterungsprozessor vorhanden ist, werden diese Befehle von Simulationsroutinen abgearbeitet, die Bestandteil des Betriebssystems sind.

3.1.4 Gleitpunktprozessor

Der Gleitpunktprozessor (GPP) steht in zwei Ausführungen zur Verfügung:

- Standardvariante
- Hochleistungsvariante .

Beide Gleitpunktprozessoren bearbeiten denselben Befehlsvorrat, nämlich

- Addieren, Subtrahieren, Multiplizieren, Dividieren mit 32-bit- und 64-bit-Gleitpunktzahlen,

- Multiplizieren und Dividieren mit 32-bit-Festpunktzahlen.

Sie sind an der universellen Schnittstelle für Erweiterungsprozessoren im Zentraleinheitsbaugruppenträger steckbar (GPP1 bzw GPP2, Bild 2.1). Die Hochleistungsvariante benötigt zwei Steckplätze, die Standardvariante benötigt nur einen Steckplatz; der zweite ist als EA-Anschlußstelle verwendbar.

Die schnellere Variante erzielt gegenüber der Standardvariante ca. 5-mal kürzere Befehlsausführungszeiten (siehe Operationszeiten im Kap. 6.4.2) durch mehrere Faktoren:

- Einsatz schnellerer Bauelemente,
- weitergehende Hardware-Unterstützung des Mikroprogramms und
- rascherer Operandentransfer zwischen Zentralprozessor und Gleitpunktprozessor.

Die Adressierung des Zentralspeichers bei Operationen des Gleitpunktprozessors geschieht unter Zuhilfenahme der Adressierungsmechanismen des Zentralprozessors. Während der Gleitpunktprozessor arbeitet, kann der Zentralprozessor keine anderen Befehle abarbeiten.

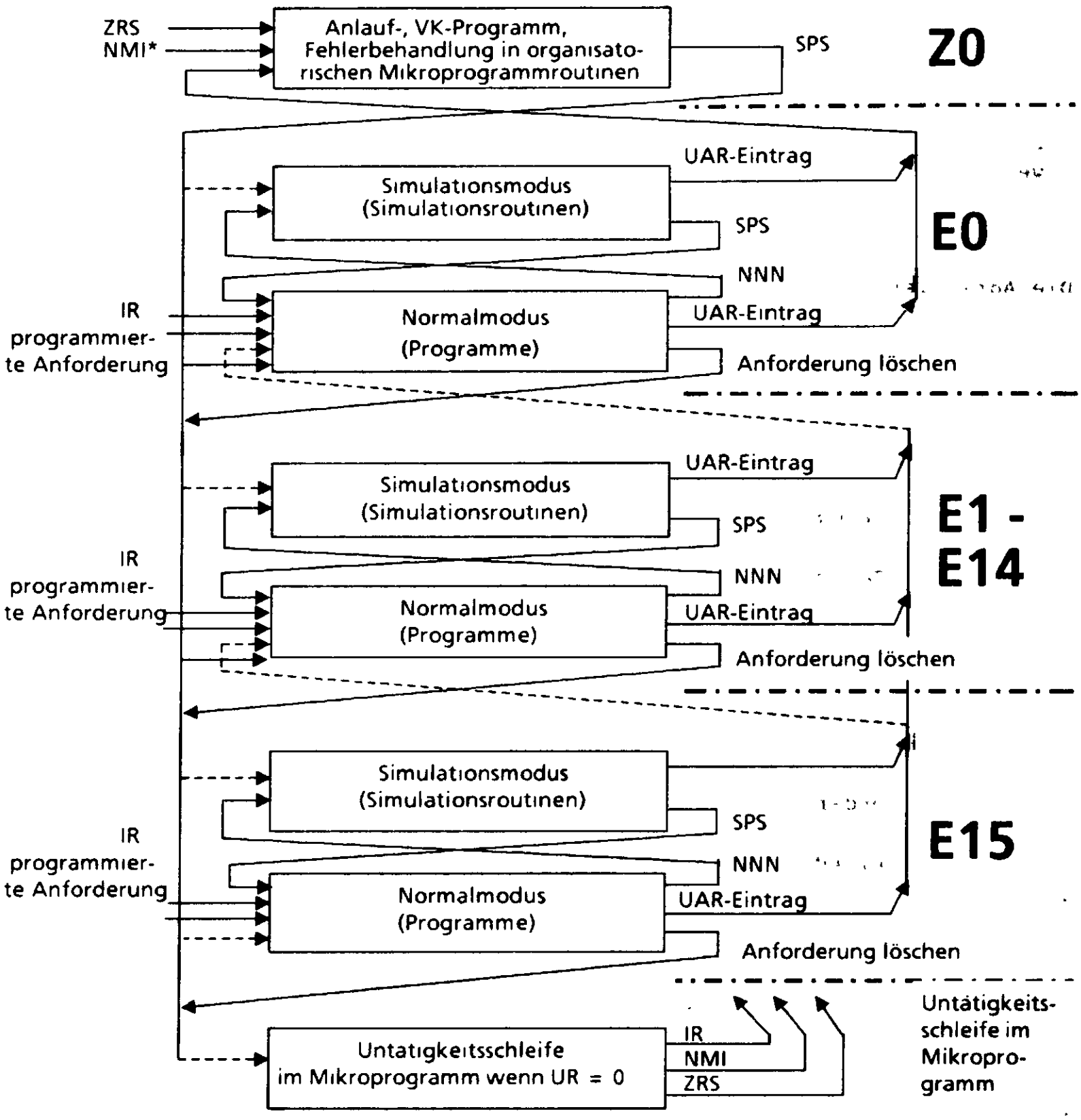
Ist kein Gleitpunktprozessor gesteckt, so führt das Auftreten eines entsprechenden Befehls zu einem "NNN", und das Betriebssystem übernimmt die softwaremäßige Ausführung des Befehls im Simulationsmodus. Auf diese Weise steht für das Programm der gleiche Befehlsvorrat zur Verfügung, unabhängig davon, ob ein Gleitpunktprozessor gesteckt ist oder nicht. Der Unterschied liegt lediglich im Durchsatz, d. h. in der Geschwindigkeit der Befehlsbearbeitung.

3.1.5 Unterbrechungsstruktur, Hardware-Prozeßverwaltung

Während des normalen Programmlaufs steht der Zentralprozessor dem jeweils aktuellen Anwenderprogramm zur Verfügung und arbeitet dieses ab. Eine Änderung dieses Zustands muß eintreten, wenn beim Programmlauf irgendwelche Besonderheiten, Fehler, Störungen oder Alarme auftreten. Um eine rasche Reaktion auf solche Unterbrechungsereignisse zu ermöglichen, ist die Zentraleinheit ZE 03 mit 16 Prioritätsebenen - E0 bis E15 - ausgestattet. Jede Ebene ist in einen Normalmodus und einen Simulationsmodus unterteilt. Festwertspeicherprogramme (Urladen, virtuelle Konsole, Basistest) laufen mit höchster Priorität im Zustand 0 (Z0). Diesem Zustand wird der Hardware-Registersatz im Zentralprozessor zugeordnet. Bild 3.3 zeigt die Prioritätszustände.

Zur Unterstützung der schnellen Umschaltung von Prozessen wurde im Zentralprozessor eine hardwaremäßige Prozeßverwaltung, auf der das Betriebssystem aufsetzen kann, implementiert.

Die prinzipielle Darstellung einer derartigen Prozeßverwaltung ist im Bild 3.4 dargelegt. Für jeden im System vorhandenen Prozeß wird im Zentralspeicher ein sogenannter Prozeßblock aufgebaut, der u. a. Informationen über Zustand des Prozesses, Priorität und auszuführende Funktionen enthält.



*STOP, FST, VC, BT, PF, ADV sind NMI-Ereignisse

Bild 3.3: Prioritätszustände und Ursachen für den Zustandswechsel

Für jede Prioritätsebene E0-E15 gibt es eine Prioritätswarteschlange. Die Priorität des Prozesses ist im Prozeßblock festgehalten. Durch die Aktivierung eines Prozesses wird entsprechend der im Prozeßblock vermerkten Priorität ein diesen Prozeßblock kennzeichnender Fädelverweis in eine der 16 Prioritätswarteschlangen eingehängt. Sofern in einer Prioritätswarteschlange einer Prioritätsebene Prozesse auf ihre Bearbeitung warten, d. h. ihre Fädelverweise in die Prioritätswarteschlange einghängt sind, wird das zugehörige Bit im Unterbrechungsregister (UR) gesetzt (z. B. Bit 10 für Prioritätsebene 10). Somit kann über das Unterbrechungsregister (UR) die jeweils höchstpriore Anforderung (Ebene) ermittelt werden, indem das erste gesetzte Bit gesucht wird. Daraufhin wird ggf. ein Zustandswechsel angestoßen. Mit diesem Verfahren einer hardwaremäßigen Warteschlangenbearbeitung ist eine einheitliche Abarbeitung von internen (Software-Anforderung) und externen (Interrupt) Anforderungen gewährleistet.

Trifft z. B. ein Interrupt eines peripheren Geräts (externe Anforderung) ein, wird mittels des mitgelieferten Vektors über eine Vektorliste der zugehörige Prozeßblock im Zentralspeicher angesprochen. Je nach Priorität wird ein Fädelverweis (Anfangsadresse des Prozeßblocks) in einen der 16 Warteschlangen eingehängt (Bild 3.4). Die Adresse des Prozeßblocks ergibt sich aus der Vektorliste und wird durch den Vektor + 256 (dezimal) ermittelt. Der erste Prozeßblock der jeweils höchstprioren Warteschlange wird abgearbeitet. Bei Programmende wird der Fädelverweis des abgearbeiteten Prozeßblocks aus dem Warteschlangenkopf ausgehängt und das nachfolgende Element der Warteschlange, das jetzt erstes Element wird, kommt zum Ablauf. Gleiches gilt für die Behandlung interner, vom Betriebssystem über den Befehl "Aktiviere Prozeß" angestoßener Anforderungen.

Diese Art der Speicherung von peripheren Unterbrechungsanforderungen ermöglicht es, sich auf eine Interrupt-Leitung an der EA-Schnittstelle zu beschränken, da die Speicherung der einzelnen Alarme über die Fädelung in der Prioritätswarteschlange der Prozeßverwaltung durchgeführt wird. Hierzu muß der Zentralprozessor im Mikroprogramm eine kurze Einhängeroutine durchlaufen (kein Zustandswechsel notwendig). Eine hardwaremäßige Festlegung der Priorität beim peripheren Gerät ist nicht erforderlich. Sie wird durch eine Prioritätsinformation im Prozeßblock ersetzt, nach der die Fädelung in dem der Priorität entsprechenden Warteschlangenkopf vorgenommen wird. Ankommende Interrupts der gleichen Hardware-Priorität werden am Ende der Warteschlange der entsprechenden Ebene eingehängt.

Zur Realisierung dieser vorgestellten Prozeßverwaltung ist nur das Unterbrechungsregister UR notwendig. Zwischen den Ebenen und den einzelnen Bitstellen herrscht eine feste 1 : 1 Zuordnung, d. h. mit Bitstelle i im Unterbrechungssystem wird eindeutig die Ebene i identifiziert.

Der Modus in der jeweiligen Ebene wird durch das PZR-Bit 12 bestimmt. Ein Wechsel aus dem Normalmodus in den Simulationsmode wird nur durch einen nicht interpretierbaren Befehl (NNN) verursacht.

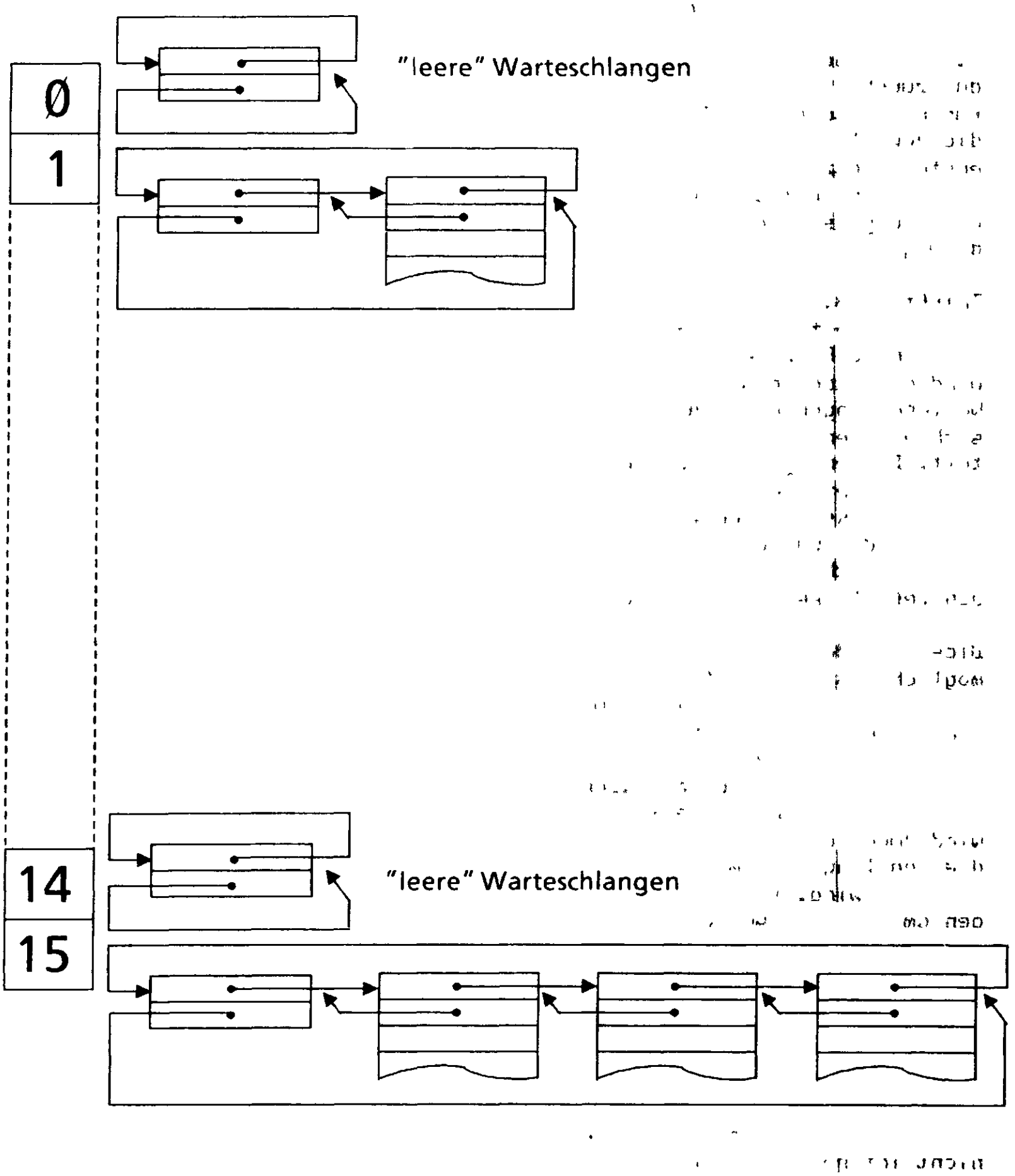


Bild 3.4: Prinzip der Prozeßverwaltung über Warteschlangenorganisation

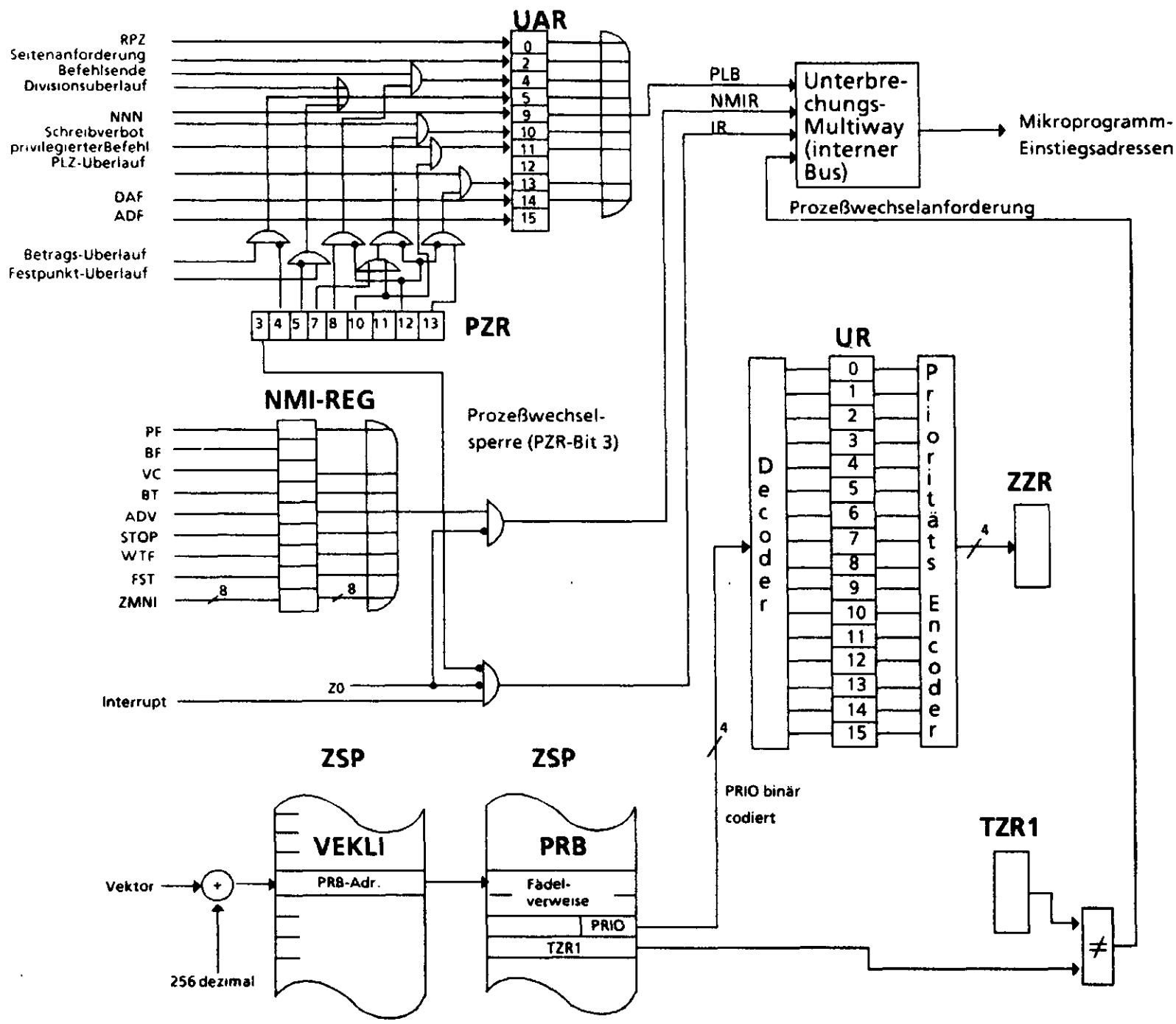


Bild 3.5: Unterbrechungssteuerung

Der Simulationsmodus wird durch den Befehl SPS (SPS ... "springe" aus dem Simulationsmodus) wieder verlassen. Dies gilt allerdings nur, wenn der Inhalt des Standardregisters, das durch das F2-Feld des Befehls SPS angegeben wird, gleich 0 ist. Andernfalls erfolgt eine Programmlaufbesonderheit (PLB)-Bearbeitung, wobei das Register, das im F2-Feld angegeben wurde, in den Unterbrechungsanzeigenwert UAW der Parametertafel PT eingetragen wird (s. Kapitel 3.1.6). Eine Programmlaufbesonderheit im Simulationsmodus wird wie eine solche im Normalmodus behandelt.

Die Bearbeitung einer Programmlaufbesonderheit wird durch Aktivieren eines höherprioren Programms durchgeführt, wodurch das ursprüngliche Programm unterbrochen wird.

Nach dem Einschalten bzw. Rücksetzen befindet sich der Zentralprozessor im Zustand 0 (Z0). In diesem Zustand werden verschiedene Anlaufroutinen durchlaufen, bevor der Befehl SPS den Zentralprozessor veranlaßt, diesen Zustand zu verlassen. Alle NMI-Ereignisse (STOP, Fortsetzungsstart FST, Urladen BT, virtuelle Konsole VC, Netzausfall PF, Adreßvergleich AIV; NMI s. Kapitel 3.3.3.3) führen ebenfalls in den Zustand Z0, ebenso Programmlaufbesonderheiten während organisatorischer Mikroprogramm-Routinen (z. B. Zustandswechsel).

Der Zentralprozessor befindet sich in der Untätigkeitsschleife, falls kein Prozeß aktiviert ist.

3.1.5.1 Unterbrechungsereignisse

Unterbrechungsereignisse unterbrechen das laufende Programm an einer definierten Stelle. Eine Unterbrechung findet normalerweise am Befehlsende statt. Es gibt einige "lange" Befehle, die auch während des Befehlsablaufs unterbrechbar sind. Bei allen Unterbrechungen ist sichergestellt, daß ein ordnungsgemäßes Fortsetzen des Programms zu einem späteren Zeitpunkt möglich ist.

Tabelle 3.4 zeigt die möglichen Unterbrechungsereignisse, wobei die Priorität bei gleichzeitig einlaufenden Ereignissen aus der Tabelle ersichtlich wird.

Name	Priorität	Erläuterungen
MINT	1	Der Mikro-Interrupt unterbricht bei freigegebenem Programmlaufzeitzähler (PLZ) ca. alle 100 us kurzzeitig das Programm, um den PLZ zu inkrementieren.
KOMSP	2	Kommunikation-Serviceprozessor (Der Serviceprozessor benötigt Informationen aus dem Zentralprozessor.)
PLB	3	Programmlaufbesonderheiten (UAR-Einträge)
NMI	4	non-mascable-interrupt STOP STOP-Taste CON Fortsetzungsstart-Taste VC Virtuelle Konsole BT Urladen PF Netzausfall ADV Adreßvergleich Diese Ereignisse führen alle in den Zustand Z0 und sind in Ebene 0 - 15 nicht maskierbar. Im Zustand Z0 müssen diese Ereignisse im Polling-Verfahren abgefragt werden.
IR	5	Interrupt-Request Ein peripheres Gerät meldet sich beim Betriebssystem durch einen Interrupt an. Dieser Interrupt ist im Zustand Z0 oder bei gesetzter Prozeßwechselsperre (PZR-Bit 3) maskiert.

Tabelle 3.4 mögliche Unterbrechungsereignisse

3.1.5.2 Unterbrechungssteuerung, Zustandswechsel

Folgende Unterbrechungsereignisse sind für die Steuerung der Unterbrechung und des Zustandswechsels zuständig:

- Programmlaufbesonderheiten (PLB)
- nicht maskierbarer Interrupt (NMI)
- Verlassen von Z0 (SPS)
- Löschen der Prozeßwechselsperre (PZWS)
- Periphere Unterbrechungsanforderungen (IR)
- Prozeßsteuerbefehle (DAP,DFP,APF,APZ)

Bild 3.6 zeigt den Umfang des Zustandswechsels für die verschiedenen Unterbrechungsereignisse.

Umfang des Zustands-wechsels (abhangig von der Ursache)	NM → SM	SM → NM	B = 0X ext NMI → Z0	B = 1X	A = 0 IR (nieder-bzw gleichprior)	A = 1	B = 00 B = 01 IR (höher-prior)	B = 10	B = 11	M = 1	M = 0; T = X0 DAP	M = 0, T = X1 M = 0; UR = 0	B = X0 APZ/APF	B = X1	B = 00 B = 10 PLB	B = 10	B = 11	B = 0X int NMI 1	B = 1X	int NMI 2	B = X0 verlassen Z0	B = X1
UAR retten															X							
BAP retten			X												X							
TZR3P retten			X												X							
BAP NNN retten	X																					
TZR 2 retten			X					X			X	X	X		X			X				
TZR 3 retten			X					X			X	X	X		X			X				
PZR retten			X					X			X	X	X		X			X				
PLZ retten				X					X	X			X			X			X			
AWR retten			X					X			X	X	X		X			X				
R0 retten	X	X		X					X	X			X				X	X	X			
R1 retten	X	X	X					X			X	X	X		X			X				
R2-R15 retten	X			X					X	X			X				X	X	X			
PRB aushangen											X	X										
M-Bit loschen										X												
A Bit loschen											X	X										
T-Bit loschen					X						X	X										
PRB einhangen					X			X					X		X							
M Bit setzen						X																
A Bit setzen					X			X					X		X							
T-Bit setzen								X					X		X							
AWR laden			0000					X			X		X		X			0000	0000		X	
TZR 3 laden			FFF0					X			X		X		X			FFF0	FFF0		X	
TZR 2 laden								X			X		X		X						X	
PZR laden			0003					X			X		X		X			0003	0003		X	
PLZ laden							0 X		0 X		0 X		0 X		0 X		0 X			0		X
R0 laden	X	X						X			X			X		X						X
R1 laden	X	X	FFFE					X			X		X			X		FFFC	FFFC		X	
R2 R15 laden		X						X			X			X			X	R6	R6			X

B Bit XX → B des neuen Prozesses

→ B des alten Prozesses

Bild 3.6 Umfang der Zustandswechselroutinen

3.1.5.3 Aufbau der Prozeßblöcke

Der Prozeßblock dient als Deskriptor eines Prozesses. In ihm sind Informationen enthalten, die es gestatten:

- einen Prozeß durch die Hardware-Prozeßverwaltung prioritätsgerecht einzuordnen
- den Status eines Prozesses festzuhalten (z. B. aktiv)
- den Bezug zum zugehörigen Programm herzustellen
- auf das Programm zur Behandlung von Besonderheiten hinzuweisen

Die Lage der Prozeßblöcke innerhalb der reell zu adressierenden 128 K*byte ist beliebig festlegbar. Die Unterbrechungssteuerung fordert lediglich einen einheitlichen, fest vereinbarten Aufbau der ersten 5 Zellen eines jeden Prozeßblocks (Bild 3.7).

Dabei sind in den beiden ersten Zellen die Fädelverweise, im dritten Wort rechtsbündig die gewünschte Priorität und linksbündig 4 Zustandsbits, im vierten Wort der Tafelzeiger 1, der auf Parametertafel und Übersetzungstafel 1 zeigt, eingetragen. Mittels der Priorität wird bei einer Aktivierung eines Prozeßblocks das zu setzende Unterbrechungsregister-Bit sowie der dazugehörige Warteschlangenkopf bestimmt.

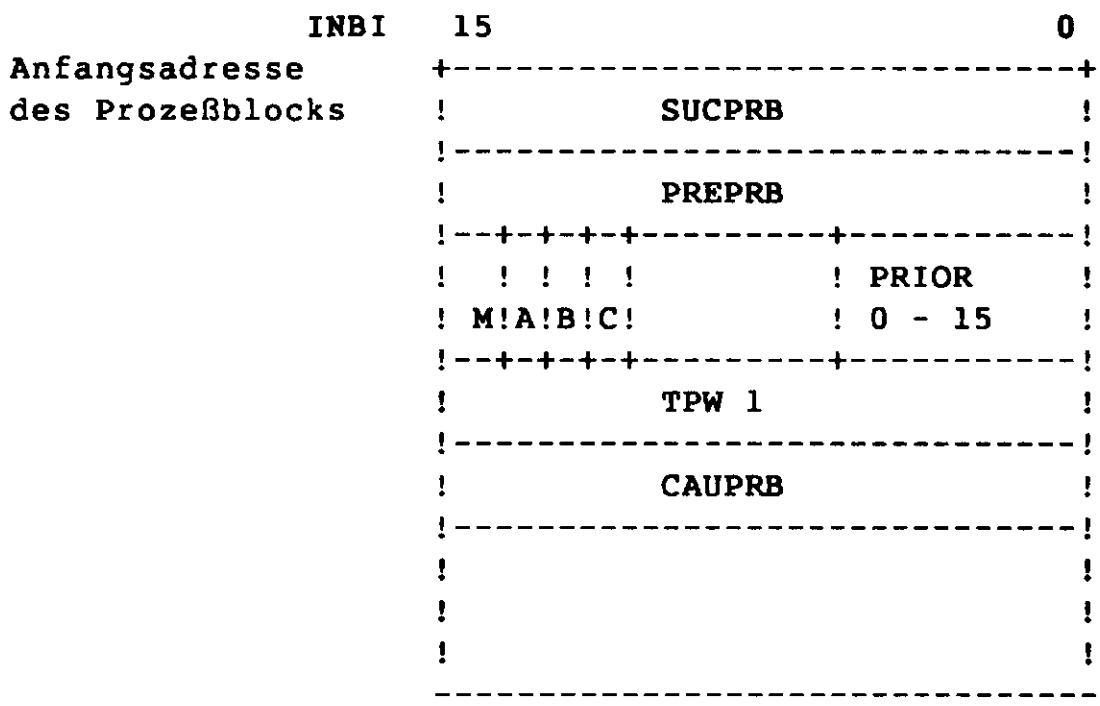


Bild 3.7: Prozeßblock

SUCPRB....succeeding PRB (Anfangsadresse des nächstfolgenden Prozeßblocks); Zweck: Fädelung (Einhängen in die Prioritätswarteschlange)
PREPRB....preceding PRB (Anfangsadresse des vorher zu bearbeitenden PRB); Zweck: Fädelung
M.....multiple bit (Mehrfachanforderung)
A.....active bit (aktiv)
B.....busy bit (tätig)
C.....check bit (Prüfbit)

PRIOR.....priority of process (Prozeß-Priorität); Zweck: M,A,B,C und PRIOR dienen als Steuerinformation für die Hardware-Prozeß-steuerung, 0 ≤ PRIOR F ≤ (hexa).

TPW 1.....tablepointerword 1 (Tafelzeigerwort 1); Zweck: Bezug zur Parametertafel eines Programms (s. Kapitel 3.1.6)

CAUPRB....causing PRB (Adresse des verursachten Prozeßblocks bei Programmlaufbesonderheit); Zweck: Bezug zum Programm, welches eine Programmlaufbesonderheit (PLB) verursacht hat.

Zustandsbits:

Sie steuern das Ein- und Aushängen des Prozeßblocks sowie den Umfang der Rett-/Laderoutine. Im einzelnen haben sie folgende Bedeutung:

o B (busy bit, Tätig-Bit)
Dieses Bit steuert den Umfang der Rett-/Laderoutine, d. h. es gibt an, ob der Prozeß beim Start bzw. bei Unterbrechung tätig war. Ein nicht mehr tätiger bzw. noch nicht tätiger Prozeß braucht nicht die vollständige Rett-/Laderoutine zu durchlaufen, wodurch Prozeßwechselroutinen ggf. beschleunigt werden. Es ist gesetzt, wenn der Prozeß an vorderster Stelle der Prioritätswarteschlange ist, d. h. wenn ihm der Zentralprozessor zugeteilt ist.

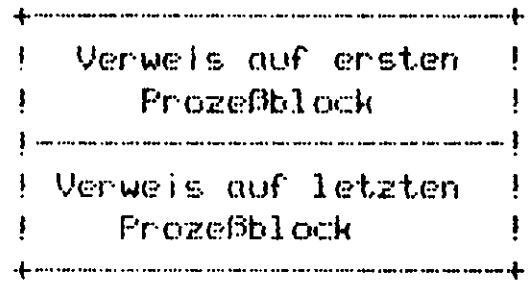
o A (active bit, Aktivierungs-Bit)
Dieses Bit ist gesetzt, wenn der Prozeßblock in der betreffenden Ablauf-warteschlange eingehängt ist, d.h. aktiviert ist.

o M (multiple bit, Mehrfach-Anforderungs-Bit)
Zusammen mit dem A-Bit wird hierfür das Problem der Mehrfachanforderung fensterfrei gelöst. Trifft ein Interrupt auf einen schon aktivierten Prozeßblock (A-Bit = 1), so wird das M-Bit gesetzt. Ist das M-Bit ge-setzt, wird der Prozeß durch den Befehl "deaktiviere aktuellen Prozeß" nicht deaktiviert, sondern er gelangt nochmals zum Ablauf. Ist das M-Bit nicht gesetzt, wird ein Prozeß durch diesen Befehl beendet, d.h. aus der Prioritätswarteschlange ausgehängt. Durch den Befehl "deaktiviere fremden Prozeß" wird auch ein mehrfach angeforderter Prozeß gänzlich deaktiviert.

o C (check-bit, Prüfbit)
Falls gesetzt, führt jede Bearbeitung dieses Prozeßblocks (aktivieren, deaktivieren, tätig werden, untätig werden) zu einer Meldung, welche die Zustandsänderung des Prozesses kennzeichnet. Diese Meldung wird z. B. von der Testanschaltung TESTAS zu Test- und Prüfzwecken verwendet.

3.1.5.4 Aufbau der Warteschlangenköpfe

Je Prioritätsebene (0-15) wird ein Warteschlangenkopf geführt, der aus zwei Zellen besteht. Die Zellenbelegung ist wie folgt definiert:



Die Warteschlangen sind doppelt gefädelt, d. h. die erste Zelle des Warteschlangenkopfs verweist auf die Anfangsadresse des ersten Prozeßblocks, die zweite Zelle des Warteschlangenkopfs verweist auf die Anfangsadresse des in der Warteschlange an letzter Stelle stehenden Prozeßblocks.

Falls in einer Warteschlange kein Prozeßblock eingehängt ist, verweist die erste Zelle des Warteschlangenkopfs auf sich selbst und die zweite Zelle auf die erste Zelle. Die Warteschlangenköpfe stehen in den Zellen 0 bis 31 des Zentralspeichers (Kapitel 3.1.7).

3.1.6 Parametertafel (PT)

Die Parametertafel dient dazu, alle für den Progammlauf der in Ebene 0-15 laufenden Programme notwendigen Informationen aufzunehmen. Es sind dies:

- die Inhalte der Standardregister im Normal- und Simulationsmodus
- die Inhalte der programmabhängigen Spezialregister
- Software-Informationen

Die Anfangsadresse einer Parametertafel ist über das Tafelzeigerwort 1 (TZW 1) bestimmt. Das 16-bit-breite TZW 1 (Zelle 3 im Prozeßblock wird unterhalb seines niederwertigsten Bits durch 6 Bits mit dem Wert "0" ergänzt, woraus sich die 22-bit-breite reelle Anfangsadresse der Parametertafel ergibt (s. 3.1.8.1).

Anfangsadresse der Parametertafel

Die Parametertafel kann somit im 64-Wort-Raster im Zentralspeicher abgelegt werden; die Parametertafel ist 128 byte lang und folgendermaßen belegt (Hardware- und Software-Zellen, Bild 3.8):

Zelle (hexa)				Zelle (dezimal)
00	PZW	Programmzustandswort		0
01				1
02	UAW	Unterbrechungsanzeigenwort		2
03	BAP/TZR3	Befehlsadressenpufferwort für TZR3		3
04	BAP/R1	Befehlsadressenpufferwort für R1		4
05	AWW	Adressierungsweichenwort		5
06	TZW2	Tafelzeigerwort 2		6
07	PLW	Programmlaufzeitwort		7
08	TZW3	Tafelzeigerwort 3		8
09	BAP/NNN	Befehlsadressenpufferwort bei NNN		9
0A		Länge des V-Teils		10
0B		Nr. der nicht geladenen Seite		11
0C		Länge des I-Teils		12
0D				13
0E		Anfangsadresse des V-Teils		14
0F	PLB-PRB	Verweis auf den PLB-PRB		15
10		Startadresse Normalmodus		16
11		VIBIAS-online-Testhilfe		17
12		Anfangsadresse des I-Teils		18
13		Wiederanlaufkennung		19
14		Adr. der Sprungleiste für Sonderroutinen		20
15		Rettzelle f. AWR bei Simulationen D,K		21
16		Rettzelle f. TZR bei Simulationen D,K		22
17		Länge des V-Teils des SCC		23
18		PRB Zeitscheibenzähler		24
19		Verweilzeit in 1/10 ms		25
1A		Zwischengespeicherter PLZ		26
1B		CPU-Zuteilzähler		27
1C		PRB-Zeitscheibenwert in Einheiten d. Zeitbasis		28
1D		Adr. f. Testpunktbearbeitung d. Anwenders		29
1E		Länge des V-Teils ohne LV-Angabe		30
1F		Überwachungsinformation		31
20		Standardregister 0		32
21		Standardregister 1		33
..			Normal-	..
..			modus	..
..				..
2F		Standardregister 15		47
30		Standardregister 0		48
31		Standardregister 1		49
..			Simulations-	..
..			modus	..
..				..
3F		Standardregister 15		63

Bild 3.8: Parametertafel

Die Bedeutung der hardware-relevanten Zellen ist folgende:

a)PZW Programmzustandswort (alle Bitnummern SIBI)
Das PZW nimmt bei einem Prozeßwechsel den Inhalt des Programmzustandsregisters PZR auf.

Bit 0,1 Ergebnisanzeigen

Bit 2 Übertrag

Bit 3 Prozeßwechselsperre, schützt gegen Prozeßwechsel wegenInterrupt (IR) oder Prozeßaktivierungsbefehlen (APZ,APF , sperren des PLZ).

Bit 4 Maske für Betragsüberlauf, bei gesetztem Bit führt der Betragsüberlauf zu einer Programmlaufbesonderheit (PLB) und einem UAR-Eintrag (UAR...Unterbrechungsanzeigenregister).

Bit 5 Maske für Festpunktüberlauf, bei gesetztem Bit führt der Festpunktüberlauf zu einer Programmlaufbesonderheit(PLB) und einem UAR-Eintrag.

Bit 7 Schreibschutz
Im nichtprivilegierten Modus ist der virtuelle Schreib-schutz in der Übersetzungstafel (ÜT) wirksam. Im privilegierten Modus schaltet das Bit 7 des Programmzustandsworts die Funktion des Schreibschutzes in die ÜT ein(Bit 7 = 1) oder aus (Bit 7 = 0). Bei eingeschaltetemSchreibschutz sind die ersten 128 Worte jedes virtuellenAdreßraums ebenfalls schreibgeschützt (Parametertafelund Übersetzungstafel 1 zur Adressierung von Daten).

Bit 8 Programmtestmodus
Wenn das Bit 8 im PZW gesetzt ist und der Rechner sichnicht im Simulationsmodus befindet, erfolgt am Ende jedes Befehls ein UAR-Eintrag.Testprogramme nutzen diese Funktion für den Befehls-trace.

Bit 10 nicht privilegierter Modus
Im nicht privilegierten Modus (Bit 10 = 1) ist der Schreibschutz automatisch eingeschaltet.
Im Simulationsmodus hat das Bit 10 keine Bedeutung.
Im nichtprivilegierten Zustand wird die Ausführung einiger Befehle mit dem UAR-Eintrag (Unterbrechungs-anzeigenregister) "Privilegverletzung" abgewiesen.
Das Unterbrechungsanzeigenregister UAR ist als Unter-brechungsanzeigenwort UAW in der Zelle 2 der Parameter-tafel enthalten (s. Punkt b). Die priviligierten Be-fehle sind:

- aktiviere Prozeß APZ, APF
- deaktiviere Prozeß DAP, DFP
- STOP

Bit 12 Simulationsmodus
Der Zentralprozessor zeigt durch das Bit 12 des Programmzustandsregisters (PZR) an, ob er sich im Normal- oder im Simulationsmodus befindet (Bit 12 = 1 bedeutet Simulationsmodus).
Der Simulationsmodus eines Prozesses kann nur über einen NNN erreicht werden. Zum Verlassen des Simulationsmodus steht der Befehl SPS (Springe aus dem Simulationsmodus) zur Verfügung. Das Bit 12 des PZR darf befehlsgesteuert (Spezialregisterbefehle) weder gesetzt noch gelöscht werden.

Bit 13 Programmlaufzeitzähler (PLZ) - Freigabe
Der Programmlaufzeitzähler begrenzt die Laufzeit eines Programms auf ca. 7 s. Danach erzeugt er, falls dieses Bit gesetzt ist, einen UAR- Eintrag (Bit 13).

Der PLZ läuft nicht im Simulationsmodus eines Prozesses, und bei PZR Bit 3 gesetzt.

Die exakte Laufzeit des PLZ kann nicht angegeben werden, weil er mit dem Grundtakt T0 betrieben wird.
Dieser Grundtakt ist, abhängig von den gerade bearbeiteten Befehlen und anderen Ereignissen (Adreßübersetzung), variabel.

Bit 14 Virtuelle Adressierung der Befehle
In der ZE 03 gibt das Bit 14 des PZR die virtuelle Adressierung der Befehle über die Übersetzungstafel ÜT3 frei.

Im Simulationsmodus laufen die Befehle grundsätzlich reell ab.

Bit 15 Virtuelle Adressierung der Operanden
Das Bit 15 des PZR gibt die virtuelle Adressierung der Operanden über die Übersetzungstafel ÜT1 frei, falls das Adressierungsweichenregister (AWR) keine Übersetzung über die Übersetzungstafel ÜT2 erzwingt.

b) UAW — Unterbrechungsanzeigenwort (alle Bitnummern SIBI)
Das UAW nimmt bei einem Prozeßwechsel den Inhalt des Unterbrechungsanzeigenregisters (UAR) auf.

Bit 0 — Rufen des Primärzustands RPZ
Mit dem Befehl RPZ kann eine Programmlaufbesonderheit und damit ein UAR-Eintrag programmiert werden.

Bit 2 — Seitenanforderung
Wenn ein Programm auf eine virtuelle Seite zugreift, die in der Übersetzungstafel ÜT nicht geladen ist, erfolgt ein Eintrag in das Bit 2 des UAR und die Programmlaufbesonderheit (PLB)-Berarbeitung.

Bit 4 — Befehlsendemodus
Wenn durch das PZR Bit 8 der Programmtestmodus freigegeben ist, erfolgt am Ende jedes Befehls ein Eintrag in das Bit 4 des UAR und die PLB-Bearbeitung.

Bit 5 — Überlauf
Die Division durch 0 und der Exponentenüberlauf bei Gleitpunktoperationen führen unmittelbar zum Setzen des UAR Bit 5.

Der Fehler Betrags- und Festpunktüberlauf führen nur zum Setzen des UAR Bit 5, wenn die Freigabe über die PZR-Bits 4 bzw. 5 gegeben ist.

Bit 9 — Nicht interpretierbarer Befehl (NNN)
Das Auftreten eines NNN bewirkt den Wechsel vom Normalmodus in den Simulationsmodus.

Befindet sich der Zentralprozessor schon im Simulationsmodus und es tritt wieder ein NNN auf, kommt es zum Eintrag in das UAR-Bit 9.

Bit 10 — Schreibschutzverstoß
Wenn bei eingeschaltetem Schreibschutz (PZR-Bit 7 = 1 oder PZR-Bit 10 = 1) versucht wird, in eine geschützte Seite (Schreibschutz S-Bit in der ÜT = 1) oder in ein Wort mit einer Adresse zwischen 0 und 127 eines virtuellen Adreßraums zu schreiben, reagiert der Zentralprozessor mit dem UAR-Bit 10.

Bit 11 — Privilegverletzung
Im nichtprivilegierten Modus (PZR-Bit 10 = 1) sind folgende Befehle verboten:

Befehle
- APZ, APF — Prozeß aktivieren
- DAP, DFP — Prozeß deaktivieren
- STOP

Verletzungen dieser Verbote führen zum Setzen des UAR-Bit 11.

Bit 12 — Adreßgleichheit
Das UAW-Bit 12 wird nicht vom Zentralprozessor, sondern vom Festspeicherprogramm der virtuellen Konsole gesetzt. Es wird benutzt, um die Gleichheit einer Adresse auf dem Bus mit einer vorgebbaren Adresse zu kennzeichnen (Kapitel 4.3). Außerdem wird diese Funktion von einigen hardwarenahen Testprogrammen zur Software-Wartung benutzt.

Bit 13 — Programmlaufzeitzähler (PLZ)-Überlauf
Der PLZ verursacht nach einer Laufzeit von ca. 7 s eine Programmlaufbesonderheit mit UAR-Bit 13 = 1. Die exakte Laufzeit kann nicht vorherbestimmt werden, weil der Zähltakt (T0) variabel (befehlsabhängig) ist.

Der PLZ kann nicht zu Laufzeitmessungen herangezogen werden, er begrenzt vielmehr fehlerhafte Programmabläufe (z. B. Endlosschleife).

Der PLZ läuft, wenn die Freigabe über das PZR-Bit 13 gegeben ist.

Bit 14 — Datenfehler
Ein Speicherzugriff (Lesen) wurde mit einem Datenfehler quittiert (FFD = 1).
Das Datum im Zentralspeicher enthält einen nicht korrigierbaren Mehrbitfehler, d. h. zwei oder mehr Bits des Datums werden von der Zentralspeicher-Korrekturlogik als falsch erkannt.

Bit 15 — Adressierfehler
Ein Speicherzugriff (Lesen/Schreiben) wurde mit Adressierfehler quittiert (FFA = 1). Gründe dafür können sein:

1. Schreiben in den Festwertspeicher
2. Zugriff auf eine nicht vorhandene Zentralspeicher-Zelle

c) BAP/TZR3 — Nach Unterbrechungen (PLB, NMI) des Programmablaufs wird das Tafelzeigeregister TZR3 in diese Zelle gerettet. Nur bei Unterbrechungen unmittelbar nach den Befehlen UCK und SPC (Sprung bzw. Rücksprung in/aus fremden Code-Adreßraum) weicht der Inhalt des Worts 3 der Parametertafel vom Inhalt des Worts 8 (TZW3) ab. Der Tafelzeiger 3 (s. Punkt h) ist schon umgeladen, der zuletzt ausgeführte Befehl steht aber in einem anderen Adreßraum.

d) BAP/R1 — Adresse des Befehls, der vor einer Unterbrechung (PLB, NMI) als letzter ausgeführt wurde.

e) AWW

Adressierungsweichenwort
Gesetzte Bits im AWW veranlassen den Zentralprozessor bei Speicherzugriffen über ein Register (indirekte Adressierung, indizierte Adressierung) die Adreßübersetzung über die Übersetzungstafel 2 vorzunehmen.

Die Bitnummern im AWW (SIBI) entsprechen der Registernummer.

f) TZW2

Tafelzeigerwort 2
Der Tafelzeiger 2 adressiert die Übersetzungstafel ÜT2. Adreßübersetzungen über die ÜT2 werden durchgeführt, wenn für das Register, über das indirekt oder indiziert adressiert wird, im Adressierungsweichenregister das zugehörige Bit gesetzt ist.

g) PLW

Programmlaufzeitwort
Nach der Unterbrechung eines Programms (PLB, NMI, IR) wird der aktuelle Wert des Programmlaufzeitzählers PLZ in das PLW gerettet.
Mit der Fortsetzung eines Programms wird der PLZ aus dem PLW geladen.
PLW = 0 begrenzt die Programmlaufzeit auf ca. 7 s.

h) TZW3

Tafelzeigerwort 3
Der Tafelzeiger 3 adressiert die Übersetzungstafel ÜT3. Die Übersetzung der virtuellen Befehlsadresse erfolgt grundsätzlich über die ÜT3.

i) BAP/NNN

Befehlsadreßpuffer bei einem NNN
Der Zentralprozessor wechselt nach der Bearbeitung eines nicht interpretierbaren Befehls (NNN) in den Simulationsmodus. Die Adresse des verursachenden NNN's hinterlegt er im Wort Nr.9 der Parametertafel. Die Operationscodelänge des NNN spielt dabei keine Rolle.
Das R1 (Zelle 21 der Parametertafel) des Normalmodus ist nach einem NNN undefiniert; es muß von der Routine im Simulationsmodus errechnet und in die Parametertafel eingetragen werden:
R1(NM) = BAP/NNN + Op-Codelänge des NNN

j) PLB-PRB

Programmlaufbesonderheiten-Prozeßblock
Programmlaufbesonderheiten (Anzeige im UAR) veranlassen den Zentralprozessor, den Prozeßblock zu aktivieren, der in der Zelle 15 der Parametertafel hinterlegt ist.
Die Adresse des Verursacher-Prozeßblocks wird in die Zelle "Adresse des rufenden PRB bei PLB" des Reaktionsprozeßblocks eingetragen (Verweis auf den Verursacher).
Besonderheiten: Wenn die Zelle PLB-PRB in der Para-

metertafel Null ist, kommt es zu einer ZO-internen Unterbrechung. Wenn der Reaktionsprozeß nicht höherprior als der Verursacher ist, kommt es ebenfalls zu einer ZO-internen Unterbrechung.

k) Standardregister Normalmodus
Satz von 15 gleichberechtigten Registern, die für entsprechende Befehle zu Registerpaaren oder -quartetten zusammengefaßt werden kann.
Das Register R1 dient als Befehlsadreßregister.

l) Standardregister Simulationsmodus
Register, die im Simulationsmodus relevant sind, mit der gleichen Bedeutung und Funktion wie die Register im Normalmodus.

In den Zentraleinheiten der Modellereihe SICOMP ist ein Registersatz hardwaremäßig im Zentralprozessor realisiert. Ein Wechsel vom Normal- in den Simulationsmodus oder umgekehrt sowie ein Prozeßwechsel enthalten das Retten bzw. Laden dieses Registersatzes aus der Parametertafel.

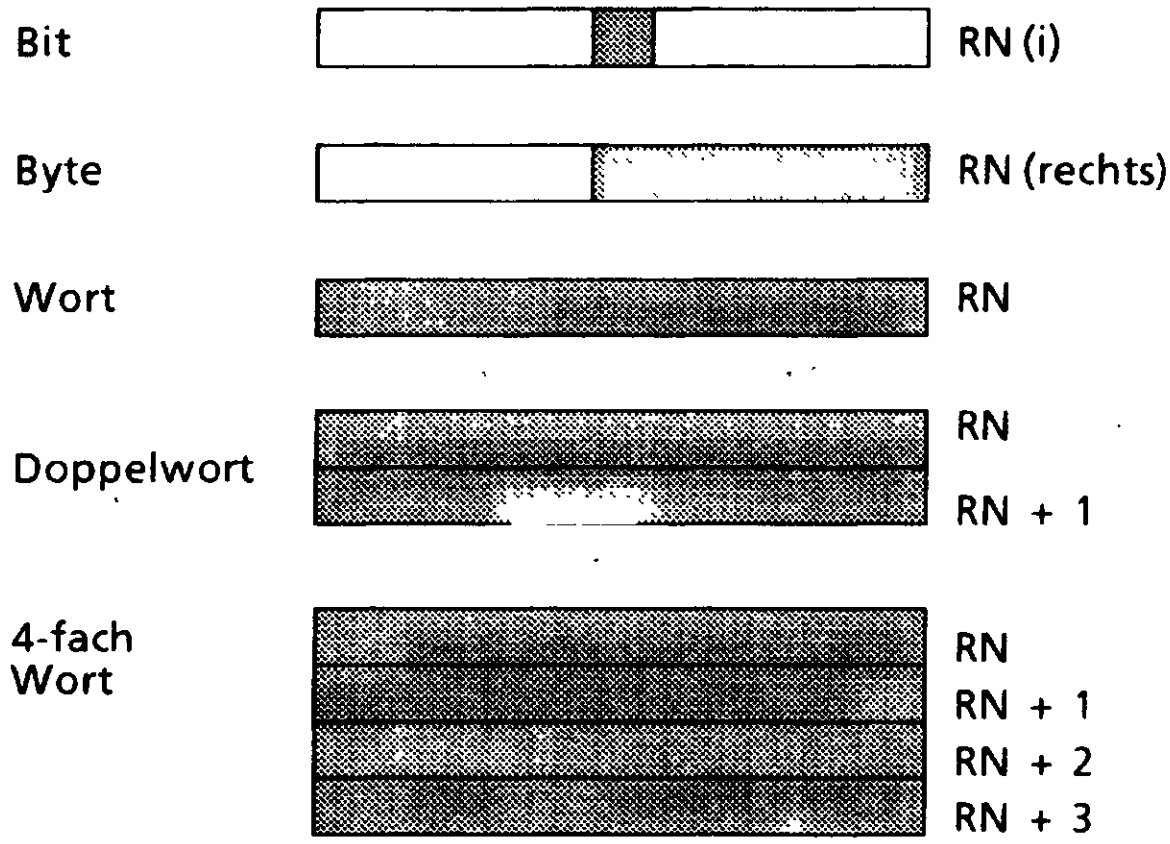


Bild 3.9: Verwendbarkeit der Standardregister

3.1.7 Hardware-Verständigungsbereich

In diesem Kapitel sind die Zentralspeicherzellen aufgelistet, die von der Hardware, dem Festspeicherprogramm, sowie der Systemsoftware zur Hinterlegung variabler Daten benötigt werden.

Dieser Speicherbereich, der die Speicherzellen mit den physikalischen Adressen 000000H bis 0005FFH umfaßt, wird als HW-VB (Hardware-Verständigungsbereich) bezeichnet. Außer dem Festspeicherprogramm, Hardware-Testprogrammen (z. B. ZEWAM), Systemprogrammen (z. B. ORG) und der Peripherie darf niemand schreibend auf diesen Bereich zugreifen.

Die Zuordnung der nachfolgend mit einem symbolischen Namen charakterisierten Speicherzellen zur physikalischen Adresse kann der Speicherbelegungstafel entnommen werden (Bild 3.10).

Anf.-Adres. hexa	Anf.-Adres. dez.	Bezeichnung Aufteilung	Inhalt	Länge (Zellen)
0000	0	PTZWS		32
:	:		>> Warteschlangen-	
001F	31		köpfe	
0020	32	PRBNOP	NOP-Prozeßblock	16
0030	48	PRBBAS	Basis-Prozeßblock	16
0040	64	PRBERR	Worst-case-PRB	16
0050	80	PRBVKX	VC-Pozeßblock	16
0060	96	PRBSOND	Sonder-PRB	16
0070	112	PRBSPAS	SPAS-Prozeßblock	16
0080	128	PRBNAUW	Netzausfallwarnung	16
0090	144	PRBX1	-PRB	16
00A0	160	PRBX2	>> reserviert	16
00B0	176	PRBX3		16
00C0	192	PRBZIG	Zeitgeber-PRB	16
00D0	208		reserviert	10
00DA	218	PHYSPBZIG	Zeitgeber-PHYSPB	18
00EC	236		reserviert	4
00F0	240	PRBSTRT	Startprozeßblock	14
00FE	254	NMIZIEL	Ziel-Adr. f. NMI	1
00FF	255	NMIWERT	Zusatz-NMI's	1
0100	250	VEKLI	PRBNOP-Adr. (32)	256
0101			PRBBAS-Adr. (48)	
0102			PRBERR-Adr. (64)	
0103			PRBVKX-Adr. (80)	
0104			PRBSOND-Adr. (96)	
0105			PRBSPAS-Ad. (112)	
0106			PRBNAUW-Ad. (128)	
0107			PRBX1-Adr. (144)	
0108			PRBX2-Adr. (160)	
0109			PRBX3-Adr. (176)	

Anf.-Adres. hexa	Anf.-Adres. dez.	Bezeichnung Aufteilung	Inhalt	Länge (Zellen)
010A			PREZIG-Adr. (192)	
010B			reserviert	
010F				
0110			reserviert für Spannungwieder-	
0117			kehr E-Rahmen	
0118			reserviert für Spannungsausfall	
011F			E-Rahmen	
0120			frei	
01FF				
0200	512	BUCH	Buchführung	256
0300	768	VORSPA	Vorspann	256
0304		PRONAX	Hilfspuffer für	
0306			Programmnamen	
0340		URLUFT	Übersetzungstafel	
037F				
0400	1024	MAKENN	Maschinenkennung	1
0401	1025	VKGERA	VC-EA-Adresse	1
0402	1026	PRGERA	BT-EA-Adresse	1
0403	1027	PRGERA	LIST-EA-Adresse	1
0404	1028	MDGERA	sonst. EA-Adresse	1
0405	1029	URDIP	BT-Codierschalter	1
0406	1030	LAREG	Lampenregister	1
0407	1031	WZBAS	Status-Zelle	1
0408	1032	ANLAWE	Anlaufsteuerung	1
0409	1033	VKGECOP	Kopie von VKGERA	1
040A	1034	VKGERES	VK-Reserve	1
040B	1035	SPGERA	SPAS-EA-Adresse	1
040C	1036	BTRESE	RESE f. Urlader	1
040D	1037	PASSWORD	Password	3
040F	1039			
0410	1040	PRONA	Programmname	3
0413	1043	ASTADR	Ausgabestand	1
0414	1044	Startdaten aktuelles Programm	Start-R1	1
0415	1045		Restart-R1	1
0416	1046		PF-R1	1
0417	1047		TZR	1
0418	1048	COPSTALI	Startdaten 1. Prog	40
041F	1055			

Anf.-Adres. hexa	Anf.-Adres. dez.	Bezeichnung Aufteilung	Inhalt	Länge (Zellen)
0420	1056		–	
:	:		>> Startdaten 2. Prog:	
0427	1063		–	
0428	1064		–	
:	:		>> Startdaten 3. Prog:	
042F	1071		–	
0430	1072		–	
:	:		>> Startdaten 4. Prog:	
0437	1079		–	
0438	1080		–	
:	:		>> Startdaten 5. Prog:	
043F	1087		–	
0440	1088	VOAIVI	–	3
0443	1091	AKAIAN		6
0449	1097	QKOPF		1
044A	1098	QAIRV		1
044B	1099	QAIRB		1
044C	1100	QZAEHL		1
044D	1101	RANZA		1
044E	1102	AKANZ		1
044F	1003	BAIRNE	>> Hilfszellen	2
0451	1105	BAIRPO		2
0453	1107	FORFUF		8
0458	1115	INOFUF		7
0462	1122	SEAFUF		16
0472	1138	SLCOPJ		8
047A	1146	BURETT		2
047C	1148	RESULT		2
047E	1150	OFFWERT		1
047F	1151	YESAIR	–	1
0480	1152	MERR0	–	1
0481	1153	MERR1	>> Hilfszellen	1
0482	1154	FEPU	–	8
048A	1162	URAIR	UR-Hilfszelle	1
048B	1163	ZZRAIR	ZZR-Hilfszelle	1
048C	1164	UROLD	gerettetes UR	1
048D	1165	ZZROLD	gerettetes ZZR	1
048E	1166	TERAM	Hilfszeslle	1
048F	1167		reserviert	1
0490	1168	PARBUF	Parameterbuffer	40
04B8	1208	INOBUFF	Sterne auf	2
			Bedienung	
04BA	1210	INBUFF	Bedienpuffer	40
04E2	1250	OUTBUFF	Standardausgabe-	40
			puffer	

Anf.-Adres. hexa	Anf.-Adres. dez.	Bezeichnung Aufteilung		Inhalt	Länge (Zellen)
04F8	1272	PHYBAS		Physikalischer Parameterblock	18
050A	1290	ERRBUFF		Fehlermeldungs-puffer	38
0530	1328	PHYSPB		Physikalischer Parameterblock	18
0542	1346	NMIADR		reserviert	2
0544	1348	ZOADR		reserviert	2
0546	1350	MEGRUP	-		1
0547	1351	MECHEP	!		1
0548	1352	TPZHLNUM	!		1
0549	1353	TPZHLCOU	!		1
054A	1354	TPSHOWRE	!	Hilfszellen	1
054B	1355	TPFUMERE	>>	für	1
054C	1356	TPMERKRE	!	Checkpoints	1
054D	1357	TPSPAS	!		1
054E	1358	TPSPASA	!		1
054F	1359	TPMERKER	-		1
0550	1360	AKTTSFUME	!	Funktionsmerker	1
0551	1361	AKTTSSHOW	!	Registerkennung	1
0552	1362	AKTTSNUMB	!	Param. NUMBER	1
0553	1363	AKTTSLEVE	!	Param. LEVEL	1
0554	1364	AKTTSMASK	!	Param. MASK	1
0555	1365	AKTTSVALU	!	Param. VALUE	1
0556	1366	AKTTSCOUN	!	Param. COUNTER	1
0557	1367	AKTTSXXXX	!	reserviert	1
0558	1368	AKTTH1PAR	!	HW-Parameter 1	1
0559	1369	AKTTH2PAR	!	HW-Parameter 2	1
055A	1370	TPP0	-	Checkpoint 0	10
:	:		>>	(Aufbau wie	:
0563	1379		-	AKT...)	
0564	1380	TPP1	-	Checkpoint 1	10
:	:		>>	(Aufbau wie	:
056D	1389		-	AKT...)	
056E	1390	TPP2	-	Checkpoint 2	10
:	:		>>	(Aufbau wie	:
0577	1399		-	AKT...)	
0578	1400	TPP3	-	Checkpoint 3	10
:	:		>>	(Aufbau wie	:
0581	1409		-	AKT...)	

Anf.-Adres. hexa	Anf.-Adres. dez.	Bezeichnung Aufteilung	Inhalt	Länge (Zellen)
0582	1410		--	
	:		>> Keller	: 122
05FC	1532		--	
05FC	1532	STACK	--	1
05FD	1533	TEIUZ	>> Zellen für ZE-Testprogramme	1
05FE	1534	FEZEI		1
05FF	1535	FEZHL	--	1
0600	1536	STEUZ	Steuerzellen für ZE-Testprogramme	16
0610	1552	FREEMEM	Fehlerpuffer für ZE-Testprogramme	n
0600	1536	ab hier:	frei	

1 Zelle = 16 bit

Bild 3.10: Speicherbelegungstafel

Im folgenden werden einige wichtige Elemente dieses Bereichs beschrieben:

PTZWS
Warteschlangenköpfe der 16 Prioritätszustände. Jeder Kopf enthält zwei Zellen:

- Vorwärtsverweisadresse
- Rückwärtsverweisadresse.

PRBNOP
Prozeßblock für Sonderzustände

PRBBAS
Basisprozeßblock. Nach dem Rücksetzen der Anlage sind alle peripheren Geräte mit Ausnahme der virtuellen Konsole, des Zeitgebers und der Serviceprozessoranschaltung auf diesen Prozeßblock eingestellt. Dieser PRB steht dem Anwender zur Verfügung (beispielsweise zur Initiierung peripherer Geräte).

PRBERR
Fehlerprozeßblock für die peripheren Geräte, falls sie sich nicht über ihren eigenen Geräte-Prozeßblock mit der Zentraleinheit verständigen können.

PRBVKX
Prozeßblock für die virtuelle Konsole

PRBSOND
Sonderprozeßblock

PRBSPAS
Prozeßblock für die Serviceprozessoranschaltung

PRBNAUW
Prozeßblock für Netzausfallwarnung

PRBX1 ... PRBX3
Drei für Systemprogramme reservierte Prozeßblöcke

PRBZIG
Prozeßblock für den Zeitgeber

PHYSPBZIG
PHYSPB für die Zeitgeberaufträge

UHRZT
4 Zellen, in denen die Uhrzeit geführt wird

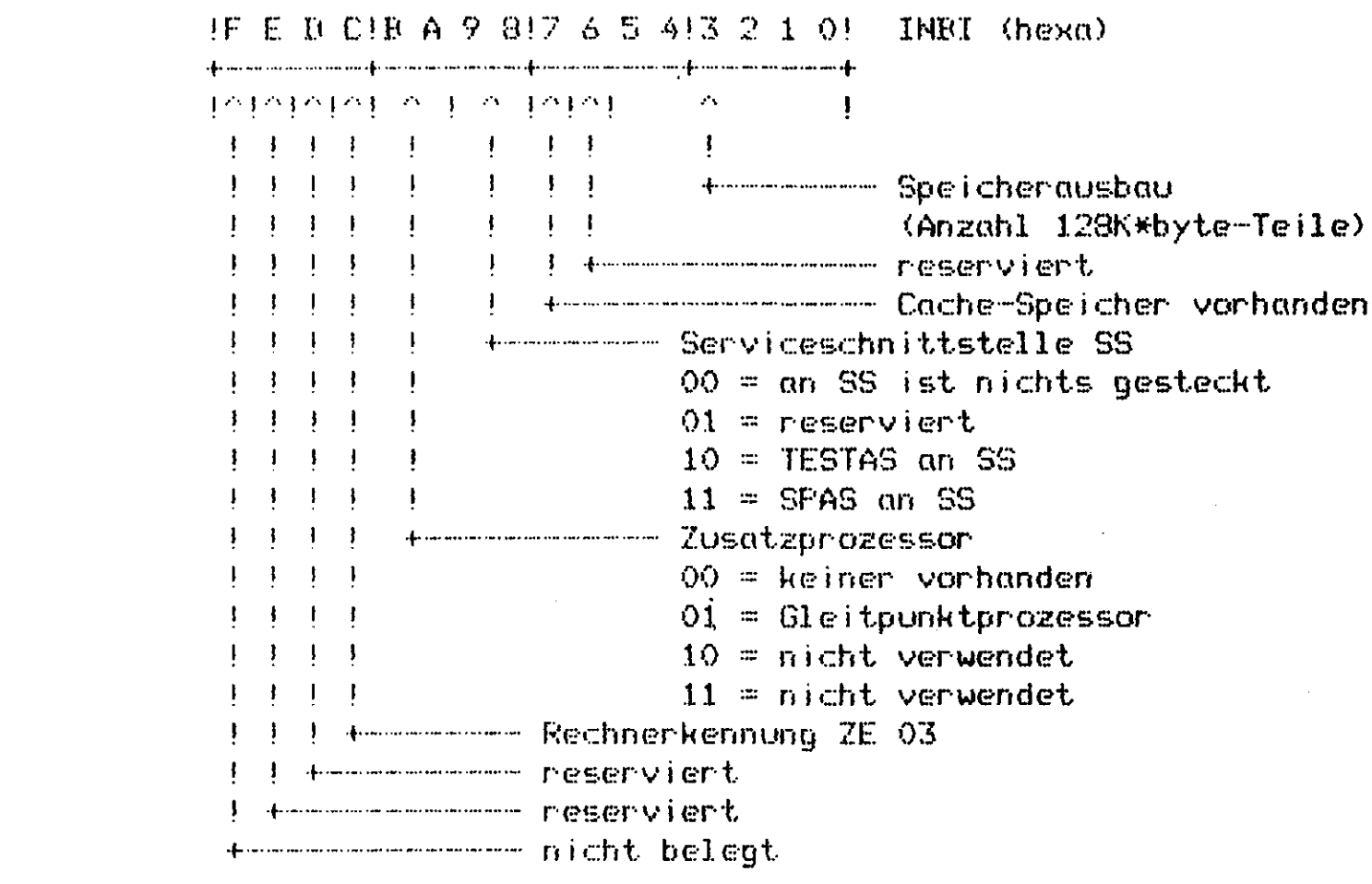
PRBSTRT
Startprozeßblock; über ihn werden urgeladene Programme gestartet.

VEKLI
256 Zellen umfassende Verweisliste, die den EA-Vektoren 0 ... 255 eine 16-bit-breite reelle Adresse zuordnet. Diese Adresse weist auf den jeweils zugehörigen Prozeßblock PRB. Die Vektoren 0 bis 10 werden durch das Festspeicherprogramm mit den entsprechenden PRB-Adressen geladen, während die übrigen Vektoren mit der Adresse des PRBBAS vorbesetzt werden.

BUCH
Enthält bei Random-Datenträgern in 256 Zellen die Buchführung mit den zum Laden erforderlichen Parametern aller auf dem Datenträger vorhandenen, urladefähigen Programme.

VORSPA
Enthält in 256 Zellen den Programmvorspann des zuletzt urgeladenen Programmblocks.

MAKENN
In dieser Zelle wird die Rechnerkennung und der Speicherausbau hinterlegt (Maschinenkennung):



VKGERA
Enthält die EA-Geräteadresse des aktuellen VK-Geräts. Nach Spannungs- und Pufferbatterie-Ausfall wird diese Zelle mit der über Referenzschalter eingestellten EA-Adresse der virtuellen Konsole geladen. In allen anderen Anlauffällen bleibt die Zelle unverändert. Ein Ändern der VK-Geräteadresse ist jederzeit durch Kommando möglich.

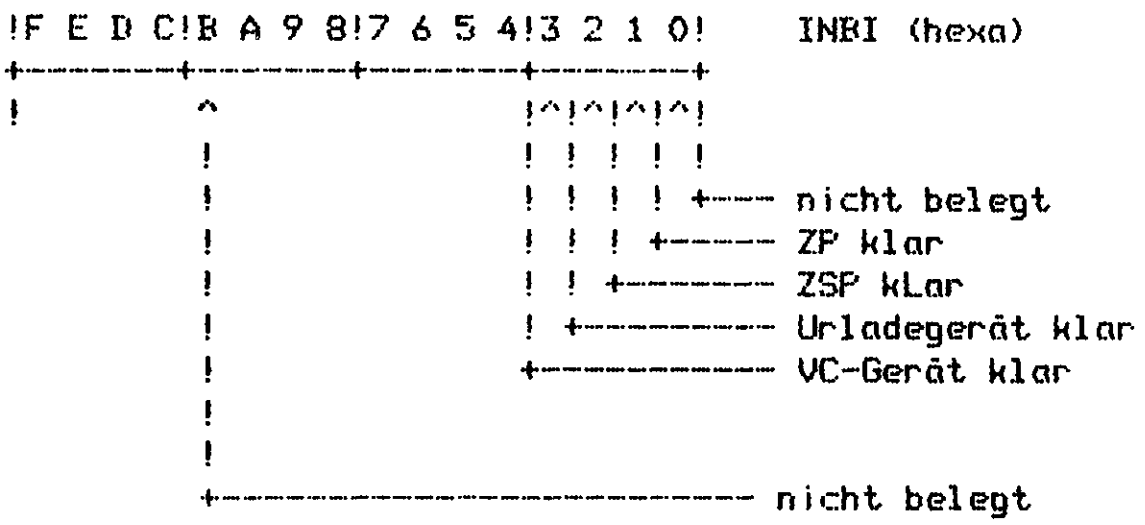
URGERA
Enthält die EA-Geräteadresse des aktuellen Urladegeräts. Sie wird bei jedem Urladen von Referenzschaltern auf der Zentralprozessor-Flachbaugruppe TC aktualisiert. Ist im Urladekommando keine EA-Geräteadresse angegeben, so enthält diese Zelle die über Referenzschalter voreingestellte Urlade-Geräteadresse. Wird der Urlader von Anwenderprogrammen benützt, bleibt die Zelle unverändert. Damit besteht für den Anwender die Möglichkeit, diese Zelle mit der von ihm gewünschten EA-Geräteadresse zu laden.

PRGERA
Enthält die EA-Geräteadresse des aktuellen Protokollgeräts. Sie wird über die virtuelle Konsole durch das Kommando LIST geladen und über das Kommando NOLIST gelöscht.

MDGERA
Enthält die EA-Geräteadresse des DUMP- bzw. MESSAGE-Zielgeräts (Meldegerät). Sie wird durch das entsprechende VK-Kommando eingetragen.

URDIP
Enthält die Information der DIP-Schalter für das Urladegerät.

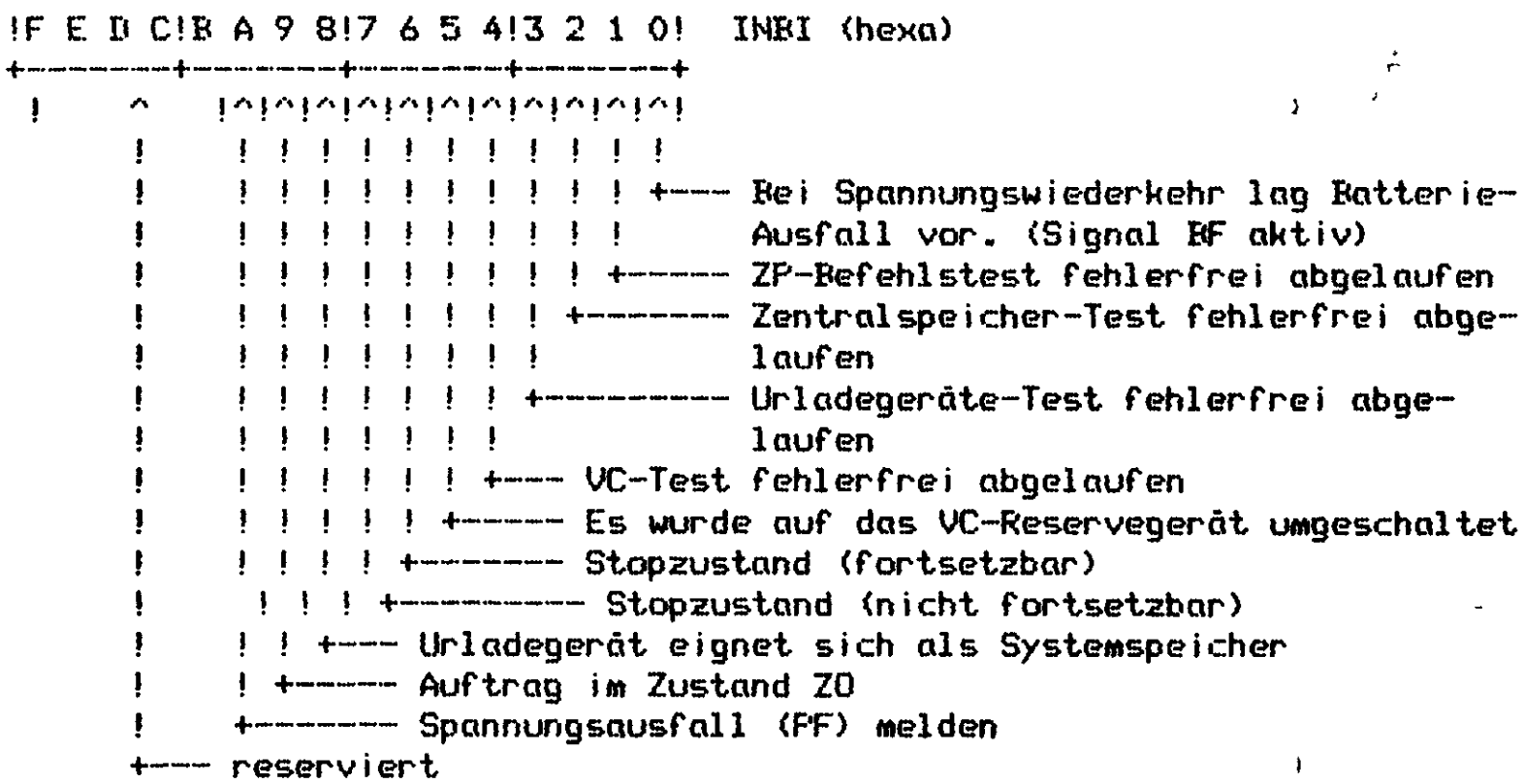
LAREG
Diese Zelle entspricht 1 : 1 dem Zustandslampenregister
(EA-Adresse 5). Dieses wird über LEDs am Systemcontainer ausgegeben.



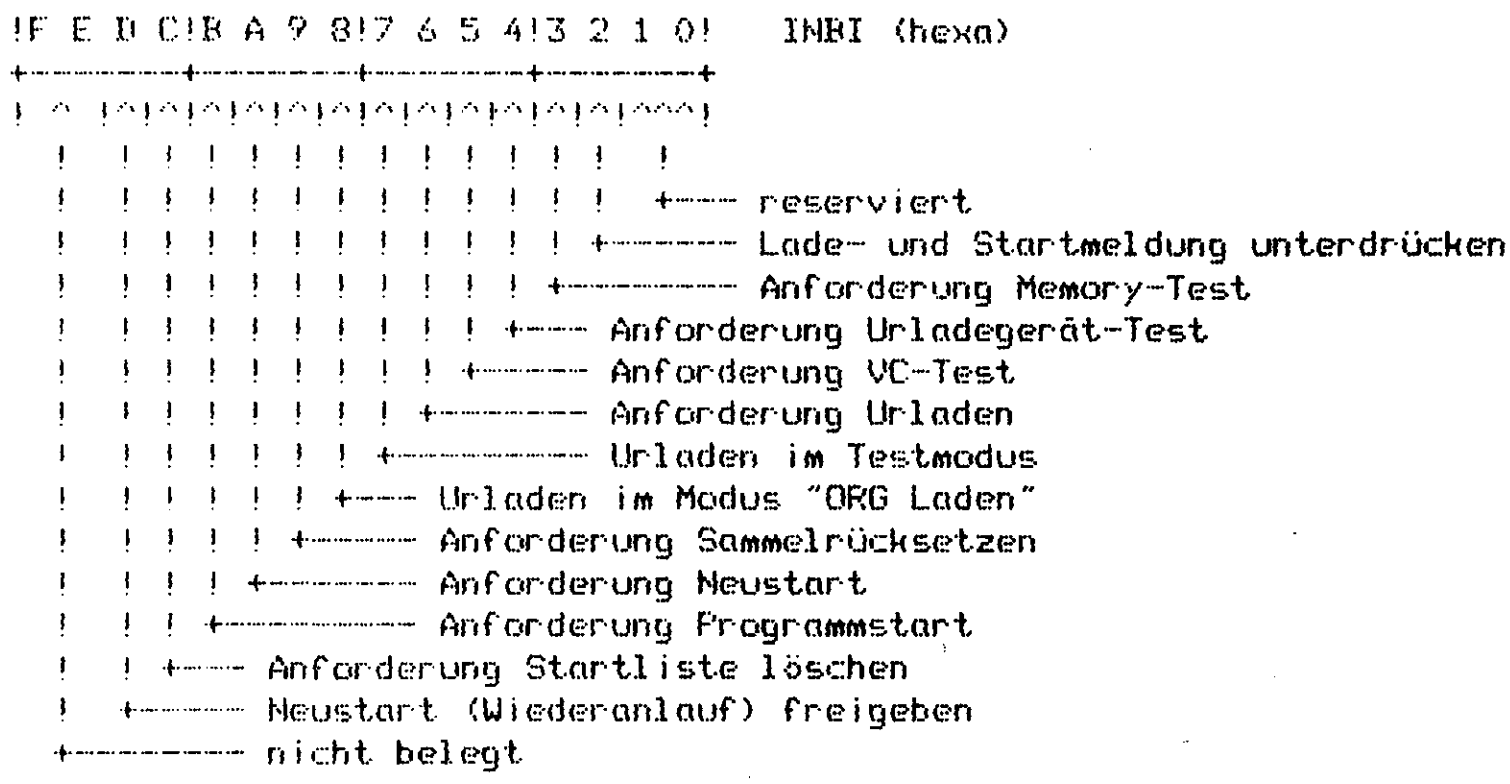
Besonderheit
Die nicht belegten Bits dürfen von Anwenderprogrammen zur Ausgabe an das Zustandslampenregister verwendet werden. Dabei müssen die folgenden Punkte beachtet werden:

- nur die INBI 0, 5 und 7 - 15 dürfen verändert werden,
- gewünschtes Bit in LAREG setzen bzw. löschen,
- Inhalt LAREG an das Zustandslampenregister ausgeben (EA-Adresse 5).

WZBAS
Diese Weichenzelle dokumentiert den aktuellen Zustand und die Vorgeschichte von Zentraleinheit (ZE), Urladegerät (BT) und virtueller Konsole (VK).



ANLAWE
Die Anlauf-Weichenzelle steuert den Anlauf in Abhängigkeit des vorliegenden NMI (s. Kapitel 3.1.9).



INTCON
Diese Zelle enthält Steuerbits für das Festspeicherprogramm VICOM.

PRONA
In drei Zellen ist der zur Identifikation des zuletzt gestarteten, urladefähigen Programms benötigte Programmname hinterlegt.

ASTADR
In diesen fünf Zellen sind der aktuelle Ausgabestand und die Startparameter des zuletzt gestarteten, urladefähigen Programms hinterlegt:

- Ausgabestand
- Programmstartadresse (START-R1)
- Neustart- (Wiederanlauf-) Adresse (RESTART-R1)
- Anfangsadresse der Netzausfallroutine (PF-R1)
- Tafelzeigeregister (TZR), das auf die Parametertafel des gestarteten Programms weist.

CODSTALI
40 Zellen umfassender Speicherbereich; enthält die Identifikations- und Startparameter der fünf zuletzt urgeladenen Programme.

VARDAT
Vom Festspeicherprogramm benutzte Hilfszellen.

PARPUF
Puffer zur Aufnahme von Parametern, die bei Lade- oder Startkommandos zur Übergabe an das zu startende Programm mit angegeben werden können. Dieser Puffer muß vom Anwenderprogramm ausgewertet werden.

PUFFER
Ein-/Ausgabepuffer für das Festspeicherprogramm.

PHYSPB
Physikalische Parameterblöcke zu den Transferprozeßblöcken des Festspeicherprogramms.

3.1.8 Adressierung

3.1.8.1 Adressierung des Zentralspeichers

Durch die 16-bit-Struktur des Rechners ist die Grenze für die Adressierung von Speicherzellen zunächst bei 64 x 1024 adressierbaren Einheiten vorgegeben, wobei eine solche Einheit ein Wort von 16 bit (= 2 byte) ist. Da dieser Speicherbereich im allgemeinen nicht ausreicht, wird eine Erweiterung des Adreßbereichs durch eine virtuelle Adressierung vorgenommen.

Bei dem komfortablen Adressierungsmechanismus der ZE 03 stehen jedem Programm mehrere unabhängige virtuelle Adreßräume von je 128 x 1024 byte zur Verfügung. Die Zuordnung der 16-bit-breiten virtuellen Adresse zur Zentralspeicheradresse erfolgt mit Hilfe von Übersetzungstafeln, die gemeinsam durch das Betriebssystem und die Hardware verwaltet werden.

Um auch verstreut liegende Bereiche im Zentralspeicher nutzen zu können, wird ein "seitenweises" Verfahren zur Adreßübersetzung angewendet. Der Adreßraum wird dazu in Bereiche von 1024 x 2 byte unterteilt, zu denen getrennt zugegriffen werden kann. Die endgültige reelle Speicheradresse ergibt sich durch Zusammenfügen eines Teils der virtuellen Adresse mit einem zweiten Teil, welcher der Übersetzungstafel entnommen wird.

Reelle Adressierung

Die 16-bit-Adresse wird nicht übersetzt, sie entspricht der Hardware-Adresse des Zentralspeichers und erfaßt die Speicherzellen 0 bis 65535. Dieser Art der Adressierung bleibt im wesentlichen dem Betriebssystem vorbehalten.

Virtuelle Adressierung

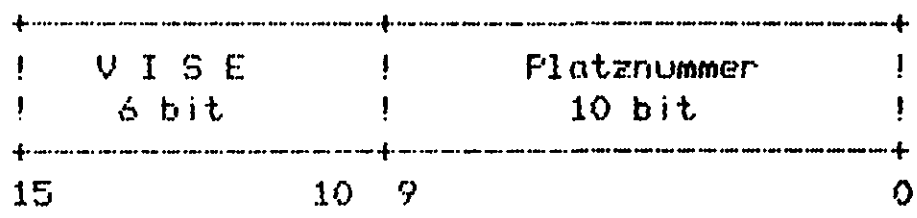
Bei dieser Adressierungsart wird die 16-bit-Adresse erst mit Hilfe von Übersetzungstafeln in eine Zentralspeicheradresse umgewandelt. Diese physikalische Adresse ist 22 bit lang und ermöglicht damit einen Zentralspeicherausbau von 8 M* byte. Die Unterscheidung, ob die 16-bit-Adresse virtuell oder reell zu interpretieren ist, wird durch die Bits 14 und 15 im Programmzustandsregister PZR und durch das Adressierungsweichenregister AWR festgelegt.

Adreßübersetzung mit Hilfe von Übersetzungstafeln

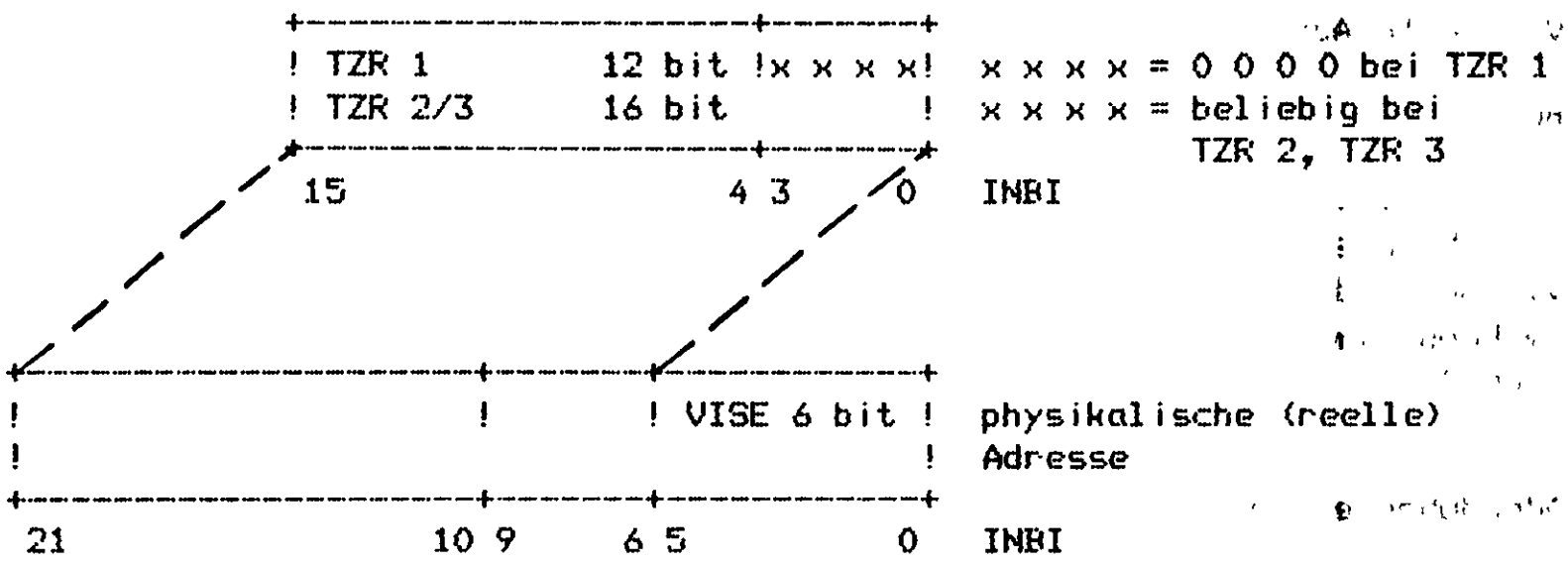
Die Umwandlung der virtuellen Adresse in eine physikalische Adresse erfolgt über Übersetzungstafeln mit Hilfe von Tafelzeigeregistern (TZR 1, TZR 2, TZR 3). Jedes Tafelzeigeregister zeigt auf die ihm zugehörige Übersetzungstafel (ÜT 1, ÜT 2, ÜT 3), wobei zu beachten ist, daß die ÜT 1 immer an einer Seitengrenze (2n x 1024 byte) beginnt. Da eine ÜT 64 Zellen umfaßt, ist es möglich, zu mehreren Adreßräumen von je 128 x 1024 byte zuzugreifen, welche im Zentralspeicher beliebig angelegt sein können. Entsprechend der Anzahl der vorhandenen Tafelzeigeregister gibt es drei solche Adreßräume. Jedem der drei Tafelzeigeregister ist ein Assoziativ-Speicherbereich zugeordnet (spezieller Speicher auf der Flachbaugruppe "Virtuelle Adressierung"), in welchem die Ergebnisse der Adreßübersetzung abgespeichert werden. Nach jedem Umladen eines Tafelzeigeregisters wird der Assoziativ-Speicher gelöscht.

Übersetzungsmechanismus

Mit den Tafelzeigeregistern werden die Anfänge der Übersetzungstafeln gefunden. Zum Auffinden der richtigen Zelle in den ÜT wird die 16-bit-breite virtuelle Adresse (d. h. jene Adresse, die im Anwenderprogramm Verwendung findet) in eine virtuelle Seitennummer (VISE) und eine Platznummer (PLAN) geteilt.



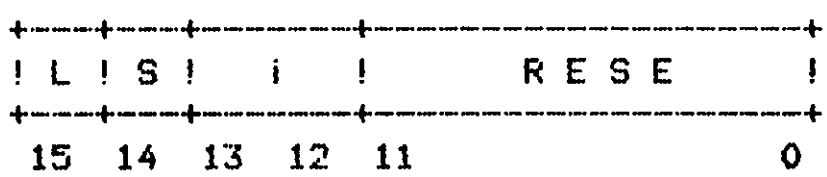
Die 6 bit der virtuellen Seite und die 12 höchstwertigen Bit des Tafelzeigeregisters 1 bzw. alle 16 bit der Tafelzeigeregister 2 oder 3 bilden zusammen die 22-bit-breite Adresse zur Adressierung der entsprechenden Zelle in der ÜT.



x x x x = 0 0 0 0 bei TZR 1
x x x x = beliebig bei TZR 2, TZR 3

physikalische (reelle) Adresse

Bei der Adressierung mit TZR 1 werden die Bits 6 bis 9 der üT-Adresse auf 0 gehalten.
Die so aufgefundene Zelle in der üT enthält die 12-bit-breite reelle Seitenadresse (RESE) und zwei Steuerbits (L,S).



i: irrelevant

Die Steuerbits haben folgende Bedeutung:

L = 0 Seite ist nicht geladen
L = 1 Seite ist geladen
S = 0 Seite hat keinen Schreibschutz
S = 1 Seite hat Schreibschutz

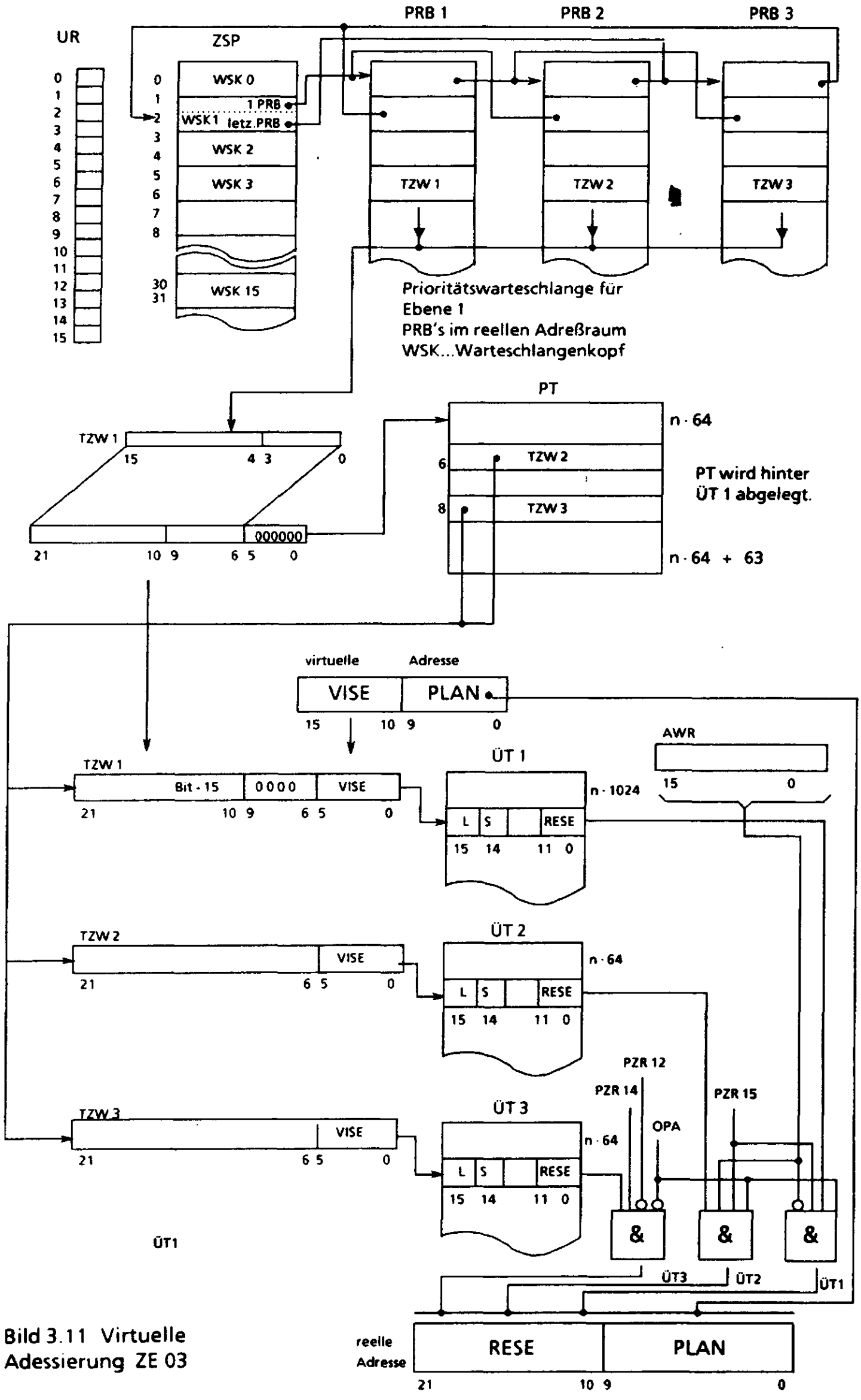
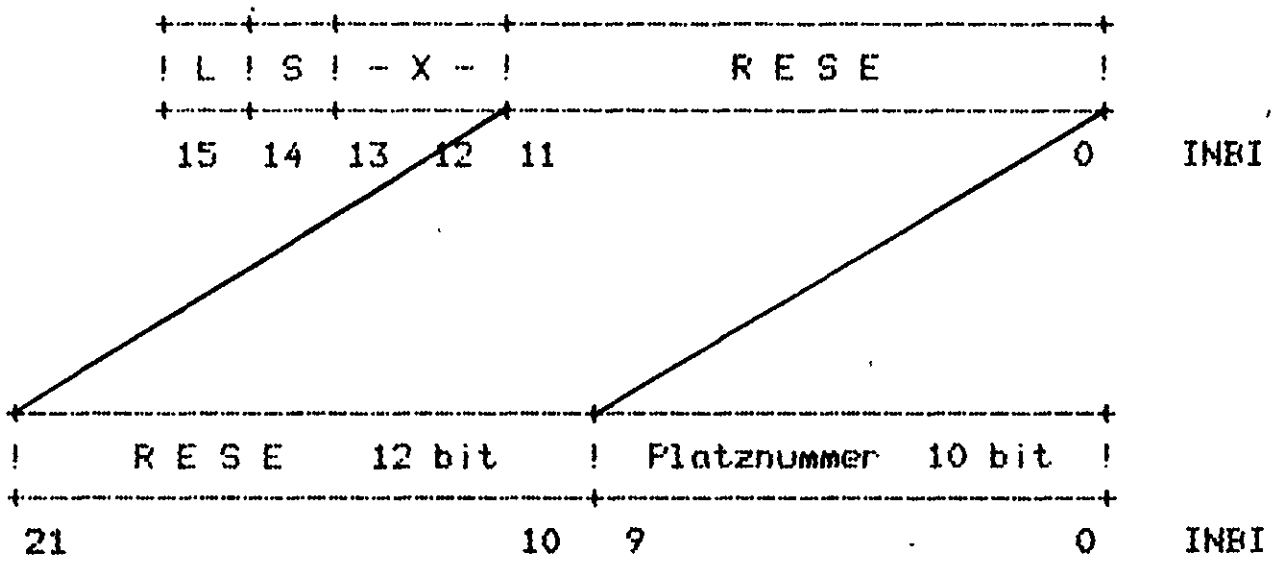


Bild 3.11 Virtuelle Adessierung ZE 03

Nunmehr wird die reelle Seitennummer (RESE) gelesen und zusammen mit der virtuellen Platznummer zur 22-bit-breiten Zentralspeicheradresse zusammengesetzt.



Wird bei einer Adreßübersetzung festgestellt, daß die Seite nicht geladen ist (Steuerbit L = 0), dann wird im Unterbrechungsanzeigenregister UAR Bit 2 gesetzt und damit ein Zustandswechsel angefordert.

Wird ein Schreiben auf eine schreibgeschützte Seite versucht, so wird das laufende Programm mit der Meldung "Schreibverbotsverstoß" im Unterbrechungsanzeigenregister UAR (Bit 10) unterbrochen.

Erhöhung der Übersetzungsgeschwindigkeit

Nach dem bisher geschilderten Übersetzungsmechanismus mußte bei jeder virtuellen Adresse ein Speicherzugriff zur Übersetzungstafel erfolgen. Im allgemeinen wird jedoch für mehrere Befehlsfolgen in den gleichen Speicherbereich, d. h. gleiche virtuelle Seite (VISE) zugegriffen. Es kann somit in vielen Fällen das Ergebnis der vorherigen Adreßübersetzung verwendet werden. Man vergleicht deshalb vor jedem Übersetzungsvorgang, ob sich die virtuelle Seitennummer geändert hat. Dieser Vergleich wird mit Hilfe eines Assoziativspeichers durchgeführt, der die Übersetzungsergebnisse aller drei Übersetzungstafeln aufnehmen kann. In diesem Speicher werden alle Ergebnisse der Adreßübersetzungen einschließlich der Steuerbits L (Seite geladen) und S (Schreibschutz) hinterlegt. Bei jedem Speicherzugriff wird festgestellt, ob die entsprechende reelle Seitennummer schon vorhanden ist (Gültig-Bit gesetzt). Ist sie nicht im Assoziativspeicher hinterlegt, wird der oben beschriebene Übersetzungsvorgang durchgeführt. Ist die reelle Seitennummer vorhanden, wird diese zusammen mit der Platznummer zur reellen Zentralspeicheradresse zusammengesetzt.

Die Gültig Bits des Assoziativspeichers werden bei jeder Änderung der Tafelzeigerregister gelöscht.
Da also während der Gültigkeit einer bestimmten Übersetzungstafel eine virtuelle Seite nur einmal übersetzt werden muß, ist insgesamt gesehen der Zeitaufwand für Adreßübersetzung für die einzelnen Befehlsausführungszeiten vernachlässigbar.

Erweiterung des Daten- und Befehlsadreßraums

Daten werden über Tafelzeigerregister TZR 1 oder TZR 2 adressiert (je nach Adressierungswechselregister AWR), Befehle werden über TZR 3 adressiert. Die Adressen von Daten werden mittels TZR 1 übersetzt, wenn im AWR kein Bit gesetzt ist.
Soll eine Zentralspeicher-Zelle im zweiten virtuellen Adreßraum (TZR 2) angesprochen werden, dann muß für das an der Adressierung beteiligte Standardregister das zugehörige Bit im AWR gesetzt werden. Damit wird die Adressenübersetzung über das TZR 2 aus der Übersetzungstafel 2 vorgenommen. Das programmspezifische AWR kann dazu vom Anwenderprogramm entsprechend der für die Adressierung vorgesehenen Register verändert werden.

Damit auch reell ablaufende Programme (Programmzustandsregister-Bits 14 und 15 gelöscht) auf Daten zugreifen können, die in einem virtuellen Adreßraum liegen (z. B. ORG auf Parameter eines Anwenderprogramms), wird bei gesetztem Bit im AWR - unabhängig davon, ob PZR-Bit 15 gesetzt ist - bei einer Adressierung durch das entsprechende Register virtuell über Übersetzungstafel 2 adressiert.

Beim Programmieren ist darauf zu achten, daß eine unkontrollierte Veränderung der Programmzustandsregister-Bits 14 und 15 durch Spezialregisterbefehle zu einem Fehllauf im Programm führen kann. Befehle können also nur im reellen oder im virtuellen Adreßraum der Übersetzungstafel 3 ablaufen.

Adreßerweiterung des Codebereichs:

Die Adressen von Befehlen werden mittels Tafelzeigeregister 3 (TZR 3) und dem Standardregister R1 übersetzt.

Ingesamt ergeben sich folgende Möglichkeiten:

Befehlsadresse		Operandenadresse		
Adressierung mit Reg. R1	PZR-Bit 14	Adressierung mit Reg. RN (N≠1)	PZR-Bit 15	AWR-Bit N
reell	0	reell	0	0
virtuell über ÜT 1	nicht möglich	virtuell über ÜT 1	1	0
virtuell über ÜT 2	nicht möglich	virtuell über ÜT 2	irrelev.	1
virtuell über ÜT 3	1	virtuell über ÜT 3	nicht möglich [1)]	

1) Ausnahme: Befehl LAC (s. Kapitel 6.4)

Um andere Code-Adreßräume anzuspringen, ist ein Unterprogrammsprung mit Kellern festgelegt, in dem automatisch der bestehende R1- und TZR 3-Inhalt abgespeichert und der neue TZR 3- und R1-Inhalt geladen werden. Für den Rücksprung ist ein Sprungbefehl vorgesehen, in dem das neue R1 und TZR 3 aus zwei aufeinanderfolgenden Zentralspeicherzellen des Kellers geladen wird (s. Kapitel 6.4).

3.1.8.2 Adressierung der Peripherie

Für die Peripherie steht ein eigener separater Adreßraum zur Verfügung. Die Umschaltung der Adreßräume erfolgt über das EA-Begleitsignal M/$\overline{IO}$, das von den Lese- und Schreibbefehlen für den EA-Bereich aktiviert wird (EAS, EAL). Im EA-Bereich werden durch die EA-Befehle nur 16 der 22 bit des Adreßbusses ausgenutzt, so daß nur die ersten 64 x 1024 Adressen dieses Adreßraums belegt werden können. Die höherwertigen Bits sind Null.

Adreßfreigabesignale $\overline{AE0}$, $\overline{AE1}$:
Aus dem gesamten EA-Adreßraum können über mehrere Adreßfreigabeleitungen Teiladreßräume gebildet werden (Vordekodierungen).
Es gibt die Signale $\overline{AE0}$, die jeweils einen Adreßraum von 4 x 1024-Adressen auswählen, und zwar oberhalb der Adresse 32 x 1024. Das ist der Adreßraum, der der Standardperipherie zugeteilt ist, wodurch sich für diese Geräte der Dekodieraufwand verringert.

$\overline{AE0}$ führt folgende Dekodierung durch:
$\overline{AE00}$ im Zentraleinheits-Rahmen: Adreßraum 32 x 1024 bis 36 x 1024-1
$\overline{AE01}$ bis $\overline{AE07}$ in den Erweiterungsrahmen: über den Drehschalter S1 auf der Busanpassung im Erweiterungsrahmen sind die Hexa-Zahlen 0 bis F einstellbar, wovon aber nur die drei niederwertigen Bits ausgewertet werden (Einstellung modulo 8). Dadurch wird sowohl die Baugruppenträger-Nummer eingestellt als auch die EA-Adresse vordekodiert. Im Erweiterungsbaugruppenträger sind also folgende Einstellungen möglich:

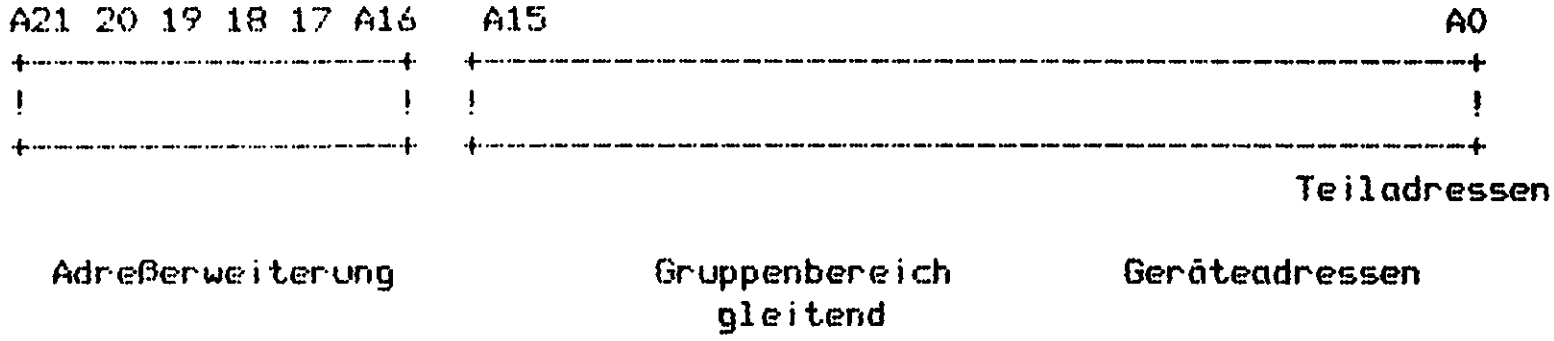
Schalter S1	BGT-Nr.	$\overline{AE0x}$	EA-Adreßbereich
0,8	0	$\overline{AE00}$	32 x 1024 - 36 x 1024-1 *)
1,9	1	$\overline{AE01}$	36 x 1024 - 40 x 1024-1
2,A	2	$\overline{AE02}$	40 x 1024 - 44 x 1024-1
3,B	3	$\overline{AE03}$	44 x 1024 - 48 x 1024-1
4,C	4	$\overline{AE04}$	48 x 1024 - 52 x 1024-1
5,D	5	$\overline{AE05}$	52 x 1024 - 56 x 1024-1
6,E	6	$\overline{AE06}$	56 x 1024 - 60 x 1024-1
7,F	7	$\overline{AE07}$	60 x 1024 - 64 x 1024-1

*) reserviert für den Rahmen der Zentraleinheit

Das Signal $\overline{AE1}$ ist fest auf 0 V gelegt, d. h. es steht nur ein Adreßbereich entsprechend 16 Adreßbits zur Verfügung.

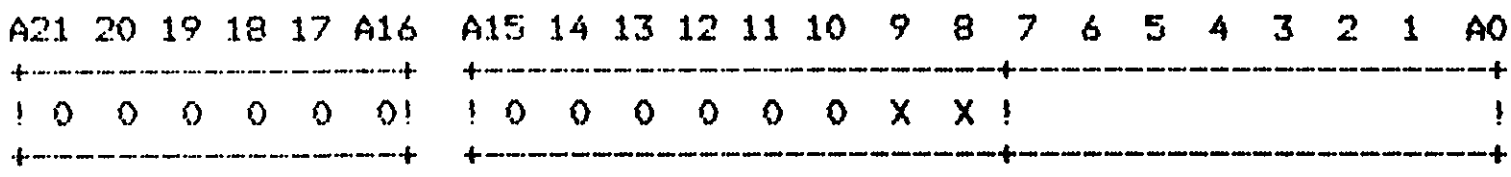
Anschaltungen, die die $\overline{AE}$-Signale benutzen, müssen nur die Adressen vergleichen, die zu ihrem Adressiervolumen gehören. Dadurch müssen nur noch 4 bit der EA-Adresse der peripheren Einheit auf der Anschaltung eingestellt werden; diese 4-bit-Adresse entspricht dem Steckplatz im Baugruppenträger (s. Adreßschema Punkt D).

Globales Adreßschema



Adreßräume A - H

A) Organisatorischer Adreßraum A



0	0	0 - 255 ZE-Service	(A0)
0	1	reserviert	(A1)
1	0	reserviert	(A2)
1	1	reserviert	(A3)

B) Reservierte Adreßräume

B

A21	20	19	18	17	A16	A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0
0	0	0	0	0	0	0	0	0	0	0	1										

Adreßraum entsprechend 10 Adreßbits

C

A21	20	19	18	17	A16	A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0
0	0	0	0	0	0	0	0	0	0	0	1										

Adreßraum entsprechend 11 Adreßbits

D

A21	20	19	18	17	A16	A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0
0	0	0	0	0	0	0	0	0	0	1											

Adreßraum entsprechend 12 Adreßbits

F

A21	20	19	18	17	A16	A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0
0	0	0	0	0	0	0	1														

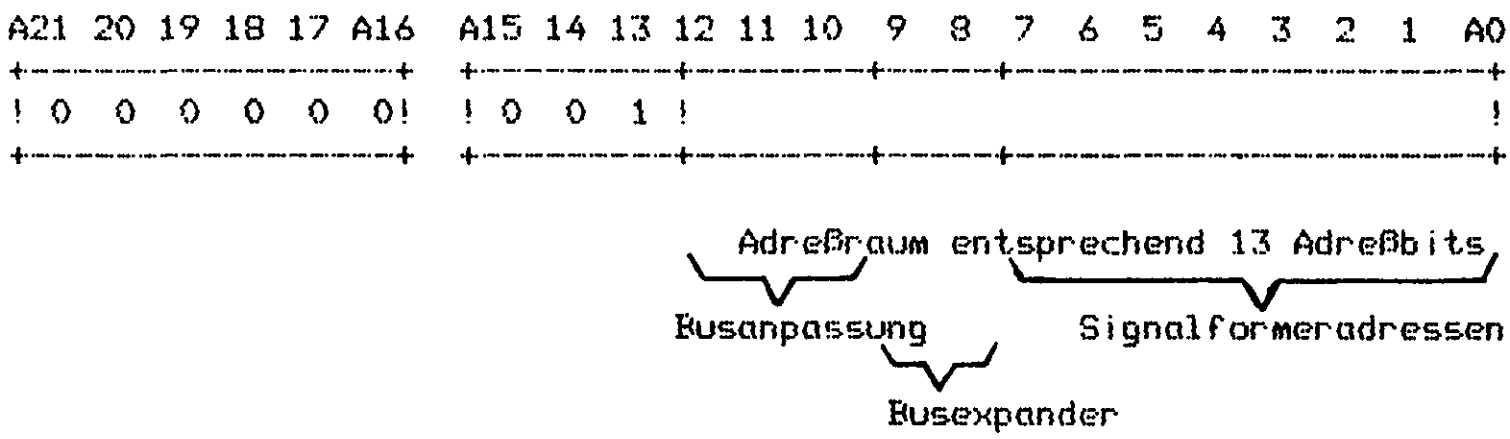
Adreßraum entsprechend 14 Adreßbits

H

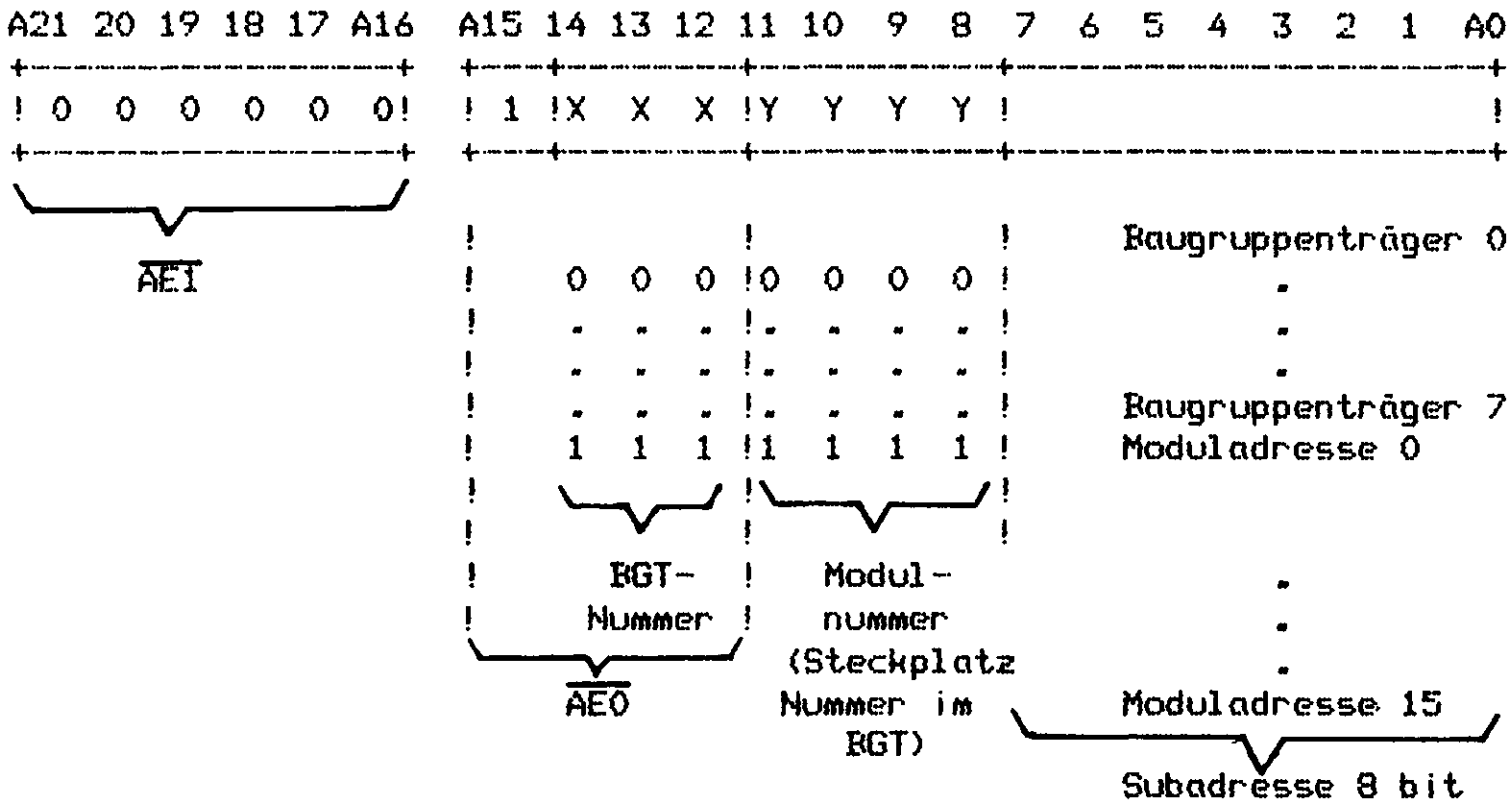
A21	20	19	18	17	A16	A15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	A0
X	X	X	X	X	X																

Adreßraum entsprechend 16 Adreßbits

C) Prozeßperipherie (Adreßraum E)



D) Standardperipherie (Adreßraum G)



Zur Reduzierung des Dekodieraufwands, vor allem bei der Standardperipherie, wird eine Verringerung der Vergleichsbreite sowie der Größe der Adreßeinstellschalter im Adreßraum G über die Vordekodierung $\overline{AE0}$ ermöglicht.

3.1.9 Ein-/Ausschaltroutinen, Netzausfall

- Einschalten
  Beim Einschalten der Anlage mittels des Schlüsselschalters "power supply" gibt die Stromversorgung Signale entsprechend Bild 3.12 ab.

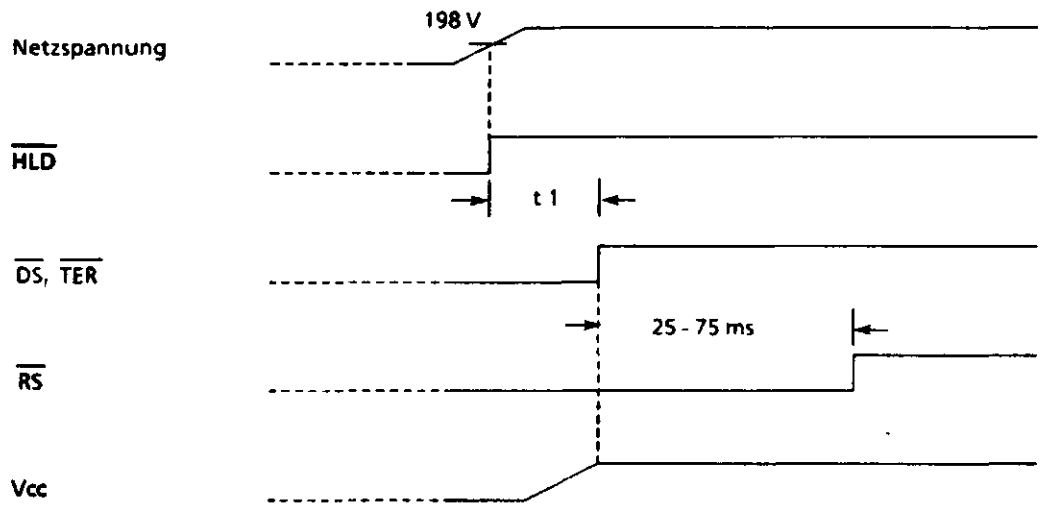


Bild 3.12 Einschalten

t_1: Einschaltzeit sekundär

HLD : Hold, Netzausfall-Frühwarnung
TER : Terminate, Netzausfall
DS : Data Save, Signal für den Hauptspeicher
RS : Reset, als RSP an die Peripherie
als RSC an den Zentralprozessor

Der Anlauf entspricht den Aktionen des Festspeicherprogramms VICOM gemäß NMIBAU.

- Rücksetzen
  Durch den Schalter "Reset" wird das Signal RSR aktiviert und es ergibt sich eine Signalsequenz lt. Bild 3.13. Dabei laufen im zweiten Teil die Signale gleich ab wie beim Einschalten; entsprechend mündet die Bearbeitung ebenfalls im NMIBAU.

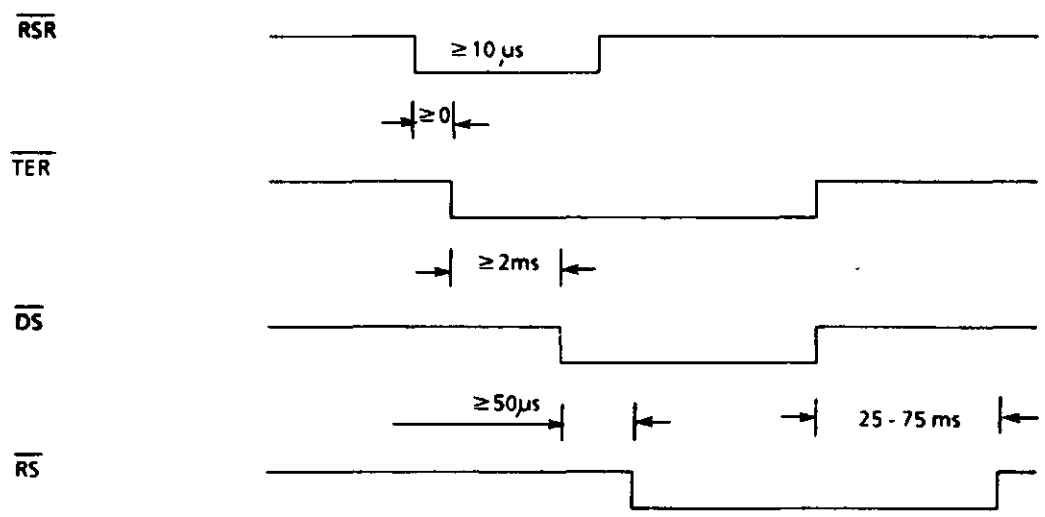


Bild 3.13 Rücksetzen

- Netzausfall

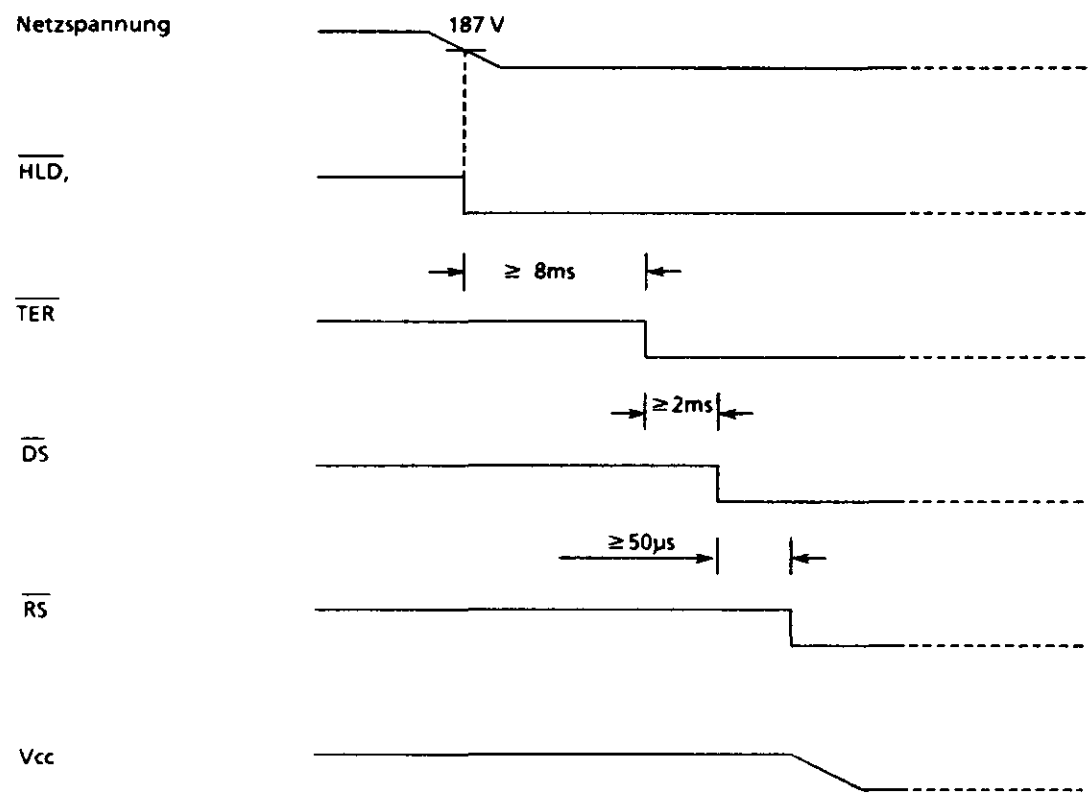


Bild 3.14 Netzausfall

Welche Aktionen auf Grund der Aktivierung der dabei auftretenden Signale ablaufen, ist den Signalbeschreibungen in Kapitel 3.3.1 zu entnehmen.

- Spannungswiederkehr
  Die Maßnahmen bei Spannungswiederkehr sind abhängig vom Signal BF (battery failure):
  Zeigt $\overline{BF}$ Batterie-Ausfall an, läuft die Sequenz entsprechend NMIBAU ab.
  Zeigt $\overline{BF}$ an, daß der Zentralspeicher gepuffert war, wird die Sequenz gemäß NMIZRS aktiviert.

Die drei erwähnten NMIs haben also folgende Bedeutung:

NMIZRS: Rücksetzen ohne Batterie-Ausfall, d. h. Spannungswiederkehr bei gepuffertem System, Speicherinformation ist noch vorhanden.

Das zuletzt gestartete urladefähige Programm wird an seiner RESTART-Adresse gestartet. Dies wird im allg. das Betriebssystem sein, das daraufhin drei Möglichkeiten hat:

- Wiederanlauf (Betriebssystem laden)

- Neustart (Betriebssystem ist noch gültig im Zentralspeicher, Listen werden zurückgesetzt und das Betriebssystem wird ge startet.)

- Wiederaufsetzen (unterbrochenes Programm läuft an einer Stelle weiter, welche kurz vor der Unterbrechungsstelle liegt und ein einwandfreies Weiterarbeiten erlaubt.)

NMIBAU: Rücksetzen mit Batterieausfall, d. h. Spannungswiederkehr bei ungepuffertem System oder entladener Pufferbatterie. Die Speicherinformation ist zerstört. Es muß neu urgeladen werden.

NMINAU: Netzausfall. Dieses NMI-Ereignis startet im höchstprioren Zustand Z0 die Netzausfallroutine.

Diese NMI-Ereignisse führen die Zentraleinheit in den Zustand Z0, wo je nach NMI-Ereignis eine bestimmte Festspeicherroutine gestartet wird.

3.2 Zentralspeicher

Der Zentralspeicher nimmt Befehle und Daten auf. Er besteht aus dem Schreib/Lesespeicher und dem Festwertspeicher.

3.2.1 Schreib-/Lesespeicher, Fehlerkorrektur

Im Schreib-/Lesespeicher sind Befehle und Daten hinterlegt, die bei Bedarf durch neue Informationen verändert werden können. Er besteht aus Speichermodulen mit 1,2 oder 4 M* byte Speicherkapazität, aufgebaut mit dynamischen 256 K* bit NMOS-RAMs. Die auf den Speichermoduln untergebrachte Fehlerkorrekturlogik erlaubt die Korrektur aller Einzelbitfehler und die Erkennung sämtlicher Doppelfehler sowie einer Vielzahl von Mehrfachfehlern. Zusätzlich zu den 16 Informationsbits sind dafür 6 Korrekturbits erforderlich. Für die Prüfung der ausgelesenen Daten werden ca. 100 ns - benötigt und weitere 100 ns, falls ein fehlerhaftes Datenbit korrigiert werden muß. In einem modulspezifischen Fehlerregister werden bis zu 15 Fehler mitgezählt sowie die oberen 12 Adreßbits (A10..A21) des letzten fehlerhaften Zugriffs gespeichert. Außerdem meldet die Korrekturlogik über die Signale ERROR (bei korrigierbarem Einzelfehler) und MERROR (bei nicht korrigierbarem Mehrfachfehler) der Speichersteuerung das A ftreten eines Fehlers (Hinweis: die Adreßinformation A10...A21 wird durch RSC oder durch Lesen des Fehlerregisters nicht rückgesetzt, der Fehlerzähler wird nur durch einen Lesezugriff rückgesetzt).

Die Speicherzugriffszeiten sind abhängig von der Betriebsart des Speichers: (Fehler-Korrektor eingeschaltet)

	fehlerfreier Fall		Fehler	
	Byte	Wort/Doppelwort	Byte	Wort/Doppelwort
Lesen	300 ns	300 ns	300 ns	300 ns
Schreiben	300 ns	150 ns	300 ns	150 ns

Die Fehklerkorrektur geschieht bei den neuen Speicherbaugruppen mit 256 k*bit übr 32 bit.

Die Speichermodule sind in den Bestückungsvarianten

- 1 M* byte
- 2 M* byte
- 4 M* byte

erhältlich. Der 4 M* byte-Modul ist eine Sandwich-Konstruktion, die 1 1/3 SEP benötigt.

Einstellung der Anfangsadresse bei den neuen Speichermoduln mit 256 k*bit-Chips

Die Einstellung der Anfangsadresse erfolgt mit 3 Schaltern des Schalterbausteins S1 auf der Grundbaugruppe., und zwar für alle Moduln einheitlich in Stufen von 1 M*byte (= kleinste Modulkapazität).
Der Adressierungsbereich eines Moduls innerhalb dews 8 M*byte Adressraums ergibt sich aus Anfangsadresse und Modulkapazität. Wegen der Wortbreite von 32 bit erübrigt sich eine Eistellung von Wort- und Doppelwortkonfiguration wie bei den bisherigen Moduln.
Dadurch wird die durch den Anwender vorzunehmende Parametrierung wesentlich vereinfacht.Bei Bestückung mit neuen Moduln unterschiedlicher Kapazität ist die Reihenfolge bei der Vorgabe der Anfangsadressen beliebig. Bei gemischter Bestückung mit alten Moduln muß die für diese geltende Vorschrift eingehalten werden.
Moduln kleinerer Kapazität haben höhere Anfangsadressen.

Positionen

8	4	2	1	Anfangsadresse (HEX)
o	o	o	o	000000 (0 MB)
o	o	o	x	080000 (1 MB)
o	o	x	o	100000 (2 MB)
o	o	x	x	180000 (3 MB)
o	x	o	o	200000 (4 MB)
o	x	o	x	280000 (5 MB)
o	x	x	o	300000 (6 MB)
o	x	x	x	380000 (7 MB)

o = Schalterpos. aus (OFF)
x = Schalterpos. ein (ON)
Schalterpos.8 ist nicht verwendet

Tabelle 3.5a Einstellung Schalter S1

Lage des Schalters S1:

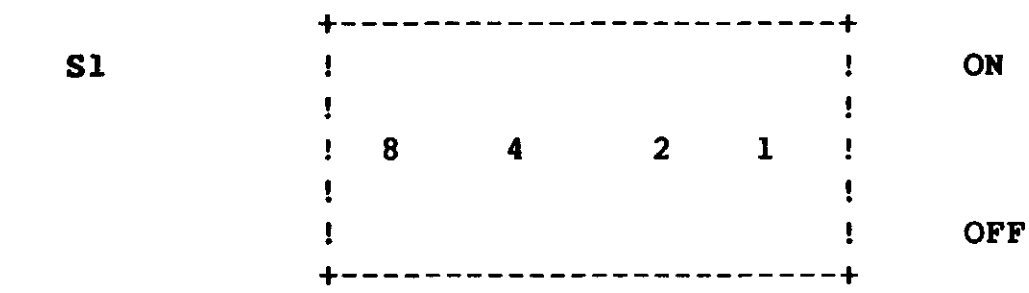


Tabelle 3.5b :Einstellung der Anfangsadresse bei den neuen Speichermoduln mit 256 k*bit-Chips

Neben dem Einstellen der Anfangsadresse muß mit einem 16-stufigen Drehschalter die Modulkennung eingestellt werden, um der Speichersteuerung das Ansprechen von Steuerwortregistern und Fehlerregistern zu ermöglichen.

Für die älteren Baugruppen mit 64 k*bit -Chips gilt:
Bei Doppelwortbetrieb müssen den zwei betroffenen Speicherwortmodulen zwei benachbarte Modulkennungen zugeteilt werden.

Die jeweils zusammengehörigen Speichermodule müssen bei Doppelwortbetrieb über gleiche Kapazität verfügen; der belegte Adreßbereich verdoppelt sich dabei.

In der folgenden Tabelle 3.5 sind die Einstellungen der Schalter S2 und S3 beschrieben:

	Schalter S2 Anfangsadresse								Schalter S3 Adreßrangierung								
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	Speichermodul
	IW		$\overline{A21}$	$\overline{20}$	$\overline{19}$	$\overline{18}$	$\overline{17}$	$\overline{16}$			$\overline{A16}$	$\overline{00}$	$\overline{19}$	$\overline{18}$	$\overline{17}$	$\overline{16}$	
Wort-an-ordnung	0		- 1 -					0			0	X	X	X	X	0	256K*byte
	0		- 2 -				0	0			0	X	X	X	0	0	512K*byte
	0		- 3 -			0	0	0			0	X	X	0	0	0	1M*byte
Dop-pel-wort-an-ordn.	X		- 2 -				0	0			X	0	X	X	0	0	256K*byte
	X		- 3 -			0	0	0			X	0	X	0	0	0	512K*byte
	X		- 4 -		0	0	0	0			X	0	0	0	0	0	1M*byte

nicht verwendet

X = Schalterstelle
0 = Schalterstelle offen
1 = Stufung 256K*byte
2 = Stufung 512K*byte
3 = Stufung 1M*byte
4 = Stufung 2M*byte

Anfangsadresse 0: alle für die Anfangsadresse zugelassenen Schalterstellen offen

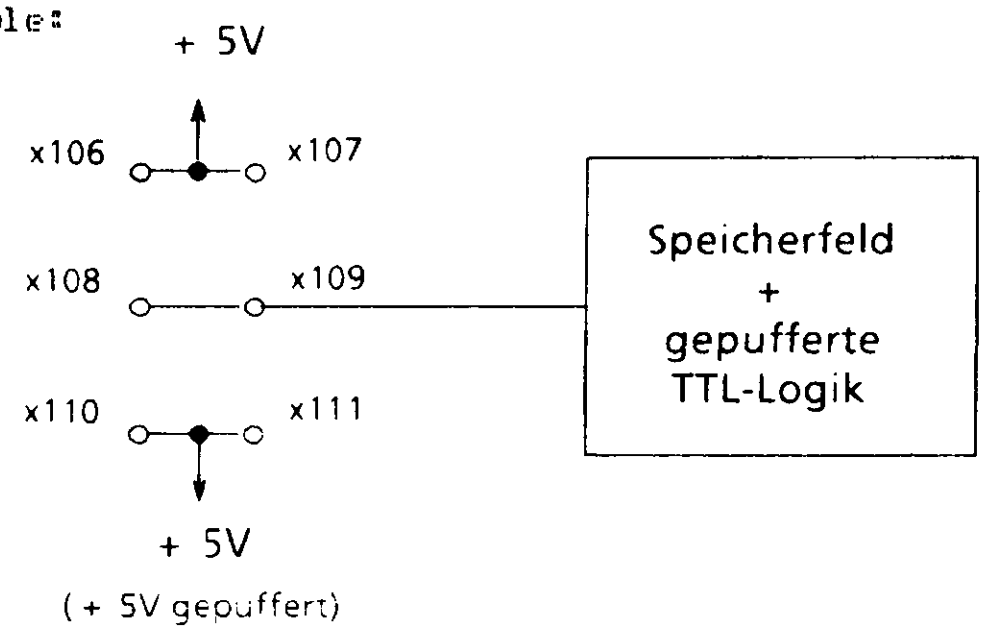
Tabelle 3.5: Schalterstellung S2, S3 für Einstellung von Anfangsadresse und Adreßrangierung

Durch die weitgehenden Funktionen auf den einzelnen Speichermodulen verbleiben der Speichersteuerung TC (timing and control) nur die Funktionen

- Auffrischen der dynamischen Halbleiterspeicher,
- Annahme und Kontrolle von Speicherzugriffen.

Die Versorgungsspannung des Zentralspeichers beträgt ausschließlich + 5V. Über einen wahlweisen Pufferzusatz kann man die im Zentralspeicher abgelegte Information bei einem Netzausfall erhalten.
Die dazu erforderlichen Verbindungen müssen auf den Speichermodulen und der Speichersteuerung individuell mit Brücken hergestellt werden:

Speichermodule:



(+ 5V gepuffert)

Brücke \ Betriebsfall	gepuffert	ungepuffert
x 106 - x 108	0	x
x 107 - x 109	0	x
x 110 - x 108	x	0
x 111 - x 109	x	0

x = Brücke eingelötet
0 = Brücke offen

Speichersteuerung:

Brücke \ Betriebsfall	gepuffert	ungepuffert
1	x	0
2	0	x

x = Brücke eingelötet
0 = Brücke offen

Dadurch können auch einzelne Speichermodule gepuffert werden, wobei immer die Module gepuffert werden müssen, die zur Aufnahme des Betriebssystems dienen.

3.2.2 Festwertspeicher

Der Festwertspeicher besteht aus zwei EPROMs, welche auf der Zentralprozessor-Flachbaugruppe TC untergebracht sind. Jedes EPROM hat eine Kapazität von 16K*byte; der Festspeicher umfaßt also 32K*byte.
Durch ihn sind die Adressen (hexa) 3FC000 bis 3FFFFF des Speicheradreßraums belegt.

Der Festwertspeicher enthält das Festspeicherprogramm VICOM; dieses ist - soweit möglich - strukturiert aufgebaut und in seinem gesamten Umfang ausschließlich mit den im Basistest getesteten Befehlen programmiert. Es nützt nur die dort getesteten Hardware-Funktionen des Zentralprozessors und des Speichers. Es besteht aus folgenden Teilen:

- Daten (z. B. Übersetzungstafel),
- Zentralprozessor - Basistest,
- Zentralspeicher - Basistest,
- NMI - Bearbeitung,
- Anlauf - Bearbeitung
- Urlader,
- ZO-Interrupt - Bearbeitung,
- VK-Kommando - Bearbeitung,
- TESTAS bzw. SPAS - Bearbeitung,
- EA - Routinen,
- diverse interne Unterprogramme.

Um die Leistungsfähigkeit von VICOM voll ausschöpfen zu können, kann man sich über die zur Verfügung stehenden Möglichkeiten durch das Hilfsprogramm VCINFO informieren. Durch Eingabe des Zeichens "?" über die virtuelle Konsole erhält man Auskunft, wie VCINFO geladen und gestartet wird. Nach dem Start ist VCINFO weitgehend selbsterklärend.

3.2.3 Cache-Speicher (schneller Pufferspeicher)

Der Cache-Speicher hat ein Speichervermögen von 2 x 1024 Zeilen, welche je ein Doppelwort (= 32 bit) enthalten (= 8 K* byte). Der Datenverkehr zwischen Zentralspeicher und Cache-Speicher wird demgemäß in 32-bit-Breite durchgeführt. Das niederwertigste Adreßbit entscheidet dabei, ob das zugehörige 16-bit-Datum (=Wort) die linke oder rechte Hälfte des Doppelworts bildet (Adreßbit 0 = 0 ... gerade Adresse, linke Hälfte; Adreßbit 0 = 1... ungerade Adresse, rechte Hälfte; jeweils INBI 0). Zur Adressierung der im Cache-Speicher enthaltenen Information werden die sogenannten Assoziativteile (Ass-Teile) verwendet. Es gibt zwei Ass-Teile, da ihre Information sowohl vom Zentralprozessor (Ass-Teil intern) als auch von der Peripherie her (Ass-Teil extern) zugänglich sein muß. Beide Ass-Teile arbeiten nach dem gleichen Prinzip. Sie enthalten ein G-Bit (Gültig-Bit) pro Doppelwort der im Cache-Speicher abgelegten Daten, welche die Gültigkeit des zugehörigen Doppelworts kennzeichnet (G-Bit = 0 ... ungültig, G-Bit = 1 ... gültig).

Die Untersuchung, ob das einer Adresse zugehörige Datum gültig im Cache steht, geschieht folgendermaßen:

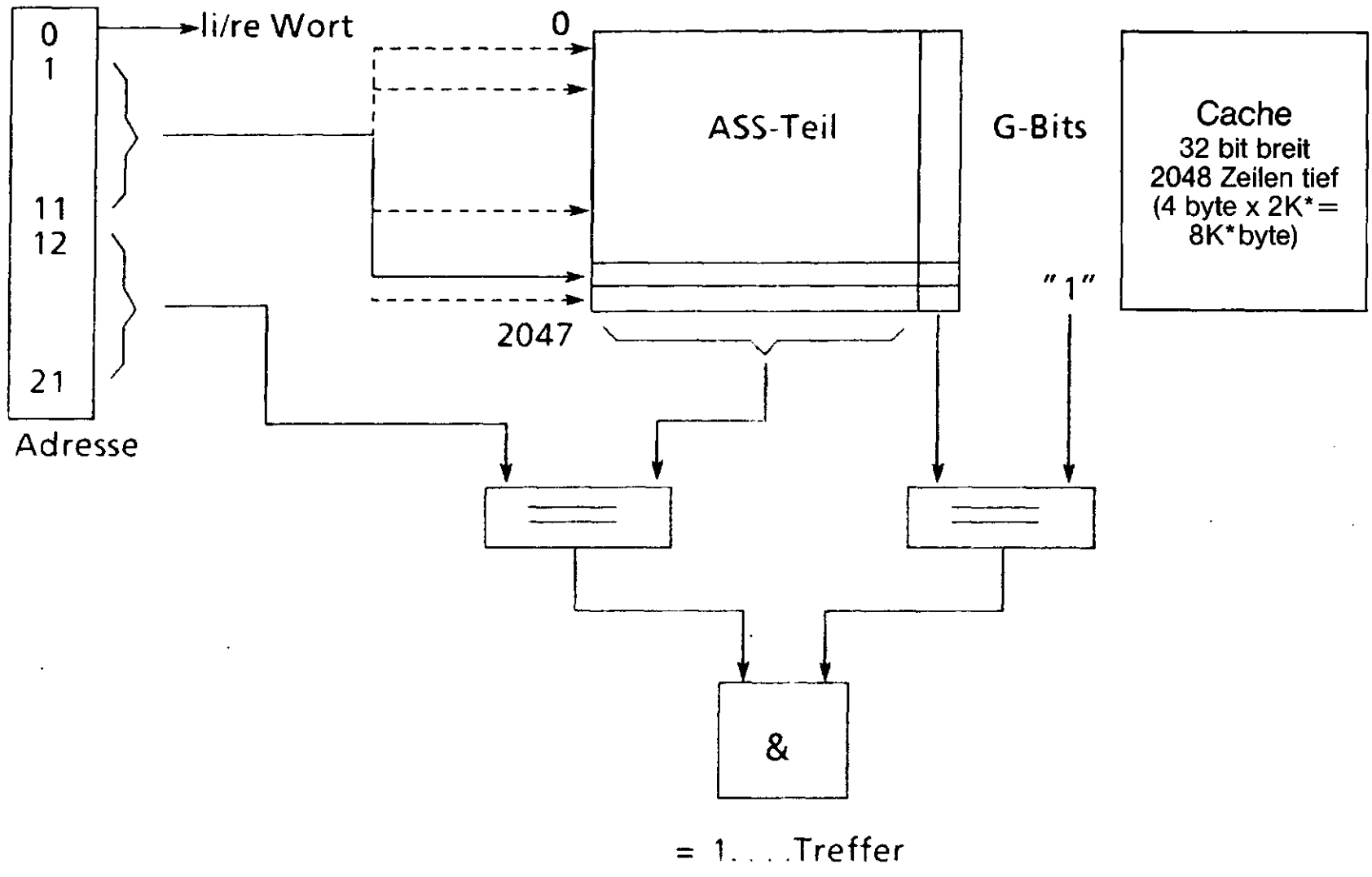


Bild 3.15 ASS-(Assoziativ-) Teil des Cache-Speichers.

Die Adreßbits INBI 1 - 11 werden verwendet, um sowohl eine Zeile des 2 x 1024 Zeilen tiefen Cache-Speichers als auch die Zeile mit der gleichen Adresse des ebenso tiefen Ass-Teils zu adressieren. Im Ass-Teil ist abgespeichert, zu welcher "Seite" (Adreßbits INBI 12 - 21) des Adreßraums das im Cache-Speicher enthaltene Datum gehört. Stimmt die im Ass-Teil enthaltene Information mit den INBIs 12 - 21 der aktuellen Adresse überein und wird auch durch das G-Bit Gültigkeit markiert, so wird Treffer gemeldet, d.h. der Zentralprozessor kann die Daten dem Cache-Speicher entnehmen. Nur der Zentralprozessor ist berechtigt und in der Lage, zum Cache-Speicher zuzugreifen. Dabei ergeben sich folgende vier Möglichkeiten:

a) Lesen eines Datums durch den ZP mit Treffer im Cache-Speicher	Datum wird dem Cache-Speicher entnommen. ZSP wird nicht angesprochen.
b) Lesen eines Datums durch den ZP ohne Treffer im Cache-Speicher	32 bit-Transfer vom ZSP in den Cache-Speicher, parallel dazu gewünschtes Datum an den ZP, Aktualisierung beider Ass-Teile.
c) Schreiben eines Datums durch den ZP mit Treffer im Cache-Speicher	Datum wird in den ZSP und Cache-Speicher geschrieben.

d) Schreiben eines Datums durch den ZP ohne Treffer im Cache-Speicher	Datum im ZSP abgelegt. Der Cache-Speicher wird nicht verändert.

ZP ... Zentralprozessor ZSP ... Zentralspeicher

Wenn durch die Peripherie Daten in den Zentralspeicher geschrieben werden, wird die ggf. betroffene Cache-Speicherzelle für ungültig erklärt, indem das zugehörige G-Bit in beiden Ass-Teilen gelöscht wird.

Bei Festspeicherzugriffen des Zentralprozessors wird der Cache-Speicher umgangen, da auf dem Festwertspeicher nicht 32-bit-breit zugegriffen werden kann. Die Grenzadresse, ab welcher diese Umgehung des Cache-Speichers einsetzt, wird mit einem DIL-Schalter auf der Cache-Flachbaugruppe eingestellt: Grenzadresse = 8 M* byte - 128 K* byte x 2 exp n (n_{hexa} = Schalter-Stellung).

Der Datenverkehr zwischen Cache-Speicher und Zentralprozessor wird aufgrund der kurzen Signalwege nicht überwacht.

Der Cache-Speicher ist als einfache Flachbaugruppe im 3fachen Europaformat ausgeführt.

Ein lesender Speicherzugriff in den Cache-Speicher dauert 200 ns.

3.3 EA-System

Die Zentraleinheiten der Modellreihe SICOMP verfügen über ein einheitliches EA-System, das durch universelle Verwendbarkeit der EA-Anschlußstellen gekennzeichnet ist. Die EA-Schnittstelle ist als Bus ausgeführt und bietet DMA-Möglichkeit für jede EA-Anschlußstelle.

Da einige Bus-Signale als Kettensignale ausgeführt sind, die von Steckplatz zu Steckplatz weitergereicht werden, kann durch Entfernen einer Baugruppe aus einem EA-Steckplatz diese Kette unterbrochen werden. Da diese Kette nicht unterbrochen werden darf, ist bei Entfernung einer Baugruppe aus einem EA-Steckplatz eine Blindbaugruppe zu stecken, welche die Signale weiterreicht.

Zusammen mit dem hardware-gesteuerten Unterbrechungssystem ist das EA-System zugleich leistungsfähig wie auch flexibel. Ein weiteres Leistungsmerkmal ist die Einheitlichkeit der Schnittstelle hinsichtlich ihrer Verwendung als Speicher-, Zentralprozessor- und Anschaltungsschnittstelle, wodurch z. B. eine wahlweise Verwendung einiger Steckplätze für Zentralspeicher oder Peripherie-Anschaltungen möglich ist.
Darüber hinaus können die Anschlußmöglichkeiten durch Verlängerung des Busses in Erweiterungsbaugruppenträger ausgedehnt werden, ohne ihre Funktionalität zu verlieren. Die Struktur der Schnittstelle bringt eine strikte Trennung der Hardware- und Software-Leistungen im EA-Verkehr mit sich. Aus diesen Gründen wird es auch in Zukunft leicht möglich sein, über entsprechende Anschaltungen neue periphere Einheiten in das Spektrum zu integrieren. Diese Möglichkeit wird durch die Verwendbarkeit von marktüblichen Prozessoren und LSI-Bausteinen zusätzlich gefördert.

3.3.1 EA-Schnittstelle

Die Systemschnittstelle enthält busförmig verdrahtete Signale, die im Baugruppenträger störsicher angeordnet sind (zwischen Nullsignalen eingebettet). Die Signalleitungen sind entweder durchgehend (z. B. Datenbus) oder als Kettensignale ausgeführt (z. B. Interrupt). Die Signale teilen sich in folgende Gruppen auf:

Datenbus $\overline{DB}$

32 bit breit (Teilmengen nutzbar: 8, 16 bit)

Adreßbus $\overline{AB}$

22 bit breit

Transfersteuersignale $\overline{TR}$, $\overline{TA}$
(Transfer Request, Transfer Acknowledge)

Jede Transferoperation (Lesen/Schreiben, Speicher/EA, DMA) wird durch das Anforderungssignal $\overline{TR}$ und das Quittungssignal $\overline{TA}$ gesteuert. Durch diese beiden Signale ist ein Handshake-Verkehr zwischen Zentraleinheit und peripherer Einheit realisiert. Die Unterscheidung in Lesen/Schreiben und Speicher/EA erfolgt durch Transferhilfssignale. Die Transfersequenzen werden zeitüberwacht.

Transferhilfssigale

o Lesen/Schreiben R/$\overline{W}$ (read/$\overline{write}$)

  Dieses Signal bestimmt die Transferrichtung. Schreiben und Lesen ist dabei jeweils vom Standpunkt desjenigen Transferpartners aus zu sehen, der den Transfer initiiert. Die Steuerung erfolgt vom Zentralprozessor (zentrale Initiative) oder von der Peripherie (DMA) aus.

o Speicher/EA-Bereich M/$\overline{IO}$ (memory/$\overline{IO}$)
  Für die Peripherie steht ein eigener Adreßraum zur Verfügung. Die Umschaltung zwischen Speicher- und EA-Adreßraum erfolgt über das Signal M/$\overline{IO}$. Das Signal wird im Zentralprozessor abhängig vom ausgeführten Befehl gebildet (Speicher- oder EA-Befehl). Dieses EA-Verfahren wird mit dem Begriff "isolated IO" bezeichnet im Gegensatz zum "memory mapped IO", das einen Teil des Zentralspeichers für die Adressierung der Peripherie benutzt.

Die Vorteile des isolierten EA-Verfahrens sind:

- reelle Adressierung der Peripherie,
- keine "Durchlöcherung" des normalen Zentralspeicher-Bereichs,
- das Adressiervolumen der Zentraleinheit steht komplett zweimal zur Verfügung: 1. für den Zentralspeicher
  2. für die Peripherie

o Adreß-Freigabe $\overline{AE0}$, $\overline{AE1}$ (address enable)
Aus dem gesamten EA-Adreßraum werden über zwei Adreß-Freigabeleitungen zwei Teiladreßräume gebildet. Steuerungen, die diese Freigabe-Signale verwenden, müssen dann nur die Adressen vergleichen, die zum Volumen ihres Teiladreßraums gehören, wenn ein EA-Transfer stattfindet; auf diese Weise werden weniger Vergleiche benötigt (Kapitel 3.1.8.2).
$\overline{AE0}$ wählt einen Adreßraum von 4 K* Adressen aus dem gesamten Adressiervolumen aus.
$\overline{AE1}$ wählt einen Teiladreßraum von 64 K* Adressen aus, d. h. die INBI 16 - 21 sind 0.

$\overline{AE0}$ und $\overline{AE1}$ werden vom Adreßbus abgeleitet und sind daher erst nach einer Verzögerungszeit von ca. 20 ns nach der Adresse gültig.

o Fehlersignale $\overline{FFA}$, $\overline{FFD}$ (fault flag address, fault flag data)
Die Fehlersignale dienen zur Meldung von Speicherfehlern (Daten- und Adressierfehler) oder zur Meldung von Besonderheiten beim EA-Verkehr (z. B. Quittungsverzug).

o Adreßsteuersignale $\overline{AC0}$, $\overline{AC1}$ (address control)
Die Adreßsteuersignale sind Begleitsignale zur Adreßinformation auf dem Adreßbus und dienen zur Festlegung des Adressiermodus auf Byte- und Doppelwortbetrieb:

$\overline{AC0}$	$\overline{AC1}$	
L	H	Wort
L	L	Doppelwort
H	H	linkes Byte
H	L	rechtes Byte

$\overline{AC0}$ bestimmt den Datentyp Wort bzw. Byte; $\overline{AC1}$ spezifiziert die Byte Adresse.

Bei Byte-Adressierung wird die Wortadresse durch $\overline{AC1}$ um 1 bit an der niederwertigen Seite ergänzt.
Bei Doppelwortadressierung (der Zentralspeicher muß dafür ausgelegt sein!) ist das niederwertigste Adreßbit irrelevant, d. h. es wird nicht ausgewertet.

Die Information auf dem Datenbus wird immer rechtsbündig angeboten und vom Zentralspeicher an die von den Adreßbegleitsignalen vorgegebene Stelle geschrieben bzw. aus der gewünschten Stelle gelesen. Bei Byte-Adressierung ist das linke Byte des Datenbus offen (tristate).

Bussteuersignale $\overline{BR}$, $\overline{BAI}$, $\overline{BAO}$, $\overline{BB}$, $\overline{BMO}$, $\overline{BMI}$

o Buseroberung
Die Initiative beim Datenverkehr über den Systembus kann von jedem Busteilnehmer ausgehen (Zentraleinheit, periphere Einheit). Damit bei gleichzeitig auftretenden Transferwünschen keine Überlagerungen oder Störungen der Signale auftreten, muß die Busbenutzung koordiniert werden.

Grundsatz: Zu einem Zeitpunkt kann nur ein Busteilnehmer einen Datentransfer durchführen. Der Teilnehmer, der einen Transfer durchführen will, muß den Bus erobert haben!

Mit dem Signal $\overline{BR}$ (bus request) wird der Bus angefordert. Das Signal $\overline{BA}$ (bus acknowledge) erlaubt die Busbelegung. $\overline{BA}$ ist ein Kettensignal und wird von jedem Busteilnehmer als $\overline{BAI}$ (bus acknowledge in) empfangen und als $\overline{BAO}$ (bus acknowledge out) weitergegeben. Mit dem Signal $\overline{BB}$ (bus busy) wird der Bus belegt.

o Block-Modus
Jeder Busteilnehmer darf den Bus im Normalfall für jeweils nur eine Transfersequenz belegen und muß ihn dann wieder freigeben. Schnelle periphere Einheiten können jedoch untereinander ein "gentlemen-agreement" treffen, indem sie mit dem Signal $\overline{BM}$ (Block-Modus) anzeigen, daß sie den Bus nach Freigabe wieder belegen wollen. Ein anderes schnelles Gerät, das $\overline{BM}$ auswertet, nimmt daraufhin vom Versuch der Buseroberung Abstand. Somit können sich über die Signale $\overline{BM0}$ und $\overline{BM1}$ periphere Geräte untereinander in ihrer Busbelegungsreihenfolge koordinieren. Geräte, die diese Blockmodussignale nicht auswerten, werden in ihrer Aktivität nicht berührt.
Aufgrund der begrenzten Summendatenrate des Busses kann ein schnelles Gerät seine hohe Einzeldatenrate nur dann erreichen und aufrechterhalten, wenn es während seiner Transferaktivität den Bus nicht mit anderen schnellen Geräten teilen muß. Teilnehmer, deren Zugriffshäufigkeit gering ist, reduzieren die erreichbare Datenrate der schnelleren Geräte nur unwesentlich. Sie können daher weiter transferieren, auch wenn ein schnelleres Geräte aktiv ist.

Mit dem Signal $\overline{BM0}$ bzw. $\overline{BM1}$ zeigt ein schnelles Gerät an, daß es von nun an vorrangig den Bus belegen will. Auf diese Weise können maximal zwei schnelle Geräte gleichzeitig am Bus aktiv sein. Vor dem ersten Transfer einer Transferfolge testet die periphere Einheit die Signale $\overline{BM0}$ und $\overline{BM1}$. Sind bereits $\overline{BM0}$ und $\overline{BM1}$ = L, so ist der Blockmodus bereits belegt und die periphere Einheit muß warten. Ist $\overline{BM0}$ = H oder $\overline{BM1}$ = H, so kann die periphere Einheit ihren Blocktransfer beginnen; sie aktiviert zusammen mit $\overline{BB}$ = L ein freies Signal $\overline{BM}$ = L und hält $\overline{BM}$ solange auf diesem Pegel, bis der ganze Block transferiert ist. Zwischendurch wird der Bus durch $\overline{BB}$ = H auch für ein eventuell vorhandenes zweites, schnelles Gerät freigegeben, das betreffende $\overline{BM}$-Signal jedoch aktiv gehalten. Auf diese Weise ist sichergestellt, daß während der Transferfolge die periphere Einheit den Bus mit $\overline{BB}$ in kurzer Zeit wieder belegen kann und nicht von anderen schnellen Geräten gestört wird. Benötigt ein schnelles Gerät für sich alleine den Vorrang am Bus, so aktiviert es beide $\overline{BM}$-Signale.

Interrupt-Signale $\overline{IR}$, $\overline{IA}$, $\overline{IL}$, $\overline{VC}$, $\overline{RT}$

Alle Teilnehmer am EA-Verkehr senden ihre Interrupt-Anforderung auf einer Sammelleitung. Die Annahme eines $\overline{IR}$ (interrupt request) zeigt die Zenraleinheit mit dem Kettensignal $\overline{IA}$ (interrupt acknowledge) an. Von der peripheren Einheit, die als erste in der Kette das Quittungssignal $\overline{IA}$ empfängt und ein $\overline{IR}$ gestellt hat, wird in einem EA-Lesezyklus der Zentraleinheit ihr Interrupt-Vektor über den Datenbus gelesen. Mit diesem Vektor identifiziert die Zentraleinheit den Interruptstellenden EA-Partner und der zugehörige Prozeß wird aktiviert.

Während dieser Phase des Vektorlesens blockiert die Zentraleinheit mit dem Signal $\overline{IL}$ (interrupt lock) alle $\overline{IR}$-fähigen EA-Moduln; es kann dort dann keine Veränderung des Interrupt-Zustands auf dem Bus vorgenommen werden (der Interrupt-Zustand ist eingefroren).

In den Zentraleinheiten der Modelreihe SICOMP sind zwei getrennte Interrupt-Kreise realsiert:

o Interrupt ($\overline{IR}$)
  Will eine periphere Einheit die Zentraleinheit über ein Ereignis informieren (z. B. Auftragsende) meldet sie sich mit dem Signal $\overline{IR}$ (interrupt request). Die Zentraleinheit liest mit einem EA-Lesezyklus den Vektor der peripheren Einheit, die den Interrupt gestellt hat. Dieser Vektor dient als Zeiger auf die Vektorliste (VEKLI) im Zentralspeicher, in der die Adresse des Geräteprozeßblocks eingetragen ist (eingetragen durch die Basisparametrierung). Den so adressierten Geräteprozeßblock hängt die Zentraleinheit (Mikroprogramm) in die Prioritätswarteschlange ein.

o Nicht maskierbarer Interrupt ($\overline{NMI}$, s. Kapitel 3.3.3.3)
  Diese Anforderung führt in den Zustand Z0 des Zentralprozessors und stößt dort ein Festwertspeicherprogramm an.

Folgende NMI-Signale sind am Bus als eigene Signalleitungen vorhanden:

o Virtuelle Konsole ($\overline{VC}$)
  Die Bedienung eines virtuellen Konsolen-Geräts führt zu einem VC-NMI, der das Festwertspeicherprogramm in der Zentraleinheit startet.

o Urladen $\overline{RT}$ (bootstrap)
  Der Urlade-NMI führt in das Festwertspeicherprogramm der Zentraleinheit, das das Urladen bearbeitet. Periphere Geräte oder Steuerungen, über die urgeladen werden kann, arbeiten mit diesem NMI.

Sondersignale

o Rücksetzen $\overline{RSP}$ (reset peripheral)
  Die Stromversorgung erzeugt beim Ein- und Ausschalten das Signal $\overline{RSP}$:

  a) Einschalten: $\overline{RSP}$ aktiv, bis die Versorgungsspannung stabil ist.

  b) Ausschalten: $\overline{RSP}$ aktiv bis zum stromlosen Zustand.

  Eine Service-Einrichtung an der Serviceprozessor-Schnittstelle kann

ebenfalls $\overline{RSF}$ erzeugen.

In der Zentraleinheit kann durch einen Schreibbefehl an die EA-Adresse 7 das Signal $\overline{RSF}$ aktiviert werden (Sammelrücksetzen der Peripherie).

$\overline{RSF}$ hat eine Mindestdauer von 5 µs.

Vor dem Rücksetzen müssen die Aktivitäten der peripheren Einheiten beendet werden, wenn Datenverluste vermieden werden sollen. Denn in den peripheren Einheiten kann das Signal $\overline{RSF}$ zu einem harten Rücksetzen führen. Das heißt, $\overline{RSF}$ wirkt auf die Rücksetzeingänge der Bauelemente, wodurch sämtliche Informationen verlorengehen.

Unabhängig von $\overline{RSF}$ kann mit einem EA-Schreibbefehl an einen logischen Kanal einer peripheren Einheiten dieser auch separat zum Rücksetzen veranlaßt werden (selektives Rücksetzen). Das Datum, das mit dem Ausgabebefehl übergeben werden muß, ist 3.

o Beenden $\overline{TER}$ (terminate)
Mit $\overline{TER}$ werden alle Busteilnehmer in den Passivzustand überführt.
$\overline{TER}$ hat eine Mindestdauer von 5 µs.

Nach der Aktivierung von $\overline{TER}$ haben die peripheren Einheiten noch mindesten 2 ms (wenn $\overline{HLD}$ auch aktiviert ist, sonst 0,5 s) Zeit, ihre Aktivitäten definiert zu beenden. Ist der Beendet-Zustand erreicht, werden von der periphere Einheit keine Datentransfers über den Bus mehr ausgeführt. Die periphere Einheit verhält sich solange passiv, bis die Zentraleinheit die Ausführung eines Auftrags fordert.

$\overline{TER}$ kann von einem Service-Mittel, über einen Ausgabe-Befehl an die EA-Adresse 6 vom Zentralprozessor oder von der Stromversorgung vor einem Spannungsausfall erzeugt werden.

Die Stromversorgung bildet 2 ms vor dem Spannungsausfall ($\overline{RSP}$) das Signal $\overline{PF}$ (power failure), aus dem das Signal $\overline{TER}$ unmittelbar abgeleitet wird.

o Anhalten $\overline{HLD}$ (hold)
$\overline{HLD}$ veranlaßt die Steuerungen, die aktuellen Tätigkeiten zu unterbrechen, bis $\overline{HLD}$ wieder deaktiviert wird. Dieses Signal stellt eine Vorwarnung dar; es wird nur von wenigen Geräten ausgewertet. Wenn das Signal $\overline{HLD}$ von der Stromversorgung erzeugt wird, wird es mindestens 10 ms vor dem Rücksetzen ($\overline{RSP}$) wegen Spannungsausfall aktiv. Geräte mit etwas längerer Reaktionszeit können sich damit rechtzeitig auf einen bevorstehenden Spannungsausfall aktiv einstellen.
Geräte mit kurzer Reaktionszeit müssen das Signal $\overline{HLD}$ nicht auswerten.
Kommt zum Signal $\overline{HLD}$ noch das Signal $\overline{TER}$, dann ist der Spannungsausfall auch durch Pufferung in der System-Stromversorgung nicht mehr überbrückbar und das Gerät muß seine Aktivitäten einstellen.
Wird $\overline{HLD}$ inaktiv ohne daß $\overline{TER}$ aktiv war, kann die periphere Einheit die Aktivitäten fortsetzen oder, falls dies nicht möglich ist, die Zentraleinheit über einen Interrupt verständigen (Auftragsabbruch).

Die Zentraleinheit erfährt $\overline{HLD}$ nur mittelbar durch den Zeitgeber. Der Zeitgeber der PROMEA® EA 01 erzeugt aus dem Signal $\overline{HLD}$ einen "normalen"

Interrupt an die Zentraleinheit mit dem Vektor 6. Dieser Interrupt aktiviert in der Zentraleinheit den Prozeßblock PRBNAUW (Prozeßblock für Netzausfallswarnung); das Betriebssystem wird über den bevorstehenden Netzausfall informiert.

o Batterie-Ausfall $\overline{BF}$ (battery failure)
Das Signal $\overline{BF}$ gibt Auskunft über den Ladezustand der Pufferbatterien. Es ist aktiv, wenn die Puffereinrichtungen entladen bzw. nicht vorhanden sind.
Beim Einschalten der Zentraleinheit hängt es unter anderem vom Signal $\overline{BF}$ ab, ob das Betriebssystem neu geladen werden muß (Wiederanlauf) oder ob das Betriebssystem noch im Zentralspeicher steht (Neustart oder Wiederaufsetzen).
Die Hardware in der Zentraleinheit aktiviert nach dem Einschalten den Urlader, wenn $\overline{BF}$ aktiv ist.

o Bussperrsignal $\overline{BL}$ (bus lock)
Das Wechseln von Baugruppen unter Spannung ist möglich, wenn vorher das Signal $\overline{BL}$ aktiviert wurde (Schalter an einer Zentralprozessor-Flachbaugruppe). $\overline{BL}$ bewirkt ein hardwaremäßiges Abschalten aller Busteilnehmer, auch während laufender Aktivitäten. Der sinnvolle Einsatz von $\overline{BL}$ setzt daher voraus, daß alle Aktivitäten des Systems beendet wurden.

o Reservierte Signale RES
Für zukünftige Erweiterungen des Systembus reservierte Signale.

o Taktsignale
Zentraltakt $\overline{CC}$ (central clock)
Systemgrundtakt mit einer Frequenz von 10 MHz zur Synchronisierung der Signalsequenzen (Bussteuerung, Speicherzugriffe).

o Versorgungsspannungen
Hauptversorgungsspannung ist +5V.
Für serielle Datenübertragungen stehen + 12V und - 12V zur Verfügung.
Optionell kann eine + 24V Spannung auf dem Bus geführt werden.

o Privatleitungen
In einem reservierten Bereich werden einige Privatleitungen geführt.

Name	Signalname	Anzahl / Richtung		deutsche Bezeichnung
$\overline{DB}$	data bus	32	TS ⟷	Datenbus
$\overline{AB}$	address bus	22	TS ⟷	Adreßbus
$\overline{TR}$	transfer request	1	OC ⟷	Transfer Anforderung
$\overline{TA}$	transfer acknowledge	1	OC ⟷	Transfer Rückmeldung
R/$\overline{W}$	read/write	1	TS ⟷	Lesen/Schreiben
M/$\overline{IO}$	memory/IO	1	TS ⟷	Adreßkennung
$\overline{FFA}$	fault flag address	1	OC ⟷	Begleitsignal Rückmeldung

Name	Signalname	Anzahl	Richtung	deutsche Bezeichnung
$\overline{FFD}$	fault flag data	1	TS/OC ⟷	Begleitsignal Rückmeldung
$\overline{AC\ 0,1}$	address control	2	TS ⟷	Adreß-Steuerung
$\overline{AE\ 0,1}$	address enable	2	⟶	Adreßfreigaben für EA-Betrieb
$\overline{BR}$	bus request	1	OC ⟷	Bus-Anforderung
$\overline{BAI}$	bus acknowledge input	1	K ⟶	Bus-Rückmeldung, Eingang
$\overline{BAO}$	" " output	1	K ⟶	Bus-Rückmeldung, Ausgang
$\overline{BB}$	bus busy	1	OC ⟷	Bus belegt
$\overline{BM\ 0,1}$	block-mode	2	OC ⟷	Blockmodus für schnelle Peripherie
$\overline{IR}$	interrupt request	1	OC ⟵	Unterbrechungsanforderung
$\overline{IA}$	interrupt acknowledge	2	K ⟶	Quittung der Unterbrechungsanforderung
$\overline{IL}$	interrupt lock	1	⟶	Unterbrechungsanforderung -Änderungssperre
$\overline{VC}$	virtual console	1	OC ⟵	Virtuelle Konsolen-Anforderung
$\overline{BT}$	bootstrap	1	OC ⟵	Urlader
$\overline{RSP}$	reset peripheral	1	OC ⟶	Rücksetzen der Peripherie
$\overline{TER}$	terminate	1	OC ⟶	Beenden
$\overline{HLD}$	hold	1	OC ⟶	Anhalten
$\overline{BF}$	battery failure	1	⟶	Batterie-Ausfall
$\overline{BL}$	bus-lock	1	⟶	Bus-Sperrsignal
$\overline{CC}$	central clock	1	⟶	Zentraler Takt
0 V				
+ 5 V				
+24 V	optionell			Versorgungsspannungen
+ 5 V	gepuffert			
+12 V				
-12 V				
PS				Privatsignale
RES				Reservierte Signale

Ruhezustand der Bussignale
H bzw. "High Z"

⟷ bidirektionale Signale
⟶ Signale zur Peripherie
⟵ Signale zum Zentralprozessor
TS Tristate-Signale
OC open collector-Signale
K Kettenleitungs-Signale

Tabelle 3.6: Systemschnittstelle

3.3.2 Peripherieklassen

Die peripheren Einheiten (besser: die Steuerungen der peripheren Geräte) sind zur Beschreibung der unterschiedlichen Leistungsfähigkeit in drei Klassen eingeteilt:

Klasse 1 — Direkter Anschluß des Geräts an die Zentraleinheit; das Gerät ist nicht DMA-fähig.

Klasse 2 — Direkter Anschluß des Geräts an die Zentraleinheit, das Gerät kann selbständig auf den Zentralspeicher zugreifen (DMA-fähig).

Klasse 3 — Die Steuerung enthält einen eigenen Prozessor, der neben dem DMA-Verkehr (reiner Datenverkehr) auch noch steuernde Aufgaben übernimmt und somit den Zentralprozessor entlastet.

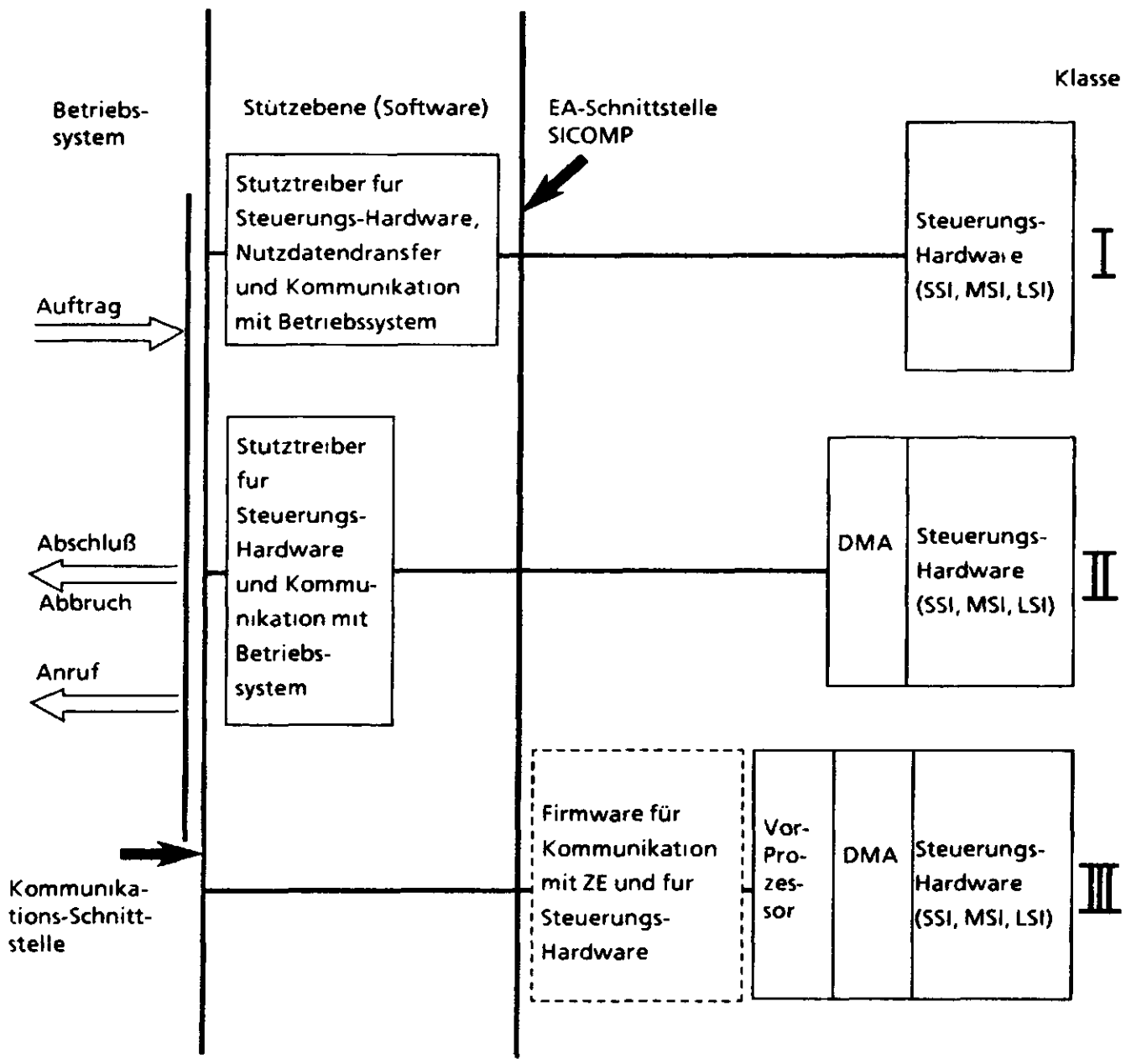


Bild 3.16 Peripherieklassen

Klasse 4 Prozeßsignalformer

Außer den Prozeßsignalformern werden zur Zeit nur Geräte der Klasse 3 an die ZE 03 angeschlossen. Die Prozeßsignalformer verkehren mit der Zentraleinheit nicht nach den im weiteren beschriebenen Prinzipien der Kommunikationsschnittstelle (KOSS), sondern nach eigenen Regeln.

Peripherie Klasse 1

Anschluß der Peripherie mit geringem Hardware-Aufwand. Die Datenraten sind meist gering.
Der Datenverkehr läuft programmgesteuert (kein DMA); d. h. der Datentransfer wird durch einen Stütztreiber durchgeführt. Dieser Stützteiber wickelt auch die Steuerung und Kommunikation mit dem Betriebssystem ab.
Die Adressierung der Geräte erfolgt über ihre EA-Adresse.
Die EA-Adresse ist auf den Steuerungen der Geräte einstellbar oder fest vorgegeben.

Peripherie Klasse 2

Anschluß von Peripherie mit niedrigem Hardware-Aufwand für Datenverkehr im Blockbetrieb. Die Unterstützung durch die Software in der Zentraleinheit (Stützteiber) ist auch in dieser Klasse notwendig. Der Nutzdatentransfer kann jedoch per DMA durch den DMA-Controller der peripheren Einheit durchgeführt werden, wodurch der Zentralprozessor entlastet wird.

Peripherie Klasse 3

Durch den Einsatz von Vorverarbeitungsprozessoren in Verbindung mit DMA-Verkehr wird eine weitere Entlastung der Zentraleinheit erreicht (Hardware und Software).

Ablauf

Die Zentraleinheit hinterlegt - wie auch bei Klasse 1 und 2 - in definierten Zentralspeicher-Zellen die Parameter zur Auftragsbeschreibung und fordert danach die periphere Einheit mit einem Ein-/Ausgabe-Befehl zur Auftragsbearbeitung auf.
Die periphere Einheit holt sich über DMA die Auftragsbeschreibung aus dem Zentralspeicher und führt den Auftrag wieder über DMA aus.
Anschließend werden Anzeigen von der periphere Einheit im Zentralspeicher abgelegt; die Zentraleinheit wird vom Auftragsende durch einen Interrupt informiert.

Periphere Geräte der Klasse 3 sind:

- o Programmierte Mehrfach-Anschaltung (PROMEA EA 01)
- o Plattenspeichersteuerungen
- o Magnetband-Kassettensteuerung
- o Datenübertragungssteuerungen

3.3.3 Die Kommunikationsmittel zwischen Zentraleinheit und peripherer Einheit

3.3.3.1 EA-Befehle

Jede periphere Einheit besitzt eine individuelle Adresse, die auf der Peripheriebaugruppe eingestellt wird. Falls das Schnittstellensignal M/$\overline{IO}$ anzeigt, daß der Adreßraum für die Peripherie angesprochen wird (M/$\overline{IO}$=L), vergleicht die periphere Einheit die Information auf dem Adreßbus der EA-Schnittstelle mit der auf ihrer Baugruppe eingestellten Adresse. Bei Gleichheit dieser Werte ist die periphere Einheit angesprochen. Die Signale $\overline{AEO}$ und $\overline{AEI}$ verringern den Vergleichsaufwand. Mittels Transfersteuersignalen kann über den Datenbus ein Datentransport zur peripheren Einheit (Schreiben) oder von der peripheren Einheit (Lesen) erfolgen.

Diese Befehle lassen sich für den Zentralprozessor als EAS (Ein-/Ausgabe schreiben, d. h. Schreiben in den Adreßraum für Peripherie) und EAL (Ein-/Ausgabe lesen, d. h. Lesen aus dem Adreßraum für Peripherie) formulieren und abarbeiten. Somit kann zentraleinheitsseitig gesteuert Information von der Peripherie abgeholt bzw. zur Peripherie transportiert werden.

Die Transfersequenz läuft nach dem Quittungsprinzip ab. Nach einem zeitlichen Vorlauf von Adreßbus $\overline{AB}$ und Steuersignalen (= Control-Bus $\overline{CB}$) leitet das Signal $\overline{TR}$ (transfer request) den Transfer ein.

Die passiven Busteilnehmer (slaves) können zur Erhöhung der Störsicherheit $\overline{AB}$ und $\overline{CB}$ zwischenspeichern, wenn $\overline{TR}$ aktiv wird.

Die Übernahme des Datums durch den angesprochenen Partner bzw. das Aufschalten der Informationen erfolgt mit dem Signal $\overline{TA}$ (transfer acknowledge).

Mit den Signalen $\overline{FFD}$ und $\overline{FFA}$ werden Transferbesonderheiten signalisiert.

Der Zentralprozessor überwacht die Zeit von $\overline{TR}$ bis $\overline{TA}$ (25 µs) und bildet Anzeigen, wenn die Quittungszeit überschritten worden ist.

Folgende Besonderheiten werden angezeigt:

a) Speicher-Verkehr

$\overline{FFA}$ Quittungsverzug (kein Speicher vorhanden)
Schreiben in den Festwertspeicher

$\overline{FFD}$ Nicht korrigierbarer Speicherfehler

Im Zentralprozessor führen diese Besonderheiten zu einem Eintrag in das Unterbrechungsanzeigenregister UAR und damit zu einer Programmlaufbesonderheit (PLB).

b) EA-Verkehr
Die Quittungsbegleiter $\overline{FFA}$ und $\overline{FFD}$ sind folgendermaßen codiert:

Code	$\overline{FFD}$	$\overline{FFA}$	PZR - Bit 0	PZR - Bit 1	Funktion
0	H	H	0	0	Quittung ohne Anzeigen
1	H	L	0	1	Quittungsverzug
2	L	H	1	0	Besonderheiten im EA-Modul
3	L	L	1	1	Vorabquittung

Die Quittungsbegleiter $\overline{FFD}$ und $\overline{FFA}$ erzeugen Ergebnisanzeigen im Programmzustandsregister PZR. Die Software kann durch die Auswertung dieser Anzeigen auf Besonderheiten reagieren.

Code 0 Normaler, fehlerfreier Abschluß einer Transfersequenz

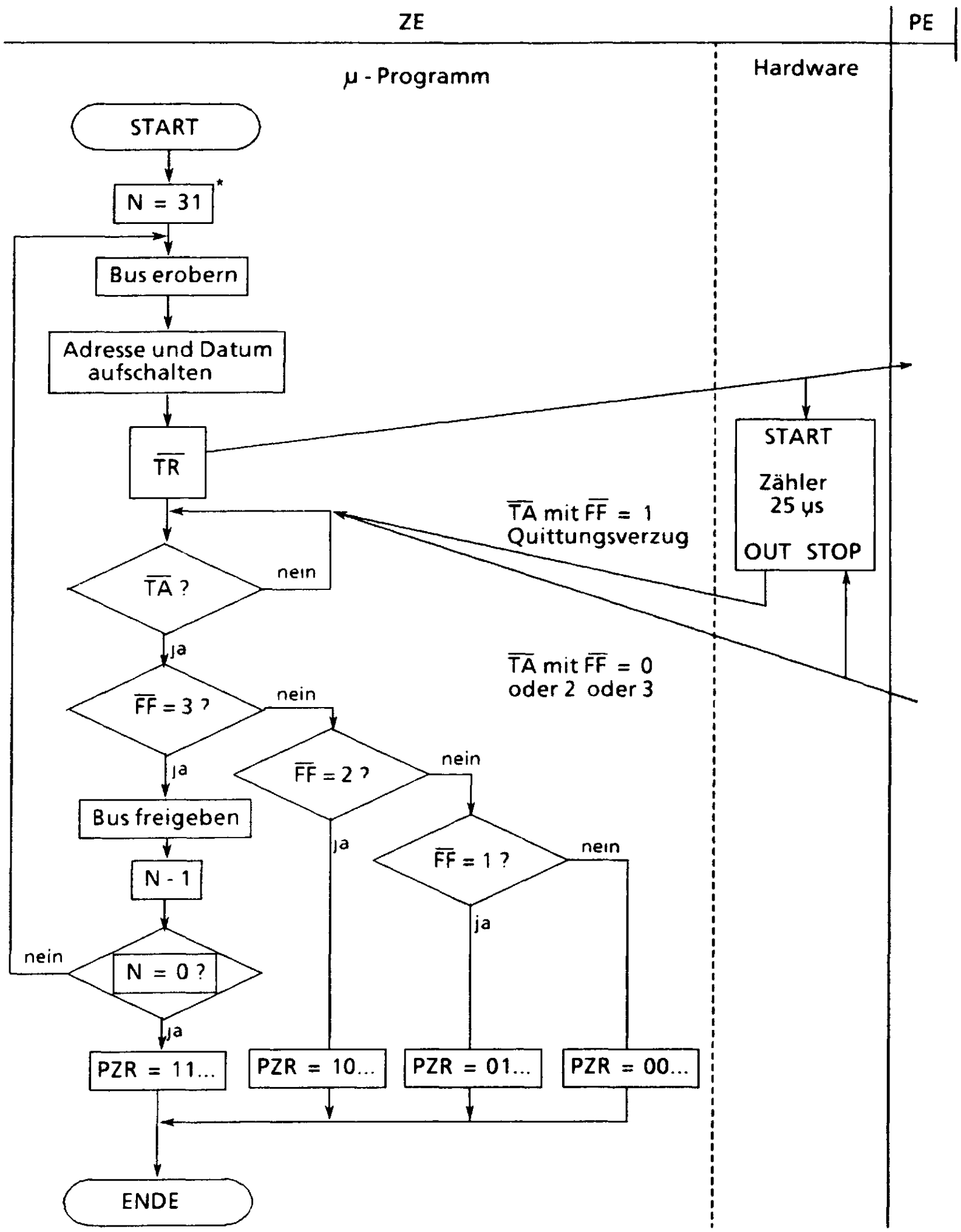
Code 1 Unter der angesprochenen Adresse befindet sich keine periphere Einheit, die Überwachung im Zentralprozessor hat angesprochen.

Code 2 Das angesprochene Gerät kann den gewünschten Transfer nicht ausführen. Die genaue Reaktion muß der Geräte-Spezifikation entnommen werden.

Code 3 Geräte die momentan nicht transferieren können, melden sich mit $\overline{TA}$ = aktiv und $\overline{FFD}$ = L, $\overline{FFA}$ = L beim Zentralprozessor. Das Mikroprogramm des Zentralprozessors wertet diese Anzeigen aus, gibt den Bus frei und versucht ihn sofort wieder zu belegen, um den Transfer noch einmal anzustoßen. Zwischendurch können aber andere periphere Einheiten den Bus belegen. Dieses Freigeben und Belegen versucht das Mikroprogramm des Zentralprozessors ca. 50 us lang und übergibt dann die PZR-Anzeige 3 (PRZ-Bit 0 und 1 = 1) an den Anwender.

Diese Vorgangsweise einer Vorabquittierung wird als "Stotterbetrieb" bezeichnet. Sollte während der 50 µs ein Transferversuch des Zentralprozessor erfolgreich verlaufen (die periphere Einheit quittiert normal), bemerkt der Anwender nichts von den erfolglosen Versuchen.

Anmerkung: Falls beim EA-Verkehr mit Prozeßperipherie irrtümlich die EA-Adresse einer angeschlossenen Einheit der Standardperipherie verwendet wird, wird dies nicht durch Anzeigen gemeldet.



* Die ZE versucht mindestens 50 us lang den Transfer durchzufuhren (abhängig davon, wie der Bus erobert werden kann)

Bild 3.17 Anzeigen im PZR beim EA-Verkehr

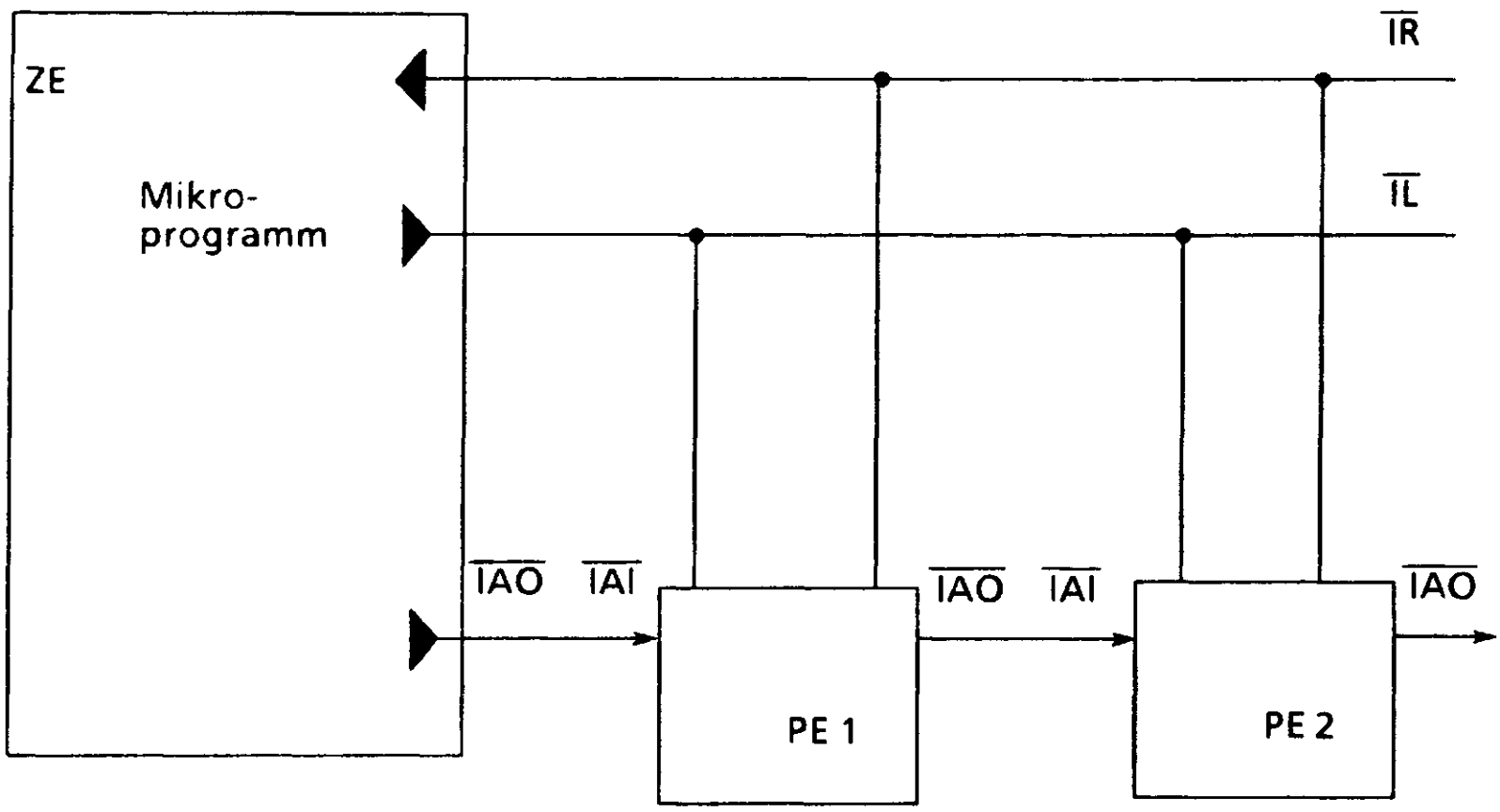


Bild 3.18 Interrupt-Logik

3.3.3.2 Interrupt

Immer dann, wenn periphere Einheiten eine Anforderung an die Zentraleinheit haben, stellen sie eine Interrupt-Anforderung. Gründe für solche Anforderungen können sein:

- Alarme von alarmbildenden Signalformern,
- Auftragsabschlußmeldung von Geräten,
- Anrufe von Geräten,

Die Anforderung an die Zentraleinheit erfolgt über das Sammelsignal $\overline{IR}$ (interrupt request). Alle Steuerungen, die einen Interrupt stellen, aktivieren das Signal $\overline{IR}$ (open collector). Das Mikroprogramm in der Zentraleinheit erobert in seiner Interrupt-Routine zunächst den Systembus und aktiviert dann das Signal $\overline{IL}$ (interrupt lock), das den aktuellen Interrupt-Zustand einfriert; neue Interrupt-Anforderungen dürfen und können nun nicht mehr gestellt werden. Aus dem Signal $\overline{IL}$ leitet die Hardware (Logik im Zentralprozessor) das Quittungssignal $\overline{IA}$ (interrupt acknowledge) ab.
Dieses Kettensignal ist über alle EA-Anschlußstellen für periphere Einheiten geschleift. Das erste Kettenglied, das die Quittung $\overline{IA}$ empfängt und einen $\overline{IR}$ gestellt hat, gibt $\overline{IA}$ nicht weiter (ähnlich dem Signal "bus acknowledge" $\overline{BA}$ bei der Buseroberung). Sie ist berechtigt, mit der folgenden EA-Lese-Sequenz ihren Interrupt-Vektor auf den Datenbus aufzuschalten. Der Zentralprozessor liest den Vektor vom Datenbus und adressiert, mit dem Vektor als Index, die Vektorliste VEKLI im Zentralspeicher.
Aus VEKLI liest der Zentralprozessor die Adresse des Geräteprozeßblocks (IOEVPRB-Adresse) und aktiviert diesen Prozeß, der die Interrupt-Ursache feststellt und darauf reagiert. Der IOEVPRB ist in jenem Bereich, dessen Struktur für alle Prozeßblöcke verbindlich festgelegt ist, genauso aufgebaut wie jeder andere Prozeßblock. Diese Maßnahmen werden durch die Prozeßsteuerhardware (Mikroprogramm) des Zentralprozessors durchgeführt. Die Vergabe der Vektoren an die Peripherie und das Eintragen der Geräteprozeßblockadressen in VEKLI geschieht durch die Basisparametrierung.

Nach der Übergabe des Vektors nimmt die periphere Einheit das Interrupt-Anforderungssignal $\overline{IR}$ weg und die Zentraleinheit deaktiviert die Signale $\overline{IL}$ und $\overline{IA}$; die Interrupt-Sequenz ist beendet.

Besondere Ereignisse, wie z. B. Netzausfall, Spannungswiederkehr, Wartungsfunktionen usw. werden nicht über das Signal $\overline{IR}$, sondern über den NMI gemeldet.

3.3.3.3 Nicht maskierbarer Interrupt (NMI)

Für höchstpriore Unterbrechungsereignisse verfügt die Zentraleinheit (Zentralprozessor) über einen speziellen, nicht maskierbaren Unterbrechungseingang. Das Signal NMI (nicht maskierbarer Interrupt) wird durch eines der folgenden Ereignisse hervorgerufen:

- Kommunikationsanforderung von Service-Einrichtungen,
- Anforderung virtuelle Konsole, Urladeanforderung,
- Netzausfall oder Batterie-Ausfall,
- Fortsetzen, Stop.

Daraufhin untersucht der Zentralprozessor - unabhängig vom aktuellen Ebenenzustand oder Stop - im Zustand höchster Priorität Z0 (Stop ist bereits Z0-Zustand) durch das Festwertspeicherprogramm die NMI-Ursache. Mit Hilfe eines Eingabebefehls an die EA-Adresse 0 wird ein externes Register (NMI-Register) gelesen, das die Unterbrechungsursache näher spezifiziert.

Aufgrund des übergebenen Bitmusters erfolgt eine Verzweigung in die verschiedenen Reaktionsprogramme. Längere Reaktionsprogramme in Z0 fragen in definierten Abständen das NMI-Register ab, um wichtige Interrupts (z. B. Netzausfall) rechtzeitig zu erkennen. Das NMI-Register ist in einem speziellen Schaltkreis eingebettet, um die Besonderheiten der von ihm festzuhaltenden Signale in geeigneter Weise zu berücksichtigen, d. h. z. B. auf Flanken, Impulse oder statische Signale zu reagieren.

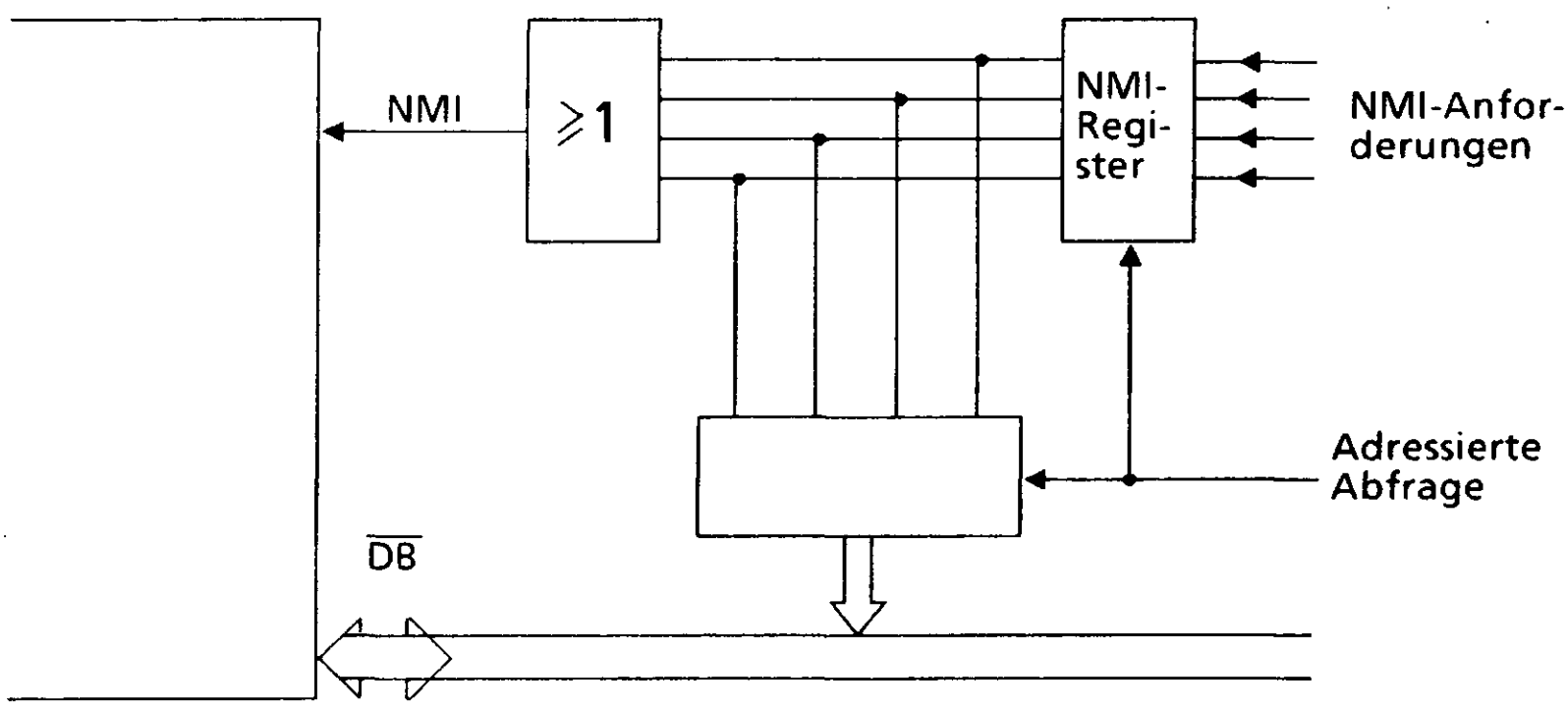


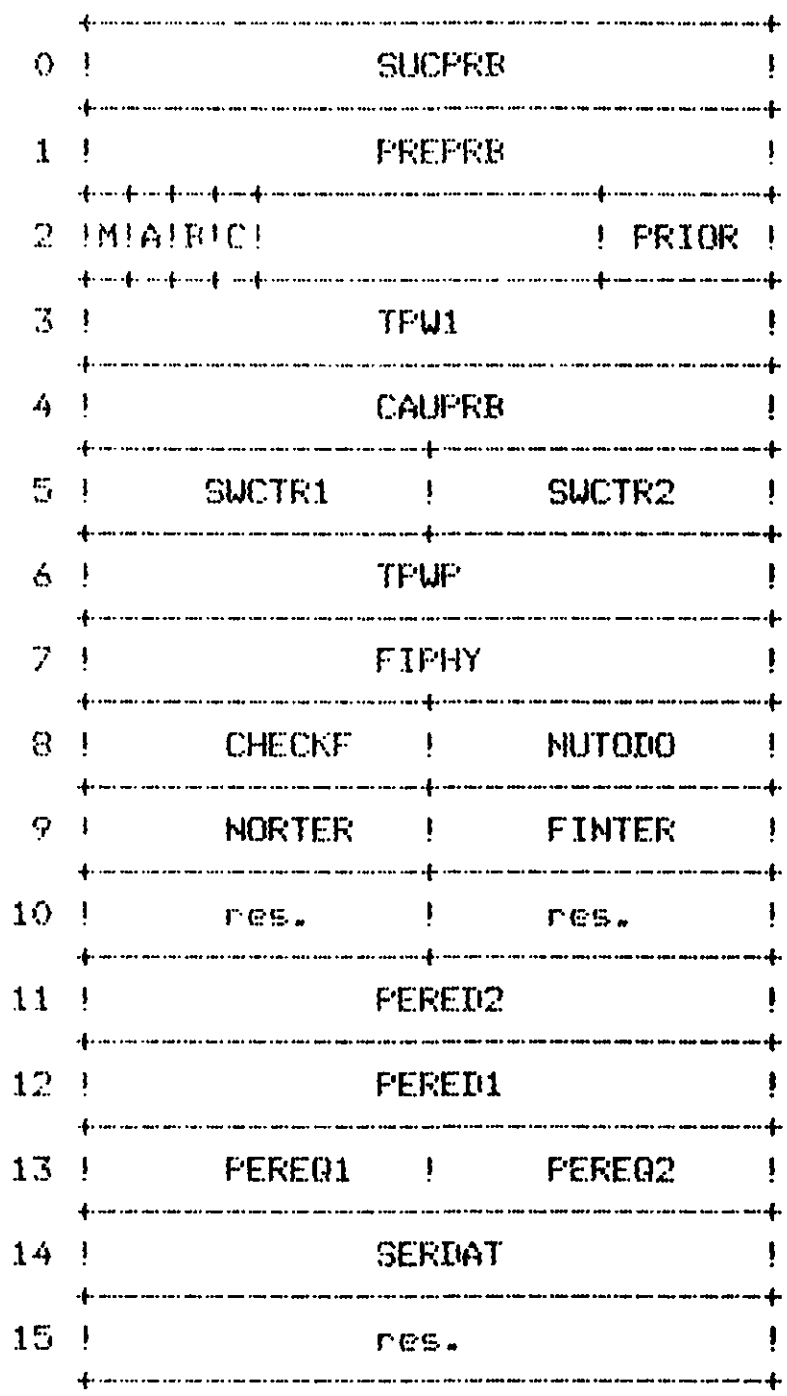
Bild 3.19: Nicht maskierbarer Interrupt (NMI)

3.3.3.4 Prozeßblock (PRB)

Um die Vorteile eines hardware-gesteuerten Unterbrechungssystems ausnützen zu können, ist eine weitgehend einheitliche Behandlung interner und externer Unterbrechungen notwendig. Die verschiedenen Vorgänge zur Informationsverarbeitung, die im Rechner ablaufen, (z. B. Anwenderprogramme, Reaktionen auf Fehler u.a.) werden als Prozesse bezeichnet. Um solche Prozesse durch Hardware und Software behandeln zu können, wird ein Feld definiert, in dem prozeßspezifische Kenngrößen festgehalten werden. Dieses Feld wird Prozeßblock (PRB) genannt. Dieser Prozeßblock ist in seiner Länge und Gesamtstruktur abhängig von der speziellen Funktion, die er charakterisiert. Ein Teil des PRB ist jedoch in jedem Fall gleich aufgebaut, egal um welche Art von PRB es sich handelt. Auf diesen Teil greift die Hardware der Unterbrechungssteuerung zu. Die Prozeßblöcke werden in zwei Gruppen unterteilt:

- PRB's für Programme
  Nach den ersten fünf Zellen folgen weitere Zellen zur Verständigung von Programmen untereinander (s. Bild 3.20 und Kapitel 3.1.5). Sie werden nur für Software-Zwecke verwendet, die Hardware greift nicht darauf zu. Die Anzahl und Bedeutung hängt vom jeweiligen Programm (Anwenderprogramm, ORG, Common Code usw.) ab.
  Bei dieser Art von PRBs dienen die ersten fünf Zellen zum Auffinden der Parametertafel PT und Übersetzungstafel ÜT 1 und bieten somit die Möglichkeit zum Auffinden des Befehlscodes und der Programmdaten.

- PRB's für periphere Geräte (Device-PRB, DEVPRB)
  Innerhalb dieser Gruppe sind zusätzlich zu den ersten fünf Zellen elf weitere Zellen in ihrer Bedeutung fest vereinbart, die u.a. auch auf den "Physikalischen Parameterblock" PHYSPB hinweisen (Kapitel 3.3.3.5). Dieser dient zur weiteren Charakterisierung des EA-Verkehrs. Der PRB selber wird als Referenzgröße benutzt, um mit ihm weitere den Prozeß charakterisierende Daten definiert festhalten zu können. Zusätzlich zu den erwähnten 16 Zellen kann der DEVPRB noch weitere Zellen umfassen, welche zur betriebssystem-internen, softwaremäßigen Verständigung verwendet werden.



SWCTR 1/2	software control 1/2 (Software-Steuerzelle 1/2)
TPWP	tablepointerword for peripheral (Tafelzeigerwort für Peripherie)
FIPHY	address pointer first PHYSPB (Adreßzeiger für den ersten PHYSPB)
CHECKF	check-field (Prüfzelle)
NUTODO	number of PHYSPBs to do (Anzahl der zu bearbeitenden PHYSPBs)
NORTER	normal termination (Abschluß)
FINTER	final termination (Beenden)
PEREQ 1/2	peripheral request 1/2 (Anforderung von der Peripherie)
PEREI 1/2	peripheral request's data 1/2 (Daten zur Anforderung von der Peripherie)
SERDAT	service data (Service Daten)
res.	reserved (reserviert)

Bild 3.20 Device process block (DEVPRB, Geräteprozeßblock)

Allgemeine Begriffe Auftrag, Anruf, Abschluß:

Wird im Zuge des EA-Verkehrs die Zentraleinheit initiativ, geschieht das in der Form der Erteilung eines "Auftrags" der Zentraleinheit an die Peripherie. Ein Auftrag wird durch einen PHYSPB näher beschrieben.
Wird im Zuge des EA-Verkehrs die periphere Einheit initiativ, geschieht dies in Form eines "Anrufs" der peripheren Einheit an die Zentraleinheit. In den Zellen PEREQ des Prozeßblocks wird die Art des Anrufs spezifiziert, danach wird der Anruf durch einen Interrupt der peripheren Einheit gestellt. Die Anrufe werden in Anrufklassen unterteilt und durch den Anrufcode präzisiert.
Nach Durchführung eines Auftrags durch eine periphere Einheit wird eine Abschlußbearbeitung durchgeführt, wobei Anzeigen abgelegt werden und ggf. ein Abschlußinterrupt gestellt wird (Kapitel 3.3.4).

Zellen SWCTR 1 und SWCTR 2
(software control 1 und 2)

Die Zellen SWCTR 1 und 2 enthalten diverse Steuerinformationen:

	INBI			
	0	00	Beendet - Interrupt unterdrücken	
		01	Beendet - Interrupt stellen	
	1	10	res.	
		11	res.	
	2	00	Kein Abbruch des Anlauftests (Wiederanlauf)	
		01	Abbruch des Anlauftests (Neustart)	
	3	10	res.	
SWCTR2		11	res.	
	4			
	5			
		nicht belegt		
	6			
	7			
	8	00	Abschlußinterrupt unterdrücken	
		01	Abschlußinterrupt stellen	
	9	10	res.	
		11	res.	
	10	00	res.	
		01	res.	
	11	10	res.	
		11	res.	
SWCTR1	12	00	keine Auftragskettung	
		01	res.	
	13	10	Auftragskettung: sequentiell abarbeiten	
		11	Auftragskettung: zeitoptimal abarbeiten	
	14	0	Adresse des PHYSPB virtuell	
		1	Adresse des PHYSPB reell	
	15	0	Verweis auf PHYSPB ungültig	
		1	Verweis auf PHYSPB gültig	

Die Bedeutung der einzelnen Steuerbits ist folgende:

- Bit 0 und 1 steuern, ob nach einem Anstoß an die periphere Einheit zum Beenden noch ein Beendet-Interrupt von der peripheren Einheit gestellt wird oder nicht.

- Bit 2 und 3 geben an, ob während des Anlauftests ein bei der peripheren Einheit eintreffender Auftrag diesen Anlauftest abbrechen darf oder ob der Auftrag zurückgewiesen wird. Der Anlauftest läuft im Rahmen eines Diagnose-Auftrags oder automatisch bei Spannungswiederkehr ab.

- Bit 8 und 9 legen fest, ob nach einem ungeketteten Auftrag bzw. dem letzten Teilauftrag einer Kette ein Abschlußinterrupt gestellt wird.

Die Bedeutung der anderen Bits ergibt sich aus ihrer Benennung.

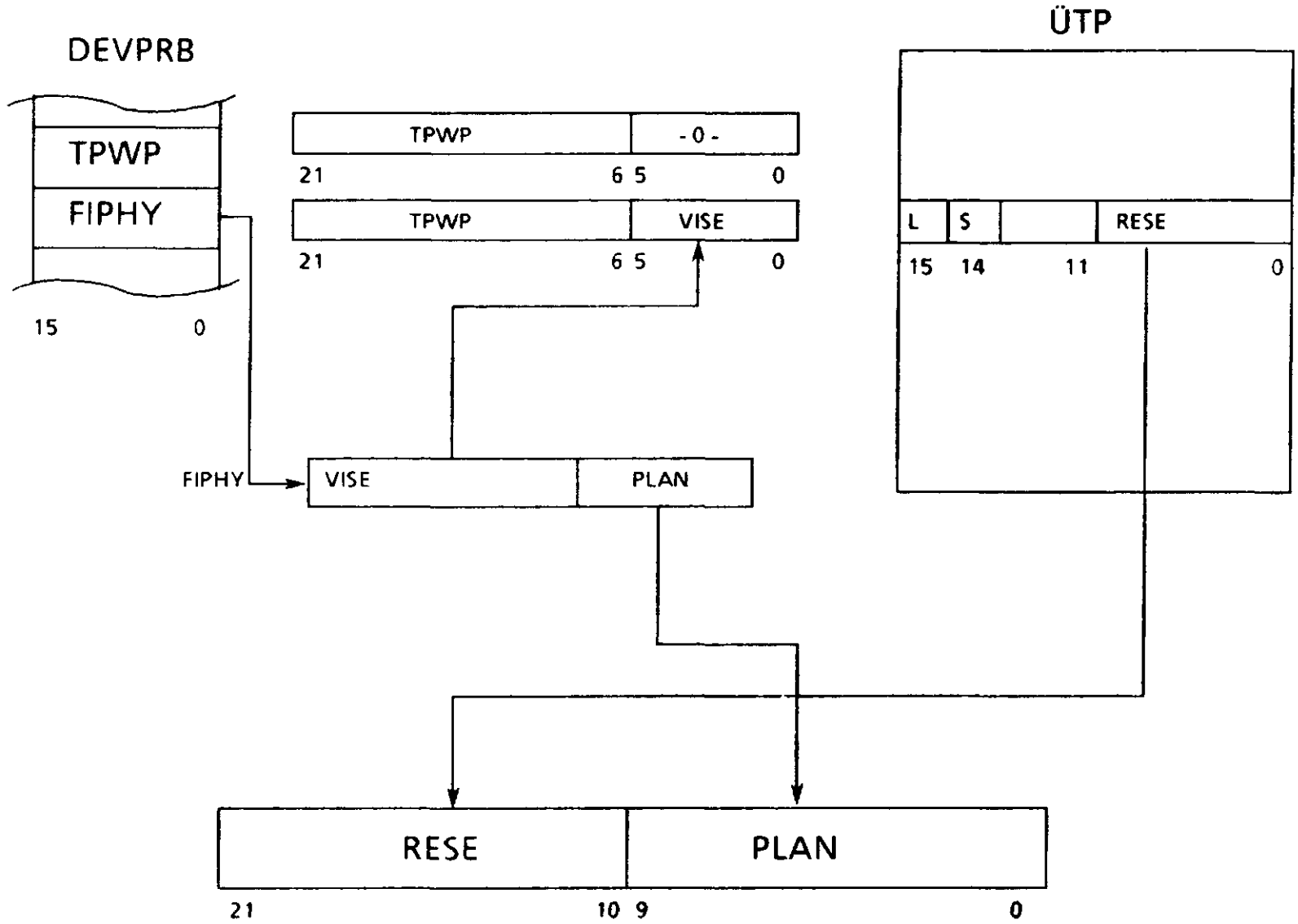


Bild 3.21 Adressierung des PHYSPB

Zellen TPWP und FIPHY

Diese beiden Zellen dienen zum Auffinden des bzw. der PHYSPBs:

TPWP ... tablepointerword for peripheral
FIPHY ... addresspointer first PHYSPB

Ist Bit 14 in SWCTR1 = 1, enthält FIPHY die reelle Anfangsadresse des ersten PHYSPB (TPWP = 0).
Ist Bit 14 in SWCTR1 = 0, ist der Inhalt von FIPHY als virtuelle Adresse zu interpretieren. Zum Errechnen der reellen Adresse läuft ein analoger Vorgang wie bei der Zentralspeicheradressierung ab:

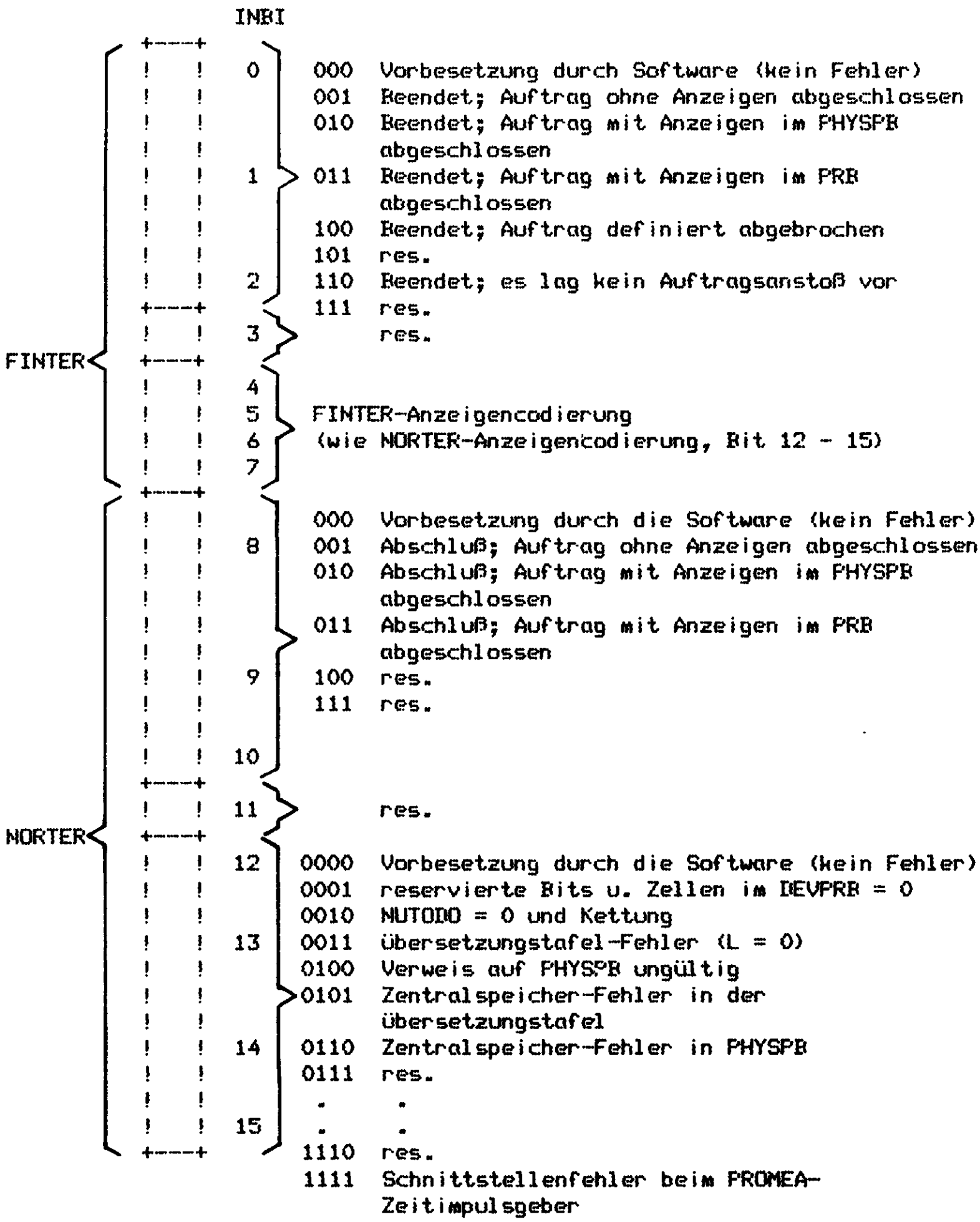
Das Bit L der Elemente der Übersetzungstafel für den PHYSPB ÜTP zeigt an, ob die Seite geladen ist (L = 1). Es wird von der peripheren Einheit überprüft. Das Schreibschutzbit (S-Bit) wird nur von der Zentraleinheit ausgewertet (analog zur Zentralspeicheradressierung).

Da dieser Adressierungsmechanismus auf der peripheren Einheit abläuft, muß die periphere Einheit mit den nötigen Daten aus dem Hauptspeicher versorgt werden und die Adreßrechnung durchführen.

Zellen CHECKF und NUTODO
(check-field und number of PHYSPBs to do)

Check-Field enthält eine Prüfsumme für Zelle 5, Zelle 6 und Zelle 7 des DEVPRB. Sie wird nach folgendem Algorithmus gebildet:
Die sechs Bytes SWCTR1, SWCTR2, TPWP (high), TPWP (low), FIPHY (high), FIPHY (low) werden byteweise summiert, wobei der bei einer Addition ggf. entstehende Übertrag sofort rechtsbündig zur Zwischensumme addiert wird. Die Endsumme wird invertiert und in das Feld CHECKF eingetragen. NUTODO gibt die Anzahl der Teilaufträge an, die im Falle einer Auftragskettung insgesamt durchzuführen sind. Jeder Teilauftrag wird durch einen PHYSPB gekennzeichnet.

Zellen NORTER und FINTER
(normal termination und final termination)



Die Zellen NORTER und FINTER enthalten Informationen für zwei verschiedene Arten eines Auftragsendes.

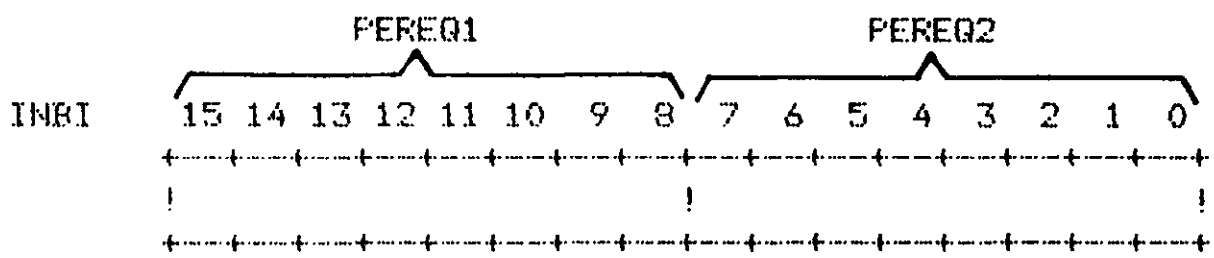
- NORTER umfaßt Meldungen nach einem normalen Abschluß des EA-Verkehrs, d. h. wenn er nicht durch Sammel- oder selektives Beenden vorzeitig beendet wurde;

- FINTER umfaßt Meldungen, wenn der EA-Verkehr durch Sammelbeenden oder selektives Beenden vorzeitig beendet wurde.

Es wird angezeigt, ob und wo Anzeigen abgelegt sind, es wird insbesondere auch angezeigt, wenn ein Sammel- oder selektives Beenden durchgeführt wurde, ohne daß ein Auftrag überhaupt angestoßen war (Code 110 in FINTER Bit 0-2).

NORTER und FINTER sind mit 0 vorbesetzt, da nur in diesem Fall die periphere Einheit die Erlaubnis hat, Anzeigen einzutragen. Die Zellen dienen als schnelle Weichen für die EA-Software.

Zellen PEREQ1 und PEREQ2
(peripheral request 1 und 2)



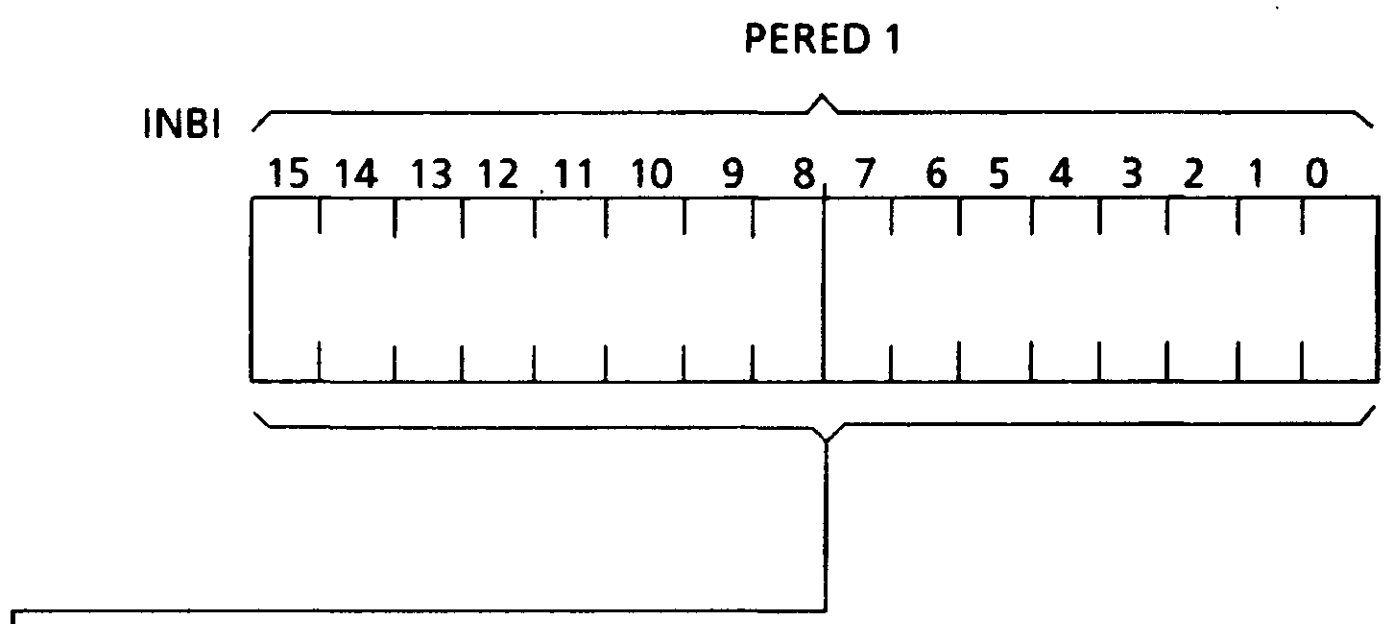
Wie bereits erwähnt sind die Anrufe in mehrere Klassen eingeteilt. Innerhalb einer Klasse sind Anrufe zusammengefaßt, die ähnliche Anforderungen in Richtung Software bzw. Zusatzinformationen (PERED, SERDAT) stellen. Wenn die periphere Einheit nach Eintrag in PEREQ und evtl. PERED und SERDAT den Grund des Anrufs eingetragen hat, stellt sie den Interrupt. Das daraufhin ablaufende Reaktionsprogramm (z. B. Treibersoftware) kann aus diesen Zellen die Interrupt-Ursache entnehmen und entsprechend reagieren.
Die Bedeutung der einzelnen Codes ist aus nachfolgender Tabelle ersichtlich. Genaue Details sind aus den Gerätebeschreibungen zu entnehmen.

PERED 1/2 ist mit 0 vorbesetzt, da nur in diesem Fall die periphere Einheit hier Eintragungen vornehmen darf.

Eintrag in PEREQ1	Eintrag in PEREQ2	Anruf Code (hexa)	Anruferereignis
		00	PEREQ 1/2 darf von PE überschrieben werden (Eintrag durch ZE)
		01	
		.	>> res.
		.	
		0E	
X		0F	Netzausfallsfrühwarnung Ende
X		10	Dateneingabe ohne Blocklängenangabe
X		11	Anzeigeneingabe
X		12	Kurzzeitwecker abgelaufen
	X	13	Zyklischer Wecker abgelaufen
X		14	Pufferspannungsausfall bei Gleitpunkt-impulsgeber der PROMEA ohne Datenverlust (Zeit noch gültig)
X		15	Pufferspannungsausfall bei Zeitimpulsgeber mit Datenverlust (Zeit einstellen)
X		16	Senderichtung frei/Puffer frei
X		17	Drohender Linetrace-Pufferüberlauf
X		18	Spannungswiederkehr am Laufwerk
		19	
		.	>> res.
		2B	
X		2C	Fehlermeldung wurde übergeben
X		2D	Mithörpuffer wurde übergeben
X		2E	Serviceprozessor-Anschaltung Information
X		2F	
X		30	Dateneingabe mit Blocklängenangabe
		31	
		.	>> res.
		.	
		3F	
X		40	Uhrzeit der ersten Externsynchronisation
		41	
		.	>> res.
		.	
		4D	
X		4E	Serviceprozessor-Anschaltung Zustandsänderung
X		4F	Virtuelle Konsole-Freigabeanforderung
X		50	Drohender Linetrace-Pufferüberlauf
		51	
		.	>> res.
		.	
		6F	
	X	70	Fehler bei Selbsttest (PRB-unspezifisch)
	X	71	Fehler bei Selbsttest (PRB-spezifisch)
	X	72	Grenzwertüberschreitung bei Inline-Statistik

Eintrag in PEREQ1	Eintrag in PEREQ2	Anruf Code (hexa)	Anruferereignis
		73	
		.	
		.	>> res.
		.	
		8F	
X		90	EAS-Datum unzulässig
X		91	Subadresse unzulässig (nur bei VK-Auftrag, Soust-NOP)
X		92	CHECKF-Fehler im PRB
		93	res.
		94	Mehrfachanstoß (Anstoß an tätige Peripherie)
		95	Fehler bei Knopfdrucktest
		96	
		.	>> res.
		99	
		A0	HSP-Fehler bei Zugriff auf VEKLI
		A2	HSP-Fehler bei Zugriff auf DEVPRB
		A3	
		.	
		.	
		.	
		.	>> res.
		.	
		.	
		.	
		FF	

Zellen PERED 1 und PERED 2
(peripheral request's data 1 und 2)



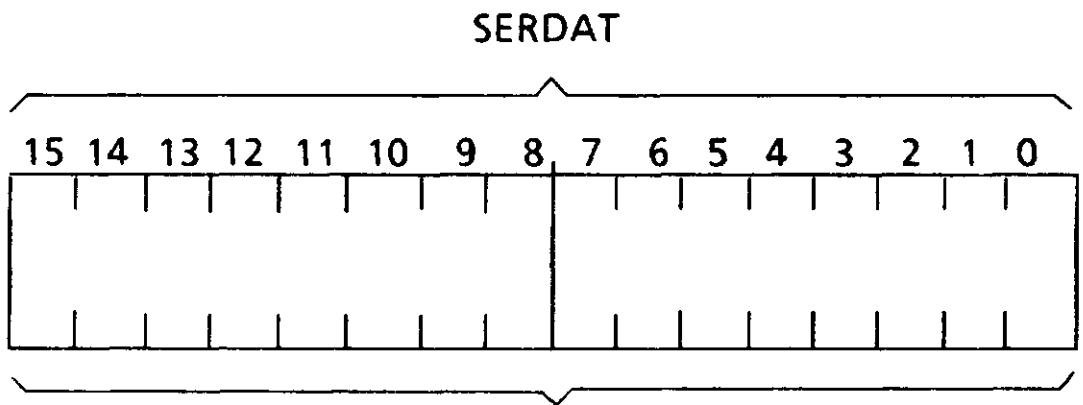
Anruf-Code	Zusatzinformation
30	Blocklänge in Byte
90 - AF	Vektor des anrufstellenden Geräts

PERED 1 enthält Zusatzinformationen zu den verschiedenen Anrufen laut PEREQ 1 und PEREQ 2. Die Bedeutung der hier verwendeten Anrufcodes ist ebenfalls aus PEREQ ersichtlich.

PERED 2 ist für zukünftige Anforderungen reserviert.

Zelle SERDAT (service data)

Periphere Geräte verfügen über Selbsttestroutinen. Ergebnisse daraus werden in SERDAT abgelegt.



Anruf-Code	Zusatzinformation
70 71	INBL 7 6 5 4 3 2 1 0 Fehlercode, der in den einzelnen Selbsttestroutinen gebildet und diesen zugeordnet werden kann.
72	INBL 15 14 13 12 11 10 9 8 Zählervariable res. Die Zählervariable gibt die Art des Fehlers an.

3.3.3.5 Physikalischer Parameterblock (PHYSPB)

Durch den PHYSPB wird in Weiterführung der im Prozessblock enthaltenen Information die Art des EA-Verkehrs näher beschrieben (z. B. serielle Standardeingabe binär, Steueroperationen für Plattenspeichereinheit/ Magnetband-Kassetteneinheit, Parametrierung der Anschaltung/Steuerung usw). Er kennzeichnet Aufträge an das periphere Gerät. Der PHYSPB bildet die dynamisch veränderliche Komponente des EA-Verkehrs im Rahmen der Kommunikationsschnittstelle. Er wird vor einem Auftrag dem DEVPRB zugeordnet. Gekennzeichnete Aufträge können gekettet werden. In diesem Fall können sich sämtliche Teilaufträge gemeinsam am Schluß mit einem einzigen Interrupt melden (Kapitel 3.3.4.1).
Die Bedeutung der Zellen des PHYSPB unterscheidet sich je nach Auftrag. Es gibt jedoch einige Zellen, deren Bedeutung einheitlich ist; das sind die ersten 5 Zellen (Zellen 0-4) und das Anzeigenfeld (Zellen 14-17, Bild 3.22).

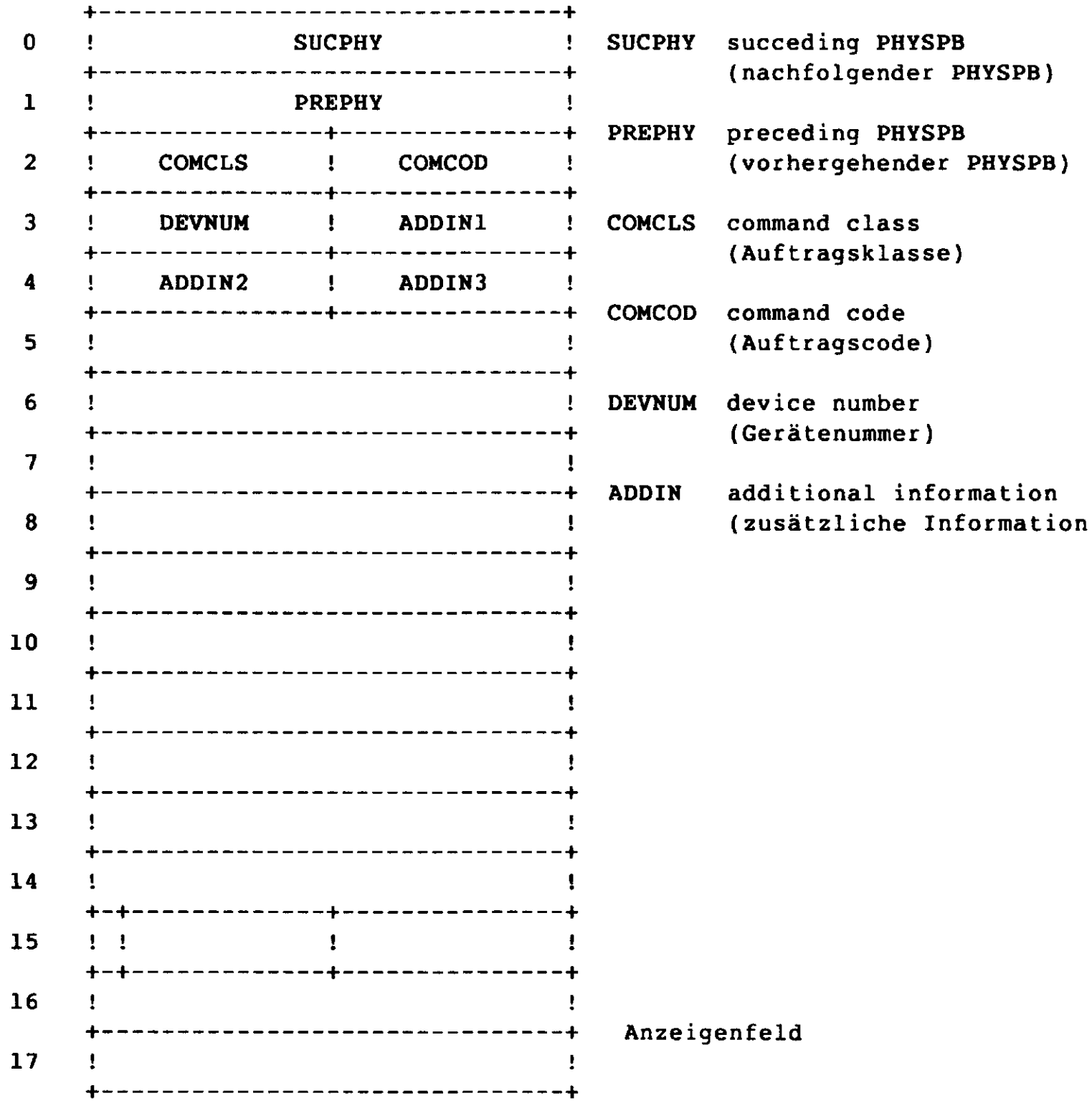
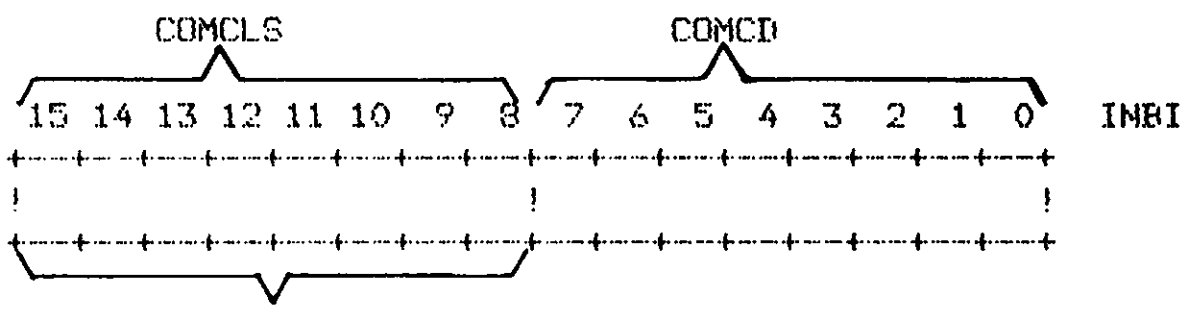


Bild 3.22 Physikalischer Parmeterblock PHYSPB

Zellen SUCPHY und PREPHY
(succeeding PHYSPB and preceding PHYSPB)

SUCPHY verweist auf die Anfangsadresse des (im Falle einer PHYSPB-Kettung) folgenden PHYSPBs, PREPHY auf die Anfangsadresse des vorausgehenden PHYSPBs.

Zellen COMCLS und COMCOD
(command class und command code)



Auftragsklassen:

hex.	
02	Diagnose/Statistik
04	Parametrierung
06	Transfer mit seriellem Zugriff
08	Urladen
0A	Steueroperationen für Plattenspeicher, Magnetband, Magnetbandkassette
0C	Steueroperationen für Datenübertragungssteuerungen, Terminals
0E	Transfer mit Direktzugriff
10	Operationen mit 2 peripheren Einheiten
12	Datenträgermanipulation
14	Firmware laden
16	Zeitfunktionen

COMCOD unterteilt - je nach Gerät - die einzelnen Auftragsklassen (COMCLS) in verschiedene Varianten, z. B. serielle Standardausgabe binär oder serielle Standardausgabe mit Steuerzeichenerkennung, jeweils aus der Auftragsklasse "Transfer mit seriellem Zugriff".

Geräteabhängige Elemente

Zelle DEVNUM (device number): z. B. Stationsadresse bei Datenübertragungssteuerungen

Zellen ADDIN 1, 2, 3 (Additional Information 1, 2, 3; auftragsabhängig)

Das Anzeigenfeld (Zellen 14 - 17) hat folgenden Aufbau

```
     !15 14          8! 7             0!  INBI
     +---+------------+----------------+
  14 !A  ! TERCOD     !     SWFLAG     !
     +---+------------+----------------+
  15 !                HWFLAG            !
     +--------------------------------+
  16 !                STATUS            !
     +--------------------------------+
  17 !             DELAY-CODE           !
     +--------------------------------+
```


Vielfach fällt auf, daß den Codierungen der gleiche Name zukommt wie Codierungen in NORTER oder FINTER.
Dabei sind die Aussage des physikalischen Parameterblocks spezifisch für einen Teilauftrag innerhalb einer Auftragskette, die Aussagen des Prozeßblocks hingegen für den Gesamtauftrag. Die Bedeutungen fallen zusammen, wenn keine Auftragskettung vorliegt.

Zellen TERCOD und SWFLAG
(termination code und software flag)

INBI

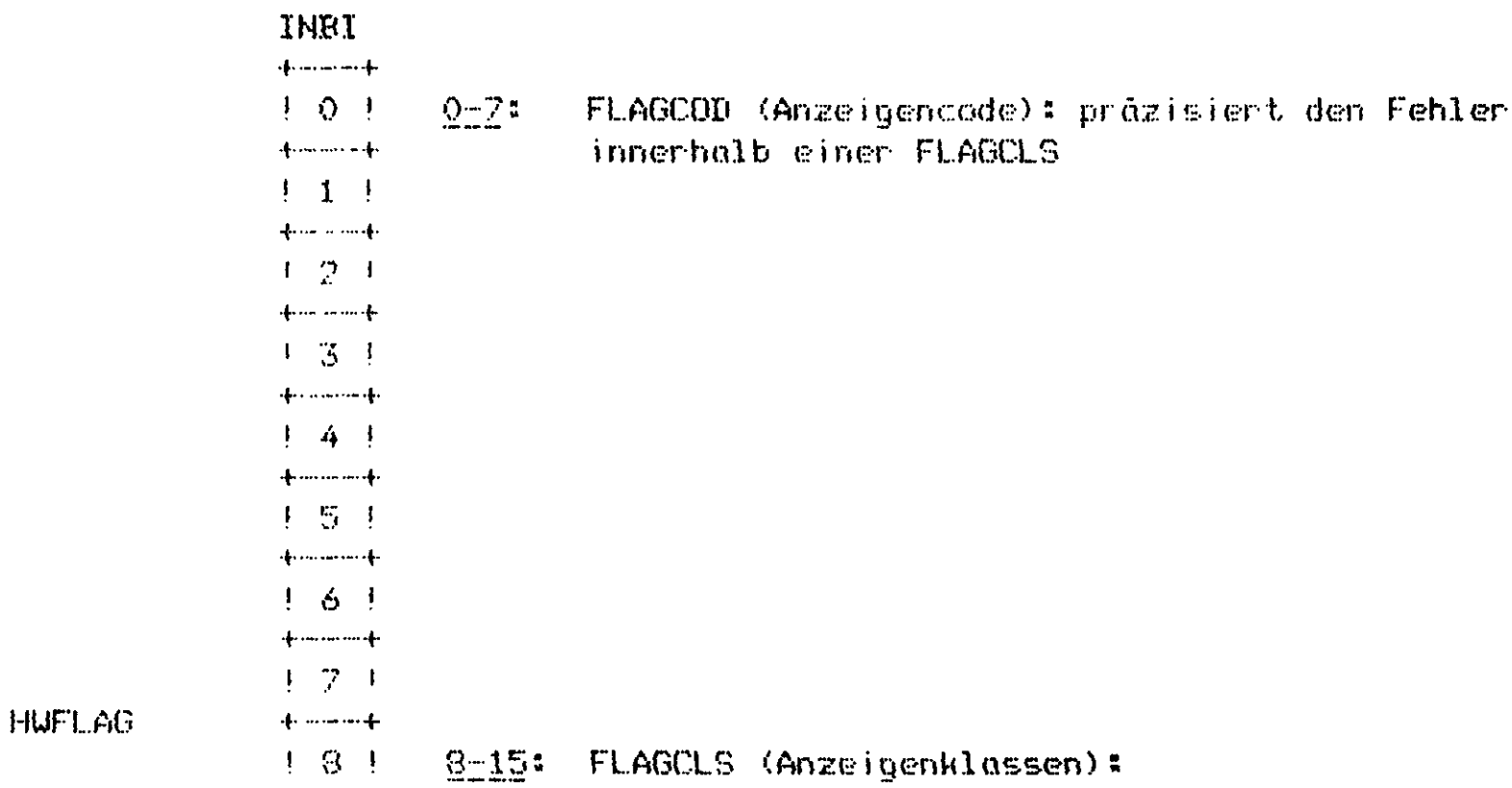
SWFLAG

! 0 !	0-7:	Informationen, die von der Peripherie aus der jeweiligen Hardware-Anzeige (HWFLAG) für die Software abgeleitet werden. Daraus kann das Betriebssystem folgendes bestimmen: - die logischen Aufrufanzeigen, - betriebssystem-interne Reaktionen und - Fehlermeldungen.
! 1 !		
! 2 !		
! 3 !		
! 4 !		
! 5 !	8-10:	000 Vorbesetzung durch Software
! 6 !		001 Auftrag ohne Anzeigen abgeschlossen
! 7 !		010 Auftrag mit Anzeigen im PHYSPB abgeschlossen
		011 res.
		100 Auftrag wurde definiert abgebrochen
! 8 !		101 res.
! 9 !		110 res.
!10 !		111 res.

TERCOD

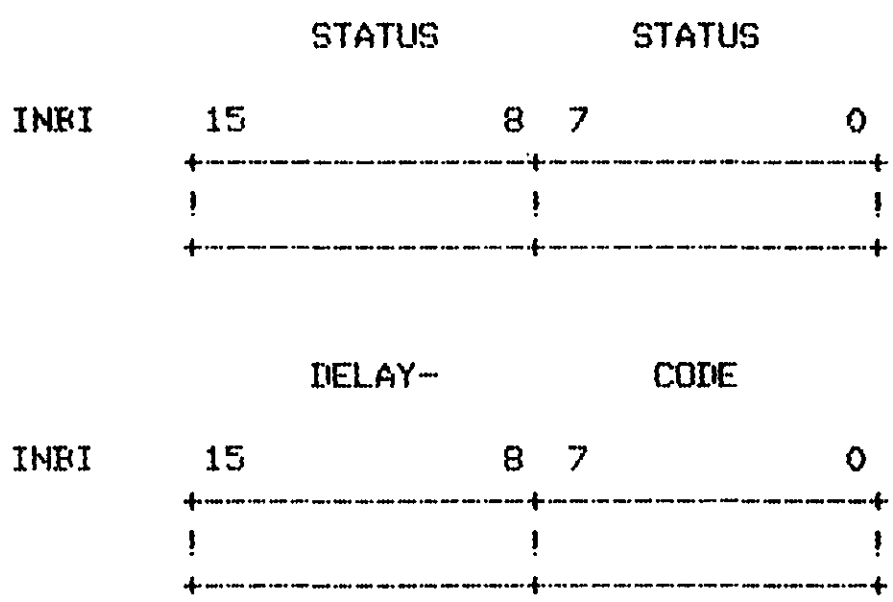
!10 !		
!11 !	11:	res.
!12 !	12:	0 Wort 16, 17 nicht relevant 1 Wort 16, 17 relevant
!13 !		
!14 !	13-14:	res.
!15 !		
	15:	Byte-Kennung für BUPOA (s. d.)

Zelle HWFLAG (hardware-flag)



Die peripheren Einheiten legen ihre Anzeige im Hardware-Flag in dual codierter Form ab. Der Aufbau dieser Anzeige orientiert sich an der Zuteilung zu einer Anzeigenklasse (linkes Byte der Anzeige) und der laufenden Fehlernummer innerhalb dieser Anzeigenklasse (rechtes Byte der Anzeige, FLAGCOD). Eine durch die periphere Einheit festgestellte Anzeigenkombination wird auf eine einzige, eindeutige Hardware-Anzeige in Form einer Fehlernummer abgebildet. Die Einzelanzeige mit dem niedrigsten Klassen-Codewert bestimmt die Anzeigenklasse der Anzeigenkombination für den Fall, daß innerhalb eines Auftrags mehrere Anzeigen aufgetreten sind.

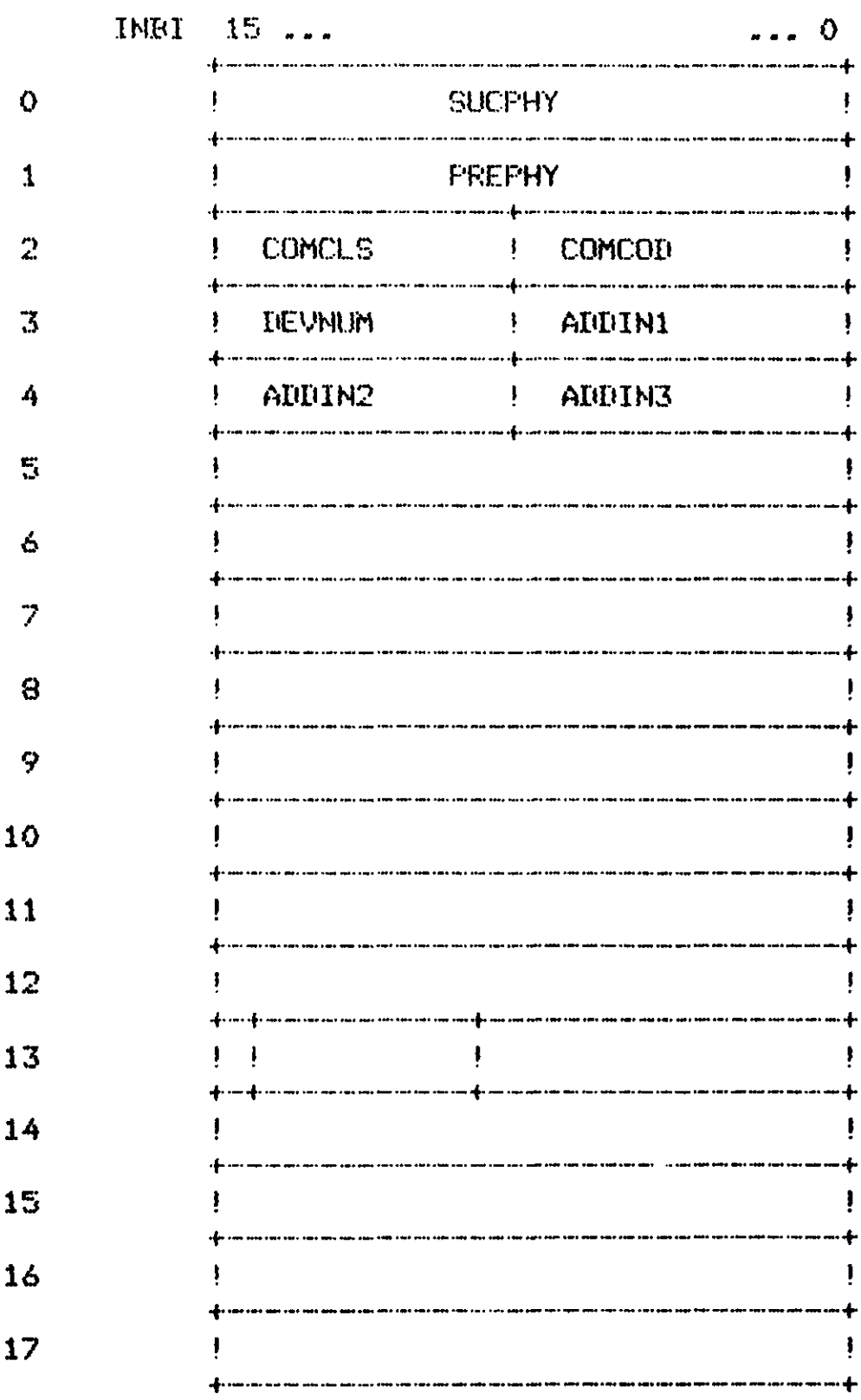
Zellen STATUS und DELAY-Code



Diese Zellen finden von Gerät zu Gerät unterschiedliche Verwendung.

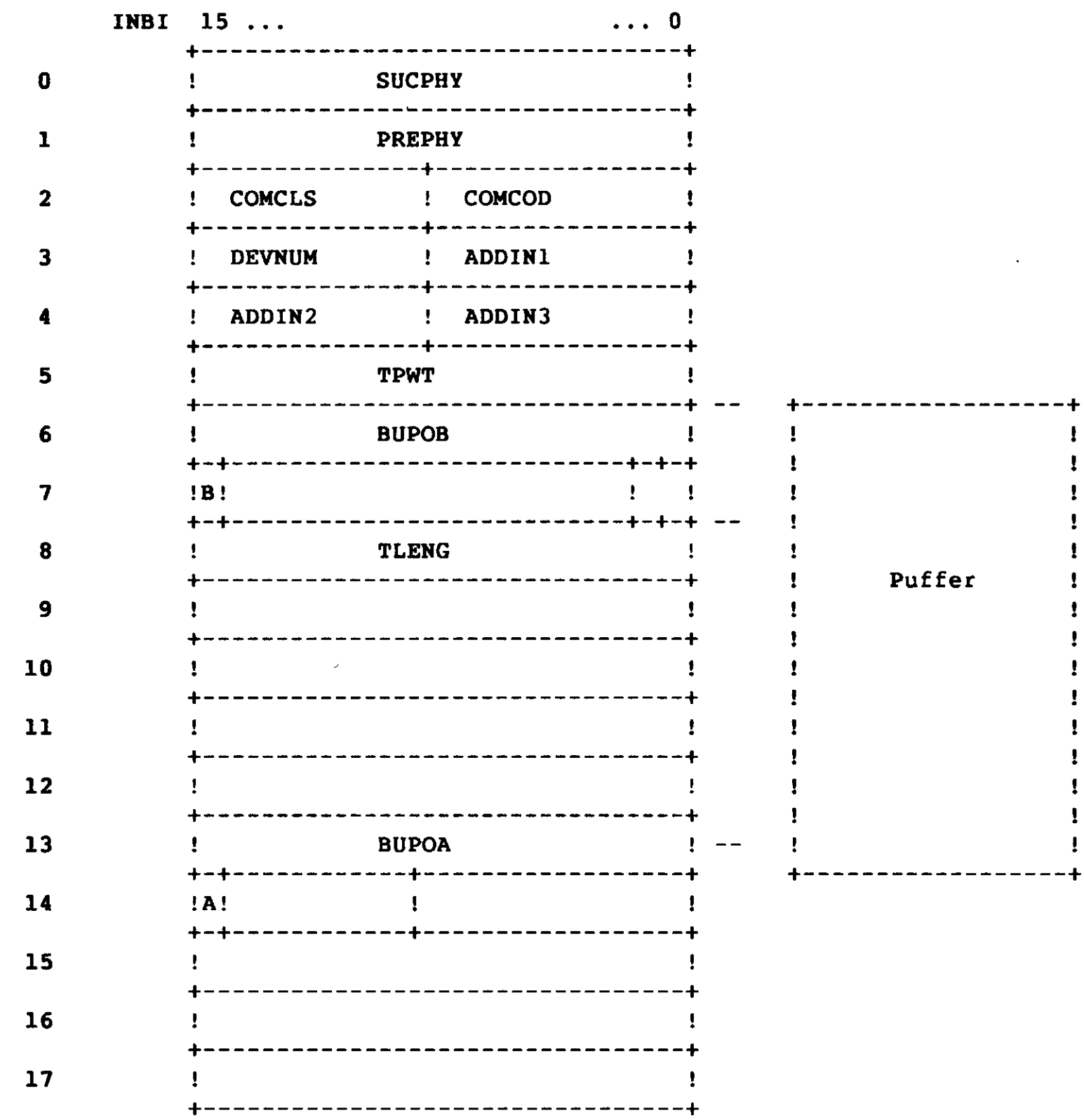
Die Bedeutung der weiteren Zellen des PHYSPB (Zellen 5-13) ist zwar im Prinzip von der Art des Transfers abhängig. Jedoch können anhand von Beispielen einige Zellenbelegungen herausgegriffen werden, die in vielen Fällen von übergreifender Bedeutung sind.

Aufbau eines physikalischen Parameterblocks (PHYSPB) ohne weitere Informationen in den Zellen 5-13:



Die Operationen sind direkt im COMCLS und COMCOD festgehalten, z. B. Steueroperationen für Plattenspeicher, Magnetband und -kassette, Datenübertragungssteuerungen, Terminals.

Aufbau eines PHYSPB mit Verweis auf Puffer im Zentralspeicher:



TPWT tablepointerword transferbuffer (Tafelanzeigewort für Transferpuffer)
BUPOB bufferpointer before transfer (Pufferzeiger vor Transfer)
BUPOA bufferpointer after transfer (Pufferzeiger nach Transfer)
TLENG tansfer length (Transferlänge in Bytes)
B Byte-Kennung für BUPOB
A Byte-Kennung für BUPOA

Dieser Typ findet Verwendung für Parametrierung (Puffer für Parameter), Transfer mit seriellem Zugriff (Transferpuffer), Datenträgermanipulationen (Puffer für Zusatzdaten) und Firmware laden (Puffer für Firmware-Vorspann).

Ein Puffer kann über den physikalischen Parameterblock (PHYSPB) sowohl virtuell als auch rein reell adressiert werden. Die reelle Adressierung ist so ausgelegt, daß auf die reellen Adressen auch mit dem virtuellen Adressierverfahren zugegriffen werden kann. Alle peripheren Einheiten beherrschen deshalb die virtuelle Adressierung, können aber dort, wo es vorgesehen ist, über die reelle Adressierung einfacher und schneller zugreifen.

Die Übersetzungstafel ÜTT (Übersetzungstafel für Transfer, Bild 3.23), auf die der Tafelzeiger TPWT (tabelpointerword for transfer, Tafelanzeiger für Transfer) im PHYSPB verweist, ist immer vorhanden. Bit 14 in Zelle 7 des PHYSPB gibt an, ob virtuell adressiert werden muß, oder ob auch reell adressiert werden kann.

Ist in Zelle 7 des PHYSPB INBI 14 = 0, so muß virtuell adressiert werden. Die Daten können über verschiedene Bereiche zu je 1024 Adressen ("Seiten") verteilt sein. Der Übergang an den Bereichsgrenzen muß nicht stetig sein.

Es ist L = 1, wenn die Seite geladen ist, d. h. das Betriebssystem hat eine gültige reelle Seite (RESE) eingetragen und im Hauptspeicher eine Seite für die periphere Einheit zur Verfügung gestellt. Die periphere Einheit darf nur auf geladene Seiten zugreifen (Lesen oder Schreiben). Das Schreibschutz-Bit S kann 0 oder 1 sein; es wird nur von der Zentraleinheit benutzt und von der peripheren Einheit nicht ausgewertet.

Ist INBI 14 = 1, so liegen die Daten in aufeinanderfolgenden "Seiten". Der Übergang an einer "Seiten"-Grenze ist stetig. In Zelle 7 des PHYSPB ist die reelle Seitennummer für die Anfangsadresse eingetragen; aber auch die Übersetzungstafel ÜTT ist gültig. Damit ist eine vereinfachte und raschere Adressierung möglich, da nicht auf die ÜTT zugegriffen werden muß.

Die Zelle für die Endadresse BUFOA wird vom Betriebssystem gleich der Anfangsadresse BUFOB vorbesetzt. Wenn Daten transferiert wurden, trägt die periphere Einheit in BUFOA einen um die Zahl der transferierten Bytes erhöhten Wert ein. Die Länge des Transferpuffers wird durch TLENG angegeben. TLENG ist 18 bit breit, und zwar 16 bit aus Zelle 8 des PHYSPBs sowie INBI 0 und INBI 1 der Zelle 7. INBI 1 aus Zelle 7 ist das höchstwertige Bit, dem INBI 0 (Zelle 7), INBI 15 (Zelle 8) bis INBI 0 (Zelle 8) folgen. Dabei ist 128 die größte zulässige Zahl, die in TLENG enthalten sein darf. Somit kann der Transferpuffer maximal 128 byte umfassen.

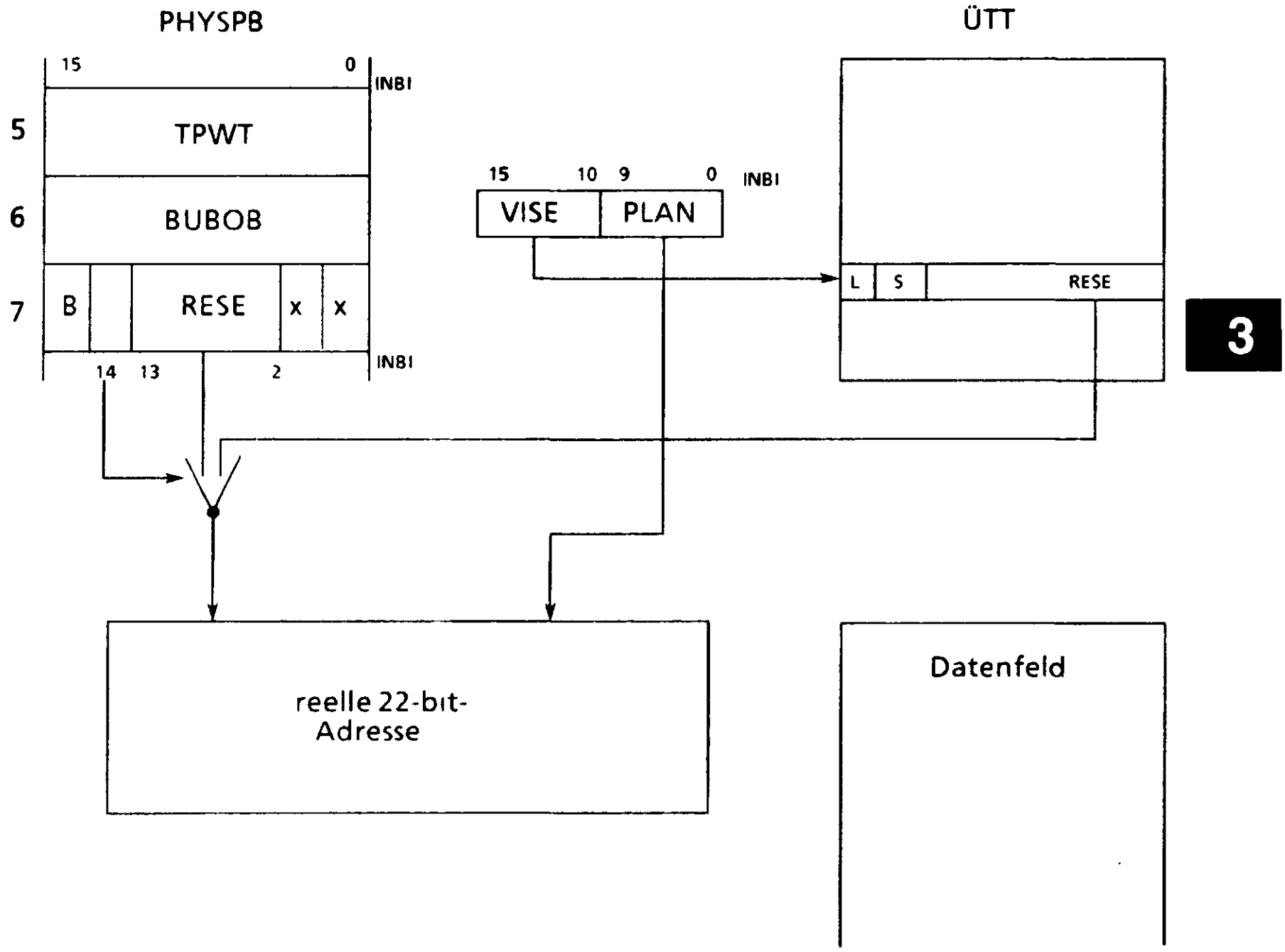


Bild 3.23: Übersetzung der Pufferadressen

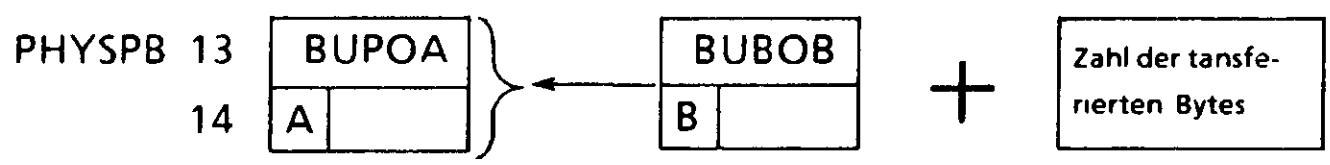
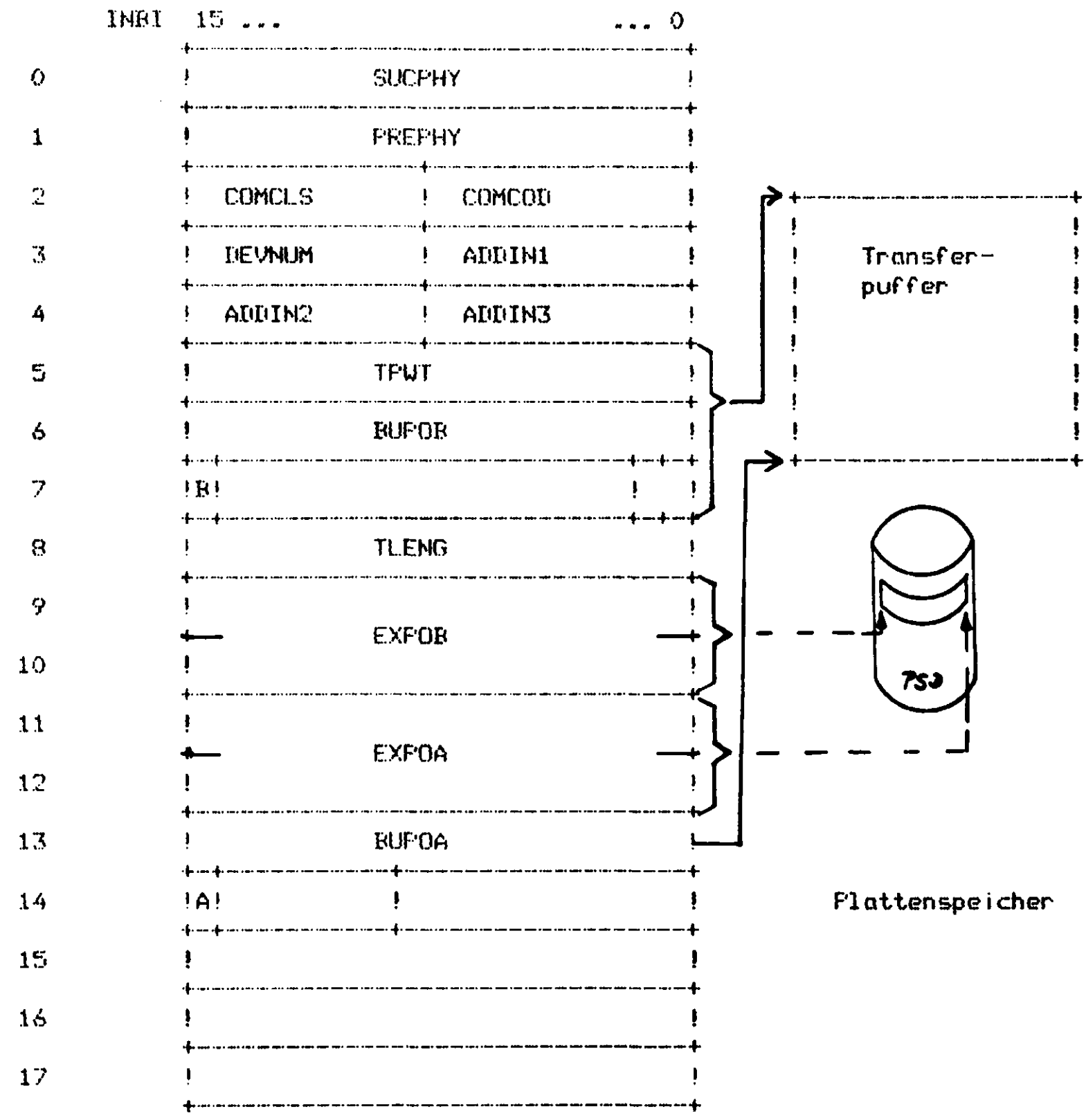


Bild 3.24: Bezeichnung der Endeadresse BUPOA

Aufbau eines PHYSPBs mit Puffer im Zentralspeicher und Externpuffer.

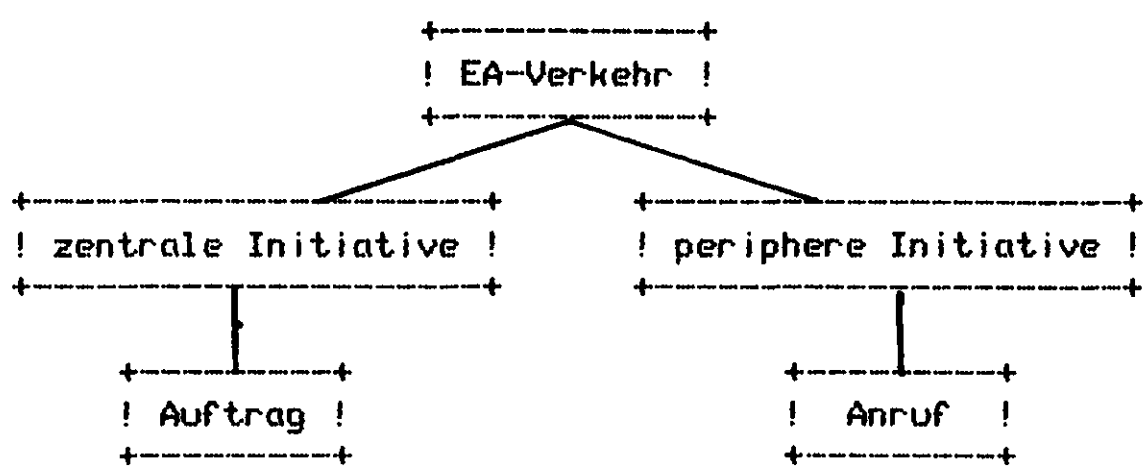


EXPOB externpointer before transfer (Externzeiger vor Transfer)
EXPOA externpointer after transfer (Externzeiger nach Transfer)

Diese Art von PHYSPBs findet Verwendung bei Transfers mit Direktzugriff und teilweise auch bei Eingabe von Urladedaten.

3.3.4 Ablauf der Kommunikation zwischen Zentraleinheit und Peripherie

Für die Abwicklung der Kommunikation zwischen Zentraleinheit und Peripherie stehen die beschriebenen Mittel zur Verfügung (PRB, PHYSPB usw). Welche dieser Mittel dabei zum Einsatz gelangen hängt davon ab, ob der EA-Verkehr durch zentrale oder durch periphere Initiative zustande kommt. Im ersten Fall spricht man von einem Auftrag der Zentraleinheit für die Peripherie, im letzteren Fall stellt die Peripherie einen Anruf an die Zentraleinheit.



Das Zusammenspiel im Rahmen des EA-Verkehrs zwischen Zentraleinheit und Peripherie hat dabei oft folgende Form:

Die Peripherie präzisiert ihren "Wunsch" durch Eintragungen im DEVPRB und tätigt dann einen Anruf durch Stellen ihres Interrupts. Daraufhin wird die Zentraleinheit ihrerseits initiativ und formuliert einen Auftrag, an dessen Ende ein Abschlußinterrupt stehen kann.

3.3.4.1 Zentrale Initiative

Ein Peripherie-Auftrag besteht im Prinzip aus folgenden Phasen:

- Auftragsvorbereitung	von ZE
- Auftragsanstoß	von ZE
- Auftragsinterpretation	von PE
- Auftragsausführung	von PE
- Anzeigenübergabe	von PE
- Unterbrechungsanforderung	von PE
- Abschlußbearbeitung	von ZE

ZE ... Zentraleinheit PE ... periphere Einheit

3.3.4.1.1 Auftragsvorbereitung

Ein Anwenderprogramm beauftragt über einen Aufruf das Betriebssystem, einen Datentransfer mit der Peripherie durchzuführen.

Das Betriebssystem bereitet den DEVPRB (device process block) und den PHYSPB (physikalischer Parameterblock) der betroffenen peripheren Einheit für den Transfer auf. Im DEVPRB werden folgende Daten für die periphere Einheit hinterlegt:

SWCTR — Die von der peripheren Einheit gewünschte Reaktionsweise wird eingetragen, z. B. Abschlußinterrupt stellen.

TPWP — Hier wird der Tafelzeiger für die Übersetzungstafel eingetragen, unter der die periphere Einheit ihren PHYSPB findet.

FIPHY — Enthält die virtuelle Adresse des PHYSPB.

CHECKF — Enthält die Prüfsumme über SWCTR, TPWP und FIPHY.

NUTODO — Enthält die Zahl der PHYSPBs, die bearbeitet werden müssen (bei Auftragskettung).

NORTER
FINTER — Sind mit 0 vorbesetzt; die periphere Einheit kann hier Anzeigen hinterlegen.

PEREQ — Wird erstmalig (Systemhochlauf) mit 0 vorbesetzt; dann können Anrufe entgegengenommen werden, nach deren Entgegennahme wieder 0 eingetragen wird.

Im PHYSPB werden folgende Daten hinterlegt:

SUCPHY,
PREPHY — Enthalten den Kettenverweis vorwärts und rückwärts bei Auftragskettung.

COMCLS — Enthält die Auftragsklasse des auszuführenden Auftrags.

COMCOD — Enthält die Auftragsvariante, die den Auftrag einer Klasse weiter spezifiziert.

DEVNUM — Enthält z. B. eine Teilgerätenummer.

ADDIN — Enthält auftragsspezifische Zusatzinformationen.

Darüber hinaus sind noch - je nach Auftragsklasse - weitere Informationen nötig, wie z. B.

TPWT — Tafelzeiger für die Übersetzungstafel des Transferpuffers

BUPOB — bufferpointer before transfer

BUPOA — bufferpointer after transfer

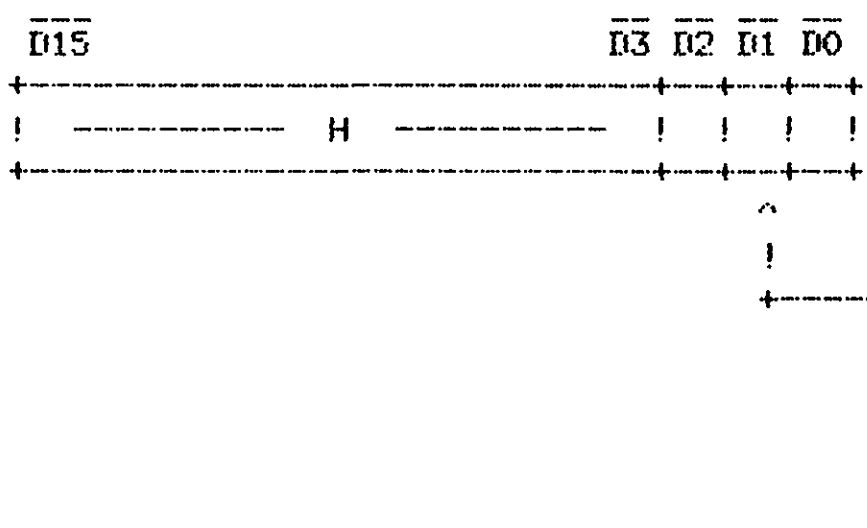
TLENG — Transferlänge in Bytes

Neben der Vorbereitung von DEVPRB und PHYSPB müssen gegebenenfalls noch andere Vorbereitungen getroffen werden, wie z. B. das Laden der Übersetzungstafeln für PHYSPB und Transferpuffer.

3.3.4.1.2 Auftragsanstoß

Nachdem die Vorbereitungen getroffen worden sind, gibt das Betriebssystem an die EA-Adresse der gewünschten peripheren Einheit einen EA-Schreibbefehl (EAS) mit dem Anstoß-Code 001 (hexa) ab.

Diese Information auf dem Datenbus wird nicht als Datum, sondern als Steuerinformation verstanden. Insgesamt gibt es dabei folgende Möglichkeiten:



000	NOP (no operation)
001	Anstoß
010	Beenden (selektiv)
011	Rücksetzen (selektiv)
100	VK-Auftrag
101	VK-Auftragsrücknahme
110	Error-Test
111	nicht belegt

VK virtuelle Konsole

Im Normalfall erhält eine periphere Einheit erst dann wieder einen neuen Auftragsanstoß, wenn sie ihren alten Auftrag abgeschlossen und das der Zentraleinheit mit Anzeigen oder Interrupt mitgeteilt hat.

Die periphere Einheit kann den EA-Transfer mit Anzeigen quittieren (FFA, FFD), die Besonderheiten beim EA-Verkehr signalisieren (s. Kapitel 3.3.3.1).

Wenn der Ausgabebefehl ohne Anzeigen abgeschlossen wird, kann das Betriebssystem davon ausgehen, daß der Ausgabebefehl von der peripheren Einheit ohne Besonderheiten angenommen wurde. In den anderen Fällen muß ein Fehlerreaktionsprogramm über das weitere Vorgehen entscheiden.

Nach erfolgtem Anstoß wartet die Zentraleinheit auf die Rückmeldung der Peripherie, die je nach Auftragsart über Anzeigen oder Interrupt erfolgen kann. Währenddessen führt die Zentraleinheit andere Aufgaben durch.

Einer Anschaltung (Steuerung), die mehrere logische Kanäle enthält, sind in der Regel mehrere EA-Adressen und auch mehrere Geräteprozeßblöcke (DEVPRBs) zugeordnet. Jeder Kanal kann unabhängig von anderen Kanälen Aufträge erhalten.

Neben dem Anstoßauftrag können noch andere Aufträge wie Beenden und Rücksetzen an eine periphere Einheit abgegeben werden. Die Zentraleinheit muß eine periphere Einheit auch dann ansprechen können, wenn diese bereits tätig ist. Nur so ist gewährleistet, daß die Zentraleinheit das Gesamtsystem führen kann. Deshalb ist es notwendig, daß die periphere Einheit auch dann noch EA-Transfers annimmt, wenn sie bereits einen Auftrag bearbeitet. Die Reaktion der peripheren Einheit ist in diesem Fall jedoch von Gerät zu Gerät unterschiedlich. Der Auftrag kann z. B. mit Anzeigen quittiert werden oder auch angenommen und zwischengespeichert werden.

Ein Auftrag kann aus mehreren Teilaufträgen bestehen, die über eigene physikalische Parameterblöcke (PHYSPB) gekettet sind. Nur für den Gesamtauftrag wird hier ein Auftragsanstoß abgegeben und ggf. erfolgt auch nur zum Schluß eine Interrupt-Meldung.

3.3.4.1.3 Auftragsinterpretation

Hat eine periphere Einheit über EA-Transfer ein Datum übernommen, so muß sie dieses zunächst dekodieren. Je nach Code ist der Auftrag mit diesem Datum schon vollständig beschrieben (z. B. Beenden, Rücksetzen, NOP) oder bedarf noch weiterer Erläuterung (Anstoß, VK-Auftrag, d. h. für virtuelle Konsole). Die zusätzliche Information findet die periphere Einheit im zugeordneten Prozessblock DEVPRB und PHYSPB.

Bei der Basisparametrierung werden den EA-Adressen Interrupt-Vektoren zugeordnet. In der Vektorliste VEKLI im Hardware-Verständigungsbereich (HW - VB) des Zentralspeichers steht zu jedem Vektor die zugehörige Prozeßblock-Adresse. Nach dem Rücksetzen ist eine periphere Einheit auf den Basis-Prozeßblock PRBBAS (Adresse 48 im Zentralspeicher) eingestellt. Sobald die periphere Einheit basisparametriert ist, kann sie ihre Prozeßblock-Adresse über ihren Vektor aus der VEKLI ermitteln. Die Prozeßblock-Adresse ist reell; alle Prozeßblöcke liegen in den ersten 28 K*byte im Hauptspeicher.

Benötigt die periphere Einheit weitere Erläuterungen, so übernimmt sie diese selbst (DMA) aus ihrem DEVPRB und dem PHYSPB. Der Übernahmezeitpunkt wird - nach ihren Erfordernissen - von der peripheren Einheit bestimmt; er muß nicht unmittelbar auf den Auftrag folgen.

Zuerst wird die Zelle CHECKF im Prozeßblock getestet. Ist die Prüfsumme ungültig, so kann der Auftrag nicht interpretiert werden, und es erfolgt ein Eintrag in den Fehler-Prozeßblock PRBERR. PRBERR ist ein Prozeßblock, der in diesem Fall zum Start eines betriebssystem-internen Reaktionsprogramms führt. Bei gültiger Prüfsumme wird der zugehörige physikalische Parameterblock PHYSPB aus den Zellen TPWP und PIPHY des Prozeßblocks ermittelt. Die Adresse kann reell oder virtuell sein; Auskunft darüber gibt die Zelle SWCTR des Prozeßblocks.

In dem oder den physikalischen Parameterblöcken (PHYSPB, Kettung) wird aus den Zellen COMCLS und COMCOD die genaue Art des Auftrags ermittelt. Die Auftragsart führt zur entsprechenden Routine im Firmware-Programm der peripheren Einheit.

Je nach der Art des Auftrags müssen noch weitere Daten aus den physikalischen Parameterblock entnommen werden, wie DEVNUM, ADDIN, TPWT, BUPOB usw. Die Bedeutung der einzelnen Zellen im physikalischen Parameterblock - abhängig vom jeweiligen Auftrag - ist geräteabhängig.

Treten während der Interpretation Besonderheiten auf, so werden Einträge im physikalischen Parameterblock (Zelle 14 bis 17) flag field und in den Prozeßblock (NORTER, FINTER) vorgenommen. Solche Besonderheiten können sein: Auftrag nicht erlaubt, Gerät unklar, Hauptspeicher-Fehler, usw.

Aufgrund der vom physikalischen Parameterblock übernommenen Daten werden Gerät und Anschaltung (Steuerung) auf den Datentransfer vorbereitet. Die

Steuerung wird beispielsweise mit Anfangsadresse und Blocklänge eines Transferpuffers geladen, das Gerät wird mit spezifischen Parametern versorgt.

3.3.4.1.4 Auftragsausführung

Sind von der peripheren Einheit alle Vorbereitungen getroffen, so wird die im Auftrag gewünschte Funktion durchgeführt. Es soll z. B. ein Datenblock - gekennzeichnet durch einen Zentralspeicher-Adreßbereich und ggf. eine Externadresse - übertragen werden.
Die Zentralspeicher-Adressen können reell oder virtuell sein. Bei virtueller Adressierung ist der Zentralspeicher in Seiten von 1 K* Adressen ("Seiten") unterteilt; an jeder Seitengrenze ist eine Adreßübersetzung notwendig. Dabei ist es möglich, daß die virtuelle Adresse durch die Software vorher bereits übersetzt wird.

Treten während der Auftragsausführungen Besonderheiten wie Speicherfehler, Übertragungsfehler usw. auf, so wird der Auftrag ggf. abgebrochen, und es werden entsprechende Anzeigen im physikalischen Parameterblock (PHYSPB) und Prozeßblock hinterlegt.

3.3.4.1.5 Anzeigenübergabe

Ist die im Auftrag gestellte Aufgabe durchgeführt oder mußte die Durchführung abgebrochen oder unterbrochen werden, weil Besonderheiten oder Fehler auftraten, die die Peripherie von sich aus nicht beheben konnte, so werden zum Schluß von der Peripherie Anzeigen im physikalischen Parameterblock und Prozeßblock hinterlegt. Diese Anzeigen erlauben es später dem Systemprogramm bzw. den Anwender oder Operator, Rückschlüsse zu ziehen, was aus dem gestellten Auftrag wurde.

Im Prozeßblock Zelle 9 (NORTER, FINTER) werden allgemeine Anzeigen hinterlegt, die eine Aussage über die Art des erreichten Abschlusses machen. Darüber hinaus kann die Peripherie im Prozeßblock Zelle 11 bis 14 noch weitere Information über peripherie-eigene Mitteilungen an die Zentraleinheit hinterlegen, die nicht in unmittelbarem Zusammenhang mit einem Auftrag stehen, z. B. Anrufereignisse. NORTER und FINTER sind vom Systemprogramm mit 0 vorbesetzt. Eintragungen erfolgen nur, wenn ein sinnvoller Abschluß bereits erreicht ist (Auftrag ausgeführt oder Anzeigen im physikalischen Parameterblock hinterlegt) oder wenn ein Anrufereignis vorliegt.

Im physikalischen Parameterblock Zelle 14 bis 17 werden auftragsspezifische Anzeigen hinterlegt, die ggf. eine Analyse des Fehlers oder der Besonderheit ermöglichen.

3.3.4.1.6 Unterbrechungsanforderung (Interrupt)

Mit einer Interrupt-Anforderung $\overline{IR}$ kann eine periphere Einheit eine Reaktion der Zentraleinheit anfordern.

Zwei Arten von Interrupt-Anforderungen können unterschieden werden:

- Anruf-Interrupt
- Auftragsende-Interrupt

Der Anruf-Interrupt dient zur Mitteilung von Wünschen der peripheren Einheit wie Alarm, Eingabewunsch, usw. Er wird im Kapitel 3.3.4.2 über periphere Initiative weiter behandelt.

Der Auftragsende-Interrupt wird zum Abschluß einer Auftragsbearbeitung gestellt. Er meldet dem Betriebssystem, daß der Auftrag beendet wurde. Die Zelle SWCTR entscheidet darüber, ob die periphere Einheit ihren Auftrag mit Interrupt abschließen soll, oder ob sie nur Anzeigen hinterlegen soll.

Soll mit Interrupt abgeschlossen werden, so sendet die periphere Einheit nach Ablage der Anzeigen das Signal $\overline{IR}$. Die Zentraleinheit leitet daraufhin eine Übernahme-Routine für den Vektor ein. Bei entsprechender Signalkombination ($\overline{IL}$, $\overline{IAI}$, $\overline{IAO}$, $\overline{TR}$) übergibt die periphere Einheit, die das Signal $\overline{IR}$ gesendet hat, ihren Vektor an die Zentraleinheit.

Der Vektor zeigt auf eine Zelle in der Vektorliste VEKLI, die die Adresse des zugehörigen Prozeßblocks enthält. Im Prozeßblock entscheidet das A-Bit (Zelle 2) darüber, ob dieser Prozeßblock bereits aktiv, d. h. in eine Warteschlange eingereiht ist. Hat die Zentraleinheit das A-Bit bereits gesetzt, so ist ein nochmaliges Einreihen nicht mehr möglich; als Kennung für weitere Interrupt-Anforderungen wird zusätzlich das M-Bit gesetzt.

War der gerufene Prozeßblock noch nicht aktiv (A = 0), so wird er von der Zentraleinheit unter seiner Priorität PRIO (Zelle 2) an das Ende seiner Prioritätswarteschlange eingereiht. Für jede Priorität existiert ein Warteschlangenkopf im Hardware-Verständigungsbereich des Hauptspeichers, der die Verweise auf das erste und letzte Glied der jeweiligen Prioritätswarteschlange enthält. SUCPRB und PREPRB (Zelle 0 und 1 im Prozeßblock) enthalten die Vor- und Rückverweise in der Warteschlange. Im TZW1 (Zelle 3 des Prozeßblocks) steht der Verweis auf eine Parametertafel im Zentralspeicher für das zugehörige Reaktionsprogramm.

Durch das Einreihen eines Prozeßblocks in seine Warteschlange - ausgelöst durch $\overline{IR}$ - ruft die Zentraleinheit das zugehörige Reaktionsprogramm auf. Es läuft ab, sobald der betreffende Prozeßblock an den ersten Platz in der Warteschlange seiner Prioritätsebene vorgerückt ist, wenn dies die momentan höchstpriore Warteschlange ist.

Bei Auftragskettung wird von der peripheren Einheit nur der letzte Auftrag einer Kette mit Interrupt abgeschlossen.

3.3.4.1.7 Abschlußbearbeitung

Nachdem die periphere Einheit ihren Auftrag ausgeführt hat, gibt sie, abhängig von der Zelle SWCTR, nur Anzeigen ab oder stellt zusätzlich noch eine Interruptanforderung.

Entsprechend kann man auch zwei Reaktionsweisen des Betriebssystems unterscheiden:

- das Polling-Verfahren,
- das Interrupt-Verfahren.

Beim Polling-Verfahren wartet das Betriebssystem auf das Eintreffen der Abschlußanzeigen, indem es NORTER und FINTER (Zelle 9 im Prozeßblock) zyklisch abfragt ("pollt"), ob deren Inhalt = 0 ist.
Sobald die Anzeigen eingetroffen sind, (Zelle 9 = 0), werden sie ausgewertet, und das Programm verzweigt entsprechend den Anzeigen. Dieses Verfahren wird hauptsächlich dann angewendet, wenn eine kurze Reaktionszeit von der peripheren Einheit zu erwarten ist; es lohnt sich dann nicht, zwischendurch ein anderes Programm anzufangen bzw. es ist nicht zulässig (z. B. Basisparametrierung).

Beim Interrupt-Verfahren muß der gewünschte Prozeßblock erst wieder durch Einreihen in die Warteschlange seiner Prioritätsebene aktiviert werden, wenn eine Interrupt-Anforderung gestellt wurde. Die Zentraleinheit arbeitet nacheinander die Prozeßblöcke der jeweils höchstprioren Warteschlange ab. Das zum Interrupt gehörige Reaktionsprogramm läuft in der Zentraleinheit ab, wenn der Prozeßblock an die erste Stelle in der Warteschlange seiner Prioritätsebene vorgerückt ist und diese momentan die höchste Priorität hat. Aus der Zelle TPW1 des Prozeßblocks wird die Parametertafel für das Programm ermittelt, daraus der Registersatz geladen und das Programm gestartet.

Das Reaktionsprogramm wertet die für die Zentraleinheit bestimmten Daten im Prozeßblock und physikalischen Parameterblock aus. Im Prozeßblock sind die Zellen 9 bis 15, insbesondere aber die Zellen NORTER, FINTER und PEREQ. Im physikalischen Parameterblock sind u. a. die Endadresse EUPOA und die Anzeigenzellen Zelle 14 bis 17 von Interesse. Was im einzelnen Fall ausgewertet werden muß, richtet sich nach der Art des vorausgegangenen Auftrags. Je nach Ergebnis der Auswertung muß zu einem Fehlerprogramm verzweigt werden, oder der Auftrag ist ohne Besonderheit erledigt und die Vorbereitung eines neuen Auftrags kann begonnen werden.

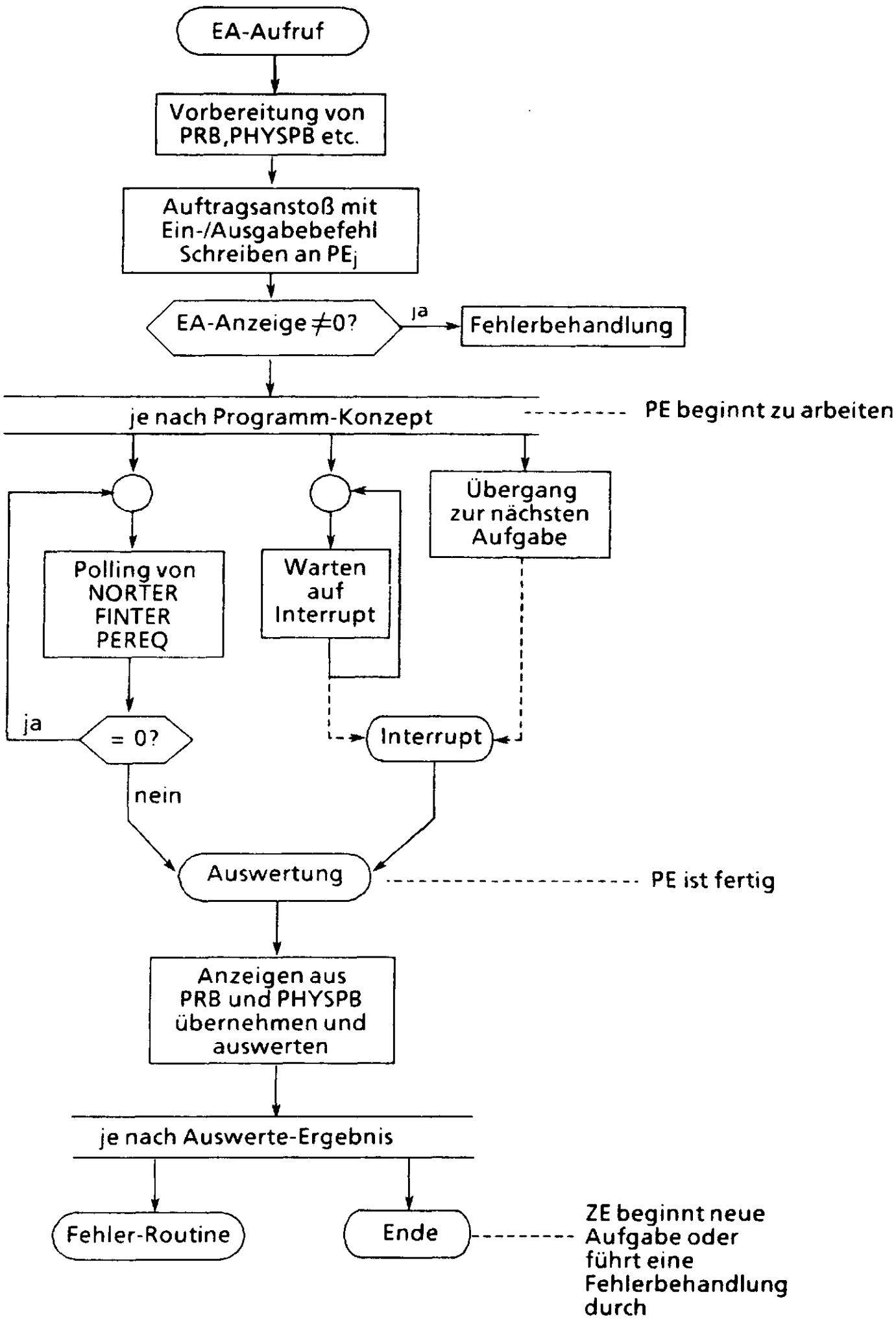


Bild 3.25 Prinzipieller Ablauf der Aktivitäten der Zentraleinheit bei EA-Verkehr mit zentraler Initiative

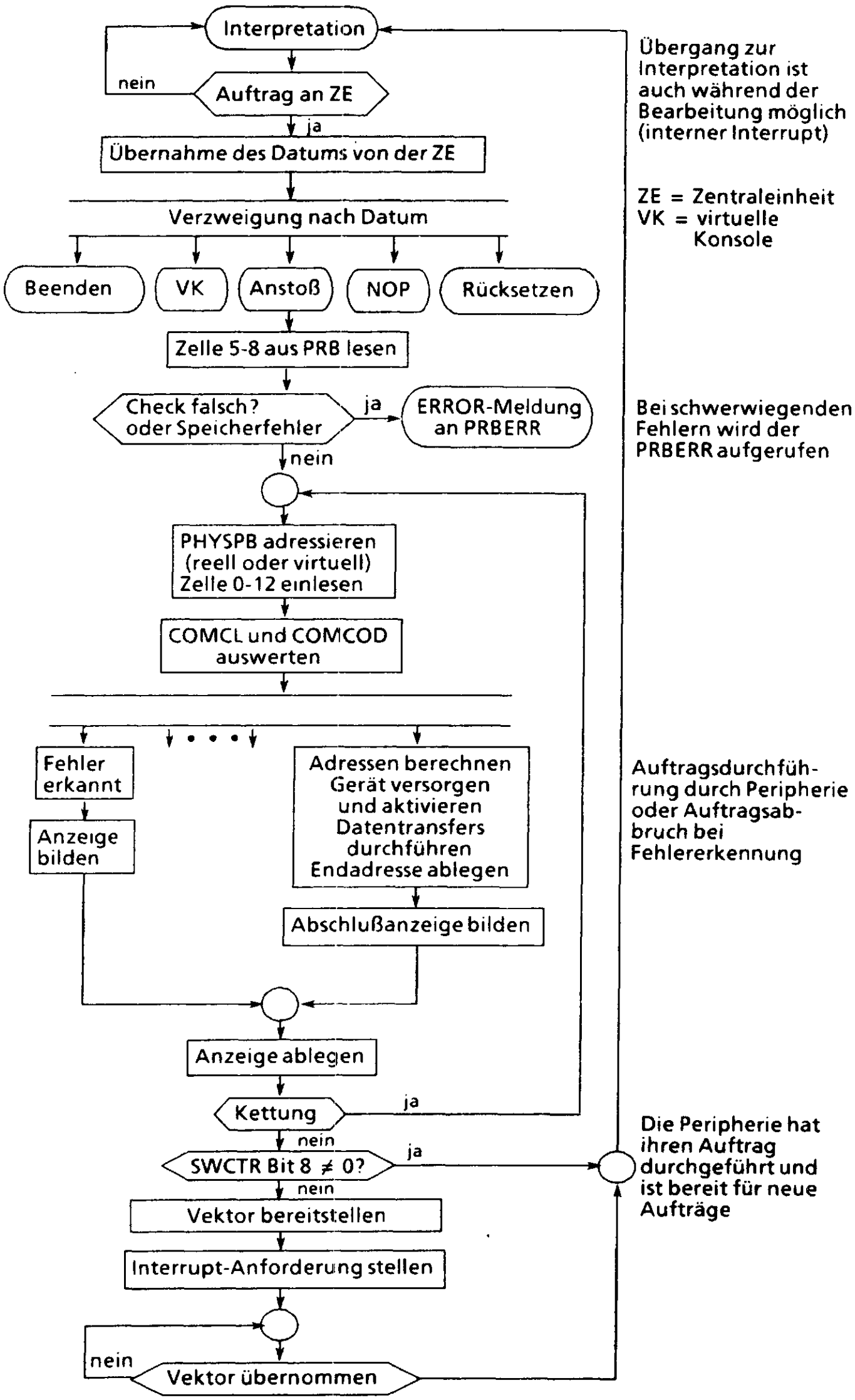


Bild 3.26 Prinzipieller Ablauf der Aktivitäten der Peripherie bei EA-Verkehr mit zentraler Initiative

3.3.4.2 Periphere Initiative

Aufgrund bestimmter Ereignisse, die innerhalb der peripheren Einheit auftreten, ist es notwendig, die Zentraleinheit zu informieren. Gründe für solche Mitteilungen können sein:

- Abschlußmeldungen nach einen Auftrag,
- zyklische Ereignisse (Weckfunktion),
- Anforderungen von Dienstleistungen der Zentraleinheit für die periphere Einheit (Stütztreiber),
- Anrufe (Tastaturbedienung),
- Alarme (z.B. Grenzwertüberschreitung, Besonderheit beim Ablauf eines Selbsttests).

Diese Ereignisse lassen sich in zwei Gruppen einteilen, nämlich Ereignisse, die auf Initiative der Zentraleinheit zurückzuführen sind (Abschlußmeldungen) und Ereignisse, die ihren Ursprung in der Peripherie haben (Alarm, Anruf). Beide Ereignisse können voneinander unabhängig auftreten und müssen deshalb auch getrennt ausgewertet werden können.

Der Prozeßblock (PRB), der der peripheren Einheit durch Basisparametrierung zugeteilt ist, ist der gemeinsame statische Verständigungsbereich zwischen Zentraleinheit und der peripheren Einheit.

In den Zellen PEREQ 1/2 (Zelle 13 des Prozeßblocks) werden der Zentraleinheit Anrufereignisse mitgeteilt. Muß der gemeldete Anruf noch weiter erläutert werden, so wird die zusätzliche Information in den Zellen PERED1, PERED2 oder SERDAT hinterlegt (Zellen 11, 12, 14 des Prozeßblocks).

Anrufereignisse werden in zwei Gruppen unterteilt, die sich im Verhalten der Zentraleinheit auf das Anrufereignis hin unterscheiden:

Gruppe a)

In dieser Gruppe sind alle Anrufereignisse zusammengefaßt, die reinen Meldecharakter haben (z. B. Grenzwertüberschreitung bei der Inline-Statistik). Die periphere Einheit erwartet keine besondere Reaktion der Zentraleinheit auf diese Anrufe; sie kann nach einem solchen Anruf ungestört weiterarbeiten und muß nicht auf einen bestimmten Auftrag der Zentraleinheit warten.

Gruppe b)

In dieser Gruppe sind alle Anrufe zusammengefaßt, die eine Reaktion des Betriebssystems erforderlich machen. Die periphere Einheit erwartet, daß die Zentraleinheit auf einen solchen Anruf auf bestimmte Weise reagiert, indem sie z. B. einen bestimmten Auftrag abgibt. Für die Reaktion der Zentraleinheit muß ggf. eine bestimmte Zeitspanne eingehalten werden. Sind einer Anschaltung/Steuerung mehrere Prozeßblöcke zugeordnet, so können auch nach einem Anruf noch Aufträge an andere Teilgeräte abgegeben

werden. Diese Aufträge dürfen nur dann bearbeitet werden, wenn sie die Information, die mit dem Anruf gekoppelt ist, nicht beeinflussen. Sonst werden solche Aufträge zwar angenommen, aber so lange zurückgestellt, bis der Anruf bearbeitet ist; anschließend werden die "wartenden" Aufträge bearbeitet.

Auf einen Anruf kann, unter der zugeordneten EA-Adresse, der erwartete oder ein unerwarteter Auftrag eintreffen. Nach einem erwarteten Auftrag gilt für die periphere Einheit der Anruf als bearbeitet.

Handelt es sich um einen unerwarteten Auftrag, so wird er trotzdem bearbeitet, wenn dies sinnvoll ist und die Anrufinformation nicht zerstört wird. Zusammen mit der Auftragsabschlußmeldung wird der Anruf erneut von der peripheren Einheit gestellt. Ist es nicht möglich, den unerwarteten Auftrag zu bearbeiten, ohne die Anrufinformation zu zerstören, wird er von der peripheren Einheit mit der Anzeige 0201 hexa in der Zelle HWFLAG (Auftrag z. Zt. nicht erlaubt) zurückgewiesen und dabei ebenfalls der Anruf wiederholt. Der Anruf wird von der peripheren Einheit so lange wiederholt, bis er bearbeitet ist, oder bis eine weitere Wiederholung von der Seite der peripheren Einheit nicht mehr sinnvoll erscheint.

Es gehören folgende Anrufe zur

Gruppe a) Kurzzeitwecker abgelaufen, zyklischer Wecker abgelaufen, Pufferspannungsausfall beim Zeitimpulsgeber ZIG, Pufferspannungswiederkehr bei ZIG, Senderichtung/Puffer frei, drohender Linetrace-Pufferüberlauf, Fehler bei Selbsttest (Prozeßblock-spezifisch oder Prozeßblock-unspezifisch), Grenzwertüberschreitung bei Inline-Statistik.

Gruppe b) Dateneingabe mit oder ohne Blocklängenangabe, Anzeigeneingabe.

Abwicklung eines Anrufs

Eine periphere Einheit hat erst dann die Berechtigung, einen Anruf abzugeben, wenn sie durch Basisparametrierung einen eigenen Geräteprozeßblock DEVPRB zugeteilt bekommen hat. Die Zentraleinheit trägt in die Zellen PEREQ1 und PEREQ2 Null ein, wenn Anrufereignisse erlaubt sind. Die periphere Einheit darf nur dann ihr Anrufereignis in PEREQ1 oder PEREQ2 eintragen, wenn das zugehörige Byte zuvor mit 0 vorbesetzt ist; der Eintrag in PEREQ muß von der peripheren Einheit byteweise vorgenommen werden (das andere Byte darf nicht verändert werden).

Ist das Byte, in das der PEREQ-Eintrag erfolgen soll, ungleich 0, so muß es die periphere Einheit zyklisch abfragen ("pollen"), bis es von der Zentraleinheit freigegeben wird (z. B. alle 100 µs bis alle 1 ms; damit die Zentraleinheit dadurch nicht unnötig belastet wird). Muß der Anruf noch genauer spezifiziert werden, so erfolgt noch ein Eintrag in die Zellen PERED1, PERED2 oder SERDAT, bevor PEREQ beschrieben wird.

Nachdem die Zelle PEREQ beschrieben ist, gibt die periphere Einheit ihre Interrupt-Anforderung $\overline{IR}$ ab. Ein Abschluß-Interrupt aufgrund einer Auftragsbearbeitung und ein Anruf-Interrupt können ggf. zu einem $\overline{IR}$ zusammengefaßt werden; grundsätzlich darf aber jeder Interupt auch einzeln abgegeben werden. Ein Anruf hat aber immer einen Interrupt zur Folge.

Aufgrund der Interrupt-Meldung holt die Zentraleinheit den Vektor der peripheren Einheit ab und aktiviert den zugehörigen Prozeßblock. Daraufhin läuft das zugehörige Reaktionsprogramm in der Zentraleinheit ab. Die Interrupt-Kennung in der Zelle PEREQ wird übernommen und ggf. die Ergänzungsinformation. Sobald das Reaktionsprogramm neue Anrufe wieder zulassen kann, werden PEREQ1 und PEREQ2 gelöscht, um damit der peripheren Einheit anzuzeigen, daß sie erneut Eintragungen vornehmen darf.

Die Abschlußanzeigen in den Zellen NORTER/FINTER und die Anrufkennung in der Zelle PEREQ werden voneinander unabhängig vom Reaktionsprogramm ausgewertet. Ist PEREQ = 0, so liegen nur Abschlußanzeigen vor, die vom zugehörigen Programm ausgewertet, umgewandelt und an den Anwender weitergegeben werden, der den EA-Aufruf abgegeben hatte. Ist NORTER/FINTER = 0, so liegen nur Anrufereignisse vor, die vom Betriebssystem ausgewertet und bearbeitet werden.

Sind sowohl PEREQ $\neq$ 0 als auch NORTER/FINTER $\neq$ 0, so lag zusammen mit einem Abschlußereignis auch ein Anruf vor. Vom Betriebssystem muß demzufolge hier eine Abschlußbearbeitung und auch eine Anrufbearbeitung durchgeführt werden.

Gehört der gemeldete Anruf zur Gruppe a), so muß sich das Systemprogramm nur merken, daß ein Anruf vorlag. Die Reaktion kann später an passender Stelle erfolgen. Anrufe der Gruppe b) fordern eine ganz bestimmte Reaktion, die vom Betriebssystem nicht immer direkt möglich ist; in solchen Fällen bleibt der Anruf unberücksichtigt und wird "vergessen". Die periphere Einheit erkennt das an der Art des nächsten Auftrags; sie wiederholt nötigenfalls den Anruf, bis die Zentraleinheit in gewünschter Weise reagiert.

Eine basisparametrierte periphere Einheit ist prinzipiell auch anrufberechtigt. Wird eine periphere Einheit selektiv beendet, so werden Anruf-ereignisse, die vor dem selektiven Beenden eingetreten waren, ungültig und führen nicht mehr zum Interrupt; die Anrufberechtigung als solche bleibt aber erhalten. Nach einem Sammelbeenden werden keine Anruf-Interrupts mehr gestellt. Anrufe sind in diesem Fall erst wieder nach einer erneuten Basisparametrierung möglich.

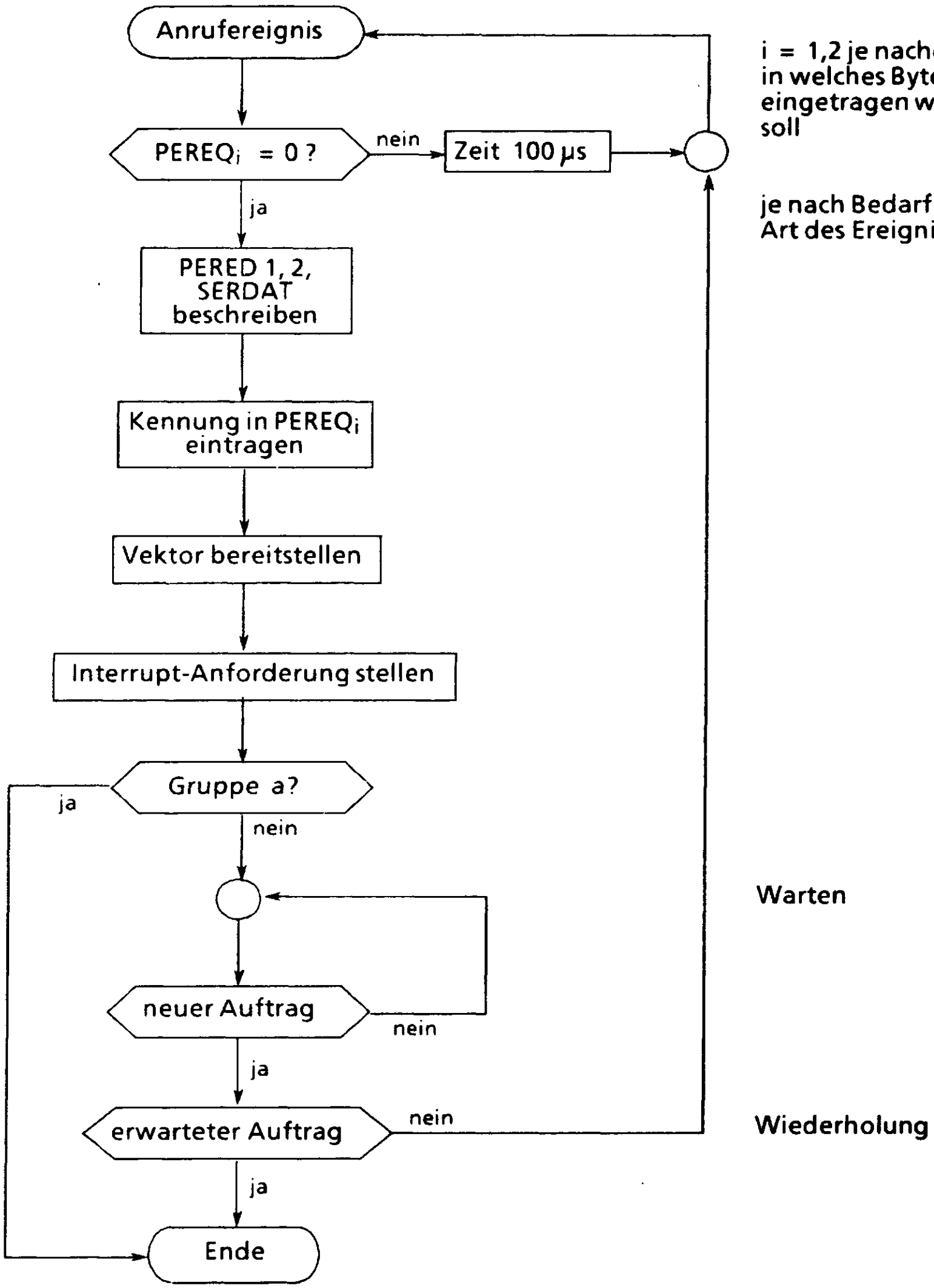


Bild 3.27 Prinzipieller Ablauf der Aktivitäten der Peripherie bei EA-Verkehr mit peripherer Initiative

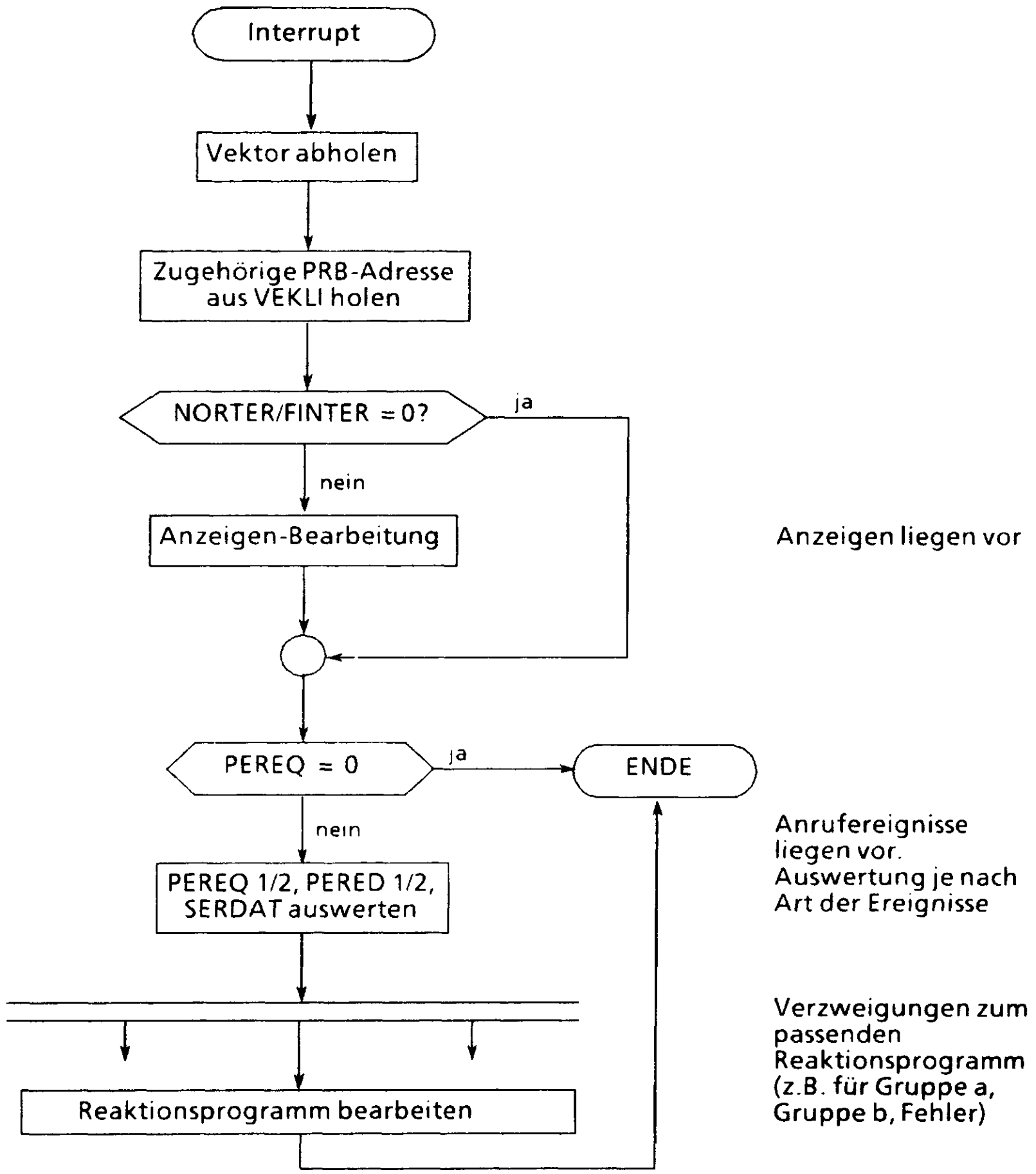


Bild 3.28 Prinzipieller Ablauf der Aktivitäten der Zentraleinheit bei EA-Verkehr mit peripherer Initiative

3.3.4.3 Direct memory access (DMA)

Der "Direkte Speicherzugriff" (DMA, direct memory access) zum Zentralspeicher entlastet bei entsprechender peripheren Intelligenz den Zentralprozessor. Die Arbeit, Zugriffe zum Zentralspeicher durchzuführen, um z. B. einen Datenpuffer vom Zentralspeicher in einen Peripherspeicher zu transferieren, muß nicht vom Zentralprozessor geleistet werden, sondern wird von intelligenten Steuerungen der peripheren Einheiten übernommen. Der Zentralprozessor wird entlastet und für andere Aufgaben frei, die Systemleistung steigt. DMA wird von peripheren Geräten der Klasse 2 und 3 beherrscht.

Vor dem eigentlichen DMA-Verkehr werden Prozeßblock und physikalischer Parameterblock von der Zentraleinheit vorbereitet, um z. B. festzulegen, ob es sich um schreibende oder lesende Zugriffe handelt und um Transferpuffergrenzen zu bestimmen. Die Anschaltungen der entsprechenden Klassen verwenden bereits zur Interpretation von Prozeßblock und physikalischem Parameterblock DMA wie auch zum Nutzdatentransfer.

Mit Beginn des DMA-Verkehrs wird die periphere Einheit zum Busmaster und übernimmt die Steuerung der Abläufe, wie sie laut Buskoordinierung (Kapitel 3.3.5) notwendig sind. Fehler- oder Endebearbeitung des DMA-Verkehrs werden wie bei allen Aufträgen ausgehend von den fault flags FFA und FFD, über Anzeigen im physikalischen Parameterblock bzw. Prozeßblock bearbeitet.

3.3.5 Buskoordinierung

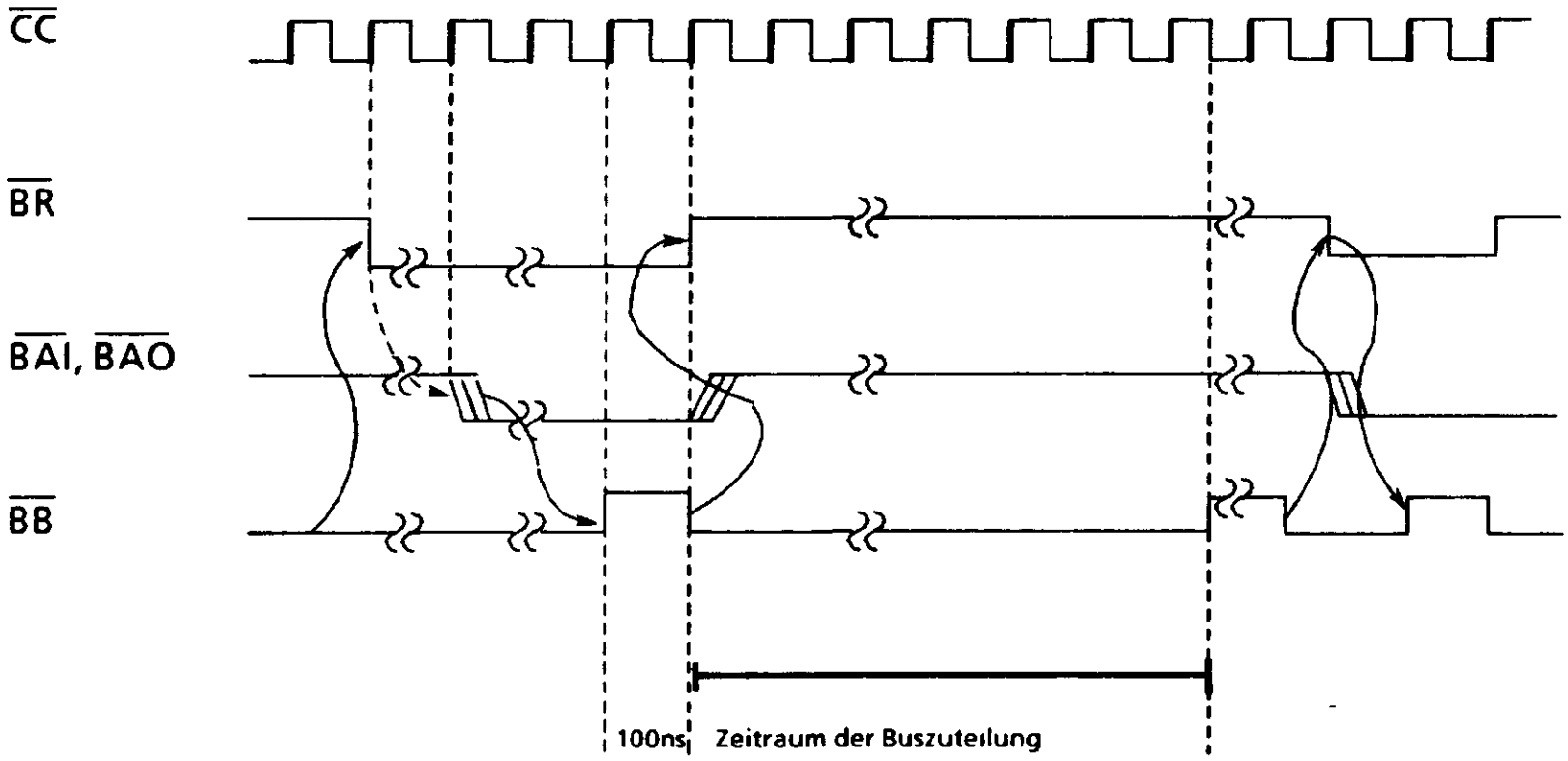
Um eine einwandfreie Abwicklung der Sequenzen des EA-Verkehrs zu gewährleisten, müssen gewisse Spielregeln befolgt werden, nach denen der EA-Verkehr am Bus anzulaufen hat. Dazu dient eine Busprioritierungslogik, welche den Systembus jeweils einem aktiven Partner (Master) zuteilt. Falls keine externen Busanforderungen vorliegen, ist das Bussystem dem Zentralprozessor zugeteilt. Die Buszuteilung erfolgt taktsynchron zum Zentraltakt $\overline{CC}$ (10 MHz).

Ein Busteilnehmer kann über das Signal $\overline{BR}$ (bus request) den Bus nur dann anfordern, wenn ein anderer Teilnehmer am Bus aktiv ist ($\overline{BB}$ = L). Die Buszuteilungslogik antwortet auf diese Anforderung mit dem Kettensignal $\overline{BA}$ (bus acknowledge). Daraufhin kann der Teilnehmer, der $\overline{BR}$ gestellt hat und $\overline{BA}$ empfängt, den Bus belegen, sobald das Bussystem frei ist. Das Freiwerden des Bussystems wird dadurch gekennzeichnet, daß das Signal $\overline{BB}$ (bus busy) = H ist. Haben zu diesem Zeitpunkt mehrere Busteilnehmer den Bus angefordert, so erhält ihn nicht unbedingt der, der die Anforderung als erster gestellt hat, sondern derjenige, der bei der steigenden Flanke des Signals $\overline{BB}$ in der Kette am weitesten vorne ist (d.h. sich im Steckplatz mit der niedrigsten Steckplatznummer befindet). Der Teilnehmer, der jetzt den Bus belegen will, zieht das Signal $\overline{BB}$ auf L. Das Bussystem ist nun solange belegt, wie dieser Pegel beibehalten wird. Ist kein anderer Teilnehmer aktiv, so belegt die Zentraleinheit mit $\overline{BB}$ = L den Bus; sie hat am Bus die niedrigste Priorität.

Bild 3.29, 3.30 zeigen Struktur und Sequenz der Buszuteilung auf der Seite der Peripherie. Die Schnelligkeit, mit der das $\overline{BA}$-Signal von Kettenglied zu Kettenglied weitergereicht wird, bestimmt die max. Länge der Kette. Deshalb darf die Verzögerungszeit des Signals $\overline{BAI}$ (bus acknowledge input) zu $\overline{BAO}$ (bus acknowledge output) nicht größer als 8 ns sein. Die max. Kettenlänge ist acht Kettenglieder. Eine größere Anzahl von Kettenteilnehmern kann durch eine Matrixkette mit übergeordneter Verteilungskette und untergeordneten Subketten erreicht werden. Diese Struktur erlaubt eine Erweiterung, die auch umfangreicheren Ausbauten genügt; sie ist in der Verdrahtung des Zentraleinheits-Rahmens realisiert.

Solange der Teilnehmer den Bus mit $\overline{BB}$ = L belegt hat, kann er über das Bussystem verfügen, d. h. er darf seine Adressen, Daten und Steuerinformationen aufschalten.

Mit Wegnahme der Busbelegung ($\overline{BB}$ = H) muß der Teilnehmer auch wieder seine Signale vom Bussystem wegschalten (Übergang in den passiven oder Tristate-Zustand).



CC Zentraltakt (10 MHz)
BR bus request
BAI bus acknowledge input
BAO bus acknowledge output
BB bus busy

Bild 3.29 Buszuteilung

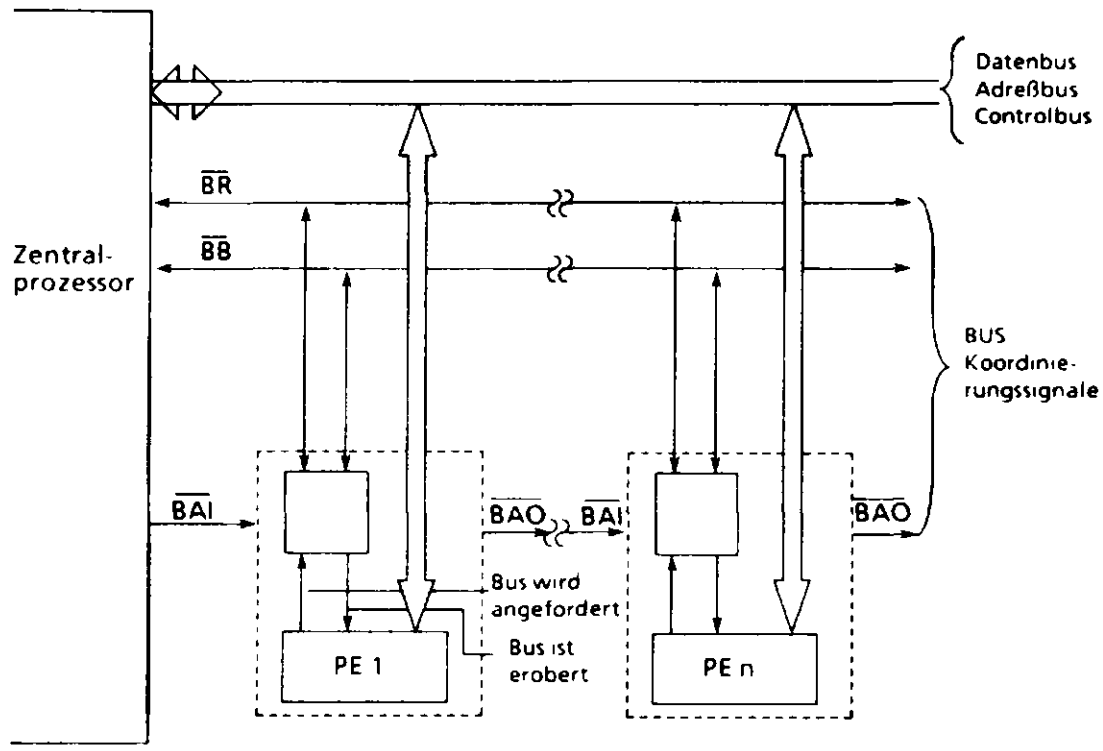


Bild 3.30 Buszuteilungslogik

Während einer Busbelegungsphase hat ein Teilnehmer prinzipiell die Möglichkeit, mehrere Datentransfers durchzuführen (burst). Im Interesse einer möglichst kurzen Buszugriffszeit sollten aber die Belegungsphasen möglichst kurz gehalten werden und es wird daher während einer Busbelegung ($\overline{BR}$ = L) nur ein Transfer ausgeführt.

Bussteuerung

Der Bus muß nur von peripheren Geräten angefordert werden, da die Zentraleinheit den Bus nur dann (automatisch) bekommt, wenn keine anderen Anforderungen vorliegen.

Vor jeder Datenübertragung über das Bussystem ist vom jeweiligen Teilnehmer mit dem Signal $\overline{BR}$ (bus request) der Bus anzufordern. Eine zentrale Prioritierungslogik liefert das Rückmeldesignal $\overline{BAI}$ (bus acknowledge input), das von den nicht anfordernden Teilnehmern als $\overline{BAO}$ (bus acknowledge output) weitergegeben wird (Kettenprioritierung).

Das Eintreffen der Rückmeldung $\overline{BAI}$ signalisiert dem anfordernden Teilnehmer die folgende Buszuteilung. Sobald der Bus freigegeben wird ($\overline{BB}$ = H), kann er seine eigene Busbelegung ($\overline{BB}$ = L) vornehmen. Die Busbelegung erfolgt taktsynchron mit dem Grundtakt $\overline{CC}$.

Nach der eigentlichen Busbelegung mit $\overline{BB}$ = L erfolgt die Aufschaltung der Busse und Steuersignale

- Adreßbus
- Datenbus
- Adreß Control
- Memory/$\overline{IO}$
- Read/$\overline{Write}$

Nach einer Bus-Einschwingzeit wird über das Signal $\overline{TR}$ (transfer request) die Übertragung aktiviert.

Der angesprochene Partner (Dekodierung des Adreßbus + M/$\overline{IO}$) schaltet daraufhin seine Daten auf den Datenbus (Eingabe) und quittiert mit $\overline{TA}$ (transfer acknowledge). Als Reaktion werden $\overline{TR}$ und die Busse sowie $\overline{TA}$ weggeschaltet und damit der Transferzyklus beendet.

Mit Hilfe der Begleitsignale $\overline{FFA}$, $\overline{FFD}$ (fault flag address, fault flag data) kann die Quittung $\overline{TA}$ durch eine Fehlermeldung gekennzeichnet werden.

Die Übergangszeit zwischen zwei Busbelegungsphasen (Busmaster-Wechsel) beträgt 100 ns.
Die Transfersequenzen sind in Bild 3.31 dargestellt.

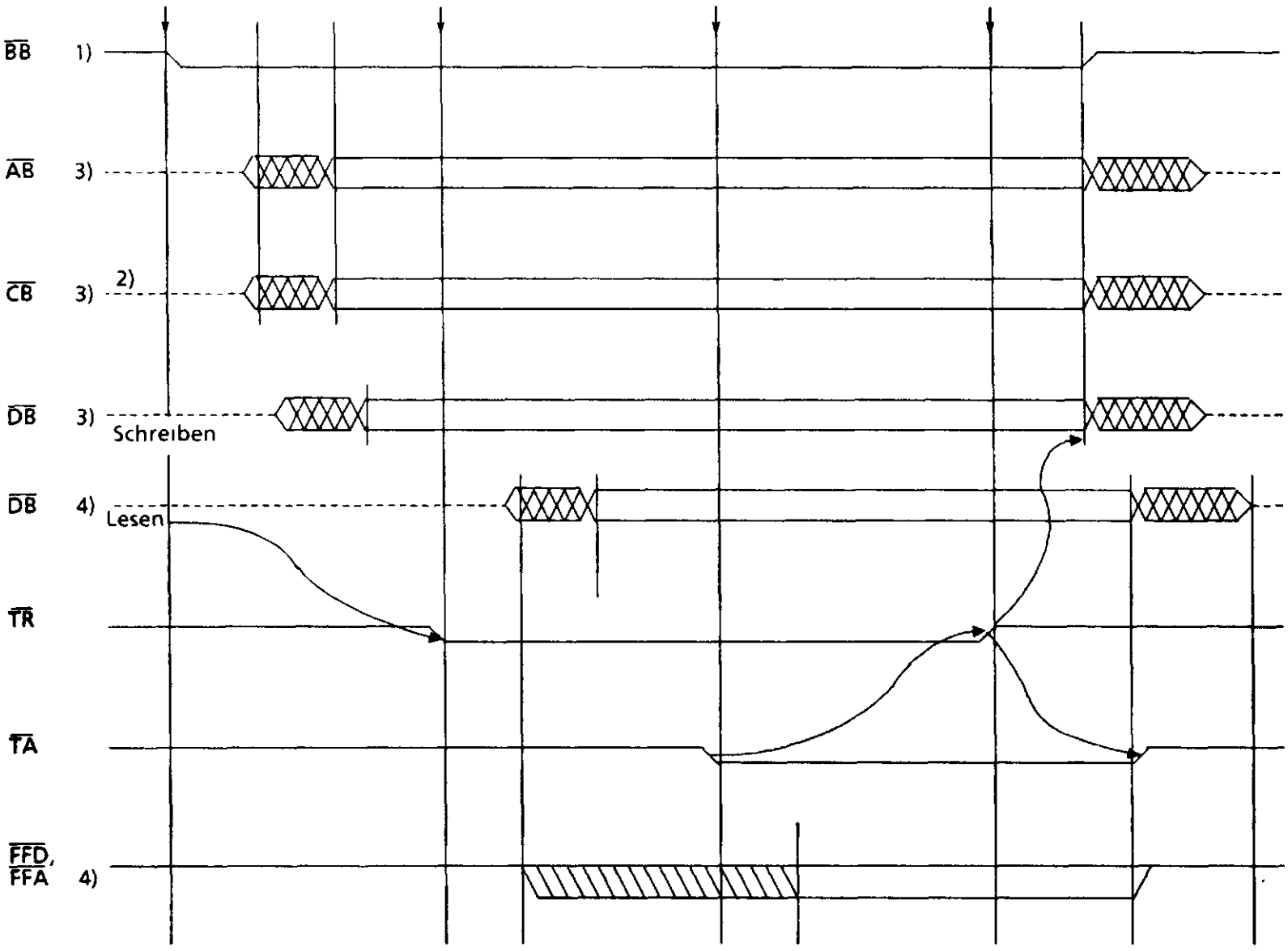


Bild 3.31 Transfer-Sequenz fur Schreiben und Lesen

Anmerkungen zu Bild 3.31:

1) Die $\overline{RB}$-Sequenz wird nur bei DMA-Verkehr relevant bzw. beim Busmaster-Wechsel

2) $\overline{CB}$ = Controlbus ($R/\overline{W}$, $\overline{AC0}$, $\overline{AC1}$, $M/\overline{IO}$)

3) Master-Signale

4) Slave-Signale

Eine spezielle Sequenz ist zur Behandlung von Interrupts realisiert. Ein Interrupt-Kreis besteht aus den beiden Signalen $\overline{IR}$ (interrupt request = Unterbrechungsanforderung) und $\overline{IA}$ (interrupt acknowledge = Unterbrechungsrückmeldung).

Auf die jeweilige $\overline{IR}$-Leitung schalten die EA-Moduln ihre Anforderungen (Bild 3.32) entsprechend einer ODER-Verknüpfung.

Das Signal $\overline{IA}$ wird in einer Kette von Teilnehmer zu Teilnehmer weitergereicht ($\overline{IAI}$ - ein, $\overline{IAO}$ - aus). Damit sich alle Busteilnehmer auf die Übergabe des Interrupt-Vektors einstellen können, wird das entkoppelte Rückmeldesignal $\overline{IL}$ (interrupt lock) von der Zentraleinheit zu allen peripheren Einheiten geführt (Bussignal). Die peripheren Einheiten dürfen dann nur ihren Interrupt-Vektor mit $\overline{TR}$ aufschalten, falls sie das Signal $\overline{IAI}$ erhalten. Andere Daten von der peripheren Einheit sind verriegelt, solange das Bussignal $\overline{IL}$ = L ansteht.

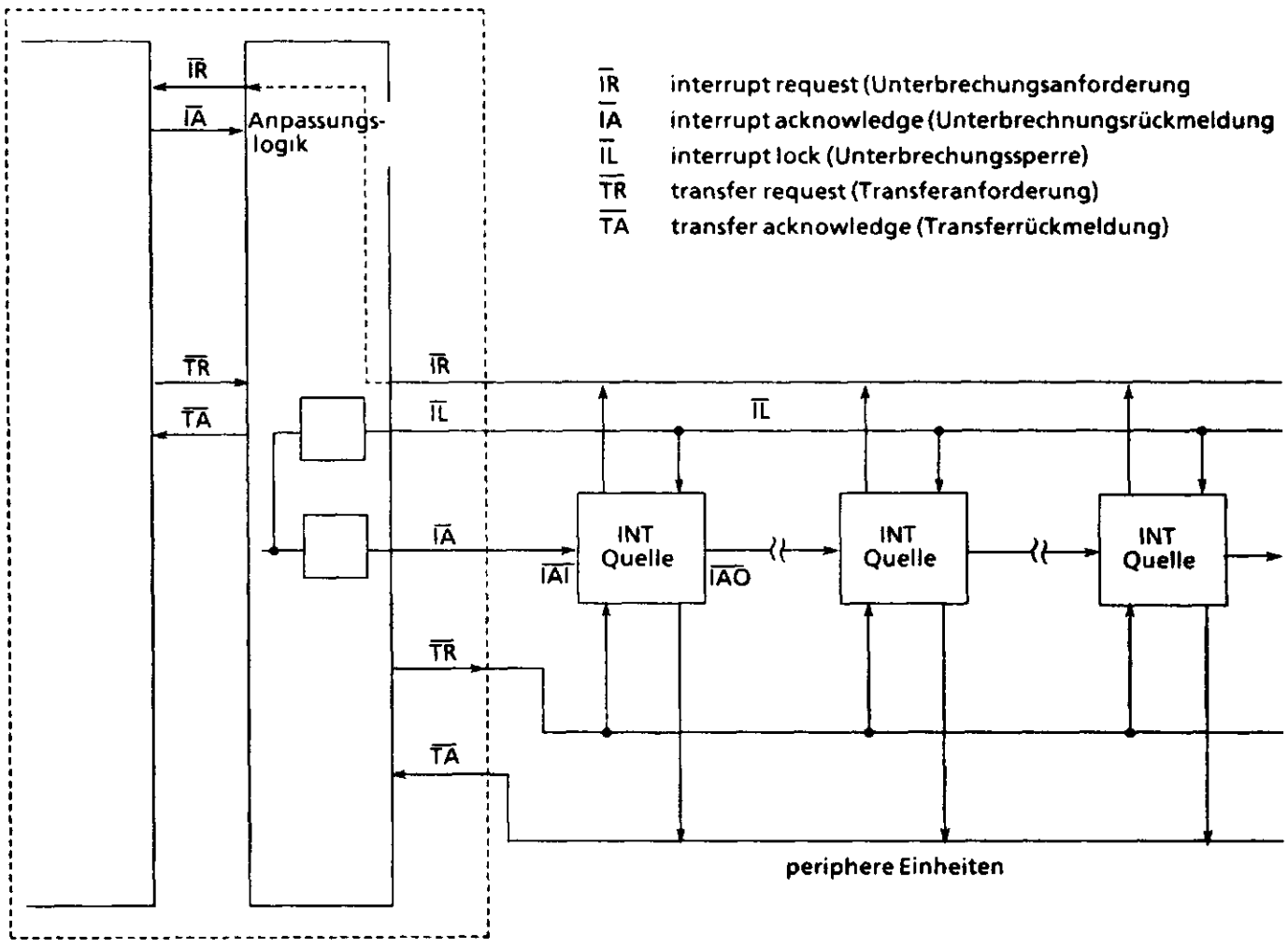


Bild 3.32 Interrupt-Logik

Zur Identifikation der Interrupt-Quelle im Zentralprozessor wird vom initiierenden Partner im EA-Verkehr ein Vektor übergeben.

Die Zentraleinheit holt den Interrupt-Vektor vom höchstprioren Teilnehmer ab, indem sie während $\overline{IL}$ = L einen EA-Lesezugriff ($\overline{TR}$ = L, M/$\overline{IO}$ = L, R/$\overline{W}$ = H) ausführt. Die Priorität ergibt sich dabei aus der Position der peripheren Einheit innerhalb der Kette der Signale $\overline{BAO}$-$\overline{BAI}$. Der höchstpriore Teilnehmer, der eine Interrupt-Anforderung gestellt hat und das Signal $\overline{IAI}$ erhält, schaltet seinen Interrupt-Vektor auf; die anderen Teilnehmer sind aufgrund des Signals $\overline{IL}$ während dieser Zeit gesperrt. Die Zentraleinheit muß während der Abgabe des Signals $\overline{IL}$ das Bussystem belegen, um zu verhindern, daß über DMA ein ungewollter EA-Zugriff stattfindet, der als Vektorzugriff mißdeutet werden könnte. Solange $\overline{IL}$ aktiv ist, sind zwar noch Speicherzugriffe möglich, nicht jedoch EA-Zugriffe.

Während der Ketteneinschwingzeit kann die Zentraleinheit noch Speichertransfers durchführen, um diese Zeit sinnvoll zu nützen. Das Signal $\overline{IL}$ und damit die Busbelegung wird jedoch möglichst kurz gehalten (typ. kleiner als 3 µs), denn während dieser Zeit können andere Busteilnehmer den Bus nicht benutzen. Schnelle Peripheriegeräte, wie z. B. Plattenspeicher, sind aber darauf angewiesen, daß sie in relativ kurzen, regelmäßigen Abständen zum Bus zugreifen können; andernfalls sinkt die Datenrate erheblich ab, z. B. durch Drehwartezeiten.

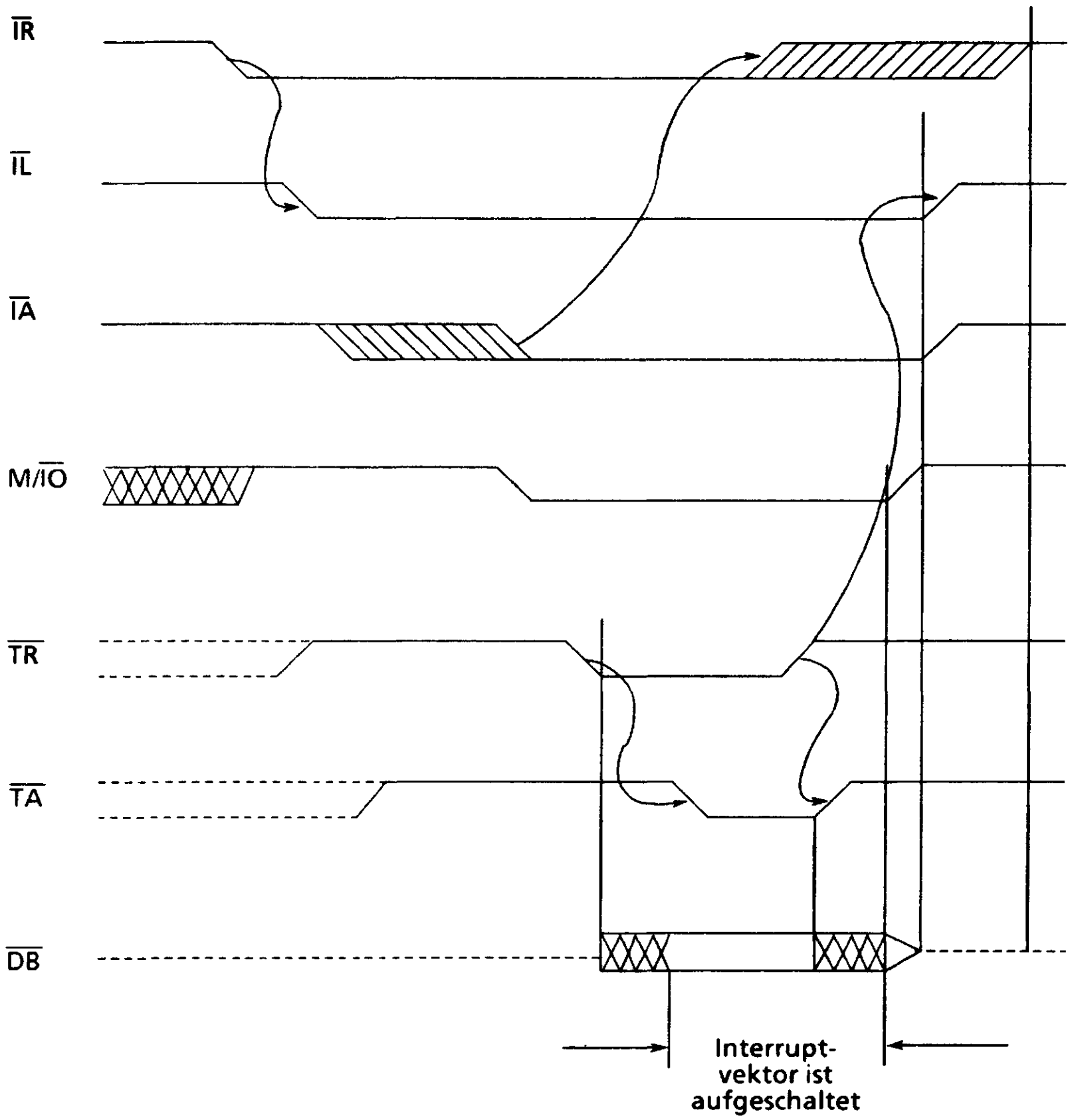
Der übergebene Interrupt-Vektor der peripheren Einheit zeigt der Zentraleinheit eindeutig an, welcher Teilnehmer eine Interrupt-Anforderung $\overline{IR}$ gestellt hat.

Die periphere Einheit, die den Interrupt-Vektor sendet, übergibt diesen nach gleichen Regeln wie beim Datentransfer, d.h., daß $\overline{TR}$ (transfer request) durch $\overline{TA}$ (transfer acknowledge) quittiert wird, sobald der Vektor stabil auf den Bus geschaltet ist.

Das Rückmeldesignal $\overline{TA}$ ermöglicht der peripheren Einheit eine Zeitdehnung bei der Vektorübergabe.

Hat eine periphere Einheit ihren Vektor übergeben, so nimmt sie ihre Interrupt-Anforderung zurück. Es ist solange $\overline{IR}$ = L, bis alle peripheren Einheiten, die eine Interrupt-Anforderung gestellt haben, ihren Vektor übergeben haben. Ist noch $\overline{IR}$ = L, nachdem die Interrupt-Routine von der Zentraleinheit beendet ist, so hat noch eine weitere periphere Einheit eine Interrupt-Anforderung gestellt und es muß eine weitere Interrupt-Routine durchgeführt werden. Dazwischen ist aber $\overline{IL}$ inaktiv, da die Zentraleinheit im Mikroprogramm weiterläuft. Dadurch ist zu diesem Zeitpunkt DMA-Verkehr möglich.

Ein aktiviertes Signal $\overline{IR}$ darf erst nach der Interrupt-Lesesequenz zurückgenommen werden.
Diese Interrupt-Sequenz ist in Bild 3.33 dargestellt.



IL	M/IO	Zugriffsart mit TR
H	X	Zugriff zum ZSP oder EA
L	H	Speicherzugriff
L	L	Zugriff zum Interrupt-Vektor

Bild 3.33 Interrupt-Sequenz

3.3.6 Basisparametrierung

Durch die Basisparametrierung werden den Anschaltungen der peripheren Geräte, d.h. den ihnen zugeordneten EA/Adressen, die entsprechenden Interrupt-Vektoren zugeordnet.

Nach dem Rücksetzen ist allen Referenzadressen der Interrupt-Vektor mit der Nummer 1 zugeteilt, d. h. der Auftrag "Basisparametrieren" erfolgt über den Basisprozeßblock. Handelt es sich hierbei um den ersten Auftrag an die Anschaltung nach dem Rücksetzen, so werden beim Abschluß des Auftrags die vorhandenen Anlauftestergebnisse im physikalischen Parameterblock PHYSPB hinterlegt.

Eine Basisparametrierung ist aus Anschaltungssicht prinzipiell immer möglich, d.h. daß bereits vergebene Vektoren überschrieben werden können (Nachbasisparametrierung). Die Abwicklung des Auftrags erfolgt dann unter dem Prozeßblock, der zur Zeit des Auftragsanstoßes aktuell war. Nach Abschluß des Auftrags ist die Anschaltung auf den neuen Prozeßblock eingestellt.

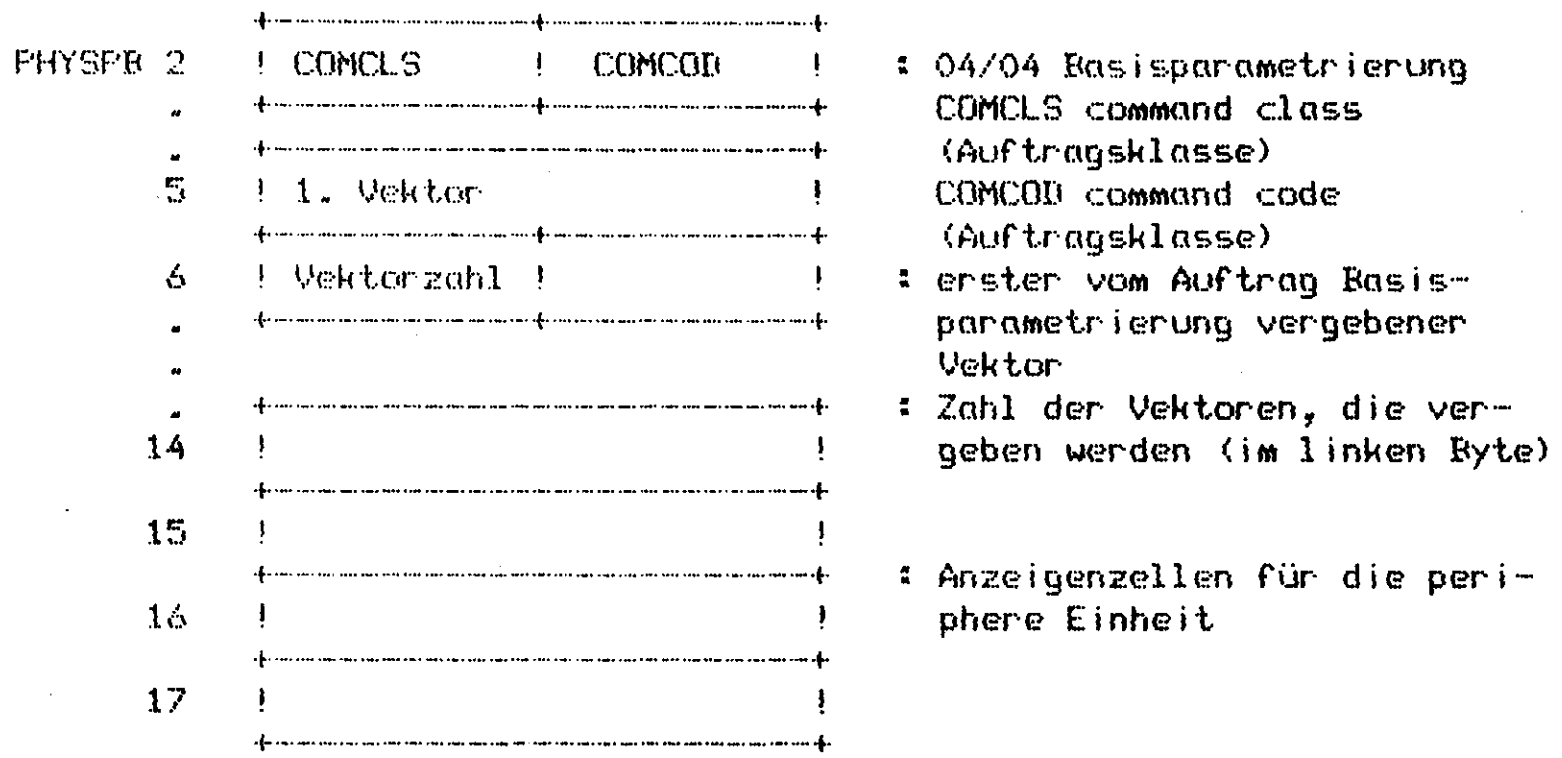
Ablauf der Basisparametrierung

Eine Basisparametrierung kann als Sammelbasisparametrierung, Einzelbasisparametrierung oder Nachbasisparametrierung durchgeführt werden. Bei einer Sammelbasisparametrierung werden mit einem Auftrag mehrere Vektoren vergeben (z. B. für periphere Einheiten mit mehreren Subadressen). Bei einer Einzelbasisparametrierung wird mit einem Auftrag nur ein Vektor übergeben. Durch eine Nachbasisparametrierung kann eine periphere Einheit, die bereits basisparametriert ist, erneut basisparametriert werden.

Nach dem Rücksetzen müssen alle peripheren Einheiten, die rückgesetzt sind, basisparametriert werden. Bevor die Basisparametrierung durchgeführt werden kann, muß vom Betriebssystem die Vektorliste VEKLI aufgebaut werden. In VEKLI werden die Adressen der zugehörigen Prozeßblöcke (Geräteprozeßblock DEVPRB) eingetragen. Nicht benutzte Plätze in VEKLI werden mit der Adresse des Sonderprozeßblocks PRBNOP (Adresse 32) geladen.

Vor der Auftragserteilung zur Basisparametrierung bereitet das Systemprogramm den Prozeßblock und den physikalischen Parameterblock PHYSPB vor. Eine rückgesetzte periphere Einheit darf zum Basisparametrieren nur auf den Basisprozeßblock PRBBAS (Adresse 48) zugreifen. Der Basisprozeßblock PRBBAS dient ausschließlich zur Basisparametrierung. In ihm steht der Verweis auf den physikalischen Prozeßblock PHYSPB. Die Zelle SWCTR gibt an, ob ein Interrupt gestellt werden soll oder nicht. NORTER und FINTER sind mit 0 vorbesetzt.

Im physikalischen Parameterblock ist der Parametrierauftrag formuliert:



Der physikalische Parameterblock (Zelle 2) enthält den Auftragscode. Der physikalische Paramterblock (Zelle 5) enthält den ersten Vektor, der mit diesem Auftrag vergeben werden soll; er wird von der peripheren Einheit der Ansprechadresse zugeordnet (EA-Adresse des Ausgabe-Befehls). Der Vektor zeigt auf einen Platz in der Vektorliste VEKLI, wo die Adressen des zugeordneten GerätеprozeßBlocks DEVPRB stehen. Zelle 6, linkes Byte des physikalischen Parameterblocks gibt an, wieviel Vektoren mit diesem Parametrierauftrag vergeben werden sollen; eine periphere Einheit mit mehreren Vektoren kann entweder auf einmal oder mit mehreren Aufträgen Basisparametrierung basisparametriert werden. Im physikalischen Parameterblock Zelle 14 - 17 kann die periphere Einheit Anzeigen hinterlegen (z. B. Anlauftestergebnisse).

Nachdem das Betriebssystem den Prozeßblock und physikalischen den Parameterblock vorbereitet hat, erfolgt der Auftragsanstoß mit dem Ausgabebefehl EAS (EA-Schreiben). Die Einheit interpretiert den Basisparametrierauftrag (PHYSPB 2) und übernimmt Vektoranfang und Vektorzahl (PHYSPB 5 und 6); in PHYSPB 14 - 17 trägt sie - falls vorhanden - Anzeigen ein. In NORTER wird die Abschlußkennung eingetragen und dann, abhängig von SWCTR, eine Interrupt-Anforderung gestellt oder nicht. Soll eine Interrupt-Anforderung gestellt werden, so wird der Vektor 1 übergeben, der auf den BasisprozeßBlock PRBBAS zeigt, wenn die periphere Einheit zuvor noch auf PRBBAS eingestellt war.

Der Auftrag ist damit von der Peripherie-Seite aus abgeschlossen. Aufgrund der übernommenen Daten (PHYSPB 5 und 6) kann die periphere Einheit später, z. B. beim nächsten Auftrag die Adresse ihres Geräteblocks aus

der VEKLI abholen. Nach erfolgter Basisparametrierung meldet sich die periphere Einheit mit ihrer individuellen Vektornummer, die sie aus PHYSPB Zelle 5 und 6 ermittelt hat.

Das Betriebssystem wartet auf den Abschluß des Basisparametrierauftrags, was durch zyklisches Abfragen von NORTER oder durch Interrupt (bisheriger Vektor) erkannt werden kann. Nach einer Ergebnisauswertung kann der nächste Auftrag formuliert werden.

Nach der Basisparametrierung erfolgt normalerweise eine Parametrierung der peripheren Einheit; daraufhin ist sie in der Lage, ihre volle Funktion im Rahmen des EA-Verkehrs anzunehmen.

4. Betriebs-,Test- und Diagnosemittel

Die Betriebs-,Test- und Diagnosemittel sind grundsätzlich Bestandteil der Modellreihe SICOMP (enthalten im Hardware-/Firmware-Speicher bzw. im Systemprogrammpaket). Eine Ausnahme bildet die für aufwendigere Diagnosen optionell einsetzbare Serviceprozessor-Anschaltung bzw. der daran anschließbare Serviceprozessor incl. seiner speziellen Diagnose-Software.

Die Betriebsmittel

- Urlader
- virtuelle Konsole (Teilfunktionen)
  dienen dazu, ein System in einen betriebsbereiten Zustand zu bringen bzw. steuernd in das laufende Betriebsgeschehen einzugreifen.

Die hardware-bezogenen Test- und Diagnosemittel

- Basistest,
- urladefähige Zentraleinheits-Testprogramme und
- virtuelle Konsole (Testfunktionen)
  weisen - ausgehend von einem minimalen Funktionsumfang - die einwandfreie Funktionsfähigkeit der für einen Systemlauf notwendigen Hardware-Komponenten Schritt für Schritt nach. Bei erkannten Fehlern liefern sie Aussagen über den Ort und die Art der defekten Komponente.

Das Hilfsmittel

- Teleservice-Anschluß
  ist kein Servicemittel im eigentlichen Sinn, sondern stellt einen besonderen Zugang (über das normale Fernsprechnetz) von einer Servicezentrale zu den lokal am System vorhandenen Betriebs- und Servicefunktionen dar.

Das optionell verfügbare, systemorientierte Servicemittel

- Serviceprozessor-Anschaltung/Serviceprozessor
  unterstützt die Diagnose in den Fällen, in denen die standardmäßig in jedem System vorhandenen Servicemittel nicht mehr ausreichen (z.B. Schnittstellenprobleme zwischen Hardware und Software).

4.1 Basistest

Der Basistest stellt die Funktionsfähigkeit eines Systems in seinen Kernfunktionen sicher. Mit dem einwandfreien Ablauf des Basistests ist gewährleistet, daß ein Betrieb der virtuellen Konsole, das Urladen und der Einsatz der Zentraleinheits-Testprogramme möglich ist. Ferner wird im Basistest der Typ und die Ausstattung der Zentraleinheit ermittelt und in der entsprechenden Zelle des Hardware-Verständigungsbereichs hinterlegt. Der Basistest läuft automatisch nach Rücksetzen des Systems, abhängig von

den im Steuerwort ANLAWE hinterlegten Steuerbits (Kapitel 3.1.7) bzw. nach einem gezielten Anstoß durch Bedienung der virtuellen Konsole. Um ein spezielles, anwenderspezifisches Anlaufverhalten zu erreichen, kann die Steuerzelle ANLAWE vom Anwender entsprechend modifiziert werden.
Der Basistest liefert zwei Arten von Fehlerhinweisen

- Gut-/Schlecht-Aussage durch die Anzeige an den Diagnose-Displays (processor check, memory check, console check, bootstrap check).
  Bei positiv verlaufenem Test leuchtet die entsprechende Lampe auf (Dauerlicht); bei negativem Verlauf, d.h. in einer Komponente wurde ein Fehler festgestellt, bleibt die der entsprechenden Komponente zugewiesene Lampe dunkel.

- Detaillierte Fehlermeldung an der virtuellen Konsole

Der Basistest umfaßt max. folgende Teiltests

- Zentralprozessor-Basistest
- Hauptspeicher-Basistest
- Anschaltungs-Basistest
- Zentralprozessor-Kontrolle.

4.1.1 Zentralprozessor-Basistest (CPU-Basistest)

Der Zentralprozessor-Basistest überprüft die Teile der Zentraleinheit, die das Festspeicherprogramm VICOM zum weiteren Ablauf benötigt. Er besteht aus mehreren Teilen:

- Zentralprozessor-Kerntest

  Der Kerntest ist Bestandteil der Hard- und Firmware (Mikroprogramm) des Zentralprozessors und wird durch VICOM nur in bestimmten Teilen gesteuert und mit Testmustern versorgt.
  Er wird durch das Hardware-Signal RSC (reset CPU, reset central processing unit, Rücksetzen Zentralprozessor) initiiert und durch den Befehl KTE (Kerntest-Ende) in VICOM beendet bzw. abgebrochen.
  Der Kerntest umfaßt folgende Teile:

  . Test von Zentralprozessor-Teilen durch das Mikroprogramm
  . Summenprüfung eines Teils des Festspeichers für das Mikroprogramm
  . Signaturüberwachung des Zentralprozessor-Basistests bis zum KTE-Befehl per Hardware (Die Signatur ist eine spezielle Art einer Prüfsumme).
  . Zeitüberwachng des Zentralprozessors-Befehlstests bis zum KTE-Befehl per Hardware.

  Im Fehlerfall bewirkt der "Kerntest" einen Hardware-Halt (Lampe "STOP" an).

- Zentralprozessor-Anlauftest

Der CPU-Anlauftest wird bei jedem nicht maskierbaren Interrupt (NMI) außerhalb Z0 initiiert. Er stellt die Interpretation des NMI-Ereignisses sicher. Er setzt den Befehl SPR/CRX als funktionsfähig voraus. Darauf aufbauend werden die Befehle LAF/RAI (F2=1), UND/RAI (F2=1) und LAF/RR, sowie die Sprungmaskenauswertung im Befehl SPR/CRX getestet. Der Test ist so strukturiert, daß 1-bit-Fehler in der Dekodierung des F0-, F1- und F2-Felds eines Befehls sowie in der Erkennung des F3-Felds bis auf wenige Ausnahmen zum "STOP" führen.

- Zentralprozessor-Befehlstest/RAM-Test/PROM-Test

Der Zentralprozessor-Befehlstest wird nur beim NMI "RS mit BF" durchgeführt. Beim NMI "RS ohne BF" wird der Kerntest mit dem Befehl KTE abgebrochen. Der Zentralprozessor-Befehlstest ist Bestandteil des Kerntests. Er testet den Teil des Zentralprozessor-Befehlssatzes, der vom Programm VICOM und den urladefähigen Zentraleinheits-Testprogrammen benötigt wird. Den Abschluß des Zentralprozessor-Befehlstests bildet der KTE-Befehl mit Signaturübergabe an den Kerntest, der damit beendet wird.
Im Teil RAM-Test (Zentralspeicher-Test) wird die ECC (error correcting code)-Generierung im Adreßbereich 0 - (6 x 1024 - 1) mit anschliessendem Kontrollesen durchgeführt und die Korrektureinrichtung aller Speichermodule zugeschaltet. Ferner wird ein Datenbustest und ein Adreßbustest durchgeführt. Der Teil PROM-Test umfaßt den Test des nicht durch den Kerntest kontrollierten Festspeichers mit einer Summenprüfung, die Kontrolle der jeweiligen Summenprüfung und die Kontrolle der jeweiligen Testschrittnummern.

4.1.2 Memory-Basistest (Zentralspeicher-Basistest)

Im Rahmen des Memory-Basistests wird die Zentraleinheits-Konfiguration festgestellt und eine ECC-Generierung für den gesamten Zentralspeicher durchgeführt. Es wird der Zentralspeicher und der Cache-Speicher (falls vorhanden) geprüft.
Dieser Test wird durch die Ereignisse NMI "RS mit BF", Virtuelle-Konsole-Kommandos 'VERIFY' und 'TEST' und durch anwendergesteuerten Anlauf im Zustand Z0 bei gesetztem Inbi 3 in ANLAWE angestoßen.
Während des Tests festgestellte Fehler werden in einem Puffer hinterlegt und bei funktionsfähiger virtueller Konsole aufbereitet und an diese ausgegeben. Dieser definierte Pufferbereich liegt in dem beim RAM-Test untersuchten Bereich. Im fehlerfreien Fall wird die Lampe 'memory check' eingeschaltet.

- Zentraleinheits-Parameter-Ermittlung

  Es wird festgestellt an welcher Zentraleinheit VICOM zum Ablauf kommt und welcher Erweiterungsprozessor am Erweiterungsprozessor-Steckplatz vorhanden ist. Die ermittelte Information wird in der Zelle MAKENN des Hardware-Verständigungsbereichs hinterlegt (Kapitel 3.1.7).

- ECC-Generierung (Generierung des error correcting code)

  Für den gesamten Speicherausbau wird die ECC-Generierung durchgeführt. Der ursprüngliche Speicherinhalt wird überschrieben.

- Speichertest

  Der Speichertest prüft die prinzipielle Funktion der ersten 256 K* byte (falls vorhanden, sonst nur 128K*byte). Damit wird sichergestellt, daß der Speicherbereich für den Ablauf der Zentraleinheits-Prüfprogramme fehlerfrei ist.

- Cache-Speicher-Test

  Es wird geprüft, ob ein Cache-Speicher vorhanden, das entsprechende Bit in der Zelle MAKENN des Hardware-Verständigungsbereichs gesetzt und der Cache eingeschaltet ist. Danach wird ein Minimaltest des Cache-Speicher durchgeführt, der die prinzipielle Funktionsfähigkeit sicherstellt. Bei aufgetretenem Fehler wird der Cache ausgeschaltet, das Kennbit in MAKENN bleibt gesetzt. Bei funktionsfähiger virtueller Konsole wird ein Fehler des Cache-Speichers auch dort gemeldet.

4.1.3 Device-Basistest (Anschaltungs-Basistest)

Im Device-Basistest werden die Anschaltungen der virtuellen Konsole und des Urladegeräts sowie die Geräte selbst durch eine Ein- bzw. Ausgabe eines Testmusters getestet.
Fehler werden soweit wie möglich über die virtuelle Konsole bzw. das Ersatzgerät gemeldet. Außerdem werden die Diagnose-Displays console check

bzw. bootstrap check zu Beginn des Tests ausgeschaltet und im fehlerfreien Fall wieder eingeschaltet.
Den Kern des Device-Basistests bildet der anschaltungsinterne Anlauftest, der durch ein Sammelrücksetzen gestartet wird. Die Testergebnisse werden in einem RAM auf der Anschaltung zwischengespeichert. Sie werden durch die Anschaltungen beim ersten Auftrag nach dem Sammelrücksetzen an den Basisprozeßblock übergeben und in aufbereiteter Form über die virtuelle Konsole gemeldet. Für bestimmte Anlauffälle kann über die Zelle ANLAWE der anschaltungsinterne Anlauftest durch Sammelbeenden vor dem ersten Auftrag abgebrochen werden.

- VK-Test (Test der virtuellen Konsole)

  Der erste Auftrag nach dem Sammelrücksetzen im Anlauf an die virtuelle Konsole ist die "VK-Freigabe". Bei diesem Auftrag werden die Anlauftest-Ergebnisse übergeben. Abschließend erfolgt die VICOM-Anfangsmeldung und danach die Testmeldung, bei der die Zeichen der Spalten 2 bis 5 der ISO-7-bit Zeichenmatrix spaltenweise hintereinander ausgegeben werden. Im Fehlerfall wird auf das an den Codierschaltern eingestellte Ersatzgerät der virtuellen Konsole umgeschaltet. Gelingt dies nicht, arbeitet das Festwertspeicherprogramm VICOM trotzdem weiter (urladen usw). Die virtuelle Konsole-Berechtigung kann von einem beliebigen virtuellen konsole-fähigen Gerät erobert werden.

- BT-Test (Test des Urladegeräts)

  Der erste Auftrag nach dem Sammelrücksetzen im Anlauf an das Urladegerät ist das Einlesen der Buchführung bzw. des ersten Programmvorspanns. Bei diesem Auftrag werden die Anlaufergebnisse übergeben. Implizit durch den Eingabeauftrag selber erfolgt ein minimaler Gerätetest; ein Datenträger mit ordnungsgemäßem Urladeformat wird vorausgesetzt.

4.1.4 CPU-Kontrolle (Zentralprozessor-Kontrolle)

Programmlaufbesonderheiten während eines Laufs im höchstprioren Zustand Z0 werden vom Zentralprozessor durch Ablegen des Unterbrechungsanzeigenregister UAR im Standardregister R0 registriert. VICOM prüft das R0 in unregelmäßigen Zeitabständen und springt bei R0 = 0 in die Fehlerbearbeitung des Zentralprozessor-Basistests. Der Fehler wird gemeldet; der Zentralprozessor geht in den Basistest - Fehlerstop.

4.2 Urladefähige Zentraleinheits-Testprogramme

Im Systemprogrammpaket (SPP) sind standardmäßig die urladefähigen Zentraleinheits-Prüfprogramme hinterlegt und in der Buchführung (s. Kapitel 4.4 Urladen) als Testprogramme gekennzeichnet.
Diese Testprogramme können abhängig vom Bit "Testmodus" im Steuerwort ANLAWE in zwei Modi laufen (s. Kapitel 3.1.7):

- Normalmodus
  Ein einzelnes Testprogramm wird von Hand geladen und zum Ablauf gebracht.

- Testmodus
  Es wird ein automatischer Ablauf aller Testprogramme, vom Urlader gesteuert, durchgeführt.

Folgende Testprogramme stehen zur Verfügung:

- ZEWAM
  Es wird die gesamte Befehlsbearbeitung aller implementierten Befehle getestet.

- ZUWAM
  Es wird die Prioritätssteuerung der Zentraleinheit überprüft.

- MEWAM
  Es wird der gesamte Speicherausbau überprüft. Außerdem werden die vorhandenen Fehlerkorrektur- und Fehlerspeichereinrichtungen überprüft.

- TAWAM
  Es wird die Testanschaltung (TESTAS) geprüft.

- ZIWAM
  Es wird der Zeitimpulsgeber (ZIG) geprüft.

4.3 Virtuelle Konsole

4.3.1 Virtuelles Konsol-Gerät (VK-Gerät)

Die virtuelle Konsole (VK) besteht aus dem VK-Gerät (i. allg. Sichtgerät, z. B. DS 075-B) mit zugehöriger Anschaltung und dem im Festspeicher hinterlegten VK-Programm, das im Festspeicherprogramm VICOM enthalten ist.
Ihr Haupteinsatzgebiet ist die Software-Diagnose hardware-naher Programmsysteme, insbesondere der Betriebssystem-Software und der Zentraleinheits-Testprogramme.
Für den Test von Anwenderprogrammen unter einem Betriebssystem ist die virtuelle Konsole nicht das geeignete Hilfsmittel. Dazu stehen Testprogramme zur Verfügung (z. B. TEST-M, TESTS-M, DEBUG-M).
Die Funktionen der virtuellen Konsole sind durch Software realisiert und ein unmittelbarer Hardware-Eingriff ist nicht gegeben (Ausnahme: Funktionen der Testanschaltung TESTAS).
Die Funktionen der virtuellen Konsole lassen sich in sechs Klassen einteilen:

- organisatorische Funktionen,
- Daten-Ein-/Ausgabe-Funktionen,
- Lade- und Start-Funktionen,
- Zentraleinheits-Test-Funktionen,
- Debugging-Funktionen (Fehlersuch-Funktionen)
- Fehlerreaktionsfunktionen.

Ein VK-Gerät ist i. allg. ein Sichtgerät - aber auch Teleservice oder die Serviceprozessor-Anschaltung (SPAS) kann ein VK-Gerät sein. Das Gerät muß bestimmten Anforderungen genügen, d.h. es muß VK-fähig sein. Der entsprechenden Gerätebeschreibung ist zu entnehmen, ob sich das jeweilige Gerät als VK-Gerät eignet oder nicht.

Das VK-Gerät bzw. die zugehörige Anschaltung müssen im laufenden System zwischen Bedienungen an VICOM und sonstigen Eingaben unterscheiden. Geräteseitig ist dies durch zwei unterschiedliche Betriebsarten realisiert, nämlich den VK-Modus und den Normalmodus. Der VK-Modus kann nur bei einer bestehenden VK-Berechtigung eingenommen werden. Die VK-Berechtigung wird einzig durch VICOM per VK-Freigabe erteilt und per VK-Sperre entzogen.

Die Eingabe über ein VK-Gerät im VK-Modus (VK-Kommando) hat den nicht maskierbaren Interrupt (NMI) "VC" und damit eine Unterbrechung des Zentralprozessors und dessen Übergang in den Zustand Z0 zur Folge. Der VK-Modus kann jederzeit angefordert werden. Die virtuelle Konsole kann auch im Stop-Zustand bedient werden.

Es ist stets genau ein Gerät das derzeit gültige und damit aktuelle VK-Gerät. In bestimmten Anlauffällen ist das an den VK-Codierschaltern eingestellte Gerät (definiert durch die EA-Geräteadresse) das aktuelle VK-Gerät.

4.3.2 Kommandos

Die virtuelle Konsole (VK) wird über Kommandos bedient. Jedes Kommando löst eine definierte Funktion aus. Das Ende der Kommandoausführung (mit oder ohne Fehler) wird gemeldet.

Ein Kommando belegt maximal genau eine Bildschirmzeile. Der allgemeine Aufbau läßt sich wie folgt darstellen:

Kommandowort parameter parameter ... parameter;

Trennzeichen zwischen "Kommandowort" und "parameter" und zwischen "parameter" untereinander sind ein oder mehrere Blanks.
Den Kommandoabschluß bildet ein Semikolon (;) oder ein <ETX> (autom. generiert durch die Datenübertragung).
Jedes VK-Kommando wird eingeleitet mit dem Kommandowort, das das jeweilige Kommando identifiziert. Dieses Kommandowort muß immer linksbündig in der Kommandozeile stehen. Das erste Zeichen des Worts muß immer geschrieben werden, die weiteren Zeichen jedoch nur, soweit sie zur eindeutigen Unterscheidung von allen anderen Kommandoworten notwendig sind.
Für Kommandowörter sind also Abkürzungen möglich. Dabei können sogar Zeichen mitten aus dem Wort weggelassen werden.
Es kann also z. B. anstelle von CHANGE auch CHG oder CANE geschrieben werden. Die Kurzform muß aber immer so beschaffen sein, daß dem VK-Programm eine eindeutige Unterscheidung von den anderen Kommandoworten möglich ist. Es ist daher unzulässig, das Kommando STOP durch STP abzukürzen, da diese Zeichenkombination auch im Kommandowort STEP enthalten ist.
Zu beachten ist, daß diese Regel von einigen Kommandos unterlaufen wird (s. Kapitel 4.3.3 "Kommandoübersicht", Kommandos: GO und STEP).

Zum Schutz vor schwerwiegenden Fehlbedienungen im laufenden System sind bestimmte Kommandos nur im Stop-Zustand möglich.

Darüber hinaus erfordern einige Kommandos eine nochmalige Quittierung, die angefordert wird mit der Meldung

VICOM: REPLY YES/NO !

Jetzt ist nur eines der folgenden Kommandos möglich:

YES;	---	Das die Quittung erfordernde Kommando wird ausgeführt.
NO;	---	Das die Quittung erfordernde Kommando wird nicht ausgeführt. VICOM ist wieder kommandobereit.

4.3.3 Kommandoübersicht

Kommando	kurz	Bedingung	Kommando-Erklärung (im Abschnitt)	
ACTIVATE	ACT		Aktivierung eines checkpoints	D
APZ	APZ		Aktivierung eines Prozesses	D
BOOT	BT	Stop-Zustand	Urladen	C
BREAK	BR	Kommando in Bearbeitung	Abbruch von Aktionen des Festspeicher-programms	A
CALCULATE	CAL		Hexa-Rechner	E
CHANGE	CHA		Dateneingabe	B
CHECK	CCK	Stop-Zustand	Test einer speziellen peripheren Einheit	E
DEACTIVATE	DEA		Deaktivierung eines checkpoints	D
DELETE	DEL		Löschen eines checkpoints	D
DFP	DFP		Deaktivierung eines Prozesses	D
DISPLAY	DIS		Datenausgabe	B
DUMP	DU	Stop-Zustand	komprimierte Datenausgabe (hexadezimal)	B
END	E		Aktuellem VK-Gerät wird VK-Berechtigung entzogen	A
FORMAT	F		Formatauswahl für Datenausgabe	B
GENERATE	GEN	Stop-Zustand	ECC-Generierung des Zentralspeichers	E
GO	G	fortsetzbarer Stop	Programm wird fortgesetzt	D
INFORM	INF		Meldung eines checkpoints ohne Stop	D
INTERRUPT	ITR		Meldung eines checkpoints bewirkt Programmlaufbesonderheit	D
LIST	L		Definition eines Protokollgeräts	A
MESSAGE	M		Mitteilung an ein anderes Ein-/Ausgabe Gerät	A
NO	N	nur als "Reply"	Rücknahme eines Kommandos	
NOLIST	NL		Deaktivierung eines Protokollgeräts	A
NOREQUEST	NR		VK-Gerät gegen Deaktivierung geschützt	A
OFFSET	O		Offset-Voreinstellung für Adreßeingabe	B
PASSWORD	PA		Definition eines Paßworts (Schutz noch nicht aktiviert)	A
PRESENT	PRE		Voreinstellung der Adreßergänzung	B
PROTECT	PRO		Aktivierung des Paßwort-Schutzes	A
QUEUE	QU		Ausgabe von Fädellisten (z. B. von PREBs, PHYSPBs)	B
Q+	Q+		Vorwärtsblättern in Fädellisten	B
Q-	Q-		Rückwärtsblättern in Fädellisten	B
REQUEST	REQ		anderes Gerät kann auf Wunsch VK-Gerät werden	A
RESET	RSET	Stop-Zustand	Beenden und Rücksetzen eines peripheren Geräts (selektiv)	E
RESTART	RSTA	Stop-Zustand	Start eines Programms wie nach Spannungswiederkehr	C
SEARCH	SE		Suche nach Daten in einem Speicherbereich	B

Kommando	kurz	Bedingung	Kommando-Erklärung	(im Abschnitt)
SHOW	SW		Übersicht über checkpoints (Tabelle)	D
START	STRT	Stop-Zustand	Start eines Programms wie nach Urladen	C
STEP	S	fortsetzbarer Stop	Fortsetzung für angebbare Zahl von Befehlen	D
STOP	STOP	kein Stop-Zustand	fortsetzbarer Stop	D
STORE	STOR	Stop-Zustand	Abzug eines Speicherbereichs als urladefähiges Programm	E
TERMINATE	TER	Stop-Zustand	selektives Beenden eines peripheren Geräts	E
TEST	TST	Stop-Zustand	Anlagenselbsttest (Zentralspeicher, VK, Urlade-Gerät)	E
VC	VC		aktuelles VK-Gerät deaktiviert, neues VK-Gerät aktiviert	A
VERIFY	VER	Stop-Zustand	Test von Zentralspeicher, Urlade- und VK-Gerät	E
VIEW	VW	Stop-Zustand	Gerätekenndaten ausgeben	E
YES	Y	nur als "Reply"	Bestätigung eines Kommandos	
?	?		Anweisung zum Urladen von VCINFO	A
+	+		Vorwärtsblättern	
-	-		Rückwärtsblättern	
*	*	nur im Mode "Inkrementierende Eingabe" siehe CHANGE		B
password		nur wenn Paßwort-schutz besteht		

(im Abschnitt).... entsprechend Abschnitt im Kapitel 4.3.4

4.3.4 Kurzbeschreibung der Kommandos

In diesem Abschnitt sind die einzelnen Kommandos kurz beschrieben. Die unterstrichenen Buchstaben des Kommandoworts stellen eine mögliche Kurzform für die Eingabe dar. Sind Buchstaben doppelt unterstrichen, so sind sie die einzig mögliche Eingabeform. Angaben mit Überstrich (z. B. parameter 1) sind optionell angebbar.

Die Kommandobeschreibung ist in fünf Gruppen aufgeteilt (Abschnitte A, B, C, D, E; siehe Hinweis in der Kommandoübersicht). Innerhalb dieser Gruppen sind die Kommandos alphabetisch geordnet. Es sind dies:

- organisatorische Kommandos
- Daten-Ein-/Ausgabe
- Programminitiierung
- Debugging
- Wartungs- u. Hilfsfunktionen.

Die Abkürzung "VK" steht im folgenden für "virtuelle Konsole".

A Organisatorische Kommandos

o BREAK;

Mit diesem Kommando können laufende Aktivitäten des Festspeicherprogramms abgebrochen werden.

o END;

Mit diesem Kommando wird dem aktuellen VK-Gerät die VK-Berechtigung entzogen und ggf. der Paßwort-Schutz aktiviert (sofern im Verständigungsbereich ein "password" vorhanden ist).

Die Lampe "console check" erlischt. Das Gerät bleibt weiterhin aktuelles VK-Gerät, eine Bedienung von VICOM ist aber nicht mehr möglich.

o LIST iogeräteadresse;

Mit diesem Kommando wird die Protokollierung eingeschaltet. Die "iogeräteadresse" definiert das Protokollgerät.

o MESSAGE iogeräteadresse mitteilung

An das durch die "iogeräteadresse" definierte VK-akzeptierende Gerät wird die "mitteilung" (Zeichenfolge) ausgegeben. Ein VK-akzeptierendes Gerät ist ein Gerät, das die Steuerinformation 04 (hexa) auf dem Datenbus versteht (s. Kapitel 3.3.3.1), z. B. ein Drucker.

o NOLIST;

Mit diesem Kommando wird die Protokollierung wieder ausgeschaltet, wobei dieses Kommando (ohne die Kommandoquittung) die letzte Protokollzeile bildet.

o NOREQUEST; --- Eroberung sperren

o REQUEST; --- Eroberung freigeben

Mit diesen Kommandos kann die Eroberung softwaremäßig gesperrt (NOREQUEST) oder freigegeben (REQUEST) werden. Damit ist es möglich, eine VK-Sitzung gegen unerwünschte Unterbrechungen durch andere Anwender zu sichern. Zu beachten ist, daß eine eingeschaltete Sperre bei 'unklarem' VK-Gerät aufgehoben wird, damit im Fehlerfall die VK-Berechtigung von einem anderen VK-fähigen Gerät erlangt werden kann.

o PASSWORD password;

Mit diesem Kommando wird der Paßwort-Schutz vorbereitet, d. h. das Paßwort im Verständigungsbereich eingetragen, wenn ein "password" im Kommando angegeben ist. Ist kein "password" angegeben, wird das Paßwort im Verständigungsbereich gelöscht, so daß in Zukunft kein Paßwort-Schutz mehr besteht.
Durch Eingabe des Paßworts ist der Schutz noch nicht aktiviert (siehe PROTECT).
Das "password" darf aus maximal 6 darstellbaren Zeichen außer 'Blank' ( ), Semikolon (;) und Doppelpunkt (:) bestehen.

o PROTECT;

Durch dieses Kommando wird der Paßwort-Schutz aktiviert. Die nächste Bedienung an das VK-Programm muß nun das aktuelle Paßwort sein.

o REQUEST;

siehe NOREQUEST;

o VC iogeräteadresse;

Dieses Kommando entspricht zunächst dem Kommando END. Es wird dem aktuellen VK-Gerät die VK-Berechtigung entzogen und der Paßwort-Schutz aktiviert, sofern im Verständigungsbereich ein "password" eingetragen ist.
Anschließend wird das VK-fähige Gerät, dessen EA-Geräteadresse ("iogeräteadresse") im Kommando angegeben wurde, zum aktuellen VK-Gerät und erhält von VICOM die VK-Berechtigung.

o ?;

Mit diesem Kommando erhält der Anwender eine kurze Beschreibung, die es ihm ermöglicht, das Hilfsprogramm VCINFO urzuladen. Dieses Programm, das Bestandteil des Systemdatenträgers ist, kann nun zur weiteren Bedienerführung eingesetzt werden.

B Dateneln-/-ausgabe

Vorbemerkungen

o Adreßausdruck

In allen Kommandos zur Datenein- und -ausgabe definiert der "adreßausdruck" einen Adreßbereich von Zellen.
Der allgemeine Aufbau ist: **zellentypadreßbereich/adreßergänzung**
Blanks sind im gesamten "adreßausdruck" nicht erlaubt!

Eine Zelle kann sein:

Zellentyp	Bedeutung	Adressvolumen
ohne Angabe	Absolutes Speicherwort	0...H3FFFFF
A	Absolutes Speicherwort	0...H3FFFFF
V	Virtuelles Speicherwort	0...HFFFF
T	Parametertafel (PT)-Zelle	0...63
R	Standardregister R0-15	0...15
G	Standardregister R8-15	0...7
S	Simulationsregister	0...15
namept	Parametertafel-Spezial-register	-
namehw	Hardware-Spezialregister	-
I	Ein-/Ausgabe-Wort (Eingabe)	0...HFFFF
O	Ein-/Ausgabe-Wort (Ausgabe)	0...HFFFF
L	Ebenen-Nummer	0...15
P	Sonderfall	-

P (preferred address, Vorzugsadresse)

Wird als "zellentyp" ein P und weder "adreßbereich" noch "adreßergänzung" angegeben, so gilt stattdessen der zuletzt angegebene "adreßausdruck".

Folgende Kombinationen sind zulässig:

Kommando	-	A	V	T,R G,S	name- pt	name- hw	I	O	L	P
CHANGE	*	*	*	*	*			*		*
DISPLAY	*	*	*	*	*	*	*			*
DUMP	*	*	*							*
SEARCH	*	*	*							*
QUEUE	&	&	&						&	&
STOP	≑	≑	≑				≑	≑		
INFORM	≑	≑	≑				≑	≑		
INTERRUPT	≑	≑	≑				≑	≑		
GO TO	≑	≑	≑							
APZ	+									
DFP	+									
STORE	*	*								

* = unbeschränkt zulässig
≑ = keine Angabe von "Länge" oder "endadr" zulässig
\+ = nur reelle Adressen (0...65535) zulässig, keine Angabe von "Länge" oder "endadr" zulässig
& = Angabe "endadr" nicht zulässig, "Länge" ist die Länge eines Fädellistenelements.

Ein Adreßbereich wird definiert durch Anfangsadresse und Länge, getrennt durch". Beispiel:

anfadr"länge

oder durch Anfangsadresse und Endadresse, getrennt durch -. Beispiel:

anfadr-endadr.

Der Aufbau der Adreßergänzung ist abhängig von der Adressierungsart (Zellentyp):

*** Typ A (absolute Adressierung)

keine "adreßergänzung"!

*** Typ V (virtuelle Adressierung)

adrut

- adrut = absolute Anfangsadresse der Übersetzungstafel; angebbar über Tafelzeigeregister TZR1- oder Prozeß-block-Anfangsadresse (reell) oder Prioritätsebene, jeweils mit Nummer der Übersetzungstafel (1, 2 oder 3).

*** Typ T, R, G, S oder namept (Parametertafeladressierung)

adrpt

- adrpt = absolute Anfangsadresse der Parametertafel (PT); angebbar über TZR1- oder PRB-Anfangsadresse (reell) oder Prioritätsebene.

*** Für die Typen namehw, I, O, L existiert keine Adressergänzung.

o Adreßformate

Adressen können in verschiedenen Darstellungen ausgegeben werden. Die absolute Adresse wird immer hexadezimal ausgegeben; dazu können folgende Angaben treten:
Relative absolute Adresse (bei Offset), relative virtuelle Adresse (Betrag, hexa), Nummer des Worts in der Parametertafel PT (bei PT-Ausgaben, mit oder ohne Prioritätsebenenangabe), Standardregister R u. G, Simulationsregister S, PT-Spezialregister, EA-Adresse.

o Daten können ebenfalls auf verschiedene Arten angegeben werden (Ein- und Ausgabe):
Absolutzahl (A), Festpunktzahl (F), Doppelt-Festpunktzahl (D), Hexadezimalzahl (H), Binärzahl (M), Zeichenfolge (Z).

CHANGE adreßausdruck TO datum 1, datum 2, ..., datum n;

Die angegebenen Daten werden in dem durch den "adreßausdruck" definierten

Adreßbereich ab der niedrigsten Adresse abgelegt. Es sind maximal so viele Daten zulässig, wie der "adreßausdruck" angibt. Die einzelnen Daten werden durch je ein Komma (,) getrennt. Wird ein Datum weggelassen (d.h. am Anfang oder Ende ein einzelnes Komma oder zwischen den Daten zwei Kommas), so wird nur die Adresse weitergeschaltet und die entsprechende Zelle unverändert gelassen.

Beispiel:

Es soll die absolute Speicherzelle H1AF93 mit H5394, die nächste Speicherzelle mit 18 und die Speicherzelle H1AF96 mit HAF05 geladen werden:

CHA H1AF93"4 H5394, 18,,HAF05;

Folgende Varianten sind im speziellen möglich:

CHANGE adreßausdruck TO datentyp datum1, datum2,...,datumn;

Anstatt den Datentyp (z. B. H für 'Hexazahl') bei jedem einzelnen Datum anzugeben, kann der Typ auch einmal zu Beginn der Datenfolge definiert werden.

CHANGE adreßausdruck ALL datum;

Mit diesem Kommando werden alle Zellen des durch den "adreßausdruck" definierten Adreßbereichs mit dem "datum" überschrieben.

CHANGE adressausdruck INC;

Mit diesem Kommando wird durch den "adreßausdruck" der Adreßbereich (und dabei zunächst die Anfangsadresse) festgelegt. Im folgenden wird stets die aktuelle Adresse gemeldet und dann auf die Eingabe eines Datums gewartet (nur "datum" ohne Codewort eingeben!). Dieses "datum" wird in der aktuellen Zelle abgelegt und die Adresse inkrementiert. Wird kein "datum" eingegeben, sondern nur das Zeilenende, so wird kein Datum abgelegt, sondern nur die Adresse weitergeschaltet. In diesem Modus der "inkrementierenden Dateneingabe" ist kein anderes Kommando zulässig. Der Modus wird verlassen (d. h. das Kommando wird beendet) durch Eingabe eines Sterns (*) oder automatisch bei Erreichen des Adreßbereich-Endes.
Das D-Format (Doppelt-Festpunktzahl) ist nicht erlaubt. Im Z-Format (Zeichenfolge) dürfen maximal zwei Zeichen geschrieben werden.

CHANGE adreßausdruck INC datentyp;

Anstatt den Datentyp (z. B. H für 'Hexazahl') bei jedem einzelnen Datum anzugeben, kann der Typ auch einmal im CHANGE-Kommando definiert werden.

DISPLAY adreßausdruck;

Der Inhalt des durch den "adreßausdruck" definierten Adreßbereichs wird am Sichtgerät ausgegeben.

DUMP iogeräteadresse adreßausdruck;

Ergänzend zum DISPLAY-Kommando ist auch eine komprimierte Datenausgabe im Sinne eines Speicherabzugs (nicht im Urladeformat!) an ein serielles, VK-akzeptierendes Ausgabegerät (z. B. Drucker) möglich. Diese Ausgabe mit dem DUMP-Kommando erfolgt an das mit dem LIST-Kommando eingestellte Gerät, wenn keine "iogeräteadresse" angegeben wird, oder an das durch die "iogeräteadresse" definierte Gerät. Das DUMP-Kommando ist nur auf Speicherzellen (Zellentyp A oder V) anwendbar.

DUMP iogeräteadresse adressausdruck Z;

Für dieses Kommando gilt im wesentlichen das gleiche wie für das normale DUMP-Kommando. Zusätzlich wird hierbei der Zelleninhalt jedes Blocks als fortlaufende Zeichenfolge (ohne vorgesetzten Formattyp Z) ausgegeben. Dabei werden nicht abdruckbare Zeichen durch Punkt (.) ersetzt. Die Blockanfangsadresse steht vor den Daten in derselben Zeile, so daß diese Version übersichtlich und zeilensparend ist. Das DUMP-Kommando (mit Z) funktioniert für Drucker mit einer Druckbreite von mindestens 126 Spalten.

Beispiel:
Es soll der Inhalt der Speicherzellen H12A bis H9B06 eines definierten, virtuellen Adreßraums auf Drucker ausgegeben werden. Der Adreßraum ist durch die Übersetzungstafel bestimmt, die an der reellen Adresse H8000 beginnt. Die Protokollierung ist durch das LIST-Kommando bereits eingeschaltet:

DU VH12A-H9B06/H8000 Z;

FORMAT format1, format2,...,formatn;

Mit diesem Kommando werden die Datenformate eingestellt, in denen die Datenausgaben erfolgen.

Beispiel:
Die Datenausgabe soll in den Formaten F, H und Z erfolgen:

FORM F, H, Z;

OFFSET offset;

Vor jedem Zugriff zu einer Speicherzelle mit der Adressierungsart A oder V wird zu der angegebenen (ggf. virtuellen) Adresse der "Offset" addiert und mit dieser Adresse tatsächlich gearbeitet.

Beispiel:
Die relative Adresse 0 (laut Protokoll) liegt bei absolut 4096 (= H1000) im virtuellen Adressraum:

OFF H1000

PRESET adreßergänzung;

Um bei häufiger Adressierung mit "adreßergänzung" (z. B. virtuelle Adressierung) nicht bei jedem Kommando die "adreßergänzung" angeben zu müssen, kann sie durch das PRESET-Kommando voreingestellt werden.

Beispiel:
Es soll über die Übersetzungstafel 2 des aktuellen Programms der Ebene 14 virtuell adressiert werden. Ebenso soll die Parametertafel dieses Programms bei Registerangabe adressiert werden:

PRES L14,U2;

QUEUE adreßausdruck parameter;

Das QUEUE-Kommando bietet die Möglichkeit, sich in Fädellisten eingehängte Elemente ausgeben zu lassen. Im Gegensatz zum DISPLAY-Kommando ist hierzu nur die Anfangsadresse des jeweiligen Kopfelements notwendig.
Ausgegeben wird das erste in der Liste eingehängte Element.
Dabei erfolgt als Elementkennzeichnung die Meldung: ELEMENT 1,
Folgende den Listenaufbau beschreibende Parameter sind zulässig:

- HEADLESS: die Fädelliste hat keinen Kopf (d.h. das Kopfelement der Liste wird gekennzeichnet als Element 0 ausgegeben).
- SINGLE : die Fädelliste ist einfach gefädelt (d.h. die Elemente enthalten keine Rückwärtsverweise)
- PHYSPR : die Fädelliste ist eine PHYSPR-Liste
- TRQ : die Fädelliste ist eine Transferwarteschlange (TRWS)
- NOCHECK : es wird bei doppeltgefädelten Listen keine Kontrolle des Listenaufbaus vorgenommen (bei einfach gefädelten Listen ist der Parameter irrelevant).

Es können auch sinnvolle Kombinationen dieser Parameter eingegeben werden.

Beispiel:
Die Fädelliste, die ab der virtuellen Adresse HA1D steht und deren Übersetzungstafel sich bei der Adresse 1000 befindet, soll bearbeitet werden:

QUE VHA1D/1000"20 HL

Q+ anzahl; ---- Elementblättern in Richtung aufsteigender Elementnummern

Q- anzahl; ---- Elementblättern in Richtung fallender Elementnummern.

Diese Kommandos ermöglichen das 'Durchhangeln' durch die jeweilige Fädelliste. Es wird jeweils um die, über "anzahl" angegebene Anzahl, Listenelemente weitergeschaltet.

SEARCH adressausdruck datum1, datum2,.....datum n;

In dem durch den "adreßausdruck" definierten Adreßbereich werden die angegebenen Daten (maximal 8 Wörter) in dem durch die Reihenfolge gegebenen Zusammenhang gesucht.

Beispiel:
Im absoluten Zentralspeicher-Adreßbereich von 1000 bis 3000 sollen die

Daten 1,2,3,4,irrelevant,5,6 gesucht werden:

SEA 1000-3000 1,2,3,4,,5,6;

Eine Fortsetzung des Suchlaufs, nachdem die Daten gefunden wurden, ist ab der aktuellen Adresse möglich, indem das Kommandowort SEARCH allein (d.h. ohne "adreßausdruck") eingegeben wird.

C Programminitiierung

Diese Kommandos dienen zum Laden und Steuern von Programmen.
Das dabei angesprochene periphere Gerät wird mittels der Codierschalter der Zentralprozessor-Flachbaugruppe "Speichersteuerung" ausgewählt oder in Form einer "ioadresse".

... IO-ioadresse ...
oder
... OI-ioadresse ...

Die "ioadresse" kann hexadezimal (z. B. H8008 oder H=8008) oder absolut ohne Formatkennzeichen A (z. B. 32776) angegeben werden. Anstelle des Bindestrichs (-) kann auch ein Gleichheitszeichen (=) geschrieben werden. Der Name eines urladefähigen Programms besteht aus mindestens 1 und maximal 6 Zeichen. Werden im Kommando weniger als 6 Zeichen angegeben, füllt VICOM den Namen automatisch mit Blanks auf. Die Zeichen eines Namens (auch das 1. Zeichen) dürfen alle abdruckbaren Zeichen der Spalten 2 bis 7 der ISO-7-bit-Matrix (DIN 66003) sein (also auch Kleinbuchstaben und Sonderzeichen) außer:

- Doppelpunkt (:);
- Komma (,),
- Semikolon (;),
- Blank ( ) .

Nach jedem vollständigen, fehlerfreien Urladen eines Programms werden alle zum Start erforderlichen Daten des Programms (Name, Startadressen) in einer Startliste eingetragen. Diese Liste kann maximal fünf Programme aufnehmen. Der Eintrag erfolgt, indem die bereits eingetragenen Programme um eine Position nach hinten geschoben werden, d.h. sie werden gealtert. Dabei fällt das zuvor in Position 5 stehende Programm heraus, somit ist es in Zukunft nicht mehr startbar. Das neue Programm wird in erster Position eingetragen.

Wurde vor dem Ladevorgang die Startliste gelöscht, so ist nach dem Laden nur das soeben geladene Programm eingetragen und somit startbar. Wurde sie aber nicht gelöscht, so können max. fünf Programme gleichzeitig im Speicher stehen (natürlich nicht überlappend!), die beliebig gestartet werden können.
Mit allen Lade- und Startkommandos und mit dem Kommando TEST ist es möglich, Parameter an das gestartete Programm zu übergeben. Diese Parameter werden durch VICOM an fester Stelle im Hardware-Verständigungsbereich (HW-VB) ab der symbolischen Adresse PARPUF abgelegt. Die Parameter werden durch VICOM nicht interpretiert. Der Parameterpuffer ist 80 byte lang.

Kommandos:

BOOT iogeräteadresse;

BOOT iogeräteadresse:parameter

Die Startliste wird zunächst gelöscht, so daß nach dem Laden nur das geladene Programm eingetragen und startbar ist. Die Geräte werden sammelrückgesetzt. Der Ablauf des Ladevorgangs ist abhängig von der Art des Kommandos und des Datenträgers:

Serieller Datenträger
Das ab dem Anfang des seriellen Datenträgers gespeicherte Programm wird urgeladen.

Random-Datenträger
Das in den Zellen "Name des bevorzugten Programms" in der Buchführung der peripheren Geräte durch seinen Namen beschriebene Programm wird urgeladen. Es handelt sich dabei i. allg. um ein Betriebssystem. Ist keines eingetragen, erfolgt eine Urlader-Fehlermeldung.

BOOT iogeräteadresse NAME-name;

BOOT iogeräteadresse NAME-name:parameter

Serieller Datenträger
Das ab dem Anfang des seriellen Datenträgers gespeicherte Programm wird urgeladen, sofern sein Name mit dem angegebenen übereinstimmt, sonst erfolgt eine Urlader-Fehlermeldung.

Random-Datenträger
Der angegebene "name" wird in der Buchführung des peripheren Geräts gesucht. Ist er gefunden, wird das zugehörige Programm urgeladen. Wird der "name" nicht gefunden, erfolgt eine Urlader-Fehlermeldung.

BOOT iogeräteadresse SAVE NAME-name;

BOOT iogeräteadresse SAVE NAME-name:parameter

Im Gegensatz zum Kommando BOOT ohne SAVE wird hier die Startliste nicht gelöscht und anstelle eines Sammelrücksetzens nur das Urladegerät selektiv zurückgesetzt - ansonsten ist der Ablauf identisch mit den vorher beschriebenen Abläufen.

BOOT iogeräteadresse SAVE PAGE-seite NAME-name;

BOOT iogeräteadresse SAVE PAGE-seite NAME-name:parameter

Bestimmte Programme lassen sich mit diesem Kommando beliebig im Speicher verschieben (Abschnitt D), d.h. unabhängig von der programminternen Voreinstellung an eine frei wählbare Stelle im Zentralspeicher laden. Diese frei wählbare Stelle muß jedoch der Beginn einer "seite" (d. h. eines Adreßraums von 1024 Adressen) sein. Die reelle (absolute) "seite", ab der geladen werden soll, wird mit dem Codewort PAGE angegeben. Wird das Codewort PAGE nicht gegeben, wird an den voreingestellten Platz geladen.
"seite" kann hexadezimal (z.B. H50) oder absolut ohne Formatkennzeichen A (z. B. 80) geschrieben werden. "seite" darf nicht kleiner sein als 2.

RESTART;

R ESTART:parameter

Mit diesem Kommando wird das aktuelle Programm (Position 1 in der Startliste) wie nach einer Spannungswiederkehr neu gestartet.

START name;

START name:parameter

Mit diesem Kommando wird ein Programm gestartet als sei es urgeladen worden.

D Debugging

Die virtuelle Konsole stellt neben den bisher beschriebenen Funktionen einige spezielle Möglichkeiten zum Austesten urladefähiger Programme zur Verfügung. Der wesentliche Teil dieser Funktionen ist allerdings nur verfügbar, wenn die Testanschaltung TESTAS oder die Serviceprozessor-Anschaltung SPAS am Service-Steckplatz SS gesteckt ist. Bestimmte Funktionen setzen definitiv entweder die TESTAS oder die SPAS voraus. Sind diese Hardware-Voraussetzungen nicht vorhanden, erfolgt bei dem entsprechenden Kommando die Fehlermeldung
VICOM: NO HARDWARE !
Zum Testen dieser Programme werden Testpunkte ("checkpoints") eingerichtet.
Es gibt zu einem Zeitpunkt durch VICOM kontrolliert maximal

- vier statische checkpoints (Nr. 0..3),
- einen temporären checkpoint.

Ein temporärer checkpoint kann die Zustände annehmen

- eingerichtet und deaktiviert,
- eingerichtet und aktiviert,
- eingerichtet, aktiviert und aktuell,
- gelöscht (d.h. nicht vorhanden).

Ein temporärer checkpoint ist immer auch aktuell!

Kommandos:

ACTIVATE checkpoint;

Der (statische) "checkpoint" wird aktiviert. Das Weglassen von "checkpoint" ist gleichbedeutend mit dem Kommando ACTIVATE 0;.

APZ adresse;

Der Prozeßblock, der an der "adresse" beginnt, wird aktiviert. Ein nicht fortsetzbarer Stop wird in einen fortsetzbaren Stop umgewandelt.

DEACTIVATE checkpoint;

Der (statische) "checkpoint" wird deaktiviert. Das Weglassen von "checkpoint" ist gleichbedeutend mit dem Kommando DEACTIVATE 0;.

DEACTIVATE ALL;

Alle "checkpoints" (temporär oder statisch) werden deaktiviert.

DEACTIVATE ACTUAL;

Der aktuelle checkpoint (temporär oder statisch) wird deaktiviert.

DELETE checkpoint;

Der (statische) "checkpoint" wird gelöscht (und damit natürlich auch deaktiviert). Das Weglassen von "checkpoint" ist gleichbedeutend mit dem Kommando DELETE 0;.

DELETE ALL;

Alle checkpoints (temporär oder statisch) werden gelöscht.

DELETE ACTUAL;

Der aktuelle checkpoint (temporär oder statisch) wird gelöscht.

DFP adresse;

Der Prozeßblock, der an der "adresse" beginnt, wird deaktiviert.

GO;
=
Kurzbeschreibung: Gehe!
Dieses Kommando setzt weder TESTAS noch SPAS voraus!
Das Programm (Testobjekt) wird fortgesetzt, wobei an den gegenwärtigen Checkpoint-Zuständen keine Änderung vorgenommen wird. Dabei wird ein temporärer checkpoint bereits bei seinem Erreichen automatisch gelöscht.

GO CHECKPOINT-checkpoint;
=
Kurzbeschreibung: Gehe zum "checkpoint"!
Ist der "checkpoint" eingerichtet, werden zunächst alle checkpoints deaktiviert und dann der "checkpoint" aktiviert. Danach wird das Testobjekt fortgesetzt.

GO TO adresse;
=
Kurzbeschreibung: Gehe bis "adresse"!
Es wird ein temporärer checkpoint eingerichtet. Die Einrichtung kann am besten durch das Kommandosynonym für statische checkpoints beschrieben werden:

```
STOP IF adresse PC;
GO;
```

Das Testobjekt wird fortgesetzt.

INFORM s. STOP

INTERUPT s. STOP

SHOW;

Mit diesem Kommando werden die checkpoints in tabellarischer Form dargestellt.
In der Darstellung erscheinen immer absolute Adressen (auch wenn sie virtuell oder als EA-Adressen angegeben wurden). Der Unterschied zwischen Speicher- und EA-Adressen geht aus der Zeile HW-COND. (Hardware-Bedingung) hervor.

Die Tabelle erscheint in folgender Form am Bildschirm:

CHECKPOINT	ACTUAL	0		3
STATUS				
REAKTION				
ADRESS				
RANGE				
HW-COND.				
NUMBER				
LEVEL				
MASK				
VALUE				
COUNTER				
ACTIVATE				
DEACTIVATE				
DELETE				

STEP anzahl;
=
Kurzbeschreibung: Gehe für "anzahl" Befehle!
Es wird ein temporärer checkpoint eingerichtet. Wird "anzahl" weggelasen, gilt "anzahl" = 1. Es gilt 0 < anzahl < 256.

STOP;

Mit diesem Kommando wird entweder der Zustand "fortsetzbarer Stop" angefordert oder es bezieht sich auf einen checkpoint. In letzterem Fall hat es folgende Form (gemeinsam mit INFORM und INTERRUPT):

STOP ——— = Meldung des checkpoints mit Stop

INFORM ——— = Meldung des checkpoints ohne Stop

INTERRUPT —— = Bit 12 (SIBI) im Unterbrechungsanzeigewort UAW der aktuellen Parametertafel setzen; im Prozeßblock für Programmlaufbesonderheiten (PLB - PRB) die Adresse des verursachenden Prozeßblocks aktivieren.

4

```
_   STOP      _
<<  INFORM    >>  IF adresse rangebed hwbed steuer define;
-   INTERRUPT -
```

Dieses Kommando setzt die TESTAS oder die SPAS voraus. Der durch define" definierte checkpoint soll erreicht werden, wenn die "adresse" entsprechend der "hwbed" (Hardwarebedingungen) angesprochen wird (NMI "ADV" wird ausgelöst) und die "swbed" (Software-Bedingungen) erfüllt sind. Dann werden die "steuer"-Tätigkeiten durchgeführt.

```
_   STOP      _
<<  INFORM    >>  TRACE swbed steuer define;
-   INTERRUPT -
```

Dieses Kommando setzt die TESTAS voraus ist bei der SPAS nicht erlaubt. Der durch "define" definierte checkpoint soll nach jedem Befehlsende (NMI "ADV" wird ausgelöst) erreicht werden, wenn die "swbed" (Software-Bedingungen) erfüllt sind ("Befehl einzeln"). Dann werden die "steuer"-Tätigkeiten ausgeführt.

*** adresse

Die "adresse" wird angegeben als
o absolute Adresse (hexadezimal oder als Betrag)
o virtuelle Adresse (hexadezimal oder als Betrag)
o EA - Adresse (hexadezimal oder als Betrag)

*** rangebed

"rangebed" definiert den Adreßbereich.

RANGE-bereich

Der durch "bereich" bezeichnete Adreßbereich, innerhalb dessen die "adresse" liegt, wird überwacht.

"bereich": 1 = Einzeladresse
64 = Adreßbereich von 64 Adressen
4096 = Adreßbereich von 4096 Adressen
4K = Adreßbereich von 4096 Adressen

Der Parameter RANGE ist nur bei der SPAS erlaubt. Die TESTAS beherrscht nur Einzeladressen. Wird er weggelassen, so ist das gleichbedeutend mit der Angabe RANGE-1.

*** hwbed

"hwbed" sind Hardware-Bedingungen, mit denen die Adreßvergleich-Hardware (TESTAS bzw. SPAS) parametriert wird. Die einzelnen "hwbed" sind ODER - verknüpft. Sie werden durch Blanks getrennt.

o Bedingungen bezüglich des befehlsgesteuerten Verkehrs zwischen Zentralprozessor und Speicher:

PROGRAMCOUNTER

Der nicht maskierbare Interrupt (NMI) "ADV" wird ausgelöst, wenn der Befehlszähler gleich "adresse" ist und dieser Befehl ausgeführt ist.

READ

Der NMI "ADV" wird ausgelöst, wenn die "adresse" über Operandenzugriff des Zentralprozessors gelesen wird.

WRITTEN

Der NMI "ADV" wird ausgelöst, wenn die "adresse" über Operandenzugriff des Zentralprozessors beschrieben wird.

o Bedingungen bezüglich des organisatorischen Verkehrs zwischen Zentralprozessor und Speicher:

HWREAD

Der NMI "ADV" wird ausgelöst, wenn die "adresse" weder beim Befehls lesen noch beim Operandenlesen (also bei einem 'organisatorischen' Zugriff, z. B. Parametertafel (lesen) durch den Zentralprozessor gelesen wird.

HWWRITTEN

Der NMI "ADV" wird ausgelöst, wenn die "adresse" nicht beim Operanden schreiben (also bei einem organisatorischen Zugriff, z. B. Parameter tafel (schreiben) durch den Zentralprozessor beschrieben wird.

o Bedingungen bezüglich des Verkehrs zwischen Peripherie und Speicher

DMAREAD

Der NMI "ADV" wird ausgelöst, wenn eine periphere Einheit per DMA-Verkehr die "adresse" ausliest.

DMAWRITTEN

Der NMI "ADV" wird ausgelöst, wenn eine peripheren Einheit per DMA-Verkehr die "adresse" beschreibt.

o Bedingungen bezüglich des Verkehrs zwischen dem Zentralprozessor und dem EA-Adreßraum

IN

Der NMI "ADV" wird ausgelöst, wenn mit dem Befehl Ein-/Ausgabe Lesen (EAL) die "adresse" als EA-Adresse angesprochen wird.

OUT

Der NMI "ADV" wird ausgelöst, wenn mit dem Befehl Ein-/Ausgabe Schreiben (EAS) die "adresse" als EA-Adresse angesprochen wird.

*** swbed

"swbed" sind Software-Bedingungen, die geprüft werden, sobald die Hardware-Bedingungen erfüllt sind. Die einzelnen "swbed" sind UND-verknüpft. Sie werden durch Blanks getrennt.

NUMBER-nummer

Der checkpoint gilt bezüglich dieser Bedingung als erreicht, wenn die Hardware-Bedingungen zum sovielten Mal erfüllt sind wie "number" angibt.

LEVEL-ebene

Der checkpoint gilt bezüglich dieser Bedingung als erreicht, wenn beim Erfüllen der Hardware-Bedingungen das Zentralprozessor-Zustandsregister ZZR gleich "ebene" ist (checkpoint ist ebenenspezifisch).

MASK-maske

VALUE-wert

Der checkpoint gilt bezüglich dieser Bedingung als erreicht, wenn beim erfüllen der Hardware-Bedingungen (also nach dem Lese- bzw. Schreibvorgang) der Inhalt von "adresse" UND-verknüpft mit "maske" gleich "wert" ist.

COUNTER-zähler

Der checkpoint ist erreicht, wenn die Hardware-Bedingungen und gleichzeitig die Software-Bedingungen (s. o.) zum sovielten Mal erfüllt sind, wie der "zähler" angibt.

*** steuer

Die "steuer"-Parameter beschreiben die automatischen Tätigkeiten beim Erreichen des checkpoints.

ACTIVATE-checkpoint

Der "checkpoint" wird aktiviert.

DEACTIVATE-checkpoint

Der "checkpoint" wird deaktiviert.

DELETE-checkpoint

Der "checkpoint" wird gelöscht.

REGISTER-register

Es werden beim Erreichen des checkpoints die "register" wie bei einem DISPLAY-Kommando dargestellt. Es handelt sich dabei um die Standardregister der aktuellen Parametertafel PT der aktuellen Ebene. Als "register"-Angabe ist zulässig:
regnr"anzahl
regnr-regnr
regnr,regnr
regnr

*** define

Der "define"-Parameter erklärt die Nummer des einzurichtenden checkpoints. Eine Mehrfachangabe ist nicht zulässig.

CHECKPOINT-checkpoint

Der Checkpoint erhält die Nummer "checkpoint".

E Wartungs- und Hilfsfunktionen

CALCULATE operand1 operator operand2;

CALCULATE RESULT operator operand2;

Der "operand1" wird gemäß dem "operator" mit dem "operand2" verknüpft und das Ergebnis ausgegeben und gespeichert. Als Operanden sind u. a. 32 bitlange Hexa- oder Dezimalzahlen zulässig (beachte Ausnahmen). Als "operand1" darf auch das Codewort RESULT angegeben werden, dann ist "operand1" das Ergebnis der letzten Operation.
Als "operator" sind zulässig:
+ --> Addition
- --> Subtraktion
(--> Schiebe "operand1" um "operand2"-viele Bitstellen nach links
) --> Schiebe "operand1" um "operand2"-viele Bitstellen nach rechts

Bei (und) darf "operand2" maximal 32 sein und muß als Betragszahl angegeben werden.

CHECK iogeräteadresse TIMER;

Mit diesem Kommando wird der "Knopfdrucktest" einer Geräteanschaltung angestoßen, nachdem sie zuvor selektiv zurückgesetzt wurde. Der Knopfdrucktest hinterlegt im Fehlerfall seine ermittelten Fehlerdaten ähnlich wie bei einem Standardfehler. Diese Daten werden durch VICOM als normale IO-ERROR-Meldung am Sichtgerät dargestellt. Im fehlerfreien Fall wird der Auftrag quittiert mit VICOM! Falls das zu testende Gerät der Zeitimpulsgeber ZIG sein soll, muß der Parameter TIMER (oder stattdessen ZIG) angegeben werden.

GENERATE anzahl;

Mit diesem Kommando wird der "error-correcting-code" (Fehlerkorrektur-Code) des Zentralspeichers generiert. Der Vorgang ist zerstörungsfrei, d. h. der Speicherinhalt bleibt erhalten.
Es werden ab der Adresse 0 "anzahl" 128-K* byte-Bereiche generiert. Wird "anzahl" im Kommando weggelassen, wird die Anzahl der 128 K* Byte-Bereiche der Zelle MAKENN entnommen.

RESET iogeräteadresse;

Das durch die "iogeräteadresse" definierte Gerät (die Anschaltung) wird selektiv beendet und nach 0,5 s selektiv zurückgesetzt.

RESET;

Es wird ein Sammelbeenden und nach 0,5 s ein Sammelrücksetzen ausgelöst.

STORE iogeräteadresse adressausdruck name;

Es wird zunächst der durch die "iogeräteadresse" definierte Datenträger untersucht und das Ergebnis ausgegeben mit der Meldung

VICOM: name STORE TO IO-H=ioadresse (kommentar) ?

"kommentar":

BOOT-DEVICE	= Es handelt sich bei "iogeräteadresse" um das Urladegerät.
DIRECTORY name	= Es handelt sich nicht um das Urladegerät; der Datenträger enthält die Buchführung "name".
PROGRAM name	= Es handelt sich nicht um das Urladegerät; der Datenträger enthält das Programm "name".
entfällt	= Es handelt sich nicht um das Urladegerät; der Datenträger enthält kein sinnvolles Urladeformat.

Mit dem Kommando STORE wird der durch den "adreßausdruck" (nur Zelltyp A zulässig, d. h. absolute Adressen) definierte Speicherbereich als urladefähiges Programm mit dem Programmnamen "name" abgezogen. Ist "name" in

der Startliste eingetragen, so wird das Urladeformat mit den entsprechenden Startdaten versehen. Das Programm ist in jedem Fall als nicht startbar gekennzeichnet. Der Speicherabzug mit dem STORE-Kommando ist nur auf Random-Datenträger (im physikalischen Sinn) möglich. Der Abzug erfolgt im Sinne des Urladeformats als serielles Programm. Eine Bearbeitung der ggf. vorgefundenen Buchführung findet nicht statt. Sie wird überschrieben!

TERMINATE iogeräteadresse;

Das durch die "iogeräteadresse" definierte Gerät (die Anschaltung) wird selektiv beendet.

TEST iogeräteadresse CYCLIC;

TEST iogeräteadresse CYCLIC:parameter;

Mit diesem Kommando werden zunächst die im Festspeicher integrierten Tests initiiert.
- Zentralspeicher-Test (inklusive ECC-Generierung, d.h. Generierung des Fehlerkorrektur-Codes),
- VK-Test (Test der virtuellen Konsole),
- BT-Test (Test des Urladegeräts).

Der VK-Test wird mit dem Gerät durchgeführt, über das bedient wurde.

Danach werden bestimmte Zellen im Hardware-Verständigungsbereich vorbereitet und dann der automatische Ablauf aller definierten urladefähigen Testprogramme angestoßen. Diese Testprogramme sind in der Buchführung des Random-Datenträgers gekennzeichnet. Sie kommen in der Reihenfolge zum Ablauf, in der sie in der Buchführung stehen. Vor dem Laden jedes einzelnen Programms wird die Startliste gelöscht, so daß stets das zuletzt geladene Programm nach einem festgestellten Fehler erneut gestartet werden kann, wobei der Testmodus ausgeschaltet wird, so daß das Programm dann im "Handbetrieb" läuft.

Die Bearbeitung der Testprogramme ist sequentiell, d. h. zu einem bestimmten Zeitpunkt ist nur ein Programm geladen und aktiv. Das Programm erkennt den Fall "automatischer Test" anhand des Bit "Testmodus" in der Zelle ANLAWE. Es steht dem letzten Programm der Testprogramme frei, sich nach Abschluß der Tests selbst aus dem Testmodus zu entlassen (löschen des Bits "Testmodus") und sich im Normalmodus fortzusetzen. Diese Möglichkeit kann für ein Testbetriebssystem interessant sein. Der Parameter CYCLIC bleibt in diesem Fall wirkungslos.

Ist ein solches Programm, das von sich aus den Testmodus beendet, nicht vorhanden, so ist der Testdurchlauf beendet, sobald der Urlader keine weiteren Testprogramme mehr in der Buchführung findet. Der weitere Ablauf ist dann abhängig von der Existenz des Parameters CYCLIC:

o ohne CYCLIC:
  Das Programm, das in den Zellen "Name des bevorzugten Programms" in

der Buchführung verzeichnet ist, wird urgeladen. Das Bit "Testmodus" bleibt dabei gesetzt.
Ist kein solches Programm vorhanden (die Zellen sind 0), erfolgt die Meldung VICOM: END OF TEST!
VICOM geht in den Zustand "Nicht fortsetzbarer Stop".

o mit CYCLIC:
Es wird ein erneuter Testdurchlauf initiiert, d. h. es wird wieder das erste in der Buchführung als Testprogramm definierte Programm urgeladen usw. Der gesamte Testlauf ist also zyklisch unbegrenzt. Unterbrechbar ist er z. B. durch das Kommando STOP, während ein Testprogramm läuft.
Vor jedem Testdurchlauf wird die Zelle TEIUZ inkrementiert (Zelle 1533 des Hardware-Verständigungsbereichs im Zentralspeicher). Der Basistest wird nicht wiederholt, d. h. er wird nur einmalig vor dem ersten Testdurchlauf ausgeführt.

4

VERIFY;

Es werden die im Festspeicher integrierten Tests initiiert
- Test (inklusive ECC-Generierung, d.h. Generierung des Fehlerkorrektur-Codes)
- VK-Test (Test der virtuellen Konsole),
- BT-Test (Test des Urladegeräts).

Dabei ist zu bedenken, daß sich nach dem Test der Zentralspeicher und alle Geräte im zurückgesetzten Zustand befinden und die Anlage im Zustand "nicht fortsetzbarer Stop" ist.

VIEW iogeräteadresse TIMER;

Mit diesem Kommando werden die Gerätekenndaten (statische Komponente der Inline-Statistik) der durch "iogeräteadresse" definierten Anschaltung in aufbereiteter Form ausgegeben. Handelt es sich bei der Anschaltung um den Zeitimpulsgeber ZIG, so muß der Parameter TIMER (oder ZIG) angegeben werden.

4.3.5 Meldungen

VICOM spricht den Bediener in Form verschiedener Meldungen an:
- Meldungen mit ! - Abschluß (danach ist VICOM bedienbereit).
- Meldungen mit . - Abschluß (danach ist VICOM nicht unmittelbar bedienbereit).
- Meldungen mit ? - Abschluß (danach ist VICOM nicht unmittelbar, aber im weiteren Verlauf, bedienbar).
- Meldungen ohne Abschluß (danach ist VICOM nicht bedienbereit, die Kommandobearbeitung wird fortgeführt.)
- Error-Meldungen: Fehler im EA-Verkehr oder beim Urladen (Tabelle siehe Anhang, s. Kapitel 6.5)
- Anfangsmeldung;

```
SIEMENS SICOMP
==============
VICOM: V x.yy
VICOM: ZE ttcc       (Anlagenausstattung)

Diese Anlagenausstattung erfolgt immer als erste Meldung über das VK-Ge-
rät nach dem Basistest.
Es bedeuten:
x  = Version
yy = Ausgabestand
tt = ZE-Typ          (01 = ZE01, 02 = ZE02, 03 = ZE03)
cc = Cache-Speicher  (-C = mit Cache-Speicher)
     (wenn kein Cache-Speicher vorhanden: "cc" = 2 Blanks)

Anlagenausstattung:
FPP = floating point processor  (Gleitpunktprozessor)
TIF = testinterface (TESTAS)          an Servicesteckplatz SS
SIF = serviceinterface (SPAS)         an Servicesteckplatz SS
aaa K = Speicherausbau beträgt 2 x "aaa" K* byte (= "aaa" K*Worte)

Beispiel:

SIEMENS SICOMP
==============
VICOM:  V 1.01
VICOM:  ZE  03-C     (FPP, 512 K, TIF).
```

4.4 Urladen

Der Urlader in VICOM hat die Aufgabe, ein Programm zu laden bzw. einen automatischen Testlauf mehrerer Programme zu organisieren. Urladefähige Programme zeichnen sich durch ein besonderes Format aus, das mit dem Dienstprogramm MURL erzeugt wird.

Der Urlader behandelt grundsätzlich alle Geräte gleich, d.h. er unterscheidet nicht den Typ des Geräts. So werden auch alle Geräte mit der Externadresse versorgt, wenngleich sie diese Adresse vielleicht gar nicht verstehen und interpretieren können. Geräte, die die Externadresse nicht berücksichtigen, sind "serielle Geräte". Der Urlader unterscheidet jedoch serielle Geräte von Random-Geräten einzig nach der Tatsache, ob eine Urlade-Buchführung vorhanden ist oder nicht. Geräte mit Buchführung auf ihrem Datenträger sind für ihn Random-Geräte, während Geräte, die keine Buchführung liefern, immer serielle Geräte sind, obwohl sie durchaus die Externadresse behandeln können. So ist z. B. eine Datenübertragungssteuerung (DÜ) ein serielles Gerät und eine Floppy Disk ein serielles oder Random-Gerät, je nach dem, ob eine Buchführung vorhanden ist oder nicht.

Der Urlader wird über die Zellen ANLAWE, BTRESE, URGERA, PRONA des Hardware-Verständigungsbereichs gesteuert. Der Urlader greift verändernd in die Zellen ANLAWE und PRONA ein.

Fehler beim Urladen werden mit Standard-Fehlermeldungen unter Angabe einer Fehlernummer über die virtuelle Konsole gemeldet. Sie werden erkannt

- beim EA-Transfer
  (gemeldet mit IO-ERROR...),
- in der Struktur des Urladeformats
  (gemeldet mit BT-ERROR...).

Beim IO-ERROR wird die Lampe 'bootstrap check' ausgeschaltet, während sie beim BT-ERROR eingeschaltet bleibt. Vor dem Laden der Urlade-Buchführung wird die Lampe 'bootstrap check' ausgeschaltet und bei fehlerfreiem Transfer danach wieder eingeschaltet.

4.4.1 Urladeformat

Das Urladeformat der Modellreihe SICOMP ist ein spezielles Datenformat, das aus der Grundsprache gewonnen wird.
Es zeichnet sich aus durch
- Nutzdaten in binärem Maschinencode,
- Ladeinformationen binär in einem Vorspann,
- Blockstruktur größerer Programme (daher kann die Gesamtlänge eines Programms beliebig sein),
- keine Interpretation (Umkodieren) der Nutzdaten erforderlich (daher schneller Ladevorgang großer Datenmengen; auf ein "Urleseprogramm" kann verzichtet werden),
- wahlfreier Zugriff zu mehreren Programmen über ihre Namen (ermöglicht durch eine Urlade-Buchführung),
- Ablage der Nutzdaten im Zentralspeicher an vorbestimmter Stelle,
- unter bestimmten Voraussetzungen auch Laden an beliebiger Stelle im Speicher (Seite) möglich,
- seitenweise Verteilung der Nutzdaten über den Speicher möglich,
- Datensicherung von Buchführung, Vorspann und Nutzdaten nach zwei Verfahren

Aufbau der Datenträger

Ein urladefähiges Programm besteht aus einem oder mehreren Blöcken, die bei 0 beginnend gezählt werden. Ein Block teilt sich auf in Vorspann und Nutzdaten. Der Vorspann hat eine feste Länge von 512 byte und sollte bei Magnetschichtdatenträgern an einer Sektorgrenze beginnen. Dem Vorspann folgen unmittelbar die Nutzdaten, die max. 128 K* byte umfassen können. Die Nutzdaten (Code des zu ladenden Programms) sind auf dem Datenträger binär abgelegt (Maschinencode). Der serielle Datenträger enthält genau ein urzuladendes Programm. Der erste Block (Block 0) dieses Programms steht am Anfang des Datenträgers (Externadresse = 0). Die ggf. weiteren Blöcke folgen unmittelbar aufeinander in aufsteigender Nummernfolge. Ein Random-Datenträger kann bis zu 38 urladefähige Programme enthalten. Die Verteilung der Programme und ihrer Blöcke über den Datenträger ist beliebig. Die Namen und die Externadressen (EXAD) des Blocks 0 der Programme sind in einer Buchführung verzeichnet. Diese Buchführung steht ab der EXAD 0 auf dem Datenträger und ist ebenso wie ein Blockvorspann 512 byte lang. Der formale Aufbau ist ähnlich dem eines Vorspanns; es entfallen bei der Buchführung aber die Nutzdaten.

Das Urladeformat ist so strukturiert, daß es möglich ist:
o von einem seriellen Datenträger ein urladefähiges Programm zu laden,

- o von einem Random-Datenträger ein beliebiges urladefähiges Programm zu laden, das über seinen Namen identifiziert wird ("PRWAHL-Funktion"),

- o von einem Random-Datenträger ein bestimmtes urladefähiges Programm ohne Angabe eines Namens zu laden ("Preferred-program-Lade-Funktion"),
- o von einem Random-Datenträger mehrere, als Testprogramme gekennzeichnete, Programme zeitlich nacheinander automatisch zu laden und zum Ablauf zu bringen.

Die erfoderlichen Informationen sind - außer bei einem seriellen Datenträger - in einer Buchführung gespeichert, die sich an fester Stelle (EXAD 0) des Random-Datenträgers befindet. Bei einem seriellen Datenträger entfällt die Buchführung; das urzuladende Programm beginnt bei der quasi EXAD 0 (d. h. am Anfang des Datenträgers). Da Buchführung und Programmblockvorspann ähnlich aufgebaut sind (512 byte, Steuerwort im Wort 0), kann der Urlader ohne zusätzliche Externspeicherzugriffe anhand eines Bits in einem Steuerwort (siehe Bild 4.3) zwischen einem seriellen und einem Random-Datenträger unterscheiden.

Buchführung

Die Buchführung belegt 512 byte ab der EXAD 0 auf dem Datenträger. In ihr sind folgende Informationen gespeichert (siehe Bild 4.1):

- o Steuerwort: Unterscheidet die Buchführung vom einem Programmblockvorspann (siehe Bild 4.3)
- o Anzahl Programme: Enthält die Anzahl der auf dem Datenträger vorhandenen urladefähigen Programme
- o Name/Ausgabestand: Identifikation des Datenträgers in Bezug auf das Urladesystem (für Urlader irrelevant)
- o Programmidentifikation (PI): Die PI umfaßt für jedes urladfähige Programm jeweils 6 Wörter. Die PI aller urladefähigen Programme stehen bündig hintereinander. Ihre Anzahl ist in der Zelle "Anzahl Programme" hinterlegt. Die Reihenfolge der PI in der Buchführung beeinflußt die zeitliche Reihenfolge der Programme im automatischen Test. Im einzelnen besteht die PI aus:
- o Programmname: Name des urladefähigen Programms (genau 6 Zeichen)
- o EXAD - Anfangsblock: Externadresse (Byte-Adresse) des 1. zu ladenden Programmblocks (Block 0).
- o Name des bevorzugten Programms: Nach einem Urladeanstoß ohne Angabe eines Programmnamens bzw. nach Ablauf aller Testprogramme im nicht zyklischen automatischen Test, wird dieses Programm urgeladen.
- o Prüfmuster Buchführung: Kontrollmuster zur Datensicherung.

Programmblock

Der erste Block (Block 0) eines Programms beginnt bei Random-Datenträgern an der in der Buchführung für das jeweilige Programm hinterlegten Externadresse EXAD. Die Externadresse des nächsten Folgeblocks ist im Vorspann des gerade bearbeiteten Blocks enthalten. Die Blöcke beginnen bei Magnetschichtdatenträgern stets an einer Sektorgrenze. Bei seriellen Datenträgern stehen die Programmblöcke eines Programms beginnend ab EXAD 0 (Block 0) in aufsteigender Reihenfolge auf dem Externspeicher bündig hintereinander.

Der Programmblock besteht aus (siehe Bild 4.2):

- o Steuerwort: Unterscheidet den Programmblockvorspann von der Buchführung und steuert den Urlader in Bezug auf Folgeblock, Programmstart und -Datensicherung.
- o Blockzählung: Die einzelnen Blöcke eines Programms sind bei 0 beginnend durchnumeriert.
- o EXAD - Folgeblock: Externadresse des nächsten zu ladenden Blocks (wenn kein Folgeblock: irrelevant).
- o Programmname: Identifiziert das Programm. Der Programmname muß identisch sein mit dem Programmnamen in der Programmidentifikation PI der Buchführung (bei Random-Datenträgern). Die Programmnamen aller Blöcke müssen gleich sein.
- o Ausgabestand: beliebige Information.
- o START-R1: Adresse, an der das Programm nach dem Urladen und beim virtuelle Konsole-Kommando START gestartet werden soll. Es ist nur der Eintrag im letzten Programmblock relevant.
- o RESTART-R1: Adresse, an der das Programm bei dem nicht maskierbaren Interrupt NMI "RS ohne BF" und beim virtuelle Konsole-Kommando RESTART gestartet werden soll. Es ist nur der Eintrag im letzten Programmblock relevant.
- o PF-R1: Adresse, an der das Programm bei dem NMI "PF" (power-failure) gestartet werden soll.
  Es ist nur der Eintrag im letzten Programmblock relevant.
- o TZR: Tafelzeiger. Er weist auf die Parametertafel, mit der das Programm an den oben genannten Adressen gestartet wird. Die Parametertafel muß Bestandteil der Nutzdaten des Programms sein. In ihr wird je nach Startfall eines der oben genannten "R1" eingetragen. Die TZR-Angabe ist nur im letzten Programmblock relevant.
- o virtuelle Anfangsadresse: Mit einem Programmblock können maximal 128 K* byte Nutzdaten geladen werden. Diese Adresse gibt die relative Lage innerhalb dieses Adreßraums an, ab der die Nutzdaten in den Zentralspeicher zu laden sind. Nur wenn diese Adresse 0 ist, ist es möglich, 128 K* byte Nutzdaten zu laden. In diesem Fall liegt die absolute Anfangsadresse an einer Seitengrenze (Grenze zwischen zwei Bereichen zu je 2 K* byte).
- o Transferlänge: Anzahl der zu transferierenden Bytes (max. 128 K* byte). Die Anzahl der Nutzdatenwörter ist die Hälfte der Transferlänge.
- o Übersetzungstafel: Tafel, über die das Gerät die Nutzdaten in den Zentralspeicher lädt. Somit ist es möglich, ein Programm seitenweise über den Zentralspeicher zu verteilen (sofern das Urladegerät, d. h. die Anschaltung dazu in der Lage ist).
- o Prüfmuster Daten: Kontrollmuster zur Datensicherung der Nutzdaten.
- o Prüfmuster-Vorspann: Kontrollmuster zu Datensicherung des Vorspanns.
- o Nutzdaten: Daten, die das eigentliche urzuladende Programm sind und ohne weitere Bearbeitung oder Veränderung im Speicher abgelegt werden.

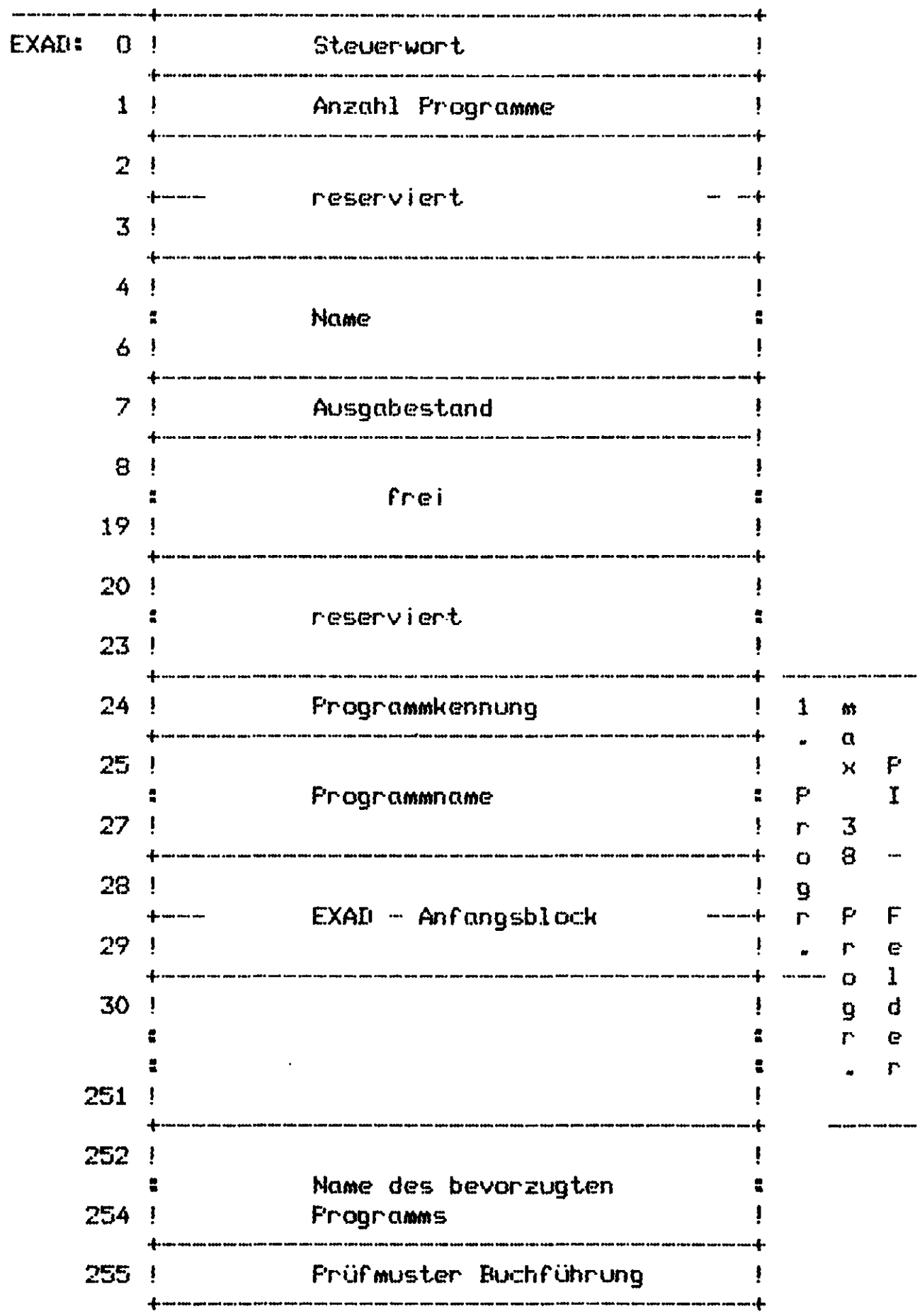


Bild 4.1 Aufbau der Buchführung

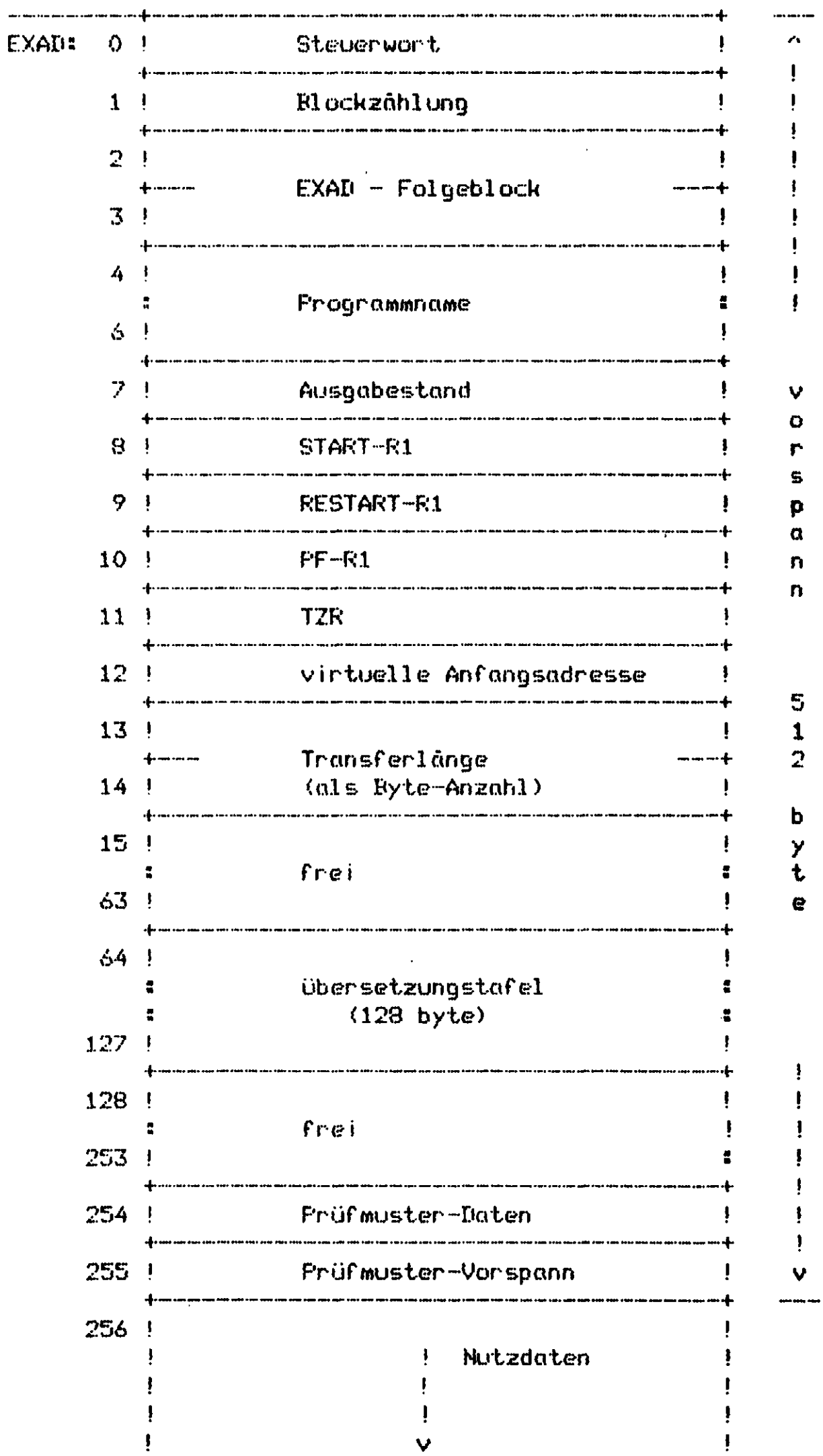


Bild 4.2 Aufbau des Programmblockvorspanns

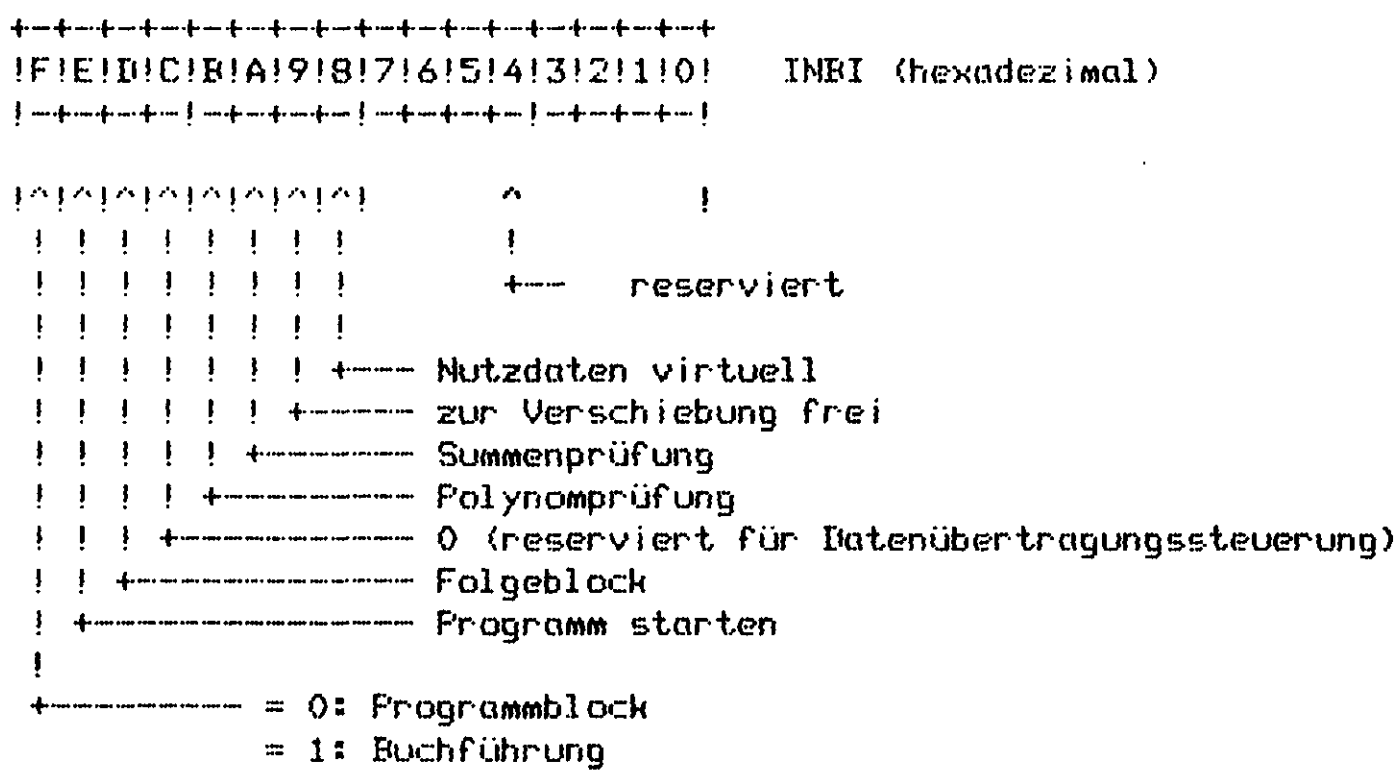


Bild 4.3 Steuerwort

Aufbau der urladefähigen Programme

Urladefähige Programme sind im allgemeinen:

- Zentraleinheits-Testprogramme,
- Organisationsprogramme (Betriebssystem),
- Hardware-nahe Dienstprogramme.

Sie können beliebig lang (max. 8060 K* byte) sein und blockweise (ein Block max. 128 K* byte) gestückelt sein.

Das Umsetzen von Grundsprache in Urladeformat und das Transferieren auf den Datenträger geschieht mit dem Dienstprogramm "MURL".

Der Programmierer eines urladefähigen Programms muß (aus der Sicht des Urladers) folgendes beachten:

- Die Start-Parametertafel muß in den Nutzdaten definiert sein, das Register mit der Startadresse (R1) ist dabei irrelevant.
- Die Übersetzungstafel, mit der der Befehlsablauf beim Start gesteuert wird, muß bei virtueller Adressierung ebenfalls Bestandteil der Nutzdaten sein.
- Die das Adressierungsverfahren bestimmenden Teile der Start-Parametertafel (z. B. AWR) sollten nie dynamisch verändert werden.

Testprogramme können in zwei Modi ablaufen:

- Normalmodus (Handbetrieb),
- Testmodus (automatischer Test).

Die Programme unterscheiden diese Modi an dem Bit "Testmodus" (INBI 7 in der Zelle ANLAWE des Hardware-Verständigungsbereichs). Ist es gesetzt, laufen die Programme im Testmodus und melden sich zum Abschluß ihres Testlaufs beim Urlader zurück. Machen sie diese Rückmeldung nicht, so müssen sie selber den weiteren Systemzustand verwalten.

4.4.2 Urladevorgang

Der Urlader wird initiiert durch den Anlaufteil, wenn in der Zelle ANLAWE das INBI 6 oder 8 gleich 1 ist. Der Urlader erwartet das Urladegerät in rückgesetztem Zustand, d.h. der Vektor muß auf den Basis-Prozeßblock zeigen. Die EA-Adresse des Urladegeräts steht in URGERA. Der Urlader wird gesteuert durch die INBI 6...8 in der Zelle ANLAWE (Zelle 1032) im Hardwae-Verständigungsbereich:

INBI 876 I Funktion

```
000       I -
001       I Urladen, beliebiges Programm
010       I -
011       I zyklischer Testlauf
100       I nur Buchführung laden
101       I Urladen, bevorzugtes Programm
110       I Urladen, bevorzugtes Programm im Testlauf
111       I nicht zyklischer Testlauf
 I
 +----- Testmodus
```

Der Inhalt der Zelle BTRESE (Zelle 1036 im Hardware-Verständigungsbereich) definiert das "verschobene Laden". In den drei Zellen PRONA (Zellen 1040 - 1042 im Hardware-Verständigungsbereich) steht der Programmname, bzw. wird durch den Urlader dort eingetragen. Außerdem übernimmt der Urlader den Eintrag des Programms in die Startliste, die ggf. zuvor gelöscht wurde (abhängig von INBI 12 in die Zelle ANLAWE) des Hardware-Verständigungsbereichs.

Der Beginn des Ladevorgangs eines Programms wird unter Angabe des Programmnamens über die virtuelle Konsole gemeldet, sofern in der Zelle ANLAWE das INBI 2 gleich 0 ist.

Programmstart

Ein Ladevorgang ist abgeschlossen, wenn das Bit "Folgeblock" im Steuerwort (siehe Bild 4.3) des zuletzt geladenen Programmblocks gelöscht ist. Ist außerdem das Bit "Programmstart" in dem Steuerwort gesetzt, wird das Programm gestartet.

Datensicherung

Um Übertragungsfehler ausschließen zu können, wird eine Prüfung der eingelesenen Daten durchgeführt. Es kommen zwei Verfahren zum Einsatz:

- Polynomprüfung
  Bei dieser Prüfung wird das Prüfmuster ermittelt nach dem Polynom

  $x^{16} + x^{12} + x^{5} + 1.$

- Summenprüfung
  Bei dieser Prüfung werden die Daten in einem Wort aufsummiert, wobei ggf. entstehende Überträge als 1 in das Summenwort addiert werden (Wort = 16 bit).

Die Polynomprüfung ist sicherer als die Summenprüfung. Sie ist aber etwa um den Faktor 20 langsamer. Während zur Datensicherung der Buchführung und des Programmblockvorspanns stets die Polynomprüfung eingesetzt wird,

kann der Programmersteller selbst entscheiden, welches Verfahren zur Sicherung der Nutzdaten eingesetzt werden soll. Die Information, nach welchem Verfahren die Nutzdaten gesichert sind, entnimmt der Urlader dem Steuerwort.
Die Bits "Summenprüfung" und "Polynomprüfung" dürfen nicht gleichzeitig gesetzt sein. Sind beide Bits gelöscht, wird keine Datensicherung der Nutzdaten vorgenommen.

4.5 Testanschaltung TESTAS

Die Testanschaltung TESTAS ist eine Hardware-Komponente der Zentraleinheit ZE 03.
Sie bildet die hardwaremäßige Voraussetzung zur Realisierung von Testpunktfunktionen (Adreßvergleiche in verschiedenen Modifikationen), ohne einen Eingriff im Testobjekt vornehmen zu müssen.

Die TESTAS ist eine 3-fache Europabaugruppe, die zur Erfüllung ihrer Funktion auf spezielle Signale der ZE 03 angewiesen ist. Sie ist daher nur an der Service-Schnittstelle betreibbar.

Die TESTAS selbst erhält ihre Bedeutung erst im Zusammenspiel mit dem entsprechenden "Testhilfeprogramm", für das die TESTAS die Hardware-Grundlage darstellt. Ein solches Testhilfeprogramm ist das Programm VICOM, das im Festspeicher der ZE 03 zur Verfügung steht. Mit Hilfe der dort verfügbaren Debugging-Funktionen können Testpunkte definiert werden. Die für die Definition eines Testpunkts verfügbarem Parameter - Hardware-Bedingung der TESTAS bzw. Software-Bedingung von VICOM - sind dem Teil virtuelle Konsole (Kapitel 4.3) zu entnehmen.

4.6 Teleservice

Unter dem Begriff versteht man den Servicezugriff zu einem Rechnersystem von einem Remote-Terminal oder Rechnersystem. Diese befinden sich in einem Servicezentrum und greifen über das öffentliche Telefonnetz zu.
Die Systeme mit den Zentraleinheiten ZE 03 sind grundsätzlich für diese Art des Servicezugriffs geeignet. Der notwendige Anschluß ist auf der PROMEA EA-01-Grundbaugruppe realisiert. Bei Abschluß eines Wartungsvertrags wird die notwendige Datenübertragungseinrichtung (Akustikkoppler) zur Verfügung gestellt.
Die prinzipielle Konfiguration ist im folgenden übersichtsbild (Bild 4.4) dargestellt.

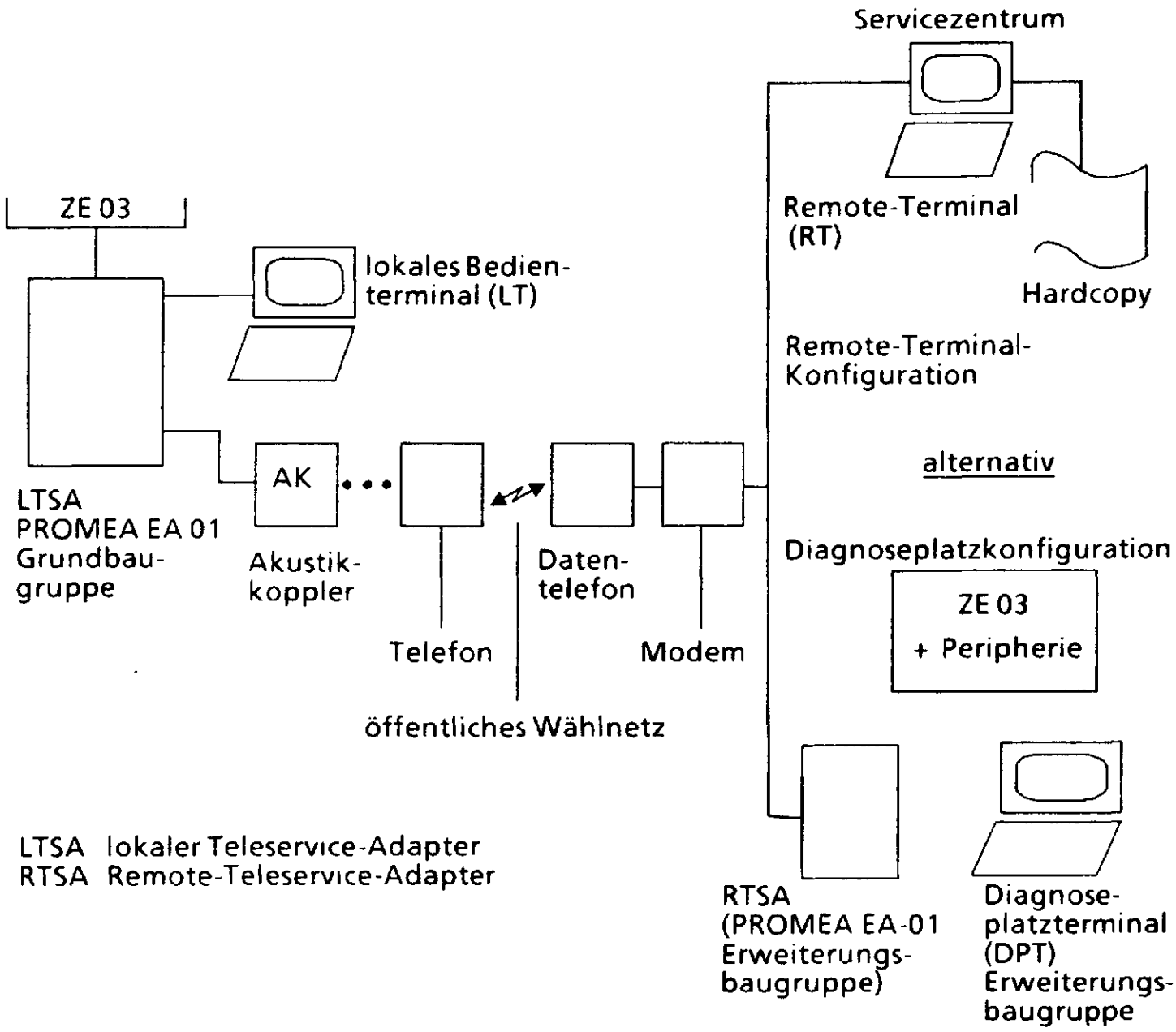


Bild 4.4 Servicezentrum

Folgende Parametereinstellungen sind per Kommando an den Teleservice-adapter möglich:

Kommando Syntax (bei Vor-einststellung)	Bedeutung	zugelassen am LT	RT	DPT
L	Lokal-Betrieb (Teleservice ist gesperrt)	x	x	
LM	Lokal + Mithören remote	x	x	
LRM	Lokal + remote + gegenseitiges Mithören	x		
LR	Lokal + remote ohne gegenseitiges Mit-hören	x		
RT	Verbindungsaufbau RT-Konfiguration	x		
DP	Verbindungsaufbau DP-Konfiguration	x		
+	Verbindungsaufbau RT-Konfiguration (Quittung durch Servicezentrum)		x	
MODOFF	Modem am LTSA ausschalten	x	x	x
N	Verbindung prüfen (Dummy)		x	SRT
message	Kommunikationsfunktion	x	x	x

Tab. 4.1 Übersicht über LTSA-Kommandos und deren Bedeutung

SRT = simuliertes Remote-Terminal
LT = lokales Terminal
RT = remote Terminal
DPT = Diagnoseplatzterminal

Nach Freigabe der Verbindung durch den Kunden können vom Spezialisten im Servicezentrum alle Aktivitäten durchgeführt werden, die auch vom lokalen Terminal aus durchführbar sind, z. B.

- Arbeiten mit VICOM
- Arbeiten mit urladefähigen Zentraleinheits-Testprogrammen
- normale Systembedienung (Lesen, Starten, Bedienung)
- Arbeiten mit Hardware-Testprogrammen
- Abruf von Statistikdaten (Fehlerstatistikprogramm) und Logbüchern.

Zusätzlich besteht die Möglichkeit, in der Diagnoseplatzkonfiguration Daten zwischen den beiden Rechnersystemen auszutauschen (Dateiübertragung, Software-Aktualisierung).

4.7 Serviceprozessor-Anschaltung und Serviceprozessor

Zur Diagnosedatenerfassung, zur dynamischen Überwachung eines Rechnersystems und für komplexe Hardware/Software-Diagnosen steht optionell als leistungsfähiges Hilfsmittel die Serviceprozessoranschaltung (SPAS) und der ggf. anschließbare Serviceprozessor (SP) zur Verfügung.

Um das dynamische Verhalten eines zu testenden Systems so wenig wie möglich zu beeinflussen, ist eine weitgehend verzögerungsfreie Erfassung von Diagnosedaten realisiert. Damit ist die Voraussetzung zur Erfassung und Analyse von zeitkritischen Vorgängen geschaffen.

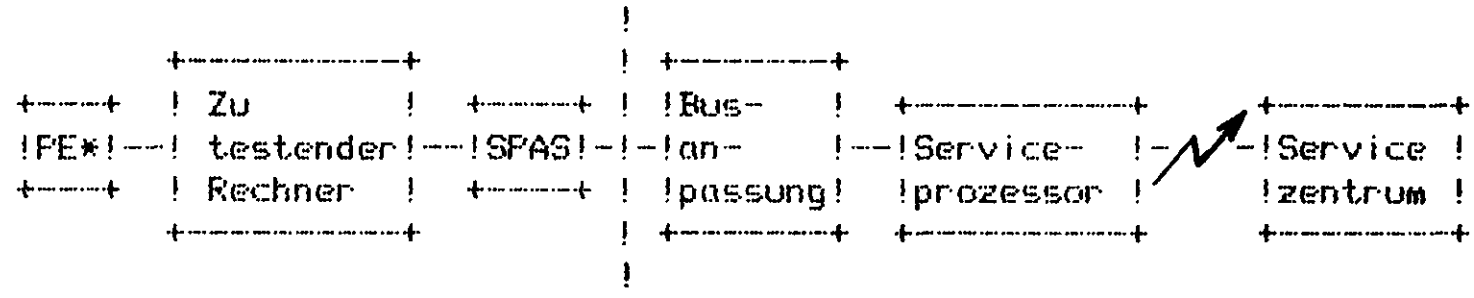
Es stehen zwei, im Leistungsumfang unterschiedliche Betriebsarten des Serviceprozessors zur Verfügung:

- "Kleine Service-Einheit"

* = periphere Einheit

Die SPAS ist in dieser "kleinen Serviceeinheit" ohne eigenen Serviceprozessor betreibbar; der zu testende Rechner wird funktionell zum Serviceprozessor und prüft und überwacht sich selbst. Bezüglich der Diagnosemöglichkeiten, vor allem bei zeitkritischen Vorgängen, stellt diese Lösung einen eingeschränkten Leistungsumfang zur Verfügung.

o "Große Service-Einheit"



* = periphere Einheit

Diese Lösung bietet den größten Leistungsumfang in den Diagnosemöglich keiten und der Diagnosetiefe. Die im zu testenden Rechner gesteckte SPAS wird über eine spezielle Busanpassungsbaugruppe mit dem Service prozessor verbunden. Als Serviceprozessor kommt jede beliebige mit Standardbetriebssystem betriebene Zentraleinheit der Modellreihe SICOMP in Betracht.

Die Baugruppe SPAS wird in den speziellen Service-Steckplatz der Zentraleinheit ZE 03 gesteckt (3-fach Europabaugruppe). Sie enthält folgende technischen Komponenten:

o Schnittstelle zum zu testenden Rechner
Sie ermöglicht sowohl die Kommunikation mit dem zu testenden Rechner als auch den physikalischen Zugriff auf Diagnosedaten wie z. B. Speicherinhalte.

o Mithörpuffer
Der Mithörpuffer zeichnet mit einer Breite von 64 bit den Datenverkehr auf den Bussen des zu testenden Rechners völlig verzögerungsfrei auf. Eingetragen wird jeweils Datum, reelle Zentralspeicher-Adresse, Zeitmarke und Zusatzinformationen über Herkunft und Bedeutung der eingetragenen Daten.

o Adreßvergleiche
Zur Überwachung des Adreßraums des zu testenden Rechners auf lesende, schreibende Zugriffe stehen vier Adreßvergleicher zur Verfügung. Jeder Adreßvergleicher kann umschaltbar entweder auf eine einzelne Adresse, einen 64-Adressen-tiefen oder einen 4 x 1024-Adressen-tiefen Adreßvergleichsraum parametriert werden.

o Testpunktbearbeitung
Diese Strategie zur Fehleranalyse in Software-Systemen baut auf dem speziellen Testbefehl auf. Nach Vorgabe von der SPAS hält die zu testende Zentraleinheit nach dem Durchlaufen eines Testbefehls an und verständigt die SPAS zur weiteren spezifischen Reaktion.

o Bedienung der virtuellen Konsole
Der Serviceprozessor erhält über die SPAS Zugriff auf das Programm VICOM des zu testenden Rechners und kann dessen Funktionen benutzen.

o Zeitgeber
Zur genauen Rekonstruktion des zeitlichen Ablaufs im zu testenden Rechner oder z. B. für Tuning-Maßnahmen steht ein hochauflösender Zeitgeber zur Verfügung.

o Steuerung
  Die Bewältigung der komplexen Aufgaben der SPAS wie z. B. Kommunikation mit dem zu testenden Rechner und dem Serviceprozessor, Verwalten und Parametrieren der Adreßvergleicher usw. übernimmt der Zentralprozessor einer ZE 01. Für seine Software stehen 32 K* byte EPROM und 32 K* byte RAM zur Verfügung.

o Schnittstelle zum Serviceprozessor
  Diese Schnittstelle dient zur logischen Kommunikation zwischen Serviceprozessor und SPAS nach Vorgabe der Geräteklasse 3.

Mit den beiden Serviceprozessor-Lösungen werden Leistungen für folgende Anwendungsfälle geboten:

- Diagnose von Hardware/Software-Vorgängen, die mit Standardmitteln nicht diagnostizierbar bzw. lokalisierbar sind.
- Systemüberwachung bei sporadischen Fehlern
- Erweiterung der Software-Testmöglichkeiten
- Ablaufverfolgung
- Auslastungsmessungen ohne Einfluß auf das Real-Zeit-Verhalten des zu testenden Rechners
- Unterstützung bei Entwicklung und Test von Anwenderpaketen.

5. Bedienelemente

5.1 Zentralprozessor

Die für den Anwender relevanten Schalter und Brücken befinden sich auf der Speichersteuerungsbaugruppe TC.

Kippschalter BL:	Der eingelegte Bus-lock-Schalter (BL) ermöglicht das Ziehen und Stecken von Baugruppen unter Spannung und verhindert, daß eventuell auf dem Systembus auftretende Störungen den Speicher zerstören. Da hierbei der Busverkehr ohne irgendwelche Rettungsmaßnahmen einfach unterbrochen wird, ist der Zeitpunkt der Betätigung dieses Schalters sorgfältig zu wählen (z. B. Zentraleinheit im Stop).
Drehschalter URL:	Einstellung der 16-bit-Adresse des Geräts, von welchem urgeladen werden soll.
Drehschalter VC:	Einstellung der 16-bit-Adresse des Geräts, welches als virtuelle Konsole arbeiten soll.
Drehschalter SP-GRENZE:	Mit diesem Schalter wird die obere Speichergrenze der zentraleinheitsspezifischen Speichersteuerung eingestellt (obere Grenze des Schreib-Lese-Speichers).

5.2 Zentralspeicher-Module

Drehschalter	Modulkennung O ... F
DIP-Schalter S 2, S 3 (64 k*byte Moduln) S 1 (256 k*byte Moduln)	Anfangsadresse des Speicherbereichs, Modulgröße, Wort-/Doppelwortbetrieb (s. Kap. 3.2)

Lötbrücken für Pufferbetrieb:

Betriebsart / Brücke	gepuffert	ungepuffert
X 35 - X 37	offen	eingelegt
X 36 - X 38	offen	eingelegt
X 37 - X 39	eingelegt	offen
X 38 - X 40	eingelegt	offen

5.3 Cache-Speicher

DIL-Schalter für Grenze der Cache-Speicher-Umgehung
Grenzadresse = 8 M* byte - 128 K* x 2^{n}
n^{hexa} = Schalterstellung

5.4 Container

Auf der Frontseite des Containers sind verschiedene Bedien- und Anzeigerichtungen untergebracht:

- Schlüsselschalter: Netz ein/aus (power supply)
  Rücksetze (reset)

- LED-Anzeigen: Stromversorgung (power supply check)
  Urladegerät (bootstrap check)
  virtuelle Konsole (console check)
  Zentralspeicher (memory check)
  Zentralprozessor (processor check)
  Stop

6. ANHANG

6.1 Technische Daten

Zentralprozessor

Prozessorstruktur	2 Prozessoren (je 16 bit parallel)
Technologie	Standard LSI (bipolar)
Adressiervolumen	8 M* byte
Wortbreite	16 bit
Befehls-look-ahead	2-fach
Befehlsvorrat	338
Aufbautechnik	3-fach Europaformat (366,7 mm x 160 mm)
Anzahl der Baugruppen	6
Ausbaumöglichkeiten	Cache-Speicher, Gleitpunktprozessoren
Cache-Speicher Lesezugriff	200 ns
Zentraltakt	10 MHz

Stromversorgung

Aufbau	Stromversorgung mit den Abmessungen des Zentraleinheits-Baugruppenträgers
Anschlußspannung	220 V (einphasig)
Spannungstoleranz	
- statisch	± 10%
- dynamisch	- 100% max. 10ms + 30% max. 5ms
Netzfrequenz	50 Hz ± 6% 60 Hz ± 5%
zulässiger Klirrfaktor	35%
Leistungsaufnahme	1850 VA

Stromaufnahme des Zentralprozessors (Werte in A)

Baugruppen	I	5 V	I
Zentralprozessor VP	I	33	I
AP	I	33	I
BA	I	33	I
SWT	I	33	I
VA	I	33	I
TC	I	33	I
Cache-Speicher	I	7	I
Standard-Gleitpunktprozessor	I	5	I
MLFB-Nummer 6AA7003-0AA	!		!
Schneller Gleitpunktprozessor	!	10	!
MLFB-Nummer 6AA7003-0FA	!		!

Speichermoduln (alt) mit 64 k*byte-Chips:

	working		stand-by	
	5 V	5 V gepuffert	5 V	5 V gepuffert
256 K* byte	3,1	1,3	2,65	0,85
512 K* byte	3,3	1,5	2,85	1,05
1024 K* byte	3,85	2	3,4	1,55

Die Aufnahmewerte für 5V sind Gesamtwerte. Sie verringern sich bei Pufferung um den Anteil von 5V gepuffert.

Speichermoduln (neu) mit 256 k*bit-Chips:

	working		stand-by	
	5 V	5 V gepuffert	5 V	5 V gepuffert
1 M * byte	3,9 A	1,4 A	2,95 A	0,45 A
2 M * byte	4,1 A	1,6 A	3,15 A	0,65 A
4 M * byte	4,6 A	2,1 A	3,65 A	1,15 A

Die Aufnahmewerte für 5V sind Gesamtwerte. Sie verringern sich bei Pufferung um den Anteil von 5V gepuffert.
Bei Pufferung ist die Stromaufnahme aus 5V gepuffert gleich dem bei 5V gepuffert angegebenen stand-by-Wert.

Busdatenrate
Alle Werte mit eingeschalteter Fehlerkorrektur
Datenrate mit Buseroberung

	Schreiben (M*byte/s)	Lesen (M+byte/s)
Doppelwort	8,0	6,2
Wort	3,6	3,1
Byte	1,8	1,55

max. Datenrate

	Schreiben (M*byte/s)	Lesen (M+byte/s)
Doppelwort	8,9	7,3
Wort	3,6	3,6
Byte	1,8	1,8

Im Burst-Mode (Dauerbelegung des Busses durch den Busmaster) kann ein Busteilnehmer die max. Datenrate erreichen.

Speichermoduln:

	Zugriffszeit (ns)	Zykluszeit (ns)
Schreiben Byte	300	550
Schreiben Wort	150	550
Schreiben Doppelwort	150	450
Lesen Byte, Wort ,Doppelwort	300	400

Korrekturmaßnahmen (ECC) verlängern die Zykluszeit für Lesezugriffe um 150 ns.
Der Zerntralspeicherbereich wird in einen ZE-spezifischen RAM- und in einenFestwetspeicherbereich unterteilt, wobei die Größe beider Bereiche durch eine Schalterstellung auf der ZE-Baugruppe wählbar ist.

Schalterstellung	Zentralspeicher (RAM)	Festwertspeicher
0	8064 K* byte	128 K* byte
1	7936 K* byte	256 K* byte
2	7680 K* byte	512 K* byte
3	7168 K* byte	1024 K* byte
4	6144 K* byte	2048 K* byte
5	4096 K* byte	4096 K* byte

Der Zentralspeicher des ZE03 kann bis 8 M*byte addressiert und ausgebaut werden.

Umgebungsbedingungen

Luftdruck (untere Grenze): 700mbar (3000m über NN)

Umgebungstemperatur (Betrieb):
o Baugruppenträger 0°C bis 55°C (SN 26556, Tabelle 1 Kennbuchstabe B)
o Container +10°C bis +35°C

Umgebungstemperatur (Stillstand und Transport):
o 40°C bis +70°C (SN 26556, Tabelle 2, Kennbuchstabe K)

Zulässige Luftfeuchte (Betrieb):
o Baugruppenträger 75% relative Feuchte
o Container 20% bis 80% relative Feuchte

Zulässige Luftfeuchte (Lagerung und Transport):
o 65% relative Feuchte

Funkentstörung: VDE 0871 Grenzwertklasse A

Zulässige Fremderschütterung: gemäß SN 29010, Teil 1-3
Prüfung nach DIN 40046, Teil 8

Luftführung: Zwangsbelüftung

Luftfilter: DIN 24185, Filterklasse EU2

Steckerbelegung der EA- und Speicher-Schnittstelle

	Stecker 1 (oben)			Stecker 2 (Mitte)				Stecker 3 (unten)		
	a	b	c	a	b	c		a	b	c
1	CC	+5V	-12V	BF	+5V	+24V			+5V	
2	VC	+5V	+12V	RES12	+5V	+5VP	-nur für Speicher-		+5V	
3	RSP	0	HLD	AE1	0	+5VP	-steckplätze		0	
4	TER	0	BL	BM0	0	BM1			0	
5	TR	0	TA	DS	0	ZSP	-			
6	BR	0	BB	ZSP	0	ZSP	!			
7	IR	0	IL	ZSP	0	ZSP	!		0	
8	IAO	0	IAI	ZSP	0	ZSP	!			
9	BAO	0	BAI	ZSP	0	ZSP	!		0	
10	R/$\overline{W}$	0	M/$\overline{IO}$	ZSP	0	ZSP	! nur für			
11	AC0	0	AC1	ZSP	0	ZSP	> Speicher-		0	
12	AB0	0	AB1	ZSP	0	ZSP	! steckplätze			
13	AB2	0	AB3	ZSP	0	ZSP	!		0	
14	AB4	0	AB5	ZSP	0	ZSP	!			
15	AB6	0	AB7	ZSP	0	ZSP	!		0	
16	AB8	0	AB9	ZSP	0	ZSP	-			
17	AB10	0	AB11	RES0	0	RES1			0	
18	AB12	0	AB13	RES2	0	RES3				
19	AB14	0	AB15	RES4	0	RES5			0	
20	AB16	0	AB17	RES6	0	RES7				
21	AB18	0	AB19	RES8	0	RES9			0	
22	AB20	0	AB21	RES10	0	RES11				
23	BT	0	AE0	DP1	0	DP2	-nur für Spei-		0	
24	FFA	0	FFD	DP3	0	DP4	-chersteckplätze			
25	DB0	0	DB1	DB16	0	DB17			0	
26	DB2	0	DB3	DB18	0	DB19				
27	DB4	0	DB5	DB20	0	DB21			0	
28	DB6	0	DB7	DB22	0	DB23				
29	DB8	0	DB9	DB24	0	DB25			0	
30	DB10	0	DB11	DB26	0	DB27			0	
31	DB12	+5V	DB13	DB28	+5V	DB29			+5V	
32	DB14	+5V	DB15	DB30	+5V	DB31			+5V	

6.2 Bitnumerierung, Abkürzungen

I 2^{15}	2^{0}I
MSB	LSB
SIBI Nr. 0	15
INBI Nr. 15	0

MSB ... most significant bit (höchstwertiges Bit)
LSB ... least significant bit (niedrigstwertiges Bit)
SIBI ... Siemens-Bitnumerierung
INBI ... internationale Bitnumerierung

6.3 Liste der vom Zentralprozessor belegten EA-Adressen

		Zugriffsart
0000	NMI-Register	lesen
0001	TC-Steuerwort	schreiben, lesen
0002	Adresse des Urladegeräts	lesen
0003	Adresse des virtuellen Konsole-Geräts	lesen
0004	reserviert	
0005	LED-Register (für Container-Anzeigen)	schreiben
0006	Sammel-TER (terminate, beenden)	
0007	Sammel-RSP (reset peripheral, Peripherie rücksetzen)	
0008-		
.		
.		
. >>	reserviert	
.		
.		
.		
001F-		
0020-	Fehlerregister der	
. >>	Speichermodule	lesen
002F-		
0030-		
.		
. >>	reserviert	
003F		
0040-		
. >>	TESTAS	
.		
004F-		
0050	Serviceprozessor-	
0051	Anschaltung SPAS	
0052-		
.		
. >>	reserviert	
.		
007F-		
0080	aufwärts: frei verfügbar	

6.4 Befehlsliste

Anmerkung:

OP .... Operand
z. B. $OP_{i,0-7}$ ... Operand i, Bit 0 - 7

Byte = 8 bit
Wort = 16 bit
Doppelwort = 32 bit

Die Formate sind in Kapitel 3.1.3 erklärt.
GPP ... Gleitpunktprozessor

6.4.1 Beschreibung der einzelnen Befehle

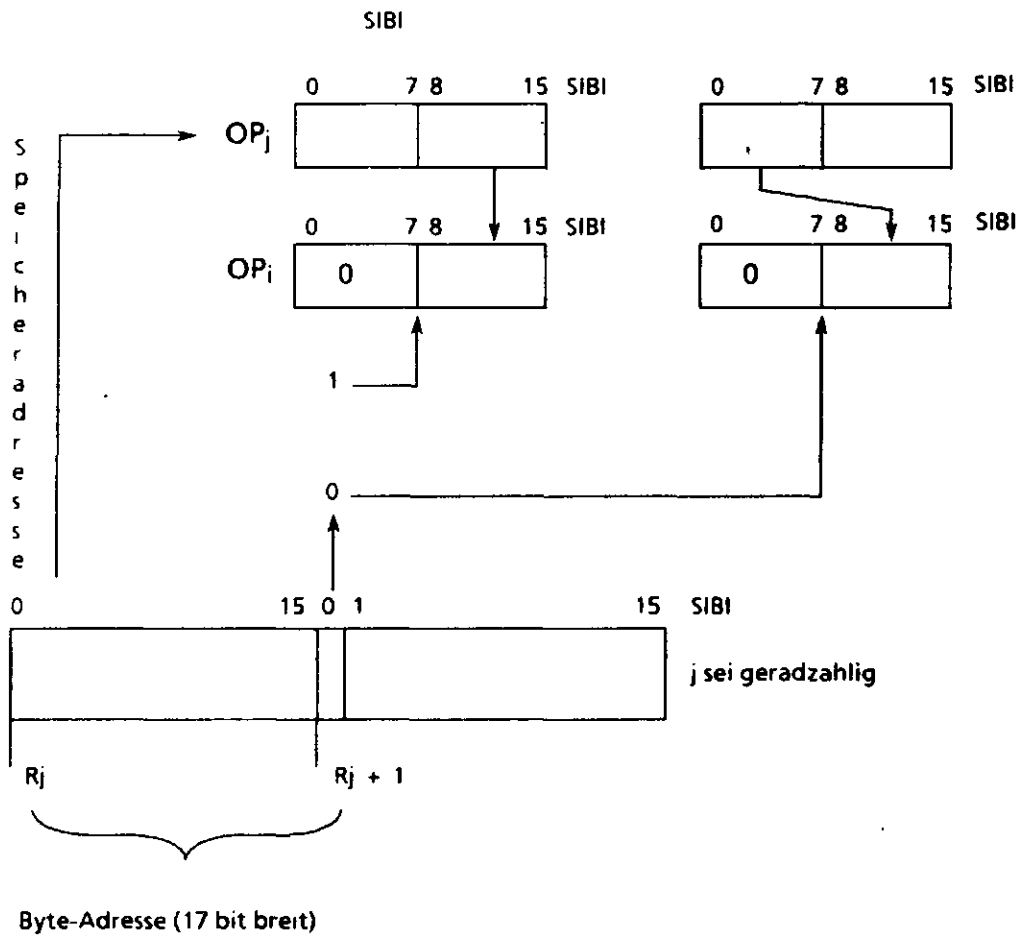
6.4.1.1 Ladebefehle

Befehl		Formate	Beschreibung	Anzeige	Bemerkung
LLB	Laden linkes Byte	RR, RA, RAI, RAX	$OP_{i,0\ \ 7} := OP_{j,0\ \ 7}$ $OP_{i,8\ \ 15} := 0$	-	-
LRB	Laden rechtes Byte	RR, RA, RAI, RAX	$OP_{i,0\ \ 15} = OP_{j,8\ \ 15}$ $OP_{i,0\ \ 7} = 0$	-	-
LAB	Laden Byte	RAI	$OP_{i,0\ \ 7} = 0$ $OP_{i,8\ \ 15} = \left\{ OP_{j,0\ \ 7} \atop OP_{j,8\ \ 15} \right\}$ siehe Text	-	-
LAF	Laden Wort	RC, RR, RA, RAI, RDA, RAX	$OP_i = OP_j$	-	-
LTF	Laden und Testen Festpunkt Wort	RR, RA, RAI RAX	$OP_i = OP_j$	F 012	-
LLF	Laden, Testen und Loschen Festpunkt Wort	RR, RA, IRAI RAX	$OP_i := OP_j$ $OP_j := 0$	F 012	-
LKF	Laden Komplement Festpunkt Wort	RC, RR, RA RAI, RAX	$OP_i := \overline{OP_j} + 1$ UE: = Übertrag aus Bit 0	F 0123	- -
LAD	Laden Doppelwort	RC, RR, RA RAI, RAX, RDA	$OP_{i,} = OP_j$	-	-
LTD	Laden und Testen Festpunkt Doppelwort	RR, RA RAI, RAX	$OP_i := OP_j$	F 012	-
LKD	Laden Komplement Festpunkt Doppelwort	RC, RR, RA RAI, RAX	$OP_i := \overline{OP_j} + 1$ UE = Übertrag aus Bit 0	F 0123	-
LAG	Laden Gleitpunkt Dopelwort	RR, RA, RAI, RAX, RDA	$OP_i = OP_j$	-	-
LTG	Laden und Testen Gleitpunkt Doppelwort	RR, RA, RAI, RAX	$OP_i := OP_j$	G 012	-
LAK	Laden Gleitpunkt Vierfachwort	RR, RA RAI, RAX, RDA	$OP_i := OP_j$	-	-
LTK	Laden und Testen Gleitpunkt Vierfachwort	RR, RA RAI, RAX	$OP_i := OP_j$	G 012	-
TAB	Tauschen Byte	RR	$OP_{i,0\ \ 7} := OP_{j,8\ \ 15}$ $OP_{i,8\ \ 15} := OP_{j,0\ \ 7}$	-	-
UBA	Umsetzen Byte-Adresse	RR	$OP_{i,16} := 0$ $O_{Pi,17\ \ 48} := OP_{j,0\ \ 31}$	-	-
LAC	Laden aus Code-Adreßraum	RAX	$OP_i := OP_j$	-	-

LAB Dieser Befehl lädt ein Byte eines Speicheroperanden in das rechte Byte des im F1-Feld adressierten Standardregisters; das linke Byte des Registers wird gelöscht.

Die Adressierung des Bytes im Speicher erfolgt über ein durch das F2-Feld adressiertes Standardregisterpaar. Das geradzahlige Register gibt die Wortadresse der Speicherzelle an. Das höchstwertige Bit im folgenden ungeradzahligen Register kennzeichnet das Byte.
Die übrigen Bits dieses Registers sind irrelevant und bleiben unverändert.

Bei jedem Durchlaufen des Befehls wird über die 17-bit-lange Byte-Adresse inkrementiert.



LAF Im RC-Format werden die 4 Bits des F2-Felds rechtsbündig übertragen, die Bits 0 bis 11 werden gelöscht. Ist die zu ladende Konstante >15, ist das RC-Format nicht möglich, sondern es muß ein LAF im RAI-Format verwendet werden, wobei im F2-Feld eine 1 stehen muß. Im darauffolgenden Wort steht die zu ladende Konstante (es wird mit R1 adressiert).

LTF Die Anzeige wird entsprechend dem Betrag und dem Vorzeichen des geladenen Operanden gesetzt.

LLF Op_j wird in das Register i transferiert und wie bei LTF getestet. Dann wird OP_j gelöscht. Wenn im RR-Format im F1- und F2-Feld das gleiche Register angegeben wird, so bleibt dieses Register unverändert. Bei A-Format Speicherschutz beachten!

LKF siehe LTF

LAD Im RC-Format werden die 4 Bits des F2-Felds rechtsbündig übertragen, die Bits 0 bis 27 werden gelöscht. Ist die zu ladende Konstante >15, ist das RC-Format nicht möglich, sondern es muß ein LAD im RAI-Format verwendet werden, wobei im F2-Feld eine 1 stehen muß. In den darauffolgenden beiden Wörtern steht die zu ladende Konstante (es wird mit R1 adressiert).

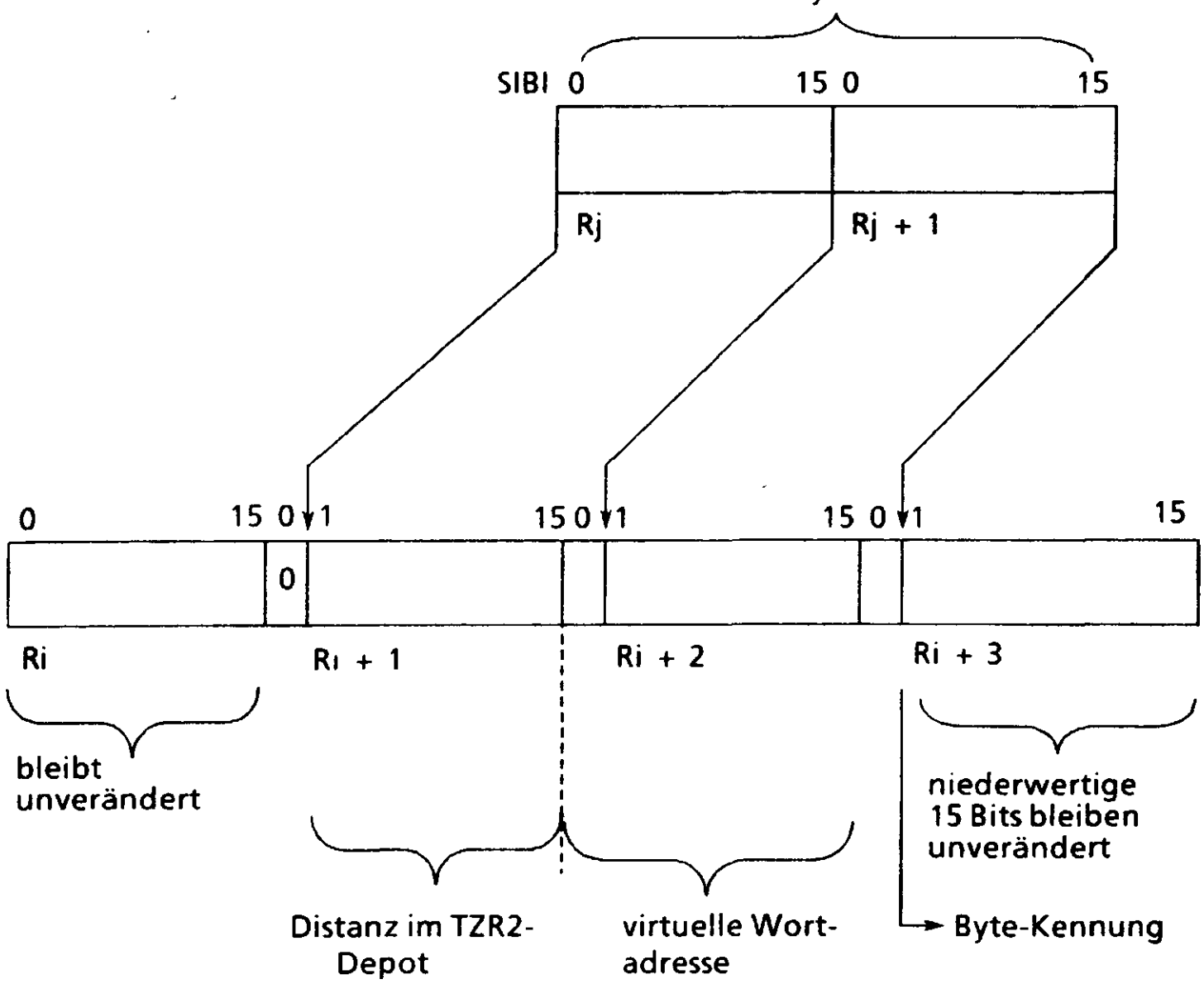
LTD Wie Befehl LAD, die Anzeige wird entsprechend dem Betrag und Vorzeichen des geladenen Operanden gesetzt.

LKD siehe LTF

LTG Die Anzeige wird entsprechend dem Betrag und dem Vorzeichen
LTK der geladenen Mantisse gesetzt. Der Exponent wird nur transferiert.

TAR Das rechte und das linke Byte des im F2-Feld genannten Registers wird in das im F1-Feld genannte Register umgeladen, wobei das linke Byte rechts und das rechte Byte links im Register plaziert wird.

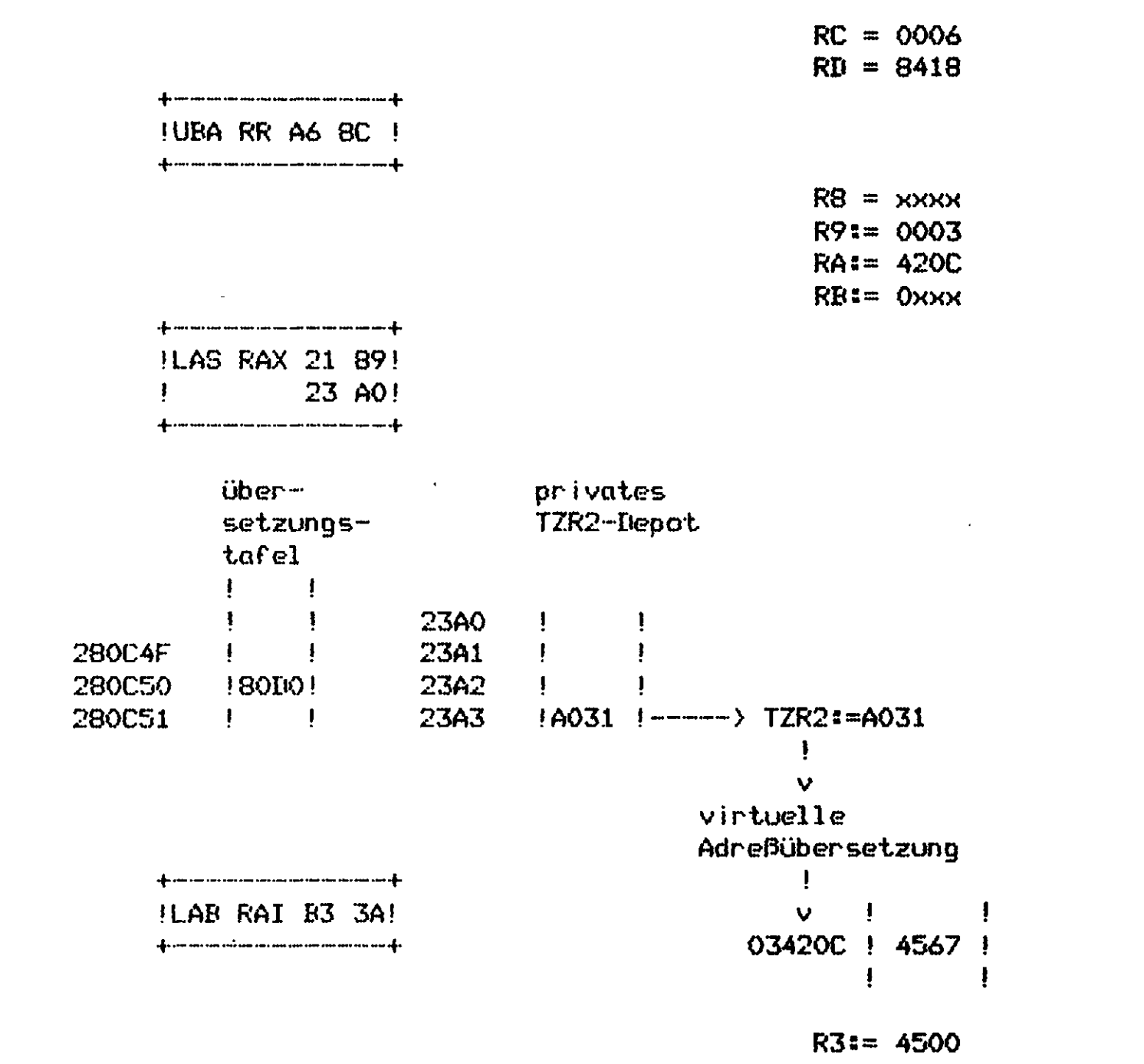
UBA Aus einem Registerpaar wird eine Byte-Adresse in ein Registerquartett in die im Siemens System 300 R übliche Form gebracht. Im F1- bzw. F2-Feld ist jeweils das erste Register anzugeben.



Anwendung des Befehls:

Der Anwender kann über diesen Befehl jede reelle Speicheradresse zusammensetzen. Dazu muß er ein privates TZR2 (Tafelzeigerregister 2)-Depot errichten, um daraus das TZR2 durch einen Befehl LAS im RAX-Format zu laden.
Im F3-Feld des LAS RAX gibt man die Anfangsadresse des Depots an, im F2-Feld das Register R_{i+1} des Befehls UBA im RR-Format. R_{i+1} gibt den Platz der Zelle im TZR2-Depot an, aus der das TZR2 geladen werden soll. R_{i+2} liefert die virtuelle Wortadresse der zusammenzusetzenden Speicheradresse und Bit 0 des R_{i+3} die Byte-Kennung.

Beispiel:



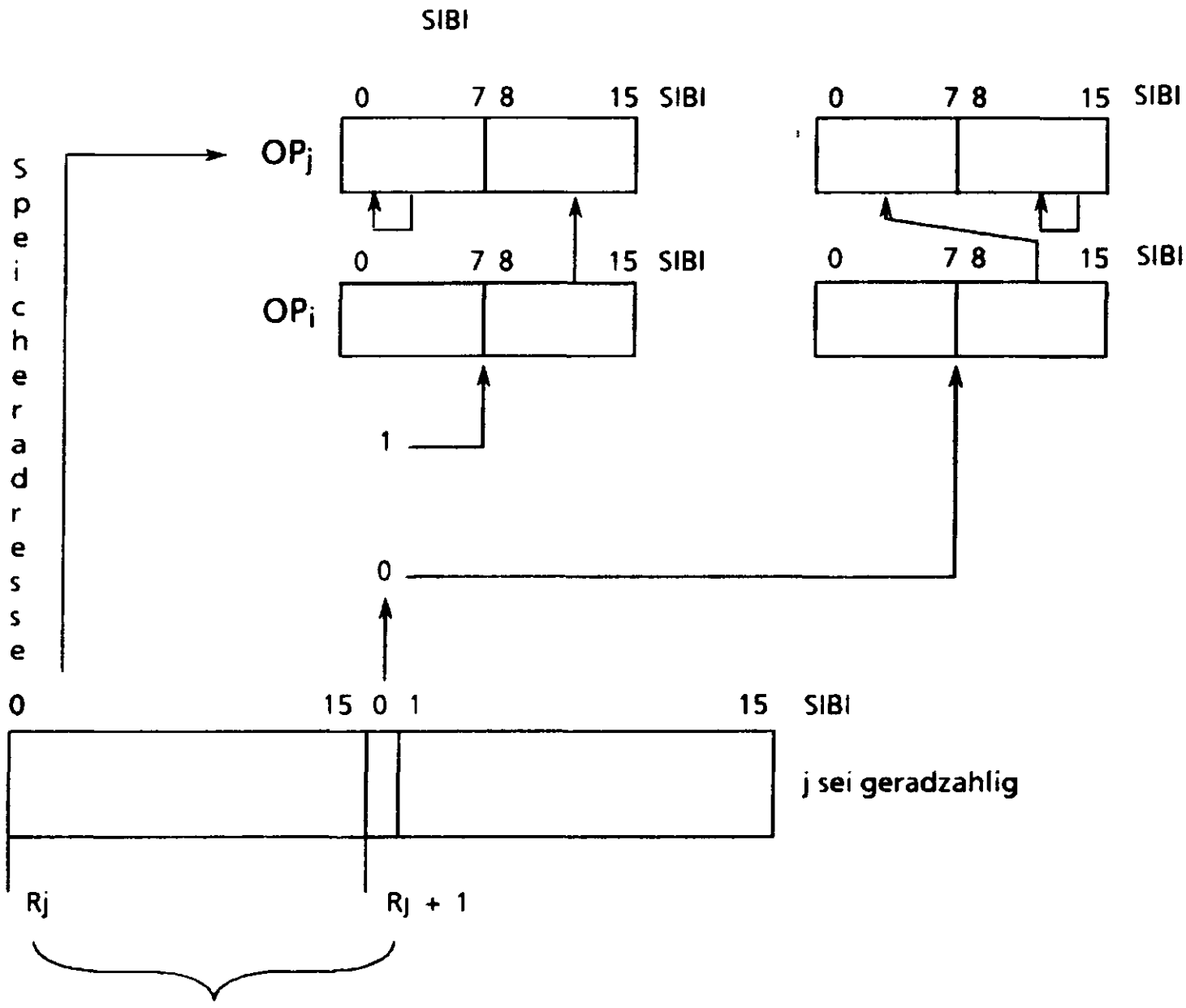
LAC Die Übersetzung der virtuellen Operandenadresse in eine reelle Adresse erfolgt über das Tafelzeigeregister 3 (TZR3) und die Übersetzungstafel 3 (ÜT3).
Falls im Programmzustandsregister (PZR) das PRZ-Bit 14=0 ist, wird die Operandenadresse reell interpretiert.

6.4.1.2 Speicherbefehle

Befehl		Formate	Beschreibung	Anzeige	Bemerkung
SLY	Speichern linkes Byte	RA, RAI, RAX	$OP_{j,0\,..\,7} := OP_{i,0\,..\,7}$ $OP_{j,8\,..\,15}$ unverändert	-	-
SRY	Speichern rechtes Byte	RA, RAI, RAX	$OP_{j,8\,..\,15} := OP_{i,8\,..\,15}$ $OP_{j,0\,..\,7}$ unverändert	-	-
SPF	Speichern Wort	RA, RAI, RDA, RAX	$OP_j := OP_i$	-	-
SPD	Speichern Doppelwort	RA, RAI, RDA, RAX	$OP_j := OP_i$	-	-
SPG	Speichern Gleitpunkt Doppelwort	RA, RAI, RDA, RAX	$OP_j := OP_i$	-	-
SPK	Speichern Gleitpunkt Vierfachwort	RA, RAI, RDA, RAX	$OP_j := OP_i$	-	-
SPB	Speichern Byte	RAI	siehe Text $\left\{ \begin{matrix} OP_{j,0\,..\,7} \\ OP_{j,8\,..\,15} \end{matrix} \right\} OP_{i,8\,..\,15}$		

SPB Dieser Befehl speichert das rechte Byte des vom F1-Feld adressierten Standardregisters in ein Byte einer Speicherzelle ab. Das nicht angesprochene Byte des Speicheroperanden bleibt unverändert. Die Adressierung des zu verändernden Bytes im Speicher erfolgt über ein durch das F2-Feld adressiertes Standardregisterpaar. Das geradzahlige Register gibt die Wortadresse der Speicherzelle an. Das höchstwertige Bit im im folgenden ungeradzahligen Register kennzeichnet das Byte. Die übrigen Bits dieses Registers sind irrelevant und bleiben unverändert.

Bei jedem Durchlaufen des Befehls wird über die 17-bit-lange Byte-Adresse inkrementiert.



6.4.1.3 Addition, Subtraktion

Befehl		Formate	Beschreibung	Anzeige	Bemerkung
ADF	Addieren Festpunkt Wort	RC, RR, RA, RAI, RAX	$OP_i := OP_i + OP_j$ UE: = Übertrag aus Bit	F 0123	-
AUF	Addieren mit Übertrag Festpunkt Wort	RR	$OP_i := OP_i + OP_j$ UE: = Übertrag aus Bit 0	F 0123	-
ADB	Addieren Betrag Wort	RC, RR, RA, RAI, RAX	$OP_i := OP_i + OP_j$ UE: = Übertrag aus Bit 0	B 023	-
AUB	Addieren mit Ubertrag Betrag Wort	RC, RR	$OP_i := OP_i + OP_j + UE$ UE: = Übertrag aus Bit 0	B 023	-
ADD	Addieren Festpunkt Doppelwort	RC, RR, RA, RAI, RAX	$OP_i := OP_i + OP_j$ UE: = Übertrag aus Bit 0	F 0123	-
ADG	Addieren Gleitpunkt Doppelwort	RR, RA, RAI, RAX	$OP_{i,8...31} : OP_{i,8...31} + OP_{j,8..31}$ $OP_{i,0...7} := \max. \{OP_{i,0..7}, OP_{j,0...7}\}$	G 0123	GPP
ADK	Addieren Gleitpunkt Vierfachwort	RR, RA, RAI, RAX	$OP_{i,8..63} : OP_{i,8...31} + OP_{j,8..63}$ $OP_{i,0...7} := \max. \{OP_{i,0...7}, OP_{j,0..7}\}$	G 0123	GPP
SBF	Subtrahieren Festpunkt Wort	RC, RR, RA, RAI, RAX	$OP_i := OP_i + \overline{OP_j} + 1$ UE : = Übertrag aus Bit 0	F 0123	-
SUF	Subtrahieren mit Übertrag, Festpunkt Wort	RR	$OP_i := OP_i + \overline{OP_j} + 1$ UE : = Übertrag aus Bit 0	F 0123	-
SBB	Subtrahieren Betrag Wort	RC, RR, RA, RAI, RAX	$OP_i := OP_i + \overline{OP_j} + 1$ UE : = Übertrag aus Bit 0	B 023	-
SUB	Subtrahieren mit Ubertrag, Betrag Wort	RC, RR	$OP_i := OP_i + \overline{OP_j} + 1$ UE : = Übertrag aus Bit 0	B 023	-
SBG	Subtrahieren Gleitpunkt Doppelwort	RR, RA, RAI, RAX	$OP_{i,8...31} : OP_{i,8..31} - OP_{j,8..31}$ $OP_{i,0...7} := \max. \{OP_{i,0..7}, OP_{j,0..7}\}$	G 0123	GPP
SBK	Subtrahieren Gleitpunkt Vierfachwort	RR, RA, RAI, RAX	$OP_{i,8...63} : OP_{i,8..31} - OP_{j,8..63}$ $OP_{i,0...7} := \max. \{OP_{i,0..7}, OP_{j,0..7}\}$	G 0123	GPP

Festpunktüberlauf tritt auf, wenn der Übertrag in die Vorzeichenstelle des Ergebnisses nicht mit dem Übertrag aus der Vorzeichenstelle übereinstimmt (Anzeige 3). Falls im Programmzustandsregister PZR das Bit 5 gesetzt ist, führt ein Festpunktüberlauf zu einer Programmlaufbesonderheit.

Betragsüberlauf tritt auf bei Bereichsüberschreitung (Addition) oder Bereichsunterschreitung (Subtraktion) (Anzeige 3).
Falls im Programmzustandsregister PZR das Bit 4 gesetzt ist, führt ein Betragsüberlauf zu einer Programmlaufbesonderheit.

Mit Übertrag: Der Anfangsübertrag ist der ursprüngliche Inhalt des Übertrag-Speichers (Bit 2 im Programmzustandsregister PZR) er hat die gleiche Wertigkeit wie das Bit 15 des Operanden. Ein etwaiger Übertrag aus Stelle 0 des Ergebnisses setzt den Übertragsspeicher erneut auf 1.

Gleitpunkt: Die Operanden können in nicht normalisierter Form vorliegen. Die Ergebnis-Mantisse ist nach der Operation normalisiert. Der Ergebnis-Exponent ist gleich dem größeren der beiden Operanden-Exponenten unter Berücksichtigung einer eventuellen Rechts- oder Linksverschiebung der Ergebnis-Mantisse. Bei auftretenden Ergebnisbesonderheiten wird das Ergebnis verändert und eventuell eine Programmunterbrechung durch einen Eintrag in das Unterbrechungsanzeigenregister UAR eingeleitet.
Die Tabelle zeigt die Zusammenhänge.

Besonderheit	UAR-Eintrag	Ergebnis		PZR 0, 1
	Bit 5	Mantisse	Exponent	Anzeige
Mantisse = 0	nein	0	-128	0
Exponent < -128	nein	0	-128	0
Exponent > +127	ja	errechn.	errechn.	3
Divisor = 0	ja	undef.	undef.	3

6.4.1.4 Multiplikation, Division

Befehl		Formate	Beschreibung	Anzeige	Bemerkung
MLF	Multiplizieren Festpunkt Wort	RC, RR, RA, RAI, RAX	i = ungerade: $OP_{i-1}, OP_i := OP_i \times OP_j$ i gerade: $OP_i, OP_{i+1} := OP_i \times OP_j$	F 012	-
MLB	Multiplizieren Betrag Wort	RC, RR, RA, RAI, RAX	i = ungerade: $OP_{i-1}, OP_i := OP_i \times OP_j$ i gerade: $OP_i, OP_{i+1} := OP_i \times OP_j$	B 02	-
MLD	Multiplizieren Festpunkt Doppelwort	RC, RR, RA, RAI, RAX	$OP_i := OP_i \times OP_j$	F 012	GPP
MLG	Multiplizieren Gleitpunkt Doppelwort	RR, RA, RAI, RAX	$OP_{i,8\ 31} := OP_{i,8\ 31} \times OP_{j,8\ 31}$ $OP_{i,0\ 7} := OP_{i,0\ 7} + OP_{j,0\ 7}$	G 0123	GPP
MLK	Multiplizieren Gleitpunkt Vierfachwort	RR, RA RAI, RAX	$OP_{i,8\ 63} := OP_{i,8\ 63} \times OP_{j,8\ 63}$ $OP_{i,0\ 7} := OP_{i,0\ 7} + OP_{j,0\ 7}$	G 0123	GPP
DVF	Dividieren Festpunkt Wort	RC, RR, RA RAI, RAX	i ungerade: $OP_{i-1}, OP_i = OP_i / OP_j$ $OP_{i-1} := Rest\ OP_i = Quotient$ i gerade: $OP_i, OP_{i+1} := OP_i, OP_{i+1}/OP_j$ $OP_i := Rest, OP_{i+1} = Quotient$	F 0123	-
DVB	Dividieren Betrag Wort	RC, RR, RA RAI, RAX	i ungerade. $OP_{i-1}, OP_i = OP_i / OP_j$ $OP_{i-1} := Rest\ OP_i := Quotient$ i gerade $OP_i, OP_{i+1} := OP_i, OP_{i+1}/OP_j$ $OP_i := Rest, OP_{i+1} = Quotient$	B 023	-
DVD	Dividieren Festpunkt Doppelwort	RC, RR, RA, RAI, RAX	$OP_i := OP_i / OP_j$ Angaben uber Dividendenlange siehe Text	F 0123	GPP
DVG	Dividieren Gleitpunkt Doppelwort	RR, RA, RAI, RAX	$OP_{i,8\ 31} := OP_{i,8\ 31} \times OP_{j,8\ 31}$ $OP_{i,0\ 7} := OP_{i,0\ 7} - OP_{j,0\ 7}$	G 0123	GPP
DVK	Dividieren Gleitpunkt Vierfachwort	RR, RA, RAI, RAX	$OP_{i,8\ 63} := OP_{i,8\ 63} / OP_{j,8\ 63}$ $OP_{i,0\ 7} := OP_{i,0\ 7} - OP_{j,0\ 7}$	G 0123	GPP

MLF Das Ergebnis hat 32 Stellen und wird rechtsbündig in zwei Registern abgespeichert. Die höherwertigen Bits des Produkts werden immer in einem Register mit geradzahliger Adresse abgelegt. Die Stelle 0 dieses Registers ist immer gleich der Stelle 1.
Ausnahme: Multiplikation zweier negativer Kleinstzahlen.

MLR wie MLF, nur werden die Operanden als vorzeichenlose Dualzahlen aufgefaßt.

MLD Das Ergebnis ist 64 bit lang und wird in einem Registerquartett abgespeichert.
Die Stellen 0 und 1 des Ergebnisses sind gleich; Ausnahme ist die Multiplikation zweier negativer Kleinstzahlen.

MLG Die Operanden müssen normalisiert sein, wenn ein norma-
MLK lisiertes Ergebnis gewünscht wird; nicht normalisierte Operanden ergeben ein zwar richtiges, aber ein nicht normalisiertes Ergebnis. Die Ergebnis-Mantisse hat die gleiche Länge wie die Operanden-Mantissen. Der Ergebnis-Exponent ist gleich der Summe der Operanden-Exponenten unter Berücksichtigung einer eventuellen Verschiebung der Ergebnis-Mantisse (Teilnormalisierung).

DVF Bei geradzahliger Adresse des Dividenden-Registers wird der Dividend als 32-stellige, vorzeichenbehaftete Dualzahl aufgefaßt.

Bei ungeradzahliger Adresse wird der Dividend als 16-stellige vorzeichenbehaftete Dualzahl aufgefaßt. In diesem Fall ergänzt das Rechenwerk den Dividenden nach links mit dem Vorzeichen auf 32 bit Länge.

Eine Korrektur des Ergebnisses oder des Restes ist nicht nötig.

Das Vorzeichen des Restes ist stets gleich dem des Dividenden.

Anzeige 3 (Divisionsfehler) tritt auf, wenn

$|\text{Dividend}| > (2^{15} - 1) \times |\text{Divisor}|$ oder wenn der Divisor 0 ist.

Anzeige 3 führt zu einer Programmunterbrechung (nicht maskierbar).

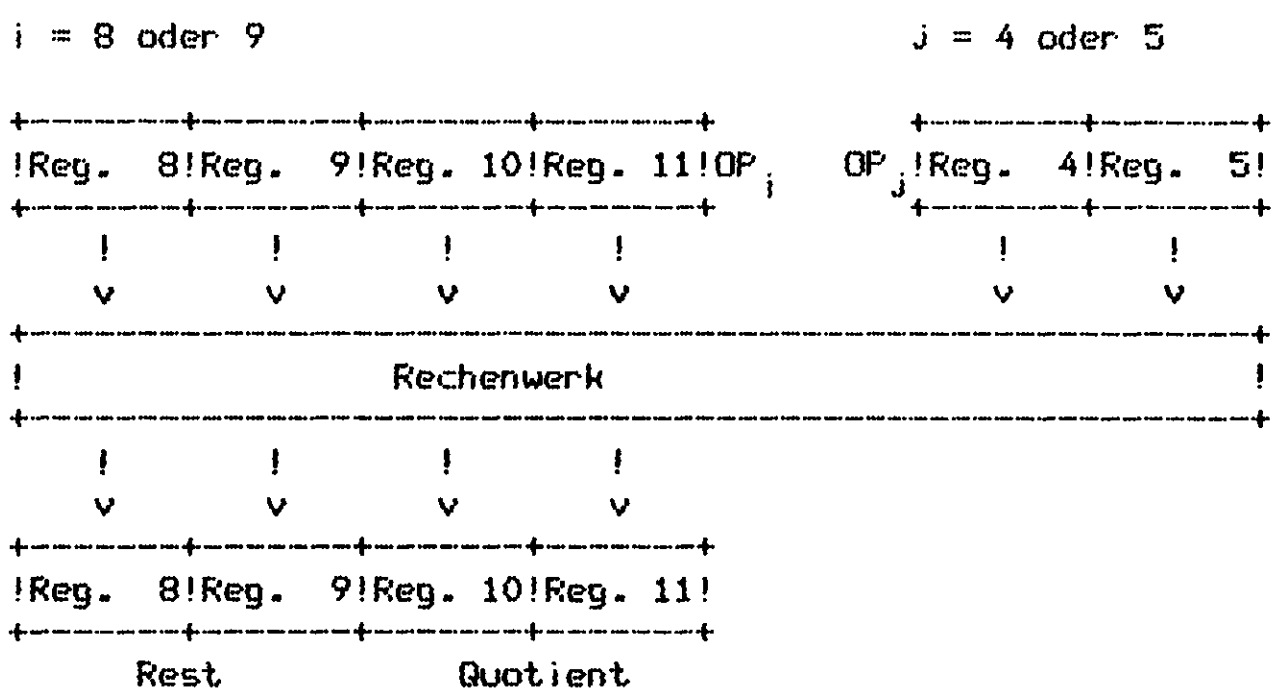
DVR Die Operanden werden als vorzeichenlose ganze Dualzahlen aufgefaßt. Bei ungeradzahliger Adresse des Dividenenregisters ist der Dividend 16 bit lang; er wird vom Rechenwerk nach links mit Nullen auf 32 bit ergänzt.

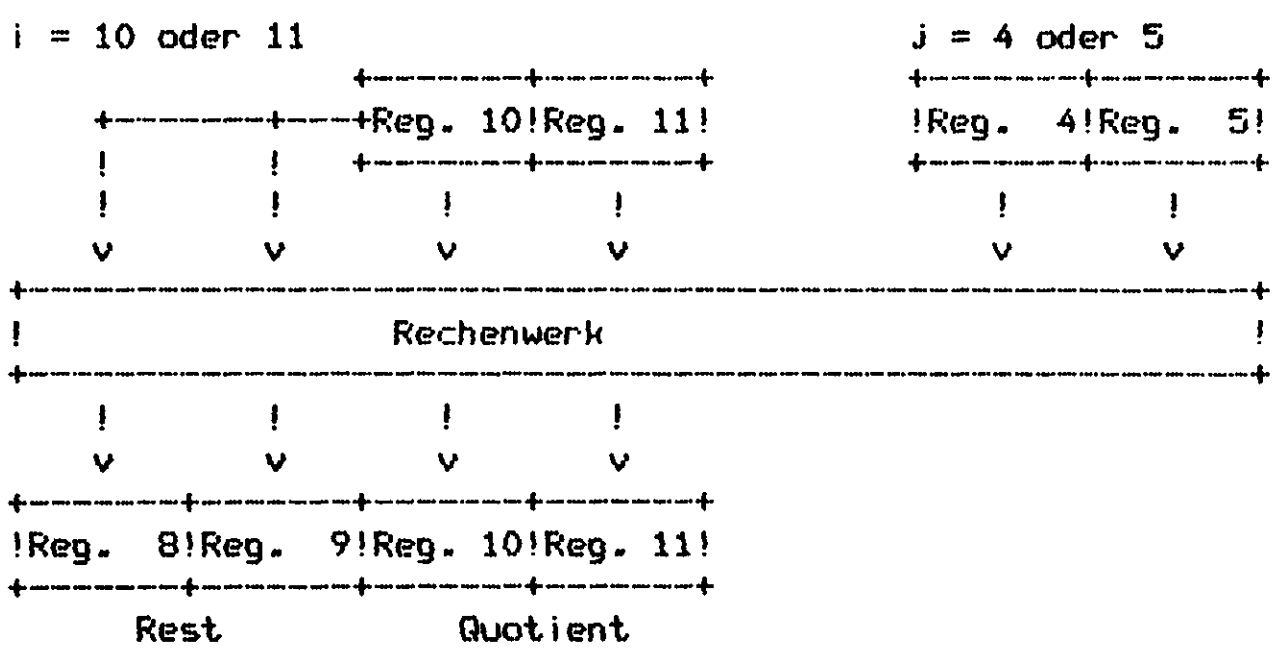
Ablauf wie DVF
Anzeige 3 (Divisionfehler) tritt auf, wenn

$|\text{Dividend}| > (2^{16} - 1) \times |\text{Divisor}|$ oder wenn der Divisor 0 ist.

Anzeige 3 führt zu einer Programmunterbrechung (nicht maskierbar).

DVD Der Dividend ist eine 64-bit- oder 32-bit-lange vorzeichenbehaftete Dualzahl, je nach dem, welche Registernummer im F1-Feld angegeben ist.
Das Rechenwerk ergänzt immer auf 64-bit-lange Dividenden. Ein Beispiel für die Angabe verschiedener Dividendenlängen ist im folgenden Bild dargestellt. Ablauf wie DVF.





Anzeige 3 (Divisionsfehler) tritt auf, wenn

$|\text{Dividend}| > (2^{32} - 1) \times |\text{Divisor}|$ oder wenn der Divisor 0 ist.

Anzeige 3 ist nicht maskierbar und führt zur Programmunterbrechung.

DVG Die Operanden müssen normalisiert sein, da eventuell mehrfach
DVK auftretende Überläufe nicht beseitigt werden und dadurch der Quotient falsch wird. Eine Restgröße wird nicht abgeliefert. Der Ergebnis-Exponent ist gleich der Differenz der Operanden-Exponenten unter Berücksichtigung einer evtl. Verschiebung der Ergebnis-Mantisse (Teilnormalisierung).

6.4.1.5 Vergleichsbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
VGF	Vergleichen Festpunkt Wort	RC, RR, RA, RAI, RAX	OP_i . VG OP_j	V 012	-
VGB	Vergleichen Betrag Wort	RC, RR, RA, RAI, RAX	OP_i . VG OP_j	V 012	-
VLB	Vergleichen Betrag linkes Byte	RR, RA, RAI, RAX	$OP_{i,0-7}$ VG $OP_{j,0-7}$	V 012	-
VRB	Vergleichen Betrag rechtes Byte	RR, RA, RAI, RAX	$OP_{i,8-15}$. VG $OP_{j,8-15}$	V 012	-
VEB	Vergleichen Betrag Byte	RAI	$OP_{i,8-15}$. VG $\begin{Bmatrix} OP_{j,0-7} \\ OP_{j,8-15} \end{Bmatrix}$	V 012	-
VBY	Vergleichen Betrag beide Bytes	RR, RA, RAI, RAX	OP_i . VG OP_j	3) 0123	-
VMS	Vergleichen mit der Maske	RC, RR, RA, RAI, RAX	OP_i . VG OP_j OP_j = Maske	2) 012	-
VMR	Vergleichen mit Maske im Register	RA, RAI, RAX	OP_i . VG OP_j OP_j = Maske	2) 012	-
VGD	Vergleichen Festpunkt Doppelwort	RC, RR, RA RAI, RAX	OP_i . VG OP_j	V 012	-
VGG	Vergleichen Gleitpunkt Doppelwort	RR, RA RAI, RAX	OP_i . VG OP_j	V 012	GPP
VGK	Vergleichen Gleitpunkt Vierfachwort	RR, RA RAI, RAX	OP_i . VG OP_j	V 012	GPP
PGF	Prüfen auf Grenzen Festpunkt	RAX	OP_i . VG OP_j OP_j = Grenzwert	7) 012	-
PGB	Prüfen auf Grenzen Betrag	RAX	OP_i . VG OP_j OP_j = Grenzwerte	7) 012	-

Die Operanden werden durch die Befehle nicht verändert.

VEB Ein Byte eines Speicheroperanden wird mit dem rechten Byte des im F1-Feld adressierten Registers betragsmäßig verglichen; das linke Byte des Registers ist nach Befehlsende undefiniert. Die Adressierung des Byte im Speicher und die Formatbearbeitung erfolgt wie bei "SPB RAI", jedoch muß im F2-Feld des "VEB" immer das gerade Register stehen.

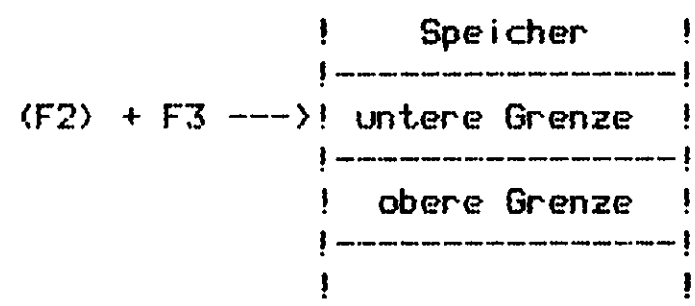
VBY Es werden jeweils Bytes verarbeitet. Anzeigeninterpretation.
Anz. 0: Die linken und die rechten Bytes der Operanden sind gleich.
Anz. 1: Die linken Bytes sind gleich.
Anz. 2: Die rechten Bytes sind gleich.
Anz. 3: Keine gleichen Bytes.

VMS Die Stellen des Operanden i werden mit einer Maske OP_j getestet. Anzeigeninterpretation:
Anz. 0: Alle getesteten Bits oder alle Maskenbits sind Null.
Anz. 1: Die getesteten Stellen des Operanden enthalten Einsen und Nullen.
Anz. 2: Alle getesteten Bits haben den Wert "1".

VMR Die Stellen des Operanden Op_j werden mit einer Maske OP; getestet. Anzeigeninterpretation wie VMS.

VGG Ablauf wie Gleitpunktsubtraktion ohne Abspeichern des
VGK Ergebnisses. Die Anzeige 0 ($OP_i = OP_j$) wird nur dann abgeliefert, wenn bei der Subtraktion die Mantisse zu Null wird; also nicht, wenn nur ein Exponent (-128 entsteht.

PGF OP_i wird auf Unter- oder Überschreitung eines Bereichs geprüft. Die beiden Grenzen stehen im Speicher.



Vorausgesetzt wird: untere Grenze (obere Grenze. Wird eine Grenzüberschreitung festgestellt, erfolgt ein Eintrag ins Unterbrechungsanzeigenregister UAR, falls im Programmzustandsregister PZR Bit 5 gesetzt ist.

Anzeigen:

0	Operand innerhalb bzw. auf Grenzen
1	Operand (untere Grenze
2	Operand) obere Grenze

PGR wie PGF, jedoch wird ein Eintrag ins Unterbrechungsanzeigenregister UAR vom Programmzustandsregister PZR Bit 4 freigegeben.

6.4.1.6 Bool'sche Befehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
UND	UND	RC, RR, RA, RAI, RAX	OP_i : = OP_i . U OP_j	L 02	-
ODR	ODER	RC, RR, RA, RAI, RAX	OP_i : = OP_i . O OP_j	L 02	-
XCR	Exklusiv ODER (Antivalenz)	RC, RR, RA, RAI, RAX	OP_i : = OP_i . X OP_j	L 02	-

6.4.1.7 Bit-Testbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
BTT	Bit testen	RR, RA, RAI, RAX, CR, CA, CAI, CAX	OP_i = k = Bitnummer $OP_{j,k}$ wird getestet	T 02	-
BTS	Bit testen und setzen	RR, RA, RAI, RAX, CR, CA, CAI, CAX	OP_i = k $OP_{j,k}$ wird getestet $OP_{j,k}$: = 1	T 02	-
BTL	Bit testen und löschen	RR, RA, RAI, RAX, CR, CA, CAI, CAX	OP_i = k $OP_{j,k}$ wird getestet $OP_{j,k}$: = 0	T 02	-
BTZ	Bit suchen	RR, RA, RAI, RAX	OP_j wird getestet OP_i = BNR	T 02	-

BTT Nach Maßgabe von OP_i = Bitnummer wird ein Bit von OP_j getestet und die Anzeigen werden entsprechend gesetzt.
C-Format: Die Bitnummer ist im Feld F1 angegeben.
R-Format: Die Bitnummer steht in dem durch F1 adressierten Register, Bitnummer > 15 werden modulo 16 interpretiert.

BTS Nach dem Testen (wie BTT) wird das Bit auf 1 gesetzt.
Bei A-Format Speicherschutz beachten!

BTL Nach dem Testen (wie BTT) wird das Bit auf 0 gesetzt.
Bei A-Format Speicherschutz beachten!

BTZ OP_j wird nach der ersten "1" von links her getestet. Die Bitnummer (BNR) der ersten "1" wird im Standardregister i abgespeichert.

BNR = 0 wird abgespeichert, wenn die erste Stelle (Bit 0) von OP_j gesetzt war oder wenn kein Bit in OP_j gesetzt war.

Anzeige 0 wird gesetzt, wenn OP_j nur Nullen enthält, Anzeige 2, wenn mind. eine "1" vorhanden ist.

6.4.1.8 Schiebebefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
SLF	Schieben links Festpunkt Wort	RC	$OP_i := OP_i . V + OP_j$ $UE := OP_{i\,(OP_j - 1)}$	F 0123	-
SRF	Schieben rechts Festpunkt Wort	RC	$OP_i . = OP_i . V- OP_j$ $UE := OP_{i\,(16-OP_j)}$	F 012	-
SHF	Schieben Festpunkt Wort	RR	$OP_i := OP_i . V \pm OP_j$ $V - UE \cdot = OP_{i,(16-OP_j)}$ $V + UE: = OP_{i,(OP_j-1)}$	F 012 0123	-
SLB	Schieben links Betrag Wort	RC	$OP_i . = OP_i . V + OP_j$ $UE: = OP_{i,(OP_j-1)}$	B 023	-
SLU	Schieben links m Ubertr Betrag Wort	RC	$OP_i := OP_i . V + OP_j$ $OP_{i,(16-OP_j)} \cdot = UE$ $UE: = OP_{i,(OP_j-1)}$	B 023	-
SRB	Schieben rechts Betrag Wort	RC	$OP_i := OP_i . V- OP_j \cdot$ $UE . = OP_{i\,(16-OP_j)}$	B 02	-
SRU	Schieben rechts mit Ubertrag Betrag	RC	$OP_i . = OP_i . V- OP_j$ $OP_{i,(OP_j-1)} := UE$ $UE = OP_{i\,(16-OP_j)}$	B 02	-
SHB	Schieben Betrag Wort	RR	$OP_i := OP_i . V \pm OP_j$ $V - UE \cdot = OP_{i,(16-OP_j)}$ $V + UE: = OP_{i,(OP_j-1)}$	B 02 023	-
SHP	Schieben rechts mit Paritatsermittlung Betrag Wort	RC	$OP_i \cdot V - OP_j$ $UE \cdot = OP_{i,(16-OP_j)}$	1) 02	-
SDL	Schieben links Betrag Doppelwort	RC	$OP_i := OP_i \cdot V + OP_j$ $UE: = OP_{i,(OP_j-1)}$	B 023	-
SDR	Schieben rechts Betrag Doppelwort	RC	$OP_i . = OP_i . V- OP_j$ $UE := OP_{i,(32-OP_j)}$	B 02	-
SDB	Schieben Betrag Doppelwort	RR	$OP_i . = OP_i . V \pm OP_j$ $V - UE = OP_{i,(32-OP_j)}$ $V + UE: = OP_{i,(OP_j-1)}$	B 02 023	-
SHD	Schieben Festpunkt Doppelwort	RR	$OP_i := OP_i . V \pm OP_j$ $V - UE := OP_{i,(32-OP_j)}$ $V + UE: = OP_{i,(OP_j-1)}$	F 012 0123	-
SLD	Schieben links Festpunkt Doppelwort	RC	$OP_i . = OP_i . V + OP_j$ $UE: = OP_{i,(OP_j-1)}$	F 0123	-
SRD	Schieben rechts Festpunkt Doppelwort	RC	$OP_i := OP_i . V- OP_j$ $UE := OP_{i,(32-OP_j)}$	F 012	-

Ablauf der Schiebebefehle

Die zu schiebenden Operanden werden als Festpunktzahlen bzw. vorzeichenlose Betragszahlen aufgefaßt. Das kann beim Links-Schieben zu Überläufen führen (bei überschreiten des Zahlenbereichs). Es werden also keine Bitmuster geschoben!
Beim Rechts-Schieben kann nie ein Überlauf auftreten, da der Betrag des Operanden gegen Null geht.

Bei den Befehlen SDL, SDR, SDB wird bei ungeradzahliger Registernummer im F1-Feld an Stelle des geradzahligen ebenfalls das ungeradzahlige Register bei der Registerpaarbildung benutzt.

Schieben Festpunkt

Links Schieben

In die Stelle niedrigster Wertigkeit werden Nullen nachgezogen. Über die Stelle 0 hinausgeschobene Stellen gehen verloren. Die letzte hinausgeschobene Stelle gelangt in den Überlaufspeicher (Programmzustandsregister PZR Bit 2).
Ändert sich während des Schiebens der ursprüngliche Wert der Stelle 0 (Vorzeichen), so wird Anzeige 3 (Überlauf) gesetzt.

Rechts Schieben

In die Stelle 0 wird das Vorzeichen nachgezogen, d.h. das Vorzeichen bleibt unverändert. Über die Stelle niedrigster Wertigkeit hinausgeschobene Stellen gehen verloren. Die letzte hinausgeschobene Stelle gelangt in das den Überlaufspeicher (Programmzustandsregister PZR Bit 2). Wurden bei negativen Operanden OP_i Einsen über die Stelle niedrigster Wertigkeit hinausgeschoben, so wird $OP_i + 1$ abgespeichert.

Schieben Betrag

Links Schieben

In die Stelle niedrigster Wertigkeit werden Nullen nachgezogen. Über die Stelle 0 hinausgeschobene Stellen gehen verloren. Die letzte hinausgeschobene Stelle gelangt in den Überlaufspeicher (Programmzustandsregister PZR Bit 2).

Wird während des Schiebens eine Eins über die Stelle 0 hinausgeschoben, so wird Anzeige 3 (Überlauf) gesetzt.

Rechts Schieben

In die Stelle 0 werden Nullen nachgezogen. Über die Stelle niedrigster Wertigkeit geschobene Stellen gehen verloren. Die letzte hinausgeschobene Stelle gelangt in den Überlaufspeicher (Programmzustandsregister PZR Bit 2).

Schieben im RR-Format

Die Schieberichtung richtet sich nach dem Vorzeichen von OP_j.
$OP_{j,0}$ = 0 Linksverschiebung
$OP_{j,0}$ = 1 Rechtsverschiebung
OP_j ist eine 16-stellige Festpunktzahl. Die maximal ausgeführte Schiebezahl ist bei 16-bit-Operanden 17, bei 32-bit-Operanden 33. In diesen Fällen ist OP_i gelöscht, und der Übertragsspeicher (Programmzustandsregister PZR Bit 2) ist mit dem Vorzeichen (Festpunkt) bzw. mit Null (Betrag) geladen.
Anzeige 3 (Überlauf) ist nur bei Linksverschiebung möglich.
Der Ablauf der Befehle entspricht den unter "Schieben Festpunkt" und "Schieben Betrag" beschriebenen Abläufen.

Schieben mit Übertrag

Der Ablauf entspricht dem unter "Schieben Betrag" beschriebenen Abläufen; jedoch wird beim ersten Schiebeschritt der Inhalt des Übertragsspeichers (Programmzustandsregister PZR Bit 2) nachgezogen.

Schieben mit Paritätsermittlung SHP

OP_i wird nach Maßgabe von OP_j im Rechenwerk nach rechts verschoben, OP_i im Standardregister bleibt dabei jedoch unverändert. Die letzte hinausgeschobene Stelle gelangt in den Übertragsspeicher (Programmzustandsregister PZR Bit 2).
Entsprechend der über die Stelle niedrigster Wertigkeit hinausgeschobenen Einsen wird die Anzeige gesetzt:
Anzeige 0: Die Anzahl der hinausgeschobenen Einsen ist ungerade.
Anzeige 2: Die Anzahl der hinausgeschobenen Einsen ist gerade.

6.4.1.9 Sprungbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
SPR	Springen bedingt	CR, CA, CAI, CDA, CAX, CRX	Wenn Bedingung erfullt: R1 : OP_j	-	-
DSP	Dekrementieren und springen	RR, RRX	OP_i : = OP_i - 1 Falls dann $OP_i \neq 0$ R 1 : = OP_j	-	-
AVS	Addieren und springen	RRX	OP_i : = OP_i + OP_j Falls dann $OP_i \leq OP_{j+1}$ R 1 : = F 3	-	-
USP	Unterprogrammsprung	RR, RAX, RRX	OP_i : = R 1 dann R 1 : = OP_j	-	-
USK	Unterprogrammsprung mit Kellern	RAI, RXAI, RXDA	OP_j : = R 1 dann R 1 : = OP_i	-	-
SKV	Springen bedingt vorwarts	CC	Wenn Bedingung erfullt ist: R 1 : = R 1 - 1 + OP_j	-	-
SKR	Springen bedingt ruckwarts	CC	Wenn Bedingung erfüllt ist: R 1 : = R 1 - 1 - OP_j	-	-
UCK	Unterprogrammsprung in anderen Codeadreß-raum	DAR	OP_i : = R 1 OP_{i-1} : = TZR 3 TZR 3 : = OP_j R 1 : = OP_{j+1}	-	-
SPC	Rucksprung aus anderem Codeadreß-raum	CAI	Wenn Bedingung erfüllt ist: TZR 3 : = OP_j R 1 : = OP_{j+1}	-	-

SPR Der Wert des Anzeigenregisters wird mit den vier Bits der Maske OP_i verglichen entsprechend der logischen UND-Verknüpfung.

Anzeige	Anzeige im PZR Bit 0 1	PZR Bit 0 1 codiert in (4) (1)
0	0 0	1 0 0 0
1	0 1	0 1 0 0
2	1 0	0 0 1 0
3	1 1	0 0 0 1

PZR ... Programmzustandsregister

Die Sprungbedingung ist erfüllt, wenn der Vergleich eine "1" ergibt. Sind alle Maskenbits 0, so entspricht dies einer Nulloperation. Sind alle Maskenbits "1", dann ergibt sich ein unbedingter Sprung.

Beispiel: Maske OP_i = 0110 besagt, daß bei Anzeige 1 oder 2 die Sprungbedingung erfüllt ist.

Bei CAI- und CDA-Format wird immer inkrementiert bzw. dekrementiert, also unabhängig von erfüllter oder nicht erfüllter Sprungbedingung.

DSP Ist bei Befehlsbeginn OP_i = 0, so erfolgt ein Sprung. Die zu dekrementierende Zahl (OP_i) wird als 16-bit-lange Betragszahl aufgefaßt.

AVS Im F2-Feld wird das gerade Register eines Registerpaars angegeben. Die Schrittweite ist im geraden Register enthalten. Op_i wird um die Schrittweite erhöht und betragsmäßig mit dem ungeraden Register verglichen, das den Endwert enthält. Im F3-Feld steht die Sprungadresse. Sobald $OP_i + OP_j > OP_{j+1}$ oder auch > FFFF wird, erfolgt kein Sprung.

USP Die Rücksprungadresse wird unter OP_i abgespeichert.

USK Die Rücksprungadresse wird im Arbeitsspeicher abgelegt. Die Sprungziel-Adresse steht entweder im Standardregister i (RAI), oder wird aus der Summe aus Inhalt des Standardregisters i plus F3-Feld (RXAI, RXDA) gebildet. Wenn im F1-Feld Standardregisters 0 angegeben ist, ist die Sprungziel-Adresse im F3-Feld allein angegeben.

SKV
SKR Bei diesen bedingten Sprungbefehlen im Format CC wird nicht wie beim Befehl SPR ein Sprungziel, sondern eine Sprungweite angegeben. Die Anwendung der Sprungbedingung ist die gleiche wie bei SPR.

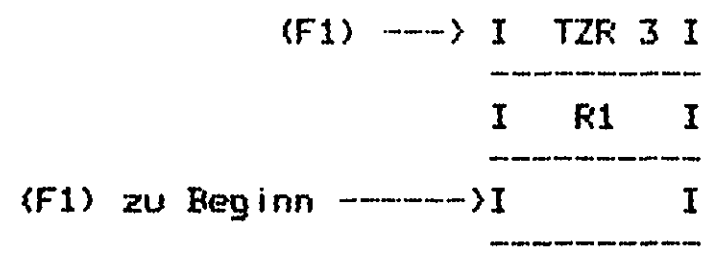
SKV (Springe kurz vorwärts):
Das F2-Feld wird zur Befehlsadresse (unverändertes R1) addiert; die Summe bildet das Sprungziel.

SKR (Springe kurz rückwärts):
Das F2-Feld wird von der Befehlsadresse (unverändertes R1) subtrahiert; die Differenz bildet das Sprungziel.

Durch diese Kurzformen der Sprungbefehle ist es möglich, über maximal 15 Zellen des Zentralspeichers zu springen.

Achtung: SKV oder SKR mit F2 = 0 springt auf sich selbst.

UCK Bei diesem Unterprogrammsprung wird im F2-Feld das gerade Register eines Registerpaars angegeben. Neben dem Sprungziel im ungeraden Register wird auch ein neues Tafelzeigeregister 3 (TZR 3) im geraden Register festgelegt. Da in einem vorher zu definierenden Kellerspeicher ("Stack", für diesen Zweck vom Anwender freizuhaltenden Bereich im Zentralspeicher) das alte Tafelzeigeregister 3 und R1 hinterlegt werden, wird das die Kelleradresse (Adresse des Kellerspeichers) enthaltende R_i zweimal dekrementiert.



Ist der Inhalt des neuen Tafelzeigeregisters 3 (TZR 3) Null, kommt es zur Programmlaufbesonderheit "Adressierungsfehler" (Unterbrechungsanzeigenregister UAR-Bit 15).

SPC Die Anwendung der Sprungbedingungen ist die gleiche wie bei SPR. Das Register im F2-Feld enthält die Kelleradresse (analog zum Befehl UCK), sie zeigt auf das neue Tafelzeigeregister 3 (TZR 3) und ist am Befehlsende um zwei inkrementiert. Die vom Befehl UCK aufgebaute Organisation des Kellerspeichers wird vorausgesetzt.

6.4.1.10 Feldsuchbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
SFG	Feld durchsuchen auf gleiches Byte	ähnlich RAI	$OP_1 \cdot VG\ OP_j$ $R_j : R_j + 1$	6) 012	-
SFU	Feld durchsuchen auf ungleiches Byte	ähnlich RAI	$OP_1 \cdot VG\ OP_j$ $R_j : R_j + 1$	5) 012	-
BZF	Feld durchsuchen auf erstes gesetztes Bit	ähnlich RAI	$OP_j = 0 \rightarrow OP_1 := OP_1 + 16$ $OP_j = 0 \rightarrow OP_1 := OP_1 + BNR$ $R_j = R_j + 1$	4) 02	-

SFG Ein Feld von Wortoperanden im Zentralspeicher wird mit den breiten Maskenbytes im Register i verglichen. Die Anfangsadresse dieses Felds steht im Standardregister j; sie wird nach jedem Vergleich um 1 erhöht. Wird eine Gleichheit im rechten oder linken Byte festgestellt, dann wird der Befehl abgebrochen. Im Register j steht dann die um 1 erhöhte Adresse des Wortoperanden, bei dem Gleichheit festgestellt wurde.

Es werden folgende Anzeigen gesetzt:

Anz. 0: Beide Bytes der Operanden sind gleich.
Anz. 1: Die linken Bytes sind gleich.
Anz. 2: Die rechten Bytes sind gleich.

Werden keine gleichen Bytes gefunden, so kommt der Befehl durch Programmlaufbesonderheit zum Ende (Speicherfehler). Der Befehl ist unterbrechbar. Beim Zustandswechsel wird bei nicht beendetem Befehl "R1 minus 1" in die zugehörige Parametertafel abgespeichert, die Befehlsadresse also auf den unterbrochenen Befehl zurückgestellt, während OP_j auf den nächsten zu bearbeitenden Wortoperanden zeigt.

SFU Der Ablauf ist wie SFG mit dem Unterschied, daß das Wortfeld auf ungleiche Bytes durchsucht wird und bei Ungleichheit eines Bytes der Befehl beendet wird.

Anzeigen:

Anz. 1: Die rechten Bytes sind ungleich.
Anz. 2: Die linken Bytes sind ungleich.
Anz. 3: Beide Bytes sind ungleich.

F

BZ Mit diesem Befehl wird die relative Bitnummer der ersten gesetzten Stelle eines Operandenfelds im Speicher ermittelt und als Befehlsergebnis abgeliefert.

In dem durch das F2-Feld adressierten Standardregister steht die Anfangsadresse des zu durchsuchenden Felds; sie wird nach jeder Operation um 1 erhöht. Der Befehl wird beendet bei dem Wortoperanden, welcher ungleich 0 ist.
In Register i steht dann die Bitadresse.

Bei jedem getesteten Wort, welches nur Nullen enthält, wird der Inhalt des Registers i um 16 erhöht. Bei dem Wort, welches ungleich 0 ist, wird die Bitnummer (BNR) zum Inhalt des Registers i addiert.

Wird kein gesetztes Bit gefunden, so tritt entweder ein Übertrag aus der höchstwertigen Stelle von OP_i auf (ohne eine Programmunterbrechung) oder eine Programmlaufbesonderheit (z. B. durch Speicherfehler); der Befehl wird beendet. Der Befehl ist unterbrechbar (siehe SFG).

Es werden folgende Anzeigen gesetzt:

Anz. 2: Das Feld enthält eine gesetzte Stelle
Anz. 0: Das Feld enthält keine gesetzte Stelle
(bei Befehlsabbruch infolge Überlauf des Registers i).

Bei Befehlsabbruch infolge Überlauf des Registers i ist dieses Register gelöscht. Bei Befehlsbeginn werden die Bits 12 bis 15 von OP_i gelöscht.

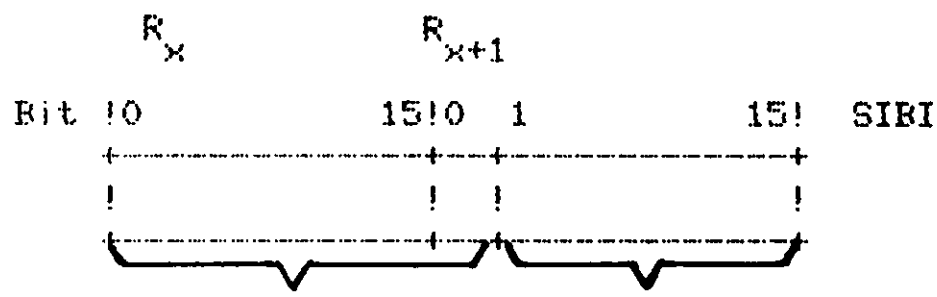
6.4.1.11 Byteadressierte Feldbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
TBF	Feld byteweise transferieren	AIAI	$OP_i = OP_j$ $R_i, R_{i+1,0} = R_i, R_{i+10} + 1$ $R_{i+11\ 15} = R_{i+11\ 15}\ 1$ $R_j, R_{j+10} = R_{j+10} + 1$ $R_{j+11\ 15} = R_{j-11\ 15} - 1$	-	-
VBF	Zwei Felder miteinander auf ungleiches Byte vergleichen	AIAI	OP_i VG OP_j $R_i, R_{i+10} = R_i, R_{i+10} + 1$ $R_{i+11\ 15} = R_{i+11\ 15} - 1$ $R_j, R_{j+10} = R_j R_{j\ 10} + 1$ $R_{j+11\ 15} = R_{j+11\ 15} - 1$	8) 012	-
SUZ	Feld byteweise auf ungleiches Byte durchsuchen	RAI	OP_i VG OP_j $R_j, R_{j+10} = R_j R_{j+10} + 1$ $R_{j+11\ 15} = R_{j-11\ 15}\ 1$	9) 123	-
SGZ	Feld byteweise auf gleiche Bytes durchsuchen	RAI	OP_i VG OP_j $R_j, R_{j+10} = R_j R_{j+10} + 1$ $R_{j+11\ 15} = R_{j-11\ 15} - 1$	10) 123	-
FMZ	Feld byteweise mit Byte füllen	RAI	$OP_i = OP_i$ $R_j, R_{j+10} = R_j R_{j+10} + 1$ $R_{j+11\ 15} = R_{j+11\ 15} - 1$	-	-

In einem byte-adressierenden Feldbefehl erfolgt die Beschreibung eines Felds durch Angabe einer Byte-Adresse und des zugehörigen Längenzählers. Die Byte-Adresse, die das erste zu bearbeitende Byte angibt, steht in einem Registerpaar.
Im geraden Register ist die Wortadresse hinterlegt, im höchstwertigen Bit des darauffolgenden ungeraden Registers die Kennzeichnung des Bytes:

0 = linkes Byte
1 = rechtes Byte

Die übrigen 15 Stellen des ungeraden Registers enthalten den Zähler, der die Länge des Felds in Bytes angibt. Im Befehl muß das gerade Register angegeben werden.



17-bit-Byte-Adresse 15-bit-Längenzähler

Nach jeder Byte-Bearbeitung wird die 17-bit-breite Byte-Adresse inkrementiert und der Längenzähler dekrementiert. Der Befehl wird spätestens beendet, wenn ein Längenzähler null geworden ist. Dann zeigt die Byte-Adresse auf das dem Feld folgende Byte. Bei einem Anfangszählerstand von Null werden Anzeigen entsprechend einer erfolglosen Zeichensuche gesetzt.

Die Feldbefehle sind an der Wort- oder Byte-Grenze unterbrechbar. Nach der Bearbeitung der Unterbrechungsanforderung wird der Befehl mit der Bearbeitung des/der folgenden Bytes auf die die Byte-Adresse weist, fortgesetzt. Für die Anzeigenbildung werden die Bytes betragsmäßig verglichen.

TBF Das durch OP_j beschriebene Feld wird in das durch OP_i beschriebene Feld umgespeichert. Falls sich die Speicherbereiche nicht überlappen, wird die durch OP_j beschriebene Zeichenfolge nicht verändert.

Wenn der Längenzähler geradzahlig ist und die beiden Felder mit linken Bytes beginnen, wird der Transfer mit einer Breite von 16 bit pro Umspeicherung vorgenommen. In allen anderen Fällen wird mit einer Breite von 8 bit umgespeichert. Im ersteren Fall läuft der Befehl wesentlich schneller ab.

VBF Das durch OP_j beschriebene Feld wird Byte für Byte mit dem durch OP_i beschriebenen Feld verglichen, ohne eine der Zeichenfolge zu verändern. Falls ein ungleiches Byte erkannt wird endet der Befehl, wobei die Byte-Adressen auf die beiden ungleichen Bytes zeigen und die Längenzähler entsprechend heruntergezählt sind.

Anzeigen:
0 beide Zeichenfolgen gleich oder Anfangswert des Längenzählers = 0.
1 erstes ungleiches Byte aus OP_i < als aus OP_j
2 erstes ungleiches Byte aus OP_i > als aus OP_j

SUZ Das durch den Operanden OP_j beschriebene Feld wird mit einem Kenn-Byte, dem rechten Byte des Registers R_i, auf Ungleichheit verglichen. Sobald ein ungleiches Byte erkannt wird endet der Befehl, wobei die Byte-Adresse auf das ungleiche Byte weist und der Längenzähler entsprechend erniedrigt ist. Das linke Byte von Register R_i ist nach Befehlsende undefiniert.

Anzeigen:
0 alle Bytes aus OP_j = Kenn-Byte oder der Anfangswert des Längenzählers = 0.
1 Kennbyte < als erstes ungleiches Byte aus OP_j
2 Kennbyte > als erstes ungleiches Byte aus OP_j.

SGZ Das durch den Operanden OP_j beschriebene Feld wird Byte für Byte mit zwei Kenn-Bytes, zuerst dem linken, dann dem rechten Byte des Registers R_i, auf Gleichheit verglichen. Sobald ein gleiches Byte gefunden wird, endet der Befehl. Byte-Adresse und Längenzähler bezeichnen dann das Byte, bei dem Gleichheit festgelegt wurde.

Anzeigen:
1 linkes Kenn-Byte = Byte aus OP_j
2 rechtes Kenn-Byte = Byte aus OP_j
3 alle Zeichen ungleich oder der Anfangswert des Längenzählers = 0

FMZ Das durch den Operanden OP_j beschriebene Feld wird mit einem Kenn-Byte, dem rechten Byte des Registers R_i, überschrieben. Das linke Byte des Registers R_i ist nach Befehlsende undefiniert.

6.4.1.12 Fädellistenbefehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
EHD	Einhangen davor	AA	Element OP_j vor Element OP_i einhängen	11) 02	-
AHE	Aushängen Element	A	Element OP_j aushängen	-	-
AHD	Aushängen danach	AA	Element nach OP_j aushängen OP_i : = ausgehängtes Element	11) 02	-

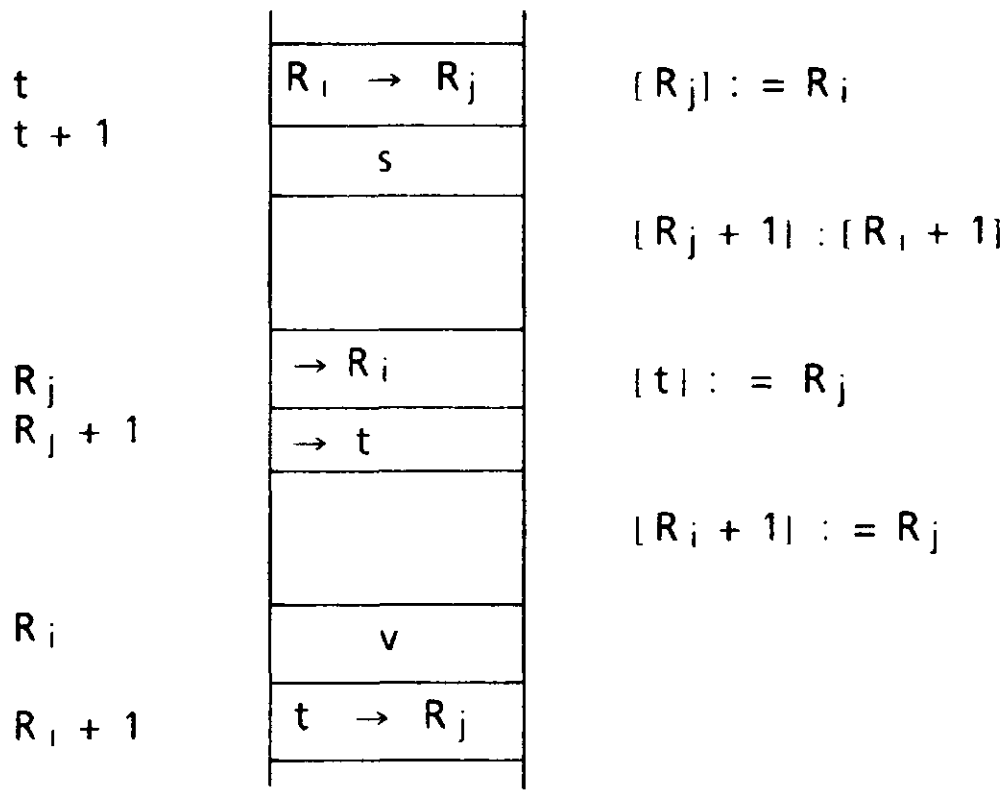
Eine Warteschlange ist wie folgt organisiert:

Adresse	Speicherzelle			
w	I	a	I	Warteschlangenkopf
w+1	I	z	I	
a	I	b	I	erstes Element
a+1	I	w	I	
	I		I	
	I		I	
	I		I	
b	I	c	I	zweites Element
b+1	I	a	I	
	I		I	
	I		I	
	I		I	
z	I	w	I	letztes Element
z+1	I	y	I	
	I		I	
	I		I	

Der Warteschlangenkopf enthält die Anfangsadresse des ersten und letzten Elements. Jedes Element enthält die Anfangsadresse des vorhergehenden und des folgenden Elements.

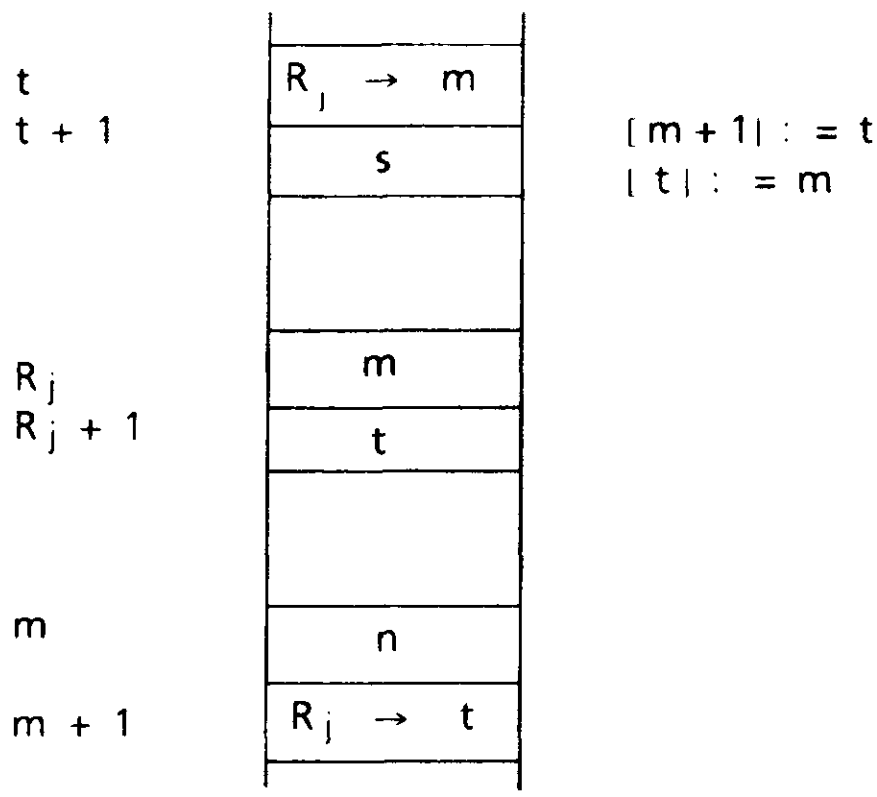
Adresse	Speicherzelle			
w	I	w	I	Kopf einer leeren Warteschlange
w+1	I	w	I	

EHD R_i enthält die Anfangsadresse des Elements, vor dem eingehängt werden soll. R_j enthält die Anfangsadresse des einzuhängenden Elements. Beide Adressen müssen im gleichen Adreßraum liegen.

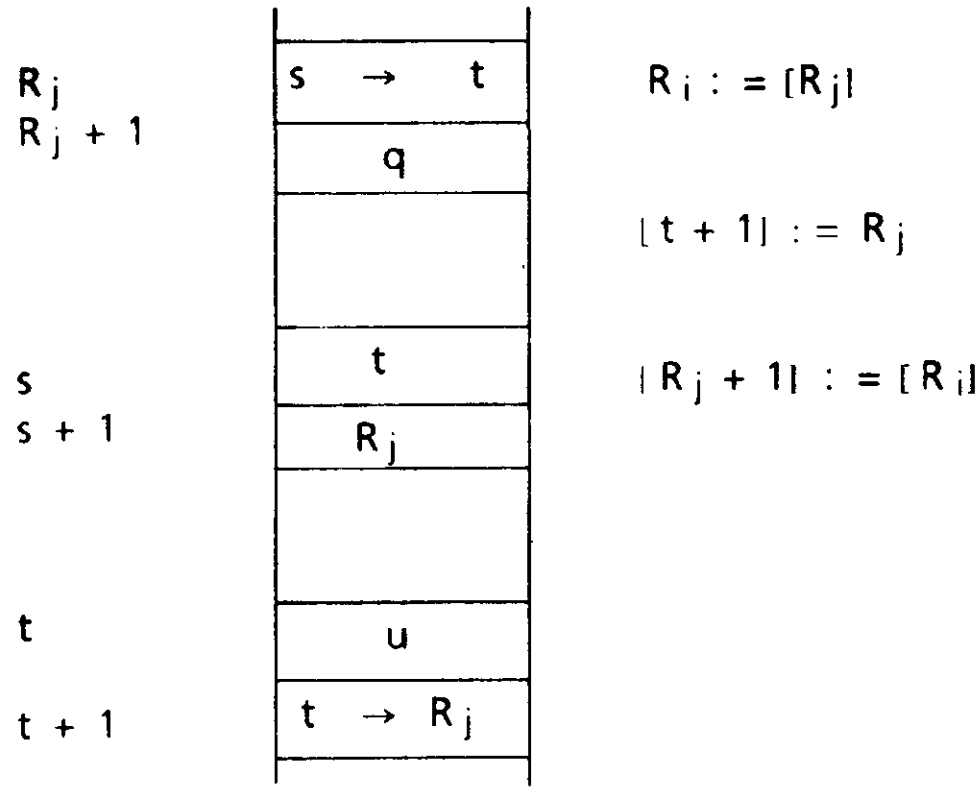


Anzeigen:
0 Warteschlange war leer
2 Warteschlange enthält ein oder mehrere Elemente

AHE R_j enthält die Anfangsadresse des auszuhängenden Elements.



AHD R_j enthält die Anfangsadresse des Elements vor dem auszuhängenden Element. In R_i wird die Anfangsadresse des ausgehängten Elements eingetragen.



Beim Versuch, aus einer leeren Warteschlange ein Element auszuhängen, wird der Warteschlangenkopf nicht verändert.

Anzeigen:
0 Warteschlange war leer
2 Warteschlange enthielt ein oder mehrere Elemente.

6.4.1.13 EA-Befehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
EAS	Schreiben EA-Bereich	AR, AAI	$OP_i := OP_j$	EA 0123	-
EAL	Lesen EA-Bereich	AR, AAI	$OP_j := OP_i$	EA 0123	-

Die EA-Adresse steht in dem durch das F1-Feld adressierten Standardregister. Quittiert die periphere Einheit die Transfersequenz für den EA-Verkehr mit "wartend", wird im Mikroprogramm eine Zeitschleife durchlaufen, die beendet wird, wenn die periphere Einheit die Aufforderung "warten" zurücknimmt. Geschieht dies nicht, wird nach ca. 50 us der Befehl mit Anzeige 3 abgebrochen. Die Zeit wird ggf. durch DMA-Zugriff anderer Bus-Teilnehmer verlängert.

Anzeigen:
0 Normaler Ablauf
1 Quittungsverzug
2 Quittung mit Fehlermeldung
3 Zeitüberschreitung beim Wartezustand.

EAS OP$_j$ wird an die EA-Adresse ausgegeben. Tritt bei der Befehlsbearbeitung ein Speicherfehler (Eintrag ins Unterbrechungsanzeigenregister UAR, Bit 2 oder 15) auf, wird die Transfersequenz für den EA-Verkehr unterdrückt.

EAL Tritt beim Lesen des Befehls oder bei Abprüfen des Vorhandenseins der Zieladresse ein Speicherfehler (Eintrag ins Unterbrechungsanzeigenregister UAR, Bit 2 oder 15) auf, wird die Transfersequenz für den EA-Verkehr unterdrückt.

Eine evtl. Schreibschutzverletzung bei Ablage des eingelesenen Datums im Zentralspeicher führt zum Eintrag von Unterbrechungsanzeigenregister-Bit 10. Die Transfersequenz für den EA-Verkehr ist dann bereits abgewickelt. Der Inhalt des Registers im F2-Feld kann in allen Fällen verändert sein.

6.4.1.14 Organisatorische Befehle

	Befehl	Formate	Beschreibung	Anzeige	Bemerkung
LAS	Laden Spezialregister	RR, RAX	$OP_i := OP_j$	-	*)
BSS	Bit setzen Spezialreg.	RAX	$OP_i := OP_i$ ODER OP_j	-	*)
BLS	Bit loschen Spezialreg.	RAX	$OP_i := OP_i$ UND OP_j	-	*)
LES	Lesen Spezialregister	RR	$OP_j := OP_i$	-	
TST	Testbefehl	Sonderformate		-	
RPZ	Rufen Primärzustand			-	
SPS	Verlassen Sondermodus			-	
STP	Stop			-	

6

*) Tritt bei der Befehlsbearbeitung ein Speicherfehler (Eintrag ins Unterbrechungsanzeigenregister UAR Bit 1, 2 oder 15) auf, laufen die Befehle als NOP ("no operation", keine Befehlsausführung) ab

LAS Das durch das F1-Feld adressierte Spezialregister wird mit dem Operanden OP_j geladen.

BSS Die in der Maske OP_j gesetzten Bits werden mit dem Inhalt des adressierten Spezialregisters ODER-verknüpft.

BLS Die in der Maske OP_j gesetzten Bits werden im adressierten Spezialregister gelöscht.

LES Das durch das F2-Feld adressierte Standardregister wird mit dem Inhalt des durch das F1-Feld adressierten Spezialregisters geladen. Wird Spezialregister 1 (Programmzustandsregister PZR linkes Byte) geladen, wird das rechte Byte im Standardregister gelöscht.

Anwendbarkeit der Befehle auf die Spezialregister:

Spezialregister	LAS BSS BLS	LES	Adresse (F1-Feld)
Programmzustandsregister	PZR	PZR	0
Programmzustandsregister (Maskenteil)	PZRLB	PZRLB	1
Unterbrechungsregister	-	UR	3
Zentralprozessor-Zustandsregister	-	ZZR	5
Tafelzeigeregister 2	TZR 2	TZR 2	8
Adressierungs-weichenregister	AWR	AWR	B
Programmlaufzeitzähler	PLZ	PLZ	C

TST Ist das Testaktivierungssignal (Signal des Zentralprozessors) aktiv, wird im Testbefehl (Zwei-Wort-Befehl) der Status: TEST ausgegeben, andernfalls läuft der Befehl als NOP ("no operation", keine Befehlsausführung) ab. Über die Art der Verwendung des F3-Felds kann der Anwender entscheiden.

STOP Durch diesen Befehl läuft der Zentralprozessor auf Stop. Es wird ein Zustandswechsel nach ZO durchgeführt. Anschließend läuft der Zentralprozessor in einer Stop-Mikroprogrammschleife, die er nur durch einen nicht maskierbaren Interrupt NMI (z. B. Fortsetzungsstart, Urladen usw.) verlassen kann.

RPZ Dieser Befehl setzt Bit 0 im Unterbrechungsanzeigenregister UAR. Damit wird der dem laufenden Programm zugehörige Programmlaufbesonderheiten-Prozeßblock (PLB - PRB) aktiviert. Somit besteht die Möglichkeit, das Programm zur Bearbeitung von Programmlaufbesonderheiten per Software zu aktivieren.

SPS Mit diesem Befehl können sowohl der Simulationsmodus als auch der ZO-Zustand verlassen werden.

Verlassen Sondermodus
Der SPS-Befehl führt im Simulationsmodus zum Wechsel in den Normalmodus (Benutzermodus), wenn die Anzeigen, welche per Software im F2-adressierten Register gesammelt wurden, gleich Null sind. Es werden die Standardregister 0, 1 des Simulationsmodus gerettet, das Bit 12 im Programmzustandsregister PZR gelöscht und die 16 Standardregister des Normalmodus aus der aktuellen Parametertafel geladen.
Ist der Inhalt des durch F2-adressierten Registers ungleich Null, so wird er in das Unterbrechungsanzeigenregister UAR geladen und der zugehörige Programmlaufbesonderheiten-Prozeßblock aktiviert. Im Normalmodus wird dieser Befehl als NNN (nicht interpretierbarer Befehl) behandelt.

Verlassen Z0-Zustand
Wird ein SPS-Befehl in Z0 abgesetzt, so verläßt das Programm durch einen Zustandswechsel Z0. Anschließend bearbeitet die Zentraleinheit den sich in Ebene 0 - 15 befindlichen höchstprioren Prozeß.

6.4.1.15 Prozeßsteuerbefehle

Befehl		Formate	Beschreibung	Anzeige	Bemerkung
APZ	Aktivieren eines Prozeßblocks nach zeitlicher Reihenfolge	A	Prozeßblock OP_j am Ende der Warteschlange einhängen. Entsprechendes Bit im Unterbrechungsregister setzen, falls die Warteschlange leer war. A-Bit im Prozeßblock setzen.	-	-
APF	Aktivieren eines Prozeßblocks an fester Stelle.	AA	Prozeßblock OP_j vor Prozeßblock OP_i einhängen. Entsprechende Bit im Unterbrechungsregister setzen, falls die Warteschlange leer war. A-Bit im Prozeßblock setzen.	-	-
DAP	Deaktivieren eines aktuellen Prozeßblocks	-	M-Bit nicht gesetzt: Aktuellen Prozeß aushängen. Entsprechendes Bit im Unterbrechungsregister löschen, falls die Warteschlange leer wird. A-Bit im Prozeßblock löschen. M- Bit gesetzt. Aktuellen Prozeß nicht aushängen. M-Bit im Prozeßblock löschen.	-	-
DFP	Deaktiviere fremden Prozeß	A	Prozeßblock OP_i aushängen. Entsprechendes Bit im Unterbrechungsregister löschen, falls die Warteschlange leer wird. A-Bit und falls gesetzt auch das M-Bit im Prozeßblock löschen.	-	-

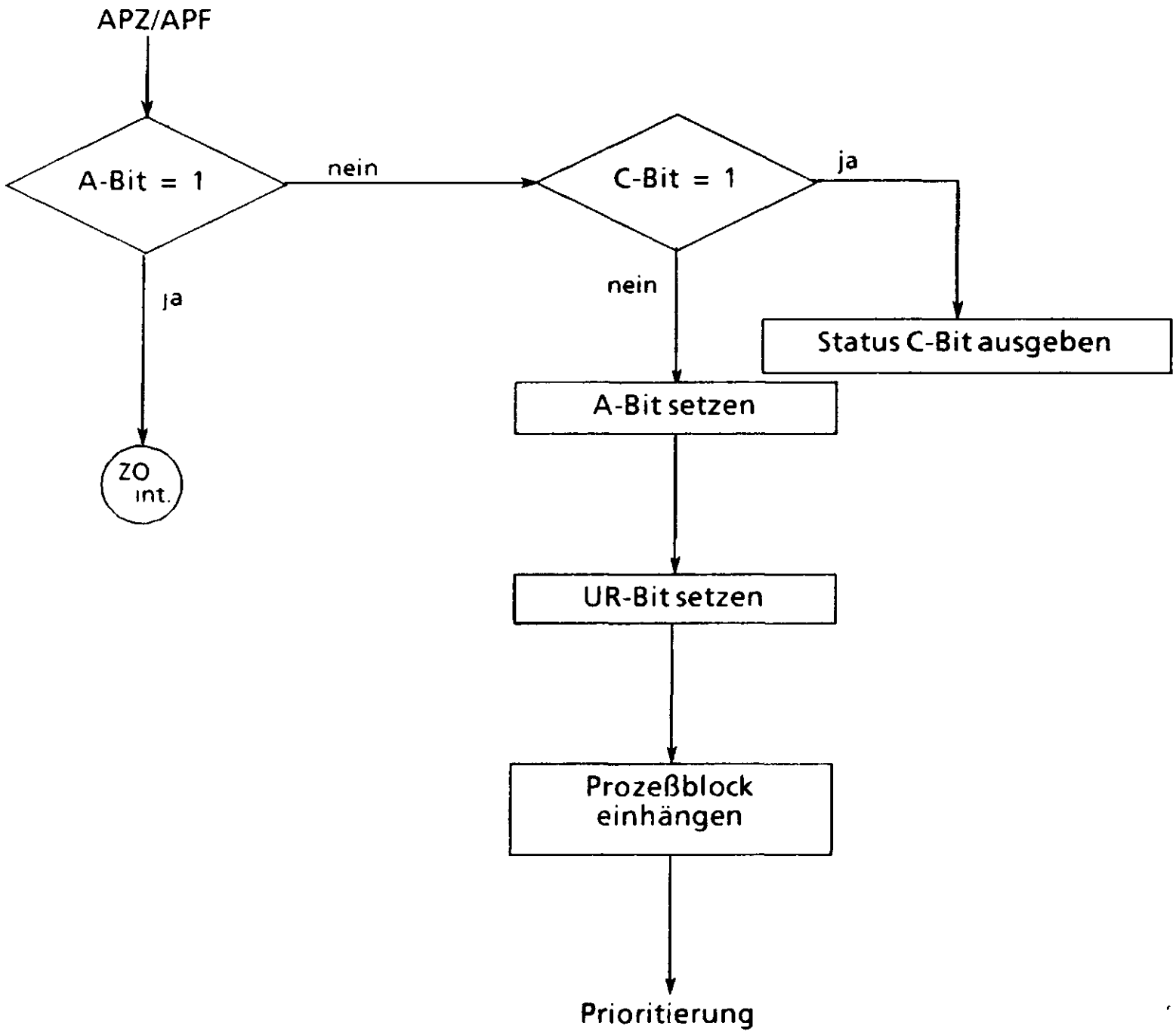
Mittels der Prozeßsteuerbefehle kann die Unterbrechungssteuerung beeinflußt werden. Diese Befehle laufen privilegiert ab.

Der in den Befehlsabläufen dargestellte Abbruch "ZO int" bedeutet einen Zustandswechsel nach ZO, der aufgrund organisatorischer Fehler im Mikroprogramm durchgeführt wird.

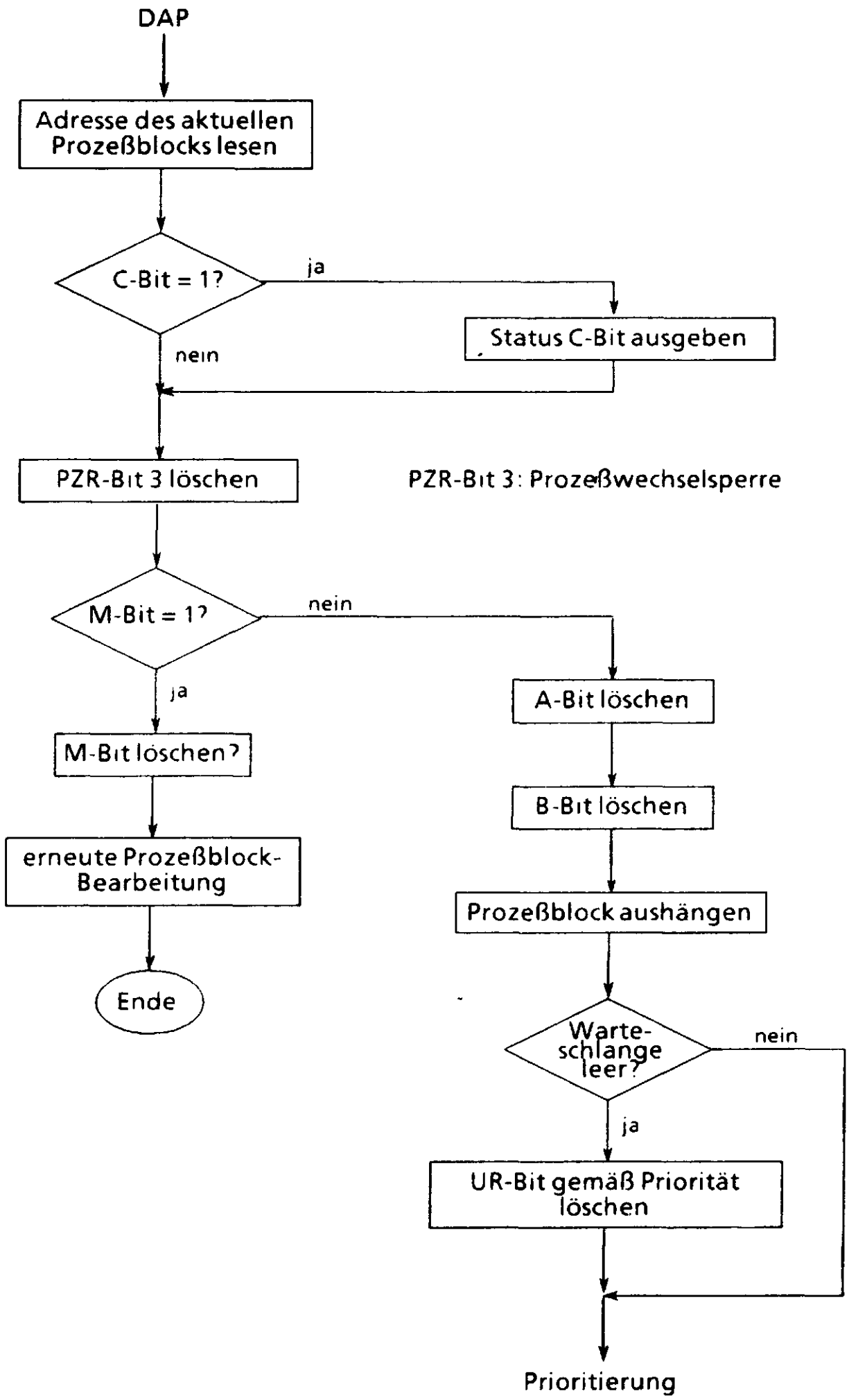
Die Befehlsabläufe werden von den im Prozeßblock befindlichen Zustandsbits (M-, A-, B-, C-Bit) gesteuert.

APZ Der zu aktivierende Prozeßblock wird mit der Prozeßblock-Anfangsadresse OP_j adressiert. Im Unterbrechungsregister (UR) wird entsprechend der im Prozeßblock (PRB) hinterlegten Priorität ein Bit gesetzt. Danach wird auf Basis der Priorität die Adresse des Warteschlangenkopfs ermittelt. Anschließend wird der Prozeßblock am Ende der Warteschlange eingehängt.

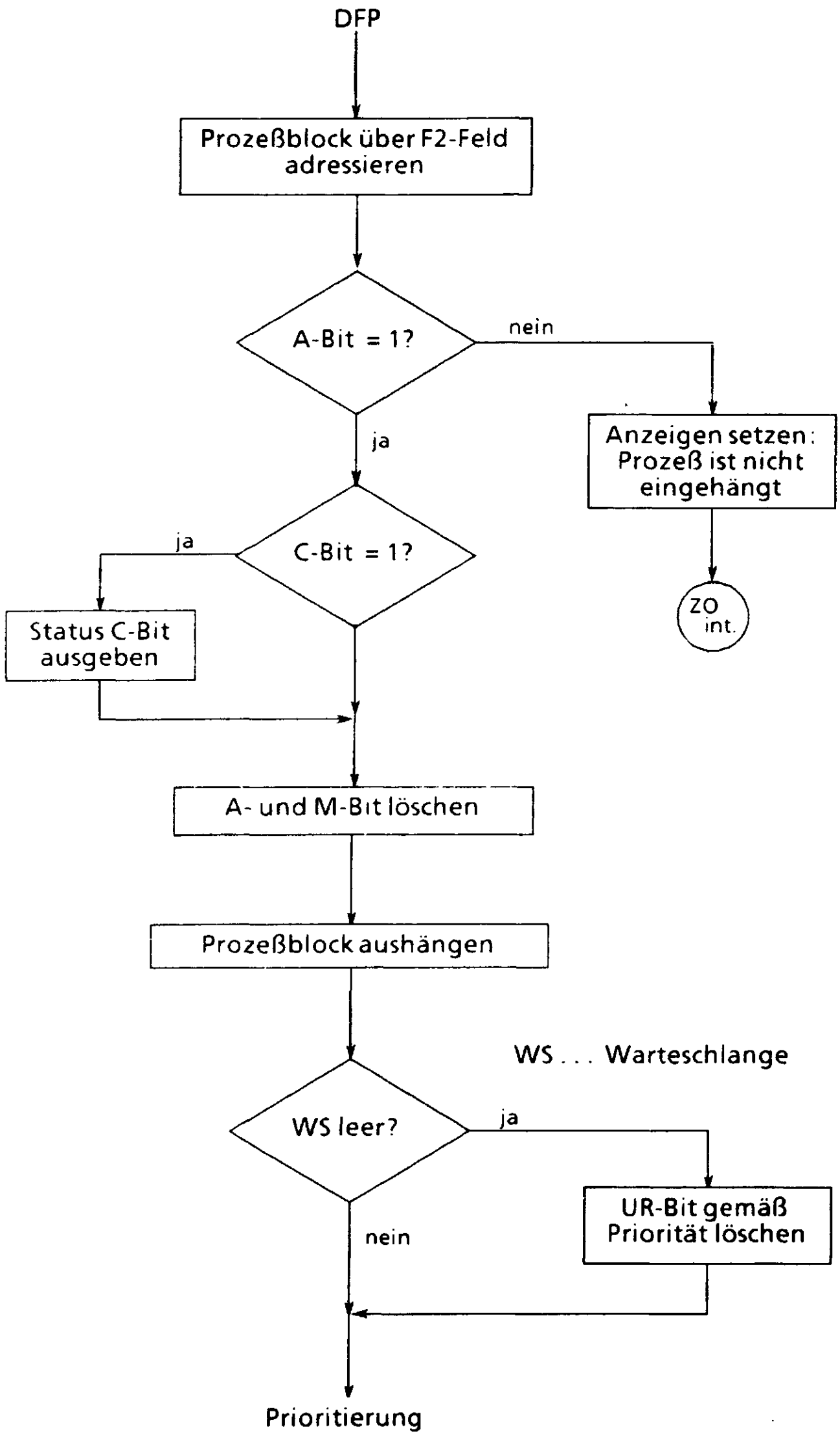
APF Der Ablauf entspricht dem des APZ, nur daß der Prozeßblock nicht am Ende der Warteschlange, sondern an vorgegebener Stelle der Warteschlange eingehängt wird.



DAP Der aktuelle Prozeßblock soll deaktiviert werden, d.h. der laufende Prozeß will den Zustand "aktiv" verlassen (sofern er nicht mehrfach angefordert wird).
Über das Zentralprozessor-Zustandsregister (ZZR) wird die Anfangsadresse des aktuellen Prozeßblocks ermittelt.



DFP Ein bestimmter Prozeßblock OP_j wird deaktiviert.



Ergebnis-anzeigen			Anzeigentyp/Ergebnis bei								
			F	B	V	L	E/A	T	G	1)	2)
Anzei-genr.	PZR-Bit 0	1	Festpunkt-Rechnung	Betrags-Befehle	Vergleichs-Befehle	Bool'sche Befehle	Ein/Ausgabe-Befehle	Bittest-Befehle	Gleitpunkt-Befehle	SHP	VMS VMR
0	0	0	=0	=0	1. Operand gleich 2. Operand	=0	---	das getestete Bit oder alle sind = 0	Mantisse = 0, Exponenten-unterlauf	Anzahl der über die Stelle 15 hinausgeschobenen Einsen war ungerade	Alle getesteten Bits oder alle Maskenbits = 0
1	0	1	<0	-	1. Operand kleiner 2. Operand	-	Quittungs-verzug	---	Mantisse <0	-	Die getesteten Stellen enthalten 1 und 0
2	1	0	>0	>0	1 Operand größer 2. Operand	≠0	Quittung mit Anzeigen	das getestete Bit = 1	Mantisse >0	Anzahl der über die Stelle 15 hinausgeschobenen Einsen war gerade	Alle getesteten Bits = 1
3	1	1	Überlauf	Überlauf	---	---	Zeitüber-schreitung bei Stotterbetrieb	---	Exponenten-Überlauf	---	---

Ergebnis-anzeigen			Anzeigentyp/Ergebnis bei								
			3)	4)	5)	6)	7)	8)	9)	10)	11)
Anzei-genr	PZR-Bit 0	1	VBY	BZF	SFU	SFG	PGF PGB	VBF	SUZ	SGZ	EHD AHD
0	0	0	Beide Bytes der Operanden sind gleich	Das Feld enthält keine gesetzte Stelle	---	Beide Bytes sind gleich	Operand innerhalb bzw auf Grenzen	Beide Zeichenfolgen gleich	Alle Bytes aus OP_j = Kennbyte	---	Warteschlange war leer
1	0	1	Die linken Bytes sind gleich	---	Die rechten Bytes sind ungleich	Die linken Bytes sind gleich	Operand kleiner als untere Grenze	erstes ungleiches Byte aus OP_i < als aus OP_j	Kennbyte <als erstes ungleiches Byte aus OP_j	linkes Kennbyte = Byte aus OP_j	---
2	1	0	Die rechten Bytes sind gleich	Das Feld enthält eine gesetzte Stelle	Die linken Bytes sind ungleich	Die rechten Bytes sind gleich	Operand großer als obere Grenze	erstes ungleiches Byte aus OP_i > als aus OP_j	Kennbyte >als erstes ungleiches Byte aus OP_j	rechtes Kennbyte = Byte aus OP_j	Warteschlange enthielt Element(e)
3	1	1	Keine Bytes sind gleich	---	Beide Bytes sind ungleich	--	---	---	---	Alle Zeichen sind ungleich	---

Codierung der Ergebnisanzeigen (Bits 0 und 1 der PZR) bei verschiedenen Befehlen

Befehlsadreßregister BAR

Das BAR, usprünglich hardwaremäßig eines der 16 Standardregister (R1), ist in der ZE 03 im RAM-Bereich des Adreßprozessors enthalten. Dynamische Befehlsmodifikation ist in der ZE 03 wegen des Befehls-look-aheads nicht möglich. Wird das BAR trotz Verbots angesprochen, läuft der Befehl undefiniert ab.

F1-Feld = 1 (d. h. Verwendung des Registers R1 über das F1-Feld) ist für folgende Befehle verboten:

LAB, LLB, LRB, LTF, LKF, LLF, LTD, LKD, LTG, LTK, TAB, UBA, LAC

SPB, SLY, SRY

ADD, ADG, ADK, SBD, SBG, SBK

MLF, MLB, MLD, MLG, MLK, DVF, DVB, DVD, DVG, DVK

VEB, VLB, VRB, VBY, VMS, VMR, VGD, VGG, VGK, PGF, PGB

BTT, BTS, BTL, BTZ

SRF, SLF, SHF, SRB, SLB, SHB, SRU, SLU, SHP, SDR, SDL, SDB,

SRD, SLD, SHD

DSP, AVS, USP, USK, UCK

SFG, SFU, BZF

SUZ, SGZ, FMZ, TBF, VBF

EHD, AHD, APF

EAL, EAS

F2-Feld = 1 (d. h. Verwendung des Registers R1 über das F2-Feld) ist für folgende Befehle verboten:

LAF RDA, LAD RDA, LAG RDA, LAK RDA

LAB, LLF RR, LLF RA, LTG RR, LTG RA, LTK RR, LTK RA, TAB, UBA

SPB, SLY RA, SRY RA, SPF RA, SPF RDA, SPD RA, SPD RDA, SPG RA, SPG RDA,

SPK RA, SPK RDA

VER

BTS RR, BTS RA, BTS CR, BTS CA, BTL RR, BTL RA, BTL CR, BTL CA

SPR CDA, USK XAI, USK XDA, UCK DAR

SFG, SFU, RZF

SUZ, SGZ, FMZ, TBF, VBF

EHD, AHE, AHD

APZ, APF, DFP

EAS AR, EAL AR, LES RR.

6.4.2 Operationszeiten

Die angegebenen Operationszeiten beziehen sich auf den Standardfall (ohne Adreßübersetzung, ohne Zentralspeicher-Korrekturmaßnahmen). Die Angaben mit Cache-Speicher setzen eine "Trefferrate" von 100% voraus, d. h. der betreffende Befehl und zugehörige Daten sind im Cache-Speicher enthalten. Die Gleitpunktzeiten beziehen sich auf die Standardvariante des Gleitpunktprozessors. Die Operationszeiten sind in Mikrosekunden angegeben.

6.4.2.1. Ladebefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	
LAB	RAI	1,20	1,55	
LLB	RR RA RAI RAX	0,40 0,60 0,60 0,65	0,55 1,30 1,30 1,65	
LRB	RR RA RAI RAX	0,40 0,60 0,60 0,65	0,55 1,30 1,30 1,65	
LAF	RC RR RA RAI RDA RAX	0,20 0,20 0,60 0,60 0,65 0,6	0,55 0,55 1,30 1,30 1,30 1,65	
LTF	RR RA RAI RAX	0,20 0,60 0,60 0,65	0,55 1,30 1,30 1,65	
LKF	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65	
LLF	RR RA RAI RAX	0,40 1,55 1,55 1,60	0,55 1,90 1,90 2.25	
LAD	RC RR RA RAI RDA RAX	0,80 0,95 1,35 1,35 1,40 1,45	1,15 1,30 2,75 2,75 2,75 3,15	

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	
LTD	RR RA RAI RAX	0,80 1,25 1,25 1,30	1,15 2,60 2,60 2,95	
LKD	RC RR RA RAI RAX	0,75 0,95 1,45 1,45 1,50	1,10 1,30 2,80 2,80 3,15	
LAG	RR RA RAI RDA RAX	0,95 1,30 1,30 1,35 1,40	0,95 2,35 2,35 2,35 2,75	
LTG	RR RA RAI RAX	1,05 1,30 1,30 1,40	1,10 2,35 2,35 2,75	
LAK	RR RA RAI RDA RAX	1,25 1,70 1,70 1,75 2,00	1,75 3,50 3,50 3,50 4,05	
LTK	RR RA RAI RAX	1,35 / 1,55 1,75 / 1,95 1,75 / 1,95 2,05 / 2,25	1,70 / 1,90 3,50 3,50 4,05	
TAB	RR	0,55	0,55	
UBA	RR	1,50 / 1,65	1,55 / 1,70	
LAC	RAX	0,85	2,20	

6.4.2.2 Speicherbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	
SPB	RAI	1,70	1,70	
SLY	RA RAI RAX	1,30 1,30 1,35	1,45 1,45 1,80	
SRY	RA RAI RAX	1,30 1,30 1,35	1,45 1,45 1,80	
SPF	RA RAI RDA RAX	1,00 1,00 1,05 1,05	1,15 1,35 1,35 1,50	
SPD	RA RAI RDA RAX	1,85 1,85 1,90 1,90	2,55 2,55 2,55 2,90	
SPG	RA RAI RDA RAX	1,65 1,65 1,70 1,70	2,00 2,00 2,00 2,35	
SPK	RA RAI RDA RAX	3,15 3,15 3,20 3,20	3,85 3,85 3,85 4,20	

6.4.2.3 Addition, Subtraktion

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	ZE03-C* (µs)	ZE03* (µs)
ADF	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65		
AUF	RR	0,20	0,55		
ADB	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65		
AUB	RC RR	0,20 0,20	0,55 0,55		
ADD	RC RR RA RAI RAX	0,95 0,95 1,45 1,45 1,50	1,30 1,30 2,80 2,80 3,15		
ADG	RR RA RAI RAX	7,90 - 37,00 8,10 - 37,20 8,10 - 37,20	7,90 - 37,00 8,90 - 38,00 8,90 - 38,00	1,90 - 5,10 2,45 - 5,55 2,45 - 5,55 2,40 - 5,50	1,90 - 5,10 3,15 - 6,25 3,15 - 6,25 3,40 - 6,20
ADK	RR RA RAI RAX	13,10 - 69,80 13,10 - 69,80 13,30 - 70,00 13,10 - 69,80	13,10 - 69,80 14,70 - 71,40 14,90 - 71,60 14,70 - 71,40	3,30-10,30 4,10-11,10 4,30-11,20 4,30-11,20	3,65-10,65 5,95-12,95 6,15-13,15 6,65-13,30
SBF	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65		
SUF	RR	0,20	0,55		
SBB	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65		
SUB	RC RR	0,20 0,20	0,55 0,55		

* Befehlszeiten des High-Performance-Gleitpunktprozessors 6AA7003-0FA

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	ZE03-C* (µs)	ZE03* (µs)
SBD	RC RR RA RAI RAX	0,95 0,95 1,45 1,45 1,50	1,30 1,30 2,80 2,80 3,15		
SBG	RR RA RAI RAX	7,90 - 37,00 8,10 - 37,20 8,10 - 37,20 8,10 - 37,20	7,90 - 37,00 8,90 - 38,00 8,90 - 38,00 8,90 - 38,00	1,90-5,10 2,45-5,55 2,45-5,55 2,40-5,50	1,90-5,10 3,15-6,25 3,15-6,25 3,40-6,20
SBK	RR RA RAI RAX	13,10 - 69,80 13,10 - 69,80 13,30 - 70,00 13,10 - 69,80	13,10 - 69,80 14,70 - 71,40 14,90 - 71,60 14,70 - 71,40	3,30-10,30 4,10-11,10 4,30-11,20 4,30-11,20	3,65-10,65 5,95-12,95 6,15-13,15 6,65-13,30

* Befehlszeiten des High-Performance-Gleitpunktprozessors 6AA7003-0FA

6.4.2.4 Multiplikation, Division

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	ZE03-C* (µs)	ZE03* (µs)
MLF	RC RR RA RAI RAX	2,80 2,80 3,15 3,15 3,20	2,95 2,95 3,85 3,85 4,05		
MLB	RC RR RA RAI RAX	2,60 (1,30/1,10) 2,60 (1,30/1,10) 2,95 (1,65/1,45) 2,95 (1,65/1,45) 3,00	2,75 (1,45/1,25) 2,75 (1,45/1,25) 3,65 (2,35/2,15) 3,65 (2,35/2,15) 3,85		
MLD	RC RR RA RAI RAX	17,60 17,40 17,60 17,60 17,60	17,60 17,40 18,20 18,20 18,20	4,55-4,55 4,55-4,55 4,90-4,90 4,90-4,90 4,85-4,85	4,90-4,90 4,90-4,90 5,90-5,90 5,90-5,90 5,85-5,85
MLG	RR RA RAI RAX	18,00 - 20,80 18,20 - 21,00 18,20 - 21,00 18,20 - 21,00	18,00 - 20,80 19,00 - 21,80 19,00 - 21,80 19,00 - 21,80	3,50-3,70 4,05-4,25 4,05-4,25 4,00-4,20	3,50-3,70 4,75-4,95 4,75-4,95 5,05-5,25
MLK	RR RA RAI RAX	50,60 - 53,10 50,60 - 53,10 50,80 - 53,30 50,60 - 53,10	50,60 - 53,10 52,20 - 54,70 52,40 - 54,90 52,20 - 54,70	6,35-6,75 7,15-7,55 7,35-7,75 7,35-7,75	6,70-7,10 8,95-9,35 9,15-9,55 9,20-9,55
DVF	RC RR RA RAI RAX	4,90-5,25 (1,85) 4,90-5,25 (1,85) 5,25 - 5,60 5,25 - 5,60 5,30 - 5,65	4,95 - 5,30 (1,90) 4,95 - 5,30 (1,90) 5,95 - 6,30 5,95 - 6,30 6,05 - 6,40		
DVB	RC RR RA RAI RAX	4,65-4,95 (1,75) 4,65-4,95 (1,75) 5,00-5,30 (2,10) 5,00-5,30 (2,10) 5,05-5,35 (2,15)	4,80 - 5,10 (1,90) 4,80 - 5,10 (1,90) 5,70 - 6,00 (2,80) 5,70 - 6,00 (2,80) 5,95 - 6,20 (3,00)		

* Befehlszeiten des High-Performance-Gleitpunktprozessors 6AA7003-0FA

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	ZE03-C* (µs)	ZE03* (µs)
DVK	RR	72,00 - 74,80	72,00 - 74,80	7,60-8,20	7,95-8,55
	RA	72,00 - 74,80	73,60 - 76,40	8,40-9,00	10,20-10,80
	RAI	72,20 - 73,00	73,80 - 76,60	8,60-9,20	10,40-11,00
	RAX	72,00 - 74,80	73,60 - 76,40	8,60-9,25	10,45-11,05
DVD	RC	21,60 - 24,70	21,60 - 24,70	7,10-7,50	7,45-7,85
	RR	21,40 - 24,50	21,40 - 24,50	7,10-7,50	7,45-7,85
	RA	21,40 - 24,50	22,20 - 26,10	7,45-7,85	8,45-8,85
	RAI	21,40 - 24,50	22,20 - 26,10	7,45-7,85	8,45-8,85
	RAX	21,40 - 24,50	22,20 - 26,10	7,40-7,80	8,40-8,80
DVG	RR	17,40 - 20,20	17,40 - 20,20	3,90-4,30	3,90-4,30
	RA	17,60 - 20,40	18,40 - 21,20	4,35-4,75	5,10-5,50
	RAI	17,60 - 20,40	18,40 - 21,20	4,35-4,75	5,05-5,45
	RAX	17,60 - 20,40	13,40 - 21,20	4,30-4,70	5,40-5,80

* Befehlszeiten des High-Performance-Gleitpunktprozessors 6AA7003-0FA

6.4.2.5 Vergleichsbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	
VEB	RAI	1,20	1,55	
VLB	RR RA RAI RAX	0,55 0,75 0,75 0,80	0,55 1,45 1,45 1,65	
VRB	RR RA RAI RAX	0,55 0,75 0,75 0,80	0,55 1,45 1,45 1,65	
VBY	RR RA RAI RAX	0,80 - 1,15 1,00 - 1,35 1,00 - 1,35 1,05 - 1,40	0,80 - 1,15 1,70 - 2,05 1,70 - 2,05 1,85 - 2,20	
VGF	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65	
VGB	RC RR RA RAI RAX	0,20 0,20 0,60 0,60 0,65	0,55 0,55 1,30 1,30 1,65	

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)	ZE03-C* (µs)	ZE03* (µs)
VMS	RC RR RA RAI RAX	0,55 0,55 0,95 0,95 1,00	0,55 0,55 1,65 1,65 1,65		
VMR	RA RA RAX	0,95 0,95 1,00	1,65 1,65 1,65		
VGD	RC RR RA RAI RAX	1,20 1,00 1,45 1,45 1,50	1,55 1,35 2,80 2,80 3,15		
VGG	RR RA RAI RAX	4,00 - 13,80 4,20 - 14,00 4,20 - 14,00 4,20 - 14,00	4,00 - 13,80 5,00 - 14,80 5,00 - 14,80 5,00 - 14,80	1,50-4,70 2,00-5,15 2,00-5,15 1,95-5,10	1,50-4,70 2,70-5,85 3,70-5,85 4,05-5,80
VGK	RR RA RAI RAX	6,20 - 27,20 6,20 - 27,20 6,40 - 27,40 6,20 - 27,20	6,20 - 27,20 7,70 - 28,60 7,90 - 28,00 7,70 - 28,60	2,50-9,50 3,30-10,30 3,50-10,50 3,50-10,40	2,85-9,85 5,10-12,10 5,30-12,30 5,35-12,35
PGF	RAX	1,10 - 2,30	2,40 - 3,60		
PGB	RAX	1,10 - 1,90	2,40 - 3,20		

* Befehlszeiten des High-Performance-Gleitpunktprozessors 6AA7003-0FA

6.4.2.6 Bool'sche Befehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
UND	RC	0,20	0,55
	RR	0,20	0,55
	RA	0,60	1,30
	RAI	0,60	1,30
	RAX	0,65	1,65
ODR	RC	0,20	0,55
	RR	0,20	0,55
	RA	0,60	1,30
	RAI	0,60	1,30
	RAX	0,65	1,65
XOR	RC	0,20	0,55
	RR	0,20	0,55
	RA	0,60	1,30
	RAI	0,60	1,30
	RAX	0,65	1,65

6.4.2.7 Bit-Testbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
BTT	RR	0,45	0,55
	RA	0,65	1,30
	RAI	0,65	1,30
	RAX	0,90	1,65
	CR	0,20	0,55
	CA	0,60	1,30
	CAI	0,60	1,30
	CAX	0,85	1,65
BTS	RR	0,65	0,65
	RA	1,80	2,15
	RAI	1,80	2,15
	RAX	1,85	2,50
	CR	0,50	0,55
	CA	1,55	1,90
	CAI	1,55	1,90
	CAX	1,80	2,45
BTL	RR	0,65	0,65
	RA	1,80	2,15
	RAI	1,80	2,15
	RAX	1,85	2,50
	CR	0,50	0,55
	CA	1,55	1,90
	CAI	1,55	1,90
	CAX	1,80	2,45
BTZ	RR	0,75	0,90
	RA	0,95	1,65
	RAI	0,95	1,65
	RAX	1,00	2,00

6.4.2.8 Schiebebefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
SRF	RC	0,90 + n x 0,15	0,95 + n x 0,15
SLF	RC	0,75 + n x 0,15	0,80 + n x 0,15
SHF	RR	links schieben:	
		1,20 + n x 0,15	1,25 + n x 0,15
		rechts schieben:	
		1,35 + n x 0,15	1,40 + n x 0,15
SRB	RC	0,75 + n x 0,15	0,80 + n x 0,15

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
SLB	RC	0,75 + n x 0,15	0,80 + n x 0,15
SHB	RR	links schieben:	
		1,20 + n x 0,15	1,25 + n x 0,15
		rechts schieben:	
		1,35 + n x 0,15	1,40 + n x 0,15
SRU	RC	0,75 + n x 0,15	0,80 + n x 0,15
SLU	RC	0,75 + n x 0,15	0,80 + n x 0,15
SHP	RC	1,40 + n x 0,15	1,45 + n x 0,15
SDR	RC	1,05 + n x 0,15	1,10 + n x 0,15
SDL	RC	1,20 + n x 0,15	1,25 + n x 0,15
SDB	RR	links schieben:	
		1,35 + n x 0,15	1,40 + n x 0,15
		rechts schieben:	
		1,35 + n x 0,15	1,40 + n x 0,15
SRD	RC	1,15 + n x 0,15	1,55 + n x 0,15
SLD	RC	1,10 + n x 0,15	1,50 + n x 0,15
SHD	RR	links schieben:	
		1,55 + n x 0,15	1,95 + n x 0,15
		rechts schieben	
		1,90 + n x 0,15	2,30 + n x 0,15

n = Schiebezahl (n) = 1)

6.4.2.9 Sprungbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
		nicht erf./erfüllt	nicht erf./erfüllt
SPR	CR	0,40 / 0,80	0,55 / 1,65
	CRX	0,60 / 0,90	1,10 / 1,65
	CA	0,40 / 1,25	0,55 / 2,45
	CAI	0,40 / 1,30	0,55 / 2,45
	CDA	0,40 / 1,25	0,55 / 2,45
	CAX	0,60 / 1,30	1,10 / 2,45
DSP	RR	0,75 / 1,15	0,75 / 1,85
	RRX	1,00 / 1,20	1,35 / 1,90

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
AVS	CRX	1,10 / 1,50	1,35 / 2,05
USF	CR	1,00	1,70
	CRX	1,00	1,70
	RAX	1,30	2,15
USK	RAI	1,20	1,90
	XAI	1,25	1,90
	XDA	1,30	1,95
SKV	CC	0,40 / 1,05	0,55 / 1,65
SKR	CC	0,40 / 1,05	0,55 / 1,65
UCK	DAR	2,65	4,05
SPC	CAI	0,60 / 2,20	1,10 / 4,65

6.4.2.10 Feldsuchbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
SFG	-	1,95 bis 2,50 + 1,8 x (W-1)	2,30 bis 2,85 + 1,8 x (W-1)
SFU	-	2,20 bis 2,35 + 1,0 x (W-1)	2,55 bis 2,70 + 1,0 x (W-1)
BZF	-	2,70 + 1,40 x (W-1)	3,05 + 1,40 x (W-1)

W = Anzahl der zu vergleichenden Operanden

6.4.2.11 Byte-adressierende Feldbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
SUZ	RAI	1,85 + 0,4 x B bzw. 2,45 + 0,4 x LZ	(2,775 bis 3,125) + 0,575 x B ≈ 2,95 + 0,575 x B bzw. (3,325 bis 3,675) + 0,575 x LZ ≈ 3,55 + 0,575 x LZ
SGZ	RAI	(1,55 bis 2,05) + 0,8 x B bzw. (2,15 bis 2,65) + 0,8 x LZ	(1,75 bis 2,60) + 1,15 x B bzw. (2,35 bis 3,20) + 1,15 x LZ

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
FMZ	RAI	1,30 + 0,70 x LZ	1,85 + 0,70 x LZ
TBF	AIAI	1,85 + 0,575 x LZ *)	2,6 + 0,675 x LZ *)
		2,1 + 1,275 x LZ	2,85 + 1,275 x LZ
VBF	AIAI	1,90 + 1,25 x B	2,60 + 1,95 x B
		bzw.	bzw.
		2,55 + 1,25 x LZ	3,95 + 1,95 x LZ

LZ = Längenzähler (Anzahl der Bytes), B = Anzahl der zu bearbeitenden Bytes

*) Dieser Wert gilt, wenn der Zähler LZ geradzahlig ist und beide Byte-Felder mit linken Bytes beginnen. In allen anderen Fällen gelten die anderen Werte.

6.4.2.12 Fädellistenbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
EHD	AA	3,20	3,70
AHE	A	2,75	3,45
AHD	AA	3,55	4,25

6.4.2.13 EA-Befehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
EAS	AR	3,05	3,20
	AAI	3,25	3,95
EAL	AR	2,00	2,35
	AAI	3,30	4,00

6.4.2.14 organisatorische Befehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
LAS	RR	1,50 - 2,15	1,70 - 2,80
	RAX	1,80 - 2,45	2,45 - 3,55

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
BGS	RAX	2,40 - 3,55	3,05 - 4,75
BLS	RAX	2,40 - 3,55	3,05 - 4,55
LES	RR	0,90 - 1,10	0,90 - 1,10
RPZ	-	0,20 + PLB + ZWS	0,20 + PLB + ZWS
SPS	-	a) 6,75 b) 2,75 + Laden	a) 13,40 b) 3,80 + Laden
TST	-	0,95	1,10

a) Verlassen des Simulationsmodus
b) Verlassen von Z0
ZWS ... Zustandswechsel

6.4.2.15 Prozeßsteuerbefehle, Zustandswechsel, Programmbesonderheit, Interrupt

Prozeßsteuerbefehle

Befehl	Format	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
APZ	A	a) 5,60 b) 5,60 + ZWS	a) 7,00 b) 7,00 + ZWS
APF	AA	a) 5,10 b) 5,10 + ZWS	a) 6,40 b) 6,40 + ZWS
DFP	AA	a) 5,40 b) 5,40 + ZWS	a) 6,70 b) 6,70 + ZWS
DAP	-	a) 6,10 b) 6,10 + ZWS c) 2,85	a) 7,00 b) 7,00 + ZWS c) 3,75

a) in Z0 bzw. bei gesetzter Prozeßwechselsperre
b) in Prioritätsebene E0 - E15
c) bei gesetztem M-Bit im Prozeßblock (Mehrfachanforderung)
ZWS ... Zustandswechsel

Zustandswechsel (ZWS)

	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
ZWS mit Retten/Laden	20,45	29,65
ZWS ohne Retten	12,85	22,05

	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
ZWS ohne Laden	17,05	20,30
ZWS ohne Retten/Laden	9,45	12,70
niederpriores Aktivieren bzw. Deaktivieren	5,70	6,40
Laden (B-Bit = 1)	7,10	15,30
Laden (B-Bit = 0)	3,70	5,95

B-Bit = busy-bit (Tätig-Bit) im Prozeßblock

Programmlaufbesonderheit (PLB)

	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
PLB	8,85 + ZWS	10,25 + ZWS

ZWS ... Zustandswechsel

Interrupt (IR)

	Zeit mit Cache-Speicher (µs)	Zeit ohne Cache-Speicher (µs)
IR	a) 5,10	a) 5,80
	b) 7,90 + ZWS	b) 9,30 + ZWS

a) Mehrfachaktivierung eines Prozeßblocks
b) Aktivieren eines Prozeßblocks
ZWS ... Zustandswechsel

6.4.3 Befehlsmatrix (Operationscodes)

Grundbefehle

sede-zimal	0	2	4	6	8	A	C	E			
dual	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1 ←		dual	
	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	Be- Forfehl mat	3	Bits 4 5 6 7	sedezimal
	NNN	USP RAX	EAL AAI	EAL AR	NNN	AVS RRX	LLB RAX	NNN	0	0 0 0 0	0
	VMR RA	LAS RAX	NNN	LAS RR	NNN	PGB RAX	LRB RAX	*)	0	0 0 0 1	1
	NNN	BLS RAX	EAS AAI	EAS AR	VMR RAX	PGF RAX	SLY RAX	*)	0	0 0 1 0	2
	NNN	BSS RAX	NNN	LES RR	VMR RAI	VEB RAI	SRY RAX	DVG RAI	0	0 0 1 1	3
	SPR CA	SPR CAX	SPR CAI	EBL RR	SDL RC	USK XDA	VLB RAX	VGG RAX	0	0 1 0 0	4
	BTL CA	BTL CAX	BTL CAI	EBS RR	SDR RC	TAB RR	VRB RAX	NNN	0	0 1 0 1	5
	BTS CA	BTS CAX	BTS CAI	SHB RR	SLU RC	UBA RR	BZF	DVG RAX	0	0 1 1 0	6
	BTT CA	BTT CAX	BTT CAI	SHF RR	SRU RC	SPS	VGG RR[1)]	LTG RR	0	0 1 1 1	7
	BTL RA	BTL RAX	BTL RAI	BTL RR	SLB RC	SPR CRX	LLB RAI	SBG RR	0	1 0 0 0	8
	BTS RA	BTS RAX	BTS RAI	BTS RR	SRB RC	USP RRX	LRB RAI	LTG RA	0	1 0 0 1	9
	BTT RA	BTT RAX	BTT RAI	BTT RR	SLF RC	DSP RRX	SLY RAI	SBG RA	0	1 0 1 0	A
	BTZ RA	BTZ RAX	BTZ RAI	BTZ RR	SRF RC	NNN	SRY RAI	LTG RAI	0	1 0 1 1	B
	DVB RA	DVB RAX	DVB RAI	DVB RR	DVB RC	SPF RDA	VLB RAI	SBG RAI	0	1 1 0 0	C
	MLB RA	MLB RAX	MLB RAI	MLB RR	MLB RC	LAF RDA	VRB RAI	LTG RAX	0	1 1 0 1	D
	DVF RA	DVF RAX	DVF RAI	DVF RR	DVF RC	RPZ	SKV CC	SBG RAX	0	1 1 1 0	E
	MLF RA	MLF RAX	MLF RAI	MLF RR	MLF RC	STP	SKR CC	ADG RR	0	1 1 1 1	F
	SPF RA	SPF RAX	SPF RAI	SDB RR	SHP RC	SPR CDA	LLB RA	NNN	1	0 0 0 0	0
	VBY RA	VBY RAX	VBY RAI	VBY RR	LAG RDA	USK XAI	LRB RA	LAG RA	1	0 0 0 1	1
	LKF RA	LKF RAX	LKF RAI	LKF RR	LKF RC	USK RAI	SLY RA	ADG RA	1	0 0 1 0	2
	LAF RA	LAF RAX	LAF RAI	LAF RR	LAF RC	LAB RAI	SRY RA	LAG RAI	1	0 0 1 1	3
	XOR RA	XOR RAX	XOR RAI	XOR RR	XOR RC	SPB RAI	VLB RA	ADG RAI	1	0 1 0 0	4
	ODR RA	ODR RAX	ODR RAI	ODR RR	ODR RC	TST	VRB RA	LAG RAX	1	0 1 0 1	5
	UND RA	UND RAX	UND RAI	UND RR	UND RC	BTL CR	SFG	ADG RAX	1	0 1 1 0	6
	VMS RA	VMS RAX	VMS RAI	VMS RR	VMS RC	BTS CR	SFU	SPG RDA	1	0 1 1 1	7
	VGB RA	VGB RAX	VGB RAI	VGB RR	VGB RC	BTT CR	LLB RR	LAG RR	1	1 0 0 0	8
	VGF RA	VGF RAX	VGF RAI	VGF RR	VGF RC	SPR CR	LRB RR	MLG RR	1	1 0 0 1	9
	SBB RA	SBB RAX	SBB RAI	SBB RR	SBB RC	USP RR	VLB RR	SPG RA	1	1 0 1 0	A
	ADB RA	ADB RAX	ADB RAI	ADB RR	ADB RC	DSP RR	VRB RR	MLG RA	1	1 0 1 1	B
	SBF RA	SBF RAX	SBF RAI	SBF RR	SBF RC	SUF RR	DVG RR[1)]	SPG RAI	1	1 1 0 0	C
	ADF RA	ADF RAX	ADF RAI	ADF RR	ADF RC	AUF RR	VGG RA[1)]	MLG RAI	1	1 1 0 1	D
	LLF RA	LLF RAX	LLF RAI	LLF RR	SUB RC	SUB RR	DVG RA1)	SPG RAX	1	1 1 1 0	E
	LTF RA	LTF RAX	LTF RAI	LTF RR	AUB RC	AUB RR	VGG RAI[1)]	MLG RAX	1	1 1 1 1	F
dual	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1 ←			
sede-zimal	1	3	5	7	9	B	D	F			

*) Befehle mit FE-Feld

Befehle mit FE-Feld

Befehl	FE-Feld Bits 0 bis 7	F0-Feld Bits 0 bis 4	F0-Feld Bits 5 bis 7							
			000	001	010	011	100	101	110	111
D-Befehle	1 1 1 0 0 0 0 1	0 0 0 0 0	LAD/RA	LAD/RAX	LAD/RAI	LAD/RR	LTD/RA	LTD/RAX	LTD/RAI	LTD/RR
		0 0 0 0 1	SPD/RA	SPD/RAX	SPD/RAI	SPD/RDA	VGD/RA	VGD/RAX	VGD/RAI	VGD/RR
		0 0 0 1 0	ADD/RA	ADD/RAX	ADD/RAI	ADD/RR	SBD/RA	SBD/RAX	SBD/RAI	SBD/RR
		0 0 0 1 1	MLD/RA	MLD/RAX	MLD/RAI	MLD/RR	DVD/RA	DVD/RAX	DVD/RAI	DVD/RR
		0 0 1 0 0	LAD/RDA	SLD/RC	SRD/RC	SHD/RR	LKD/RA	LKD/RAX	LKD/RAI	LKD/RR
		0 0 1 0 1	LAD/RC	LKD/RC	VGD/RC	ADD/RC	SBD/RC	MLD/RC	DVD/RC	NNN
		0 0 1 1 0	LAC/RAX	UCK/DAR	SPC/CAI	TBF/AIAI	VBF/AIAI	SGZ/RAI	SUZ/RAI	FMZ/RAI
		0 0 1 1 1	EHD/AA	AHE/A	AHD/AA	APZ/A	APF/AA	DAP/-	DFP/A	NNN
		0 1 0 0 0	NNN	NNN	NNN	NNN	NNN	NNN	NNN	NNN
		-	-	-	-	-	-	-	-	-
		-	-	-	-	-	-	-	-	-
		-	-	-	-	-	-	-	-	-
		1 1 1 1 1	NNN	NNN	NNN	NNN	NNN	NNN	NNN	NNN
K-Befehle	1 1 1 0 0 0 1 0	0 0 0 0 0	LAK/RA	LAK/RAX	LAK/RAI	LAK/RR	LTK/RA	LTK/RAX	LTK/RAI	LTK/RR
		0 0 0 0 1	SPK/RA	SPK/RAX	SPK/RAI	SPK/RDA	VGK/RA	VGK/RAX	VGK/RAI	VGK/RR
		0 0 0 1 0	ADK/RA	ADK/RAX	ADK/RAI	ADK/RR	SBK/RA	SBK/RAX	SBK/RAI	SBK/RR
		0 0 0 1 1	MLK/RA	MLK/RAX	MLK/RAI	MLK/RR	DVK/RA	DVK/RAX	DVK/RAI	DVK/RR
		0 0 1 0 0	LAK/RDA	NNN	NNN	NNN	NNN	NNN	NNN	NNN
		0 0 1 0 1	NNN	NNN	NNN	NNN	Zentralprozessor-Testbefehle			
		0 0 1 1 0	NNN	NNN	NNN	NNN	NNN	NNN	NNN	NNN
		-	-	-	-	-	-	-	-	-
		-	-	-	-	-	-	-	-	-
		-	-	-	-	-	-	-	-	-
		1 1 1 1 1	NNN	NNN	NNN	NNN	NNN	NNN	NNN	NNN

6.5 Fehlermeldungen beim Dialog mit der virtuellen Konsole

Diese Meldungen kennzeichnen Transferfehler, die über eine Nummer spezifiziert sind. Den Abschluß bildet ein Punkt (.) oder ein Ausrufezeichen (!).

A) IO-Error (Ein-/Ausgabefehler)-

Bei diesen Meldungen kennzeichnet die erste Ziffer die Fehlerart und die zweite Ziffer (im folgenden mit x bezeichnet) den Fehleort (oder Fehlerzeitpunkt). Es handelt sich hierbei um Fehler beim EA-Verkehr.

Fehlerort:

x = 0 : Beenden oder Rücksetzen (selektiv)
x = 1 : Urladegerät, einlesen der Buchführung (Random-Datenträger), bzw. des ersten Vorspanns (serieller Datenträger)
x = 2 : Urladegerät, einlesen Programmblock-Vorspann
x = 3 : Urladegerät, einlesen der Nutzdaten
x = 4 : virtuelle Konsole, Eingabe
x = 5 : virtuelle Konsole, Ausgabe
x = 6 : List-Gerät, Ausgabe bei LIST-Funktion
x = 7 : Ausgabegerät, Ausgabe mit PRBVKX (Prozeßblock für die virtuelle Konsole, VK-Kommandos DUMP, MESSAGE)
x = 8 : Serviceprozessor-Anschaltung SPAS, Parametrierung
x = 9 : beliebiges Ausgabegerät, Ausgabe mit PRBBAS (Basisprozeßblock) (Knopfdrucktest, VK-Kommando STORE).

VICOM: IO-ERROR 1x : H = ioad, H = ioda

Quittungsverzug beim EA-Befehl
(ioad = EA-Adresse, ioda = EA-Datum)

VICOM: IO-ERROR 2x : H = ioad, H = ioda

Quittung mit Anzeigen beim EA-Befehl
(ioad = EA-Adresse, ioda = EA-Datum)

VICOM: IO-ERROR 3x : H = ioad, H = ioda

Überschreitung der Zeitüberwachung nach Vorabquittierung ("stottern") beim EA-Befehl
(ioad = EA-Adresse, ioda = EA-Datum)

VICOM: IO-ERROR 4x : H = dat1, H = dat2, H = dat3, H = dat4

worst-case-error (Eintrag in PEREQ im Prozeßblock)
(dat1 = PERED2, dat2 = PERED1, dat3 = PEREQ, dat4 = SERDAT).

VICOM: IO-ERROR 5x : H = dat1, ... , H = dat5

Transferabschluß mit unzulässigem NORTER/FINTER im Prozeßblock
(dat1 = SWCTR, dat2 = TPWP, dat3 = FIPHY,
 dat4 = CHECKF/NUTODO, dat5 = NORTER/FINTER).

VICOM: IO-ERROR 6x : H = dat1, ... , H = dat5

Transferabschluß mit Anzeigen im PHYSPB (physikalischer Parameterblock)
(dat1 = Zelle 13 BUPOA, dat2 = Zelle 14, dat3 = Zelle 15,
 dat4 = Zelle 16, dat5 = Zelle 17, Zelle 14...17 bilden das
 Anzeigenfeld im PHYSPB).

VICOM: IO-ERROR 7x : H = ioad, H = ioda
Zeitüberschreitung (Aufträge wie VC-Eingabe und VC-Freigabe werden
zeitüberwacht)
(ioad = I/O-Adresse, ioda = I/O-Datum).

B) Urlade-Fehler

Bei diesen Fehlern handelt es sich um Fehler in der Struktur des Urlade-
formats.

VICOM: BT-ERROR 11: H = ssss, H = iiii

Fehler bei der Polynomprüfung

- der Buchführung (Random-Datenträger),
- des Programmvorspanns (serieller Datenträger).
(ssss = Soll-Prüfmuster, iiii = Ist-Prüfmuster).

VICOM: BT-ERROR 12 : H = ssss, H = iiii

Fehler bei der Polynomprüfung eines Programvorspanns.
(ssss = Soll-Prüfmuster, iiii = Ist-Prüfmuster)

VICOM: BT-ERROR 13 : H = ssss, H = iiii

Fehler bei der Polynomprüfung der Nutzdaten.
(ssss = Soll-Prüfmuster, iiii = Ist-Prüfmuster)

VICOM: BT-ERROR 14 : H = ssss, H = iiii

Fehler bei der Summenprüfung der Nutzdaten.
(ssss = Soll-Prüfmuster, iiii = Ist-Prüfmuster).

VICOM: BT-ERROR 21 : H = maxa, H = ista

Die Programmanzahl in der Buchführung ist falsch (= 0 oder > 38).
(maxa = maximale Anzahl, ista = tatsächliche Anzahl).

6.6 Stichwortverzeichnis

6.7 Abkürzungsverzeichnis

AB	Adreßbus
AC	adress control (Adreßsteuersignal)
ADDIN	additional information (Zusatz-Information)
AE	adress enable (Adreßfreigabe)
AP	Adreßprozessor
AWR	Adressierungsweichenregister
AWW	Adressierungsweichenwort
BA	Befehlsaufbereitung, Busanpassung, bus acknowledge (Busanforderungs-Rückmeldung)
BAI	bus acknowledge input (Busanforderungs-Rückmeldung-Eingang)
BAO	bus acknowlege output (Busanforderungs-Rückmeldung-Ausgang)
BAP	Busanpassung peripher, Befehlsadressen-Pufferwort
BAZ	Busanpassung zentral
BB	bus busy (Bus belegt)
BF	battery failure (Batterieausfall)
BGT	Baugruppenträger
BL	bus lock (Bussperrsignal)
BM	block mode (Busmodus)
BR	bus request (Busanforderung)
BT	bootsstrap (urladen)
BUPOA	buffer pointer after transfer (Pufferzeiger nach Transfer)
BUPOB	buffer pointer before transfer (Pufferzeiger vor Transfer)
CA	Cache-Speicher
CB	control Bus (Steuer-Bus)
CAUPRB	causing PRB (verursachender PRB)
CC	central clock (Zentraltakt)
CHECKF	checkfield (Prüffeld)
COMCLS	command class (Auftragsklasse)
COMCOD	command code (Auftrags-Code)
CPU	central processing unit (Zentralprozessor)
DB	Datenbus
DEVNUM	device number (Gerätenummer)
DEVPRB	device process block (Geräte-Prozeßblock)
DMA	direct memory access (direkter Speicherzugriff)
EA	Ein-/Ausgabe
ECC	error correcting code (Fehler-Korrekturcode)
EPROM	erasable programmable read only memory
EXPOA	externpointer before transfer (Externzeiger vor Transfer)
EXPOB	externpointer after transfer (Externzeiger vor Transfer)
FFA	fault flag address (Fehlersignal Adresse)
FFD	fault flag data (Fehlersignal Daten)
FINTER	final termination (Beenden)
FIPHY	first PHYSPB (erster PHYSPB)
FPP	Floating Point Processor
GP	Gleitpunktprozessor
HLD	hold (Anhalten)
HSP	Hauptspeicher
HW	Hardware
HWFLAG	hardware flag (Hardware-Anzeige)
HW-VB	Hardware-Verständigungsbereich
IA	interrupt acknowledge (Unterbrechungs-Rückmeldung)
IAI	interrupt acknowledge input (Unterbrechungs-Rückmeldung Eingang)
IAO	interrupt acknowledge output (Unterbrechungs-Rückmeldung Ausgang)

IL	interrupt lock (Unterbrechungs-Sperre)
IR	interrupt request (Unterbrechungs-Anforderung)
NM	Normalmodus
NMI	nicht maskierbarer Interupt
NNN	nicht interpretierbarer Befehl
NOP	no operation (keine Befehlsausführung)
NORTER	normal termination (Abschluß)
NUTODO	number of PHYSPBs to do (Anzahl der zu bearbeitenden PHYSPBs)
PE	periphere Einheit
PEREQD	peripheral request's data (Daten zur peripheren Anforderung)
PEREQ	peripheral request (periphere Anforderungen)
PHYSPB	physikalischer Parameterblock
PLAN	Platznummer
PLB	Programmlaufbesonderheit
PLW	Programmlaufzeitwort
PLZ	Programmlaufzeit-Zähler
PRB	Prozeßblock
PREPHY	preceding PHYSPB (vorhergehender PHYSPB)
PREPRB	preceding PRB (vorhergehender PRB)
PRIOR	priority of process (Prozeß-Priorität)
PROM	programmable read only memory
PS	Privatsignale
PT	Parametertafel
PZR	Programmzustandsregister
PZW	Prcgrammzustandswort
RAM	random access memory
RES	reserviert
RESE	reelle Seite
ROM	read only memory
RS	reset (Rücksetzen)
RSC	reset CPU (Rücksetzen Zentralprozessor)
RSP	reset peripheral (Rücksetzen Peripherie)
SEP	Standard-Einbauplatz
SERDAT	service data (Service-Daten)
SM	Simunlationsmodus
SPAS	Serviceprozessor-Anschaltung
SS	Service-Steckplatz, Service-Schnittstelle
STW	Steuerwerk
SUCPHY	succeeding PHYSPB (nachfolgender PHYSPB)
SUCPRB	succeding PRB (nachfolgender PRB)
SW	Software
SWCTR	software control (Software-Steuerzelle)
SWFLAG	software flag (Software-Anzeige)
TA	transfer acknowledge (Transfer-Rückmeldung)
TC	timing and control (Speichersteuerung)
TER	terminate (Beenden-Code)
TERCOD	termination code (Beenden-Code)
TESTAS	Testanschaltung
TLENG	transfer length (Tansferlänge)
TPW	table pointer word (Tafelanzeigerwort)
TPWP	table pointer word for PHYSPB (Tafelzeigerwort für PHYSPB)
TR	transfer request (Transfer-Anforderung)
TZR	Tafelzeigeregister
TZW	Tafelzeigerwort
UAR	Unterbrechungsanzeigenregister

UAW	Unterbrechungsanzeigenwort
UT	Übersetzungstafel
UTP	Übersetzungstafel für PHYSPB
UTT	Übersetzungstafel für Transferpuffer
UR	Unterbrechungsregister
VA	virtuelle Adressierung
VC	virtual console (virtuelle Konsole)
VEKLI	Vektorliste
VISE	virtuelle Seite
VK	virtuelle Konsole
VP	Verarbeitungsprozessor
VZ	Vorzeichen
WS	Warteschlange
WSK	Warteschlangenkopf
ZE	Zentraleinheit
ZIG	Zeit-Impulsgeber
ZP	Zentralprozessor
ZSP	Zentralspeicher
ZVE	Zentrale Verarbeitungseinheit
ZZR	Zentralprozessor-Zustandsregister