

AEG

Prozeß-Datenverarbeitung

Prozeßrechner

AEG 80-20

mit Zentraleinheit AEG 80-20/4

Kurzbeschreibung

Prozeßrechner

AEG 80-20

mit Zentraleinheit AEG 80-20/4

Kurzbeschreibung

Die vorliegende Veröffentlichung dient zu Informationszwecken.
Änderungen und Ergänzungen des Inhalts werden vorbehalten.
Die Ausführungen oder Teile davon sind nur verbindlich, wenn sie
von AEG-TELEFUNKEN ausdrücklich im Rahmen eines Angebotes
oder eines Vertrages zugesichert werden.
Vervielfältigungen nur mit Einverständnis der herausgebenden Stelle.

INHALTSVERZEICHNIS

	PROZESSRECHNERFAMILIE AEG 80	3
1	SYSTEMÜBERSICHT	5
2	ZENTRALEINHEIT	7
2.1	Zentralprozessor	8
2.2	Hauptspeicher	9
2.3	Datenstruktur	11
2.4	Befehlsformate	12
2.5	Befehlsliste	14
2.6	Eingabe- und Ausgabesystem	14
2.7	Unterbrechungssystem	16
2.7.1	Interne Unterbrechungen	17
2.7.2	Externe Unterbrechungen	18
3	PERIPHERE EINHEITEN	19
3.1	Prozeßperipherie	19
3.2	Datenperipherie	20
3.3	Datenfernübertragung	20
4	BEDIENUNG	23
5	BETRIEBSSYSTEM MARTOS-K	25
5.1	Unterbrechungs- und EA-Verwaltung	26
5.2	Programmverwaltung	27
5.3	Systeminitialisierung	28
5.4	Systemdienste	29
5.5	Laufzeitorganisation	30
5.6	Konvertierungsroutinen	30
6	PROGRAMMIERSYSTEM	31
6.1	Programmiersystem PRODOS	31
6.2	Programmiersystem PROGA	33
7	Dienstprogramme	35
7.1	Systemgenerierung	35
7.2	Integriertes Dienstsysteem PROG1	36
8	STANDARD-ANWENDUNGSSOFTWARE	37
8.1	ARSI80	38
8.2	AS80	38
9	ANHANG	39

PROZESSRECHNERFAMILIE AEG 80

AEG—TELEFUNKEN bietet seit 1974 eine wachsende Anzahl von Typen der AEG 80 — Prozeßrechner mit abgestuften Leistungsmerkmalen an:

AEG 80—60
AEG 80—40
AEG 80—20/4
AEG 80—20/3
AEG 80—20/2

Die Prozeßrechnerfamilie AEG 80 ist für den Einsatz in industriellen Anlagen sowie für technisch-wissenschaftliche und nachrichtentechnische Anwendungen bestimmt. In ihr sind die neuesten Erkenntnisse der Prozeßrechnertechnik mit den umfassenden Erfahrungen aus Produktion, Lieferung und Inbetriebsetzung von über 1000 Rechenanlagen einschließlich der Vorgänger AEG 60, TR 86 usw. vereinigt.

Das Hardware- und Softwarekonzept der AEG 80 — Prozeßrechner ist auf den Leistungsbereich des jeweiligen Typs abgestimmt, dabei aber weitestgehend einheitlich. Der modulare Aufbau der Hardware und der Software und die einheitlichen Schnittstellen aller AEG 80 — Prozeßrechner zur Prozeßperipherie, zur Datenperipherie, zu den peripheren Speichern und zur Datenfernübertragung gewährleisten für jeden Typ

- optimale und preisgünstige Anpassung an die Leistungsanforderungen
- Erweiterbarkeit und Konfigurierbarkeit sowie
- Austauschbarkeit von Geräten.

Alle AEG 80 — Systeme besitzen gemeinsam als besonders zu erwähnende Eigenschaften:

- höhere Leistung als die Vorgängersysteme durch die Zwei-Bus-Struktur (EA-Bus und Speicherbus) der Zentraleinheit
- mikroprogrammierter Zentralprozessor
- 16 vielseitig verwendbare, weitgehend gleichwertige Arbeitsregister
- ein leistungsfähiges Unterbrechungssystem
- hohe Betriebssicherheit durch konsequent in Normalmodus und System-Modus getrennten Betrieb
- leistungsfähige Befehle
- vielfältige Adressierungsmöglichkeiten
- der integrierte Multiplexkanal als leistungsfähiger EA-Kanal
- geringer Raumbedarf
- Einsetzbarkeit unter erschwerten Betriebsbedingungen.

Außerdem weist jeder Typ der AEG 80 — Prozeßrechner spezielle Vorteile über dem Durchschnitt seiner Preis- und Leistungsklasse auf, die AEG 80—20/4 z.B.

- Selektorkanalwerke zum Anschluß peripherer Speicher mit hoher Übertragungsgeschwindigkeit
- Verwendung problemorientierter Programmiersprachen für verschiedene Anwendungsgebiete.

SYSTEMÜBERSICHT

Die AEG 80–20/4 ist eine Weiterentwicklung der AEG 80–20/3. Ihre Zentraleinheit ist mikroprogrammiert und besitzt 16 Arbeitsregister, von denen 15 als Indexregister verwendbar sind. Die Wortlänge beträgt 16 Bits. Der Hauptspeicher kann bis 128 KBytes ausgebaut werden und ist byteweise adressierbar. Bei einem Speicherausbau von mehr als 64 KBytes ist die AEG 80–20/4 mit einer Speicherverwaltungseinheit ausgerüstet, die sich optional auch bei einem Speicherausbau bis 64 KBytes einsetzen läßt. Die Speicherverwaltungseinheit bietet einen wirkungsvollen Speicherschutz, der Speicher- und Sprungoperationen von einem Anwenderprogramm in das Betriebssystem oder in andere Anwenderprogramme verhindert.

Durch den kompakten Aufbau im INTERMAS-System ist der Prozeßrechner AEG 80–20/4 auch unter erschwerten Umweltbedingungen einsetzbar. Durch den variablen Aufbau der Anlage können viele verschiedenartige Anwendungen realisiert werden. Die einheitliche, in der Leistung abgestufte Peripherie läßt vielseitige Konfigurationsmöglichkeiten zu.

Die hohe Verarbeitungsleistung der AEG 80–20/4 wird bestimmt durch:

- die Zwei-Bus-Struktur: Speicherbus und paralleler EA-Bus
- das leistungsstarke EA-System mit automatischer Ein- und Ausgabe über den integrierten Multiplexkanal, maximal zwei Selektorkanalwerken mit je vier Unterkanälen und dem optional anschließbaren direkten Speicherzugriff
- das reaktionsschnelle Unterbrechungssystem und
- den vielseitigen Befehlsvorrat mit verschiedenen wirkungsvollen Adressierungsmöglichkeiten.

Das System AEG 80–20/4 ist softwarekompatibel zu den anderen AEG 80–20–Systemen und hardwarekompatibel zu allen AEG 80 – Bus- und Geräteschnittstellen.

Die in ihrer Hardwarestruktur begründete Leistungsfähigkeit der AEG 80–20/4 wird dem Benutzer durch die Software voll zugänglich gemacht:

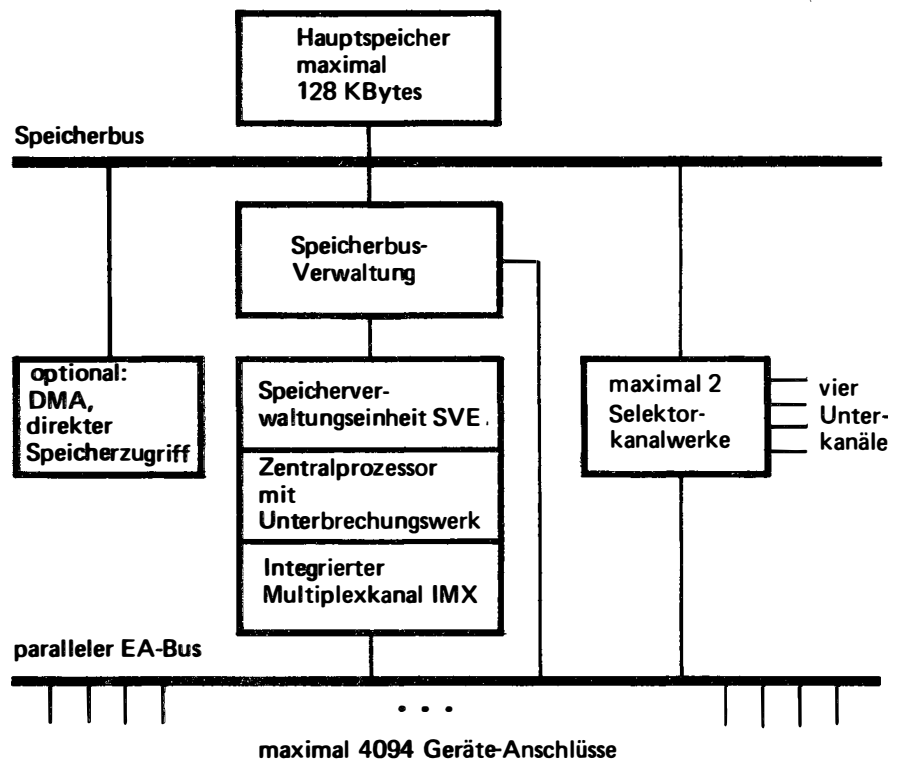
- Das modular aufgebaute Betriebssystem MARTOS–K läßt sich verschiedenen Konfigurationen und Systemanforderungen optimal anpassen. Es ist als Hauptspeicher- oder Peripheriespeichervariante lieferbar und enthält alle für die Prozeßsteuerung erforderlichen Funktionen, ferner die Funktionen zur Bedienung der Anlage in dem integrierten Dienstsysteem PROGI.
- Zur Programmierung in Assemblersprache oder Prozeß-FORTTRAN stehen das unter MARTOS–K laufende Programmiersystem PRODOS und das autonome Programmiersystem PROGA zur Verfügung. PROGA dient auch zum Systemaufbau. Beide Programmiersysteme umfassen die Dienste für Quell- und Objektbehandlung sowie zum Programmtest.

1 Systemübersicht

- Die aus den Systemen ARSI80 und AS80 bestehende Standard-Anwendungssoftware. Mit diesen modularen Programmsystemen lassen sich ohne spezielle Programmierkenntnisse optimal angepaßte Prozeßsteuerungssysteme installieren und ändern.

ZENTRALEINHEIT

Der Aufbau der Zentraleinheit AEG 80-20/4 ist in dem folgenden Bild dargestellt:



Der Zentralprozessor ist über den parallelen EA-Bus mit maximal 4094 Datenperipheriegeräten bzw. Prozeßperipheriebaugruppen verbunden. Über die 16 Datenleitungen des parallelen EA-Bus kann jeweils ein 16-Bit-Halbwort oder ein 8-Bit-Byte ausgegeben werden.

Die automatische Ein- und Ausgabe wird durch den integrierten Multiplexkanal IMX erledigt; dieser wird durch Mikroprogrammfunktionen des Zentralprozessors und den parallelen EA-Bus realisiert.

An den Speicherbus kann in maximal vier Ausbaustufen von 16 oder 32 KBytes Kernspeichermodule ein Hauptspeicher von maximal 128 KBytes als passiver Teilnehmer angeschlossen werden.

Die maximal zwei Selektorkanalwerke lassen den Anschluß von je vier Peripheriespeichern mit hoher Übertragungsgeschwindigkeit oder von schnellen Rechnerkopplungen zu. Sie erhalten die Steuerinformationen vom Rechner über den parallelen EA-Bus.

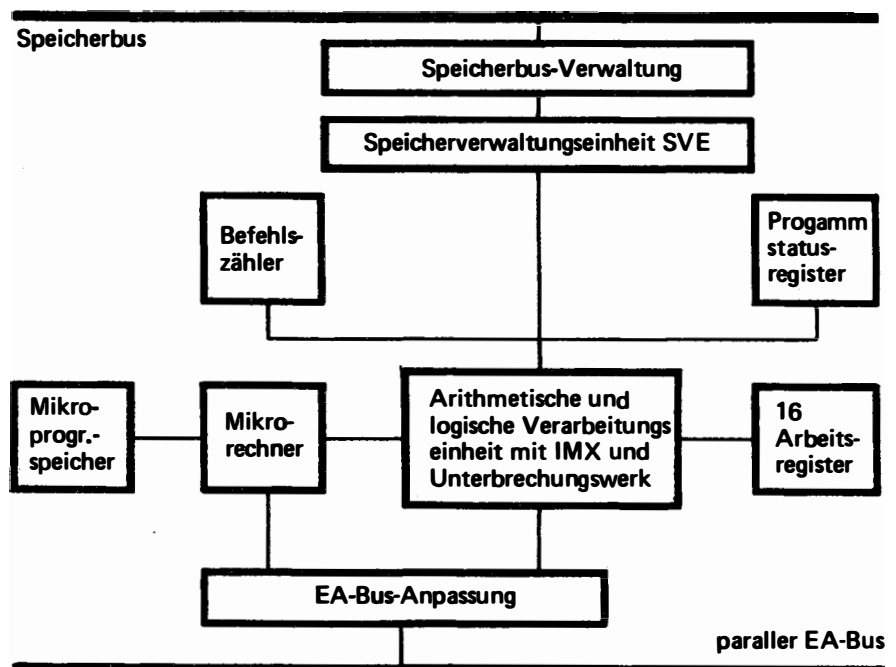
Zur Erhöhung der Ein- und Ausgabeleistung läßt sich an den Speicherbus ein direkter Speicherzugriff (DMA) als weiterer aktiver Teilnehmer anschließen.

2 Zentraleinheit

Die Speicherverwaltungseinheit SVE bildet bei einem Hauptspeicherausbau von mehr als 64 KBytes den Arbeitsadreßraum von 64 KBytes auf den größeren Speicheradreßraum ab. Außerdem schützt sie das Betriebssystem und die Anwenderprogramme untereinander gegen unzulässige Speicher- und Sprungoperationen. Das Betriebssystem steuert die SVE ebenfalls über den parallelen EA-Bus.

2.1 Zentralprozessor

Der Aufbau des Zentralprozessors ist aus folgendem Bild zu entnehmen:



Die Funktionen der Teile des Zentralprozessors gehen aus ihren Bezeichnungen und aus ihrer Anordnung innerhalb des Bildes weitgehend hervor. Ergänzend sind einige Einzelheiten zu erwähnen:

Der Mikroprogrammspeicher ist ein Festwertspeicher (Read Only Memory, ROM). Er enthält als "Firmware" die Mikroprogramme, die den Ablauf

- der Befehle,
- der automatischen Ein- und Ausgabe über den integrierten Multiplexkanal und
- der Unterbrechungen durch Ein- und Ausgabe und eventuelle Fehler

bestimmen.

Die 16 Arbeitsregister zu je 16 Bits nehmen Operanden und Resultate beim Ablauf der Befehle auf. Operanden oder Resultate von Wortlänge (32 Bits) beginnen in Arbeitsregistern mit gerader Adresse. Die Arbeitsregister 1 bis 15 können außerdem zur Indizierung von Adressen und anderen Parametern bei der Befehlsentschlüsselung benutzt werden. In den Befehlen werden die Arbeitsregister in den dafür vorgesehenen 4-Bit-Feldern als Operanden- und Resultatregister bzw. als Indexregister adressiert.

Das 16-Bit-Programmstatusregister enthält in den ersten 8 Bits des Programmstatusworts Angaben über

- Wartezustand,
- Zulässigkeit verschiedener Unterbrechungsarten,
- Unterscheidung von System-Modus und Normalmodus

und in den letzten 4 Bits Angaben über den Ablauf eines der jeweils letzten Befehle, ob z.B. das Resultat positiv, negativ oder Null war, ob es überlief oder einen Übertrag lieferte (Bedingungefeld). Diese Angaben sind zur Programmverzweigung durch bedingte Sprungbefehle ausnutzbar.

Der 16-Bit-Befehlszähler adressiert das erste Halbwort des jeweils nächsten auszuführenden Befehls im gesamten Arbeitsadreseßraum. Befehle beginnen stets auf Halbwortgrenzen, und daher ist das letzte Bit im Befehlszähler immer mit 0 besetzt. Beim Programmwechsel und bei der Ausführung eines Sprungs wird der Befehlszähler neu mit der Adresse des ersten danach auszuführenden Befehls besetzt. Sonst wird der Inhalt des Befehlszählers um 2, 4 oder 6 erhöht, je nachdem, ob der gerade ausgeführte Befehl im Hauptspeicher 1, 2 bzw. 3 Halbwörter belegt.

Die Speicherverwaltungseinheit SVE dient bei einem Ausbau des Hauptspeichers von mehr als 64 KBytes zur Abbildung des 64 KBytes-Arbeitsadreseßraums auf den größeren Speicheradreseßraum und zugleich zum Schutz des Betriebssystems sowie der Anwenderprogramme untereinander gegen unzulässige Speicher- und Sprungoperationen.

In Anlagen bis 64 KBytes kann die Speicherverwaltungseinheit optional für diesen Speicherschutz eingesetzt werden.

2.2 Hauptspeicher

Der Hauptspeicher kann in maximal vier Stufen aus Speichermodulen zu 16 oder 32 KBytes bis zur Maximalkapazität von 128 KBytes ausgebaut werden, wahlweise entweder als Kernspeicher oder Halbleiterspeicher.

Die Halbleitervariante ist preisgünstiger und benötigt weniger Platz. Sie kann optional mit Batteriepufferung (Pufferzeit für 128 KBytes 1/2 Stunde) und mit Fehlerkorrektur ausgestattet werden.

Der Hauptspeicher ist in Halbwörter zu je 16 Bits gegliedert, die jeweils in einem Speicherzyklus von 0,70 μ s gelesen bzw. geschrieben werden. Die Information wird durch je ein Paritätsbit pro Byte gesichert; die Bytes sind einzeln adressierbar.

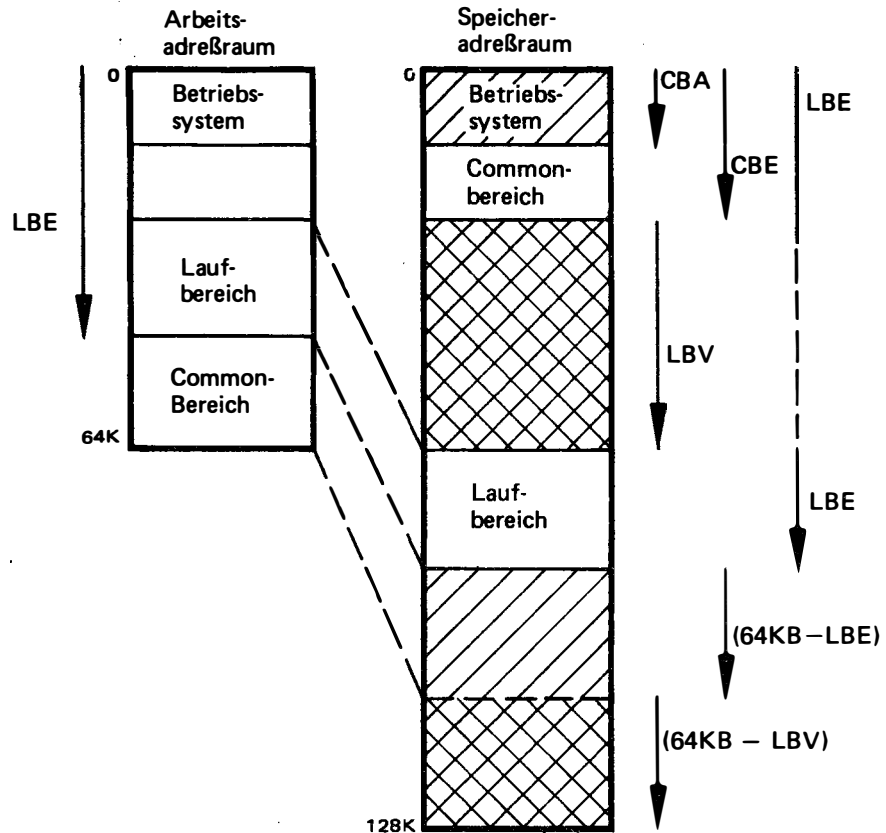
Daten von Halbwort- oder Wortstruktur (16 bzw. 32 Bits lang) sind an die Halbwortgrenzen des Hauptspeichers gebunden; dies gilt auch für die Befehle, die ein, zwei oder drei Halbwörter im Hauptspeicher belegen. Einzelne Bytes können dagegen als linke oder rechts Hälfte von Halbwörtern gelesen und geschrieben werden; die bis zu 16 Bits umfassenden Bitfelder dürfen Halbwortgrenzen überschreiten.

Die Adressen in Befehlen und Indexregistern sind 16 Bits lang, und ihr letztes Bit hat Bytewertigkeit; sie machen daher einen Arbeitsadreseßraum von 64 KBytes zugänglich. Die Speicherverwaltungseinheit SVE bildet ihn mit Hilfe der Laufbereichsverschiebung LBV auf den maximal 128 KBytes umfassenden Speicheradreseßraum ab und ermöglicht mit Hilfe der Parameter

2 Zentraleinheit

- Commonbereichsanfang CBA
- Commonbereichsende CBE
- Laufbereichsende LBE

einen einfachen und doch sehr wirkungsvollen Speicherschutz.



Anwenderprogramme haben ungehinderten Zugang mit Speicher-, Sprung- und Lese-Operationen zum

- Commonbereich (nicht schraffiert), in dem die Arbeitsadressen mit den Speicheradressen identisch sind, und zum
- eigenen Laufbereich (nicht schraffiert), in dem die SVE die Arbeitsadressen durch Addition der Laufbereichverschiebung LBV in Speicheradressen abbildet.

Anwenderprogramme können auch lesend auf das Betriebssystem ohne Veränderung der Arbeitsadressen und auf den ihnen folgenden Speicherbereich der Länge $64K - LBE$ mit Verschiebung der Arbeitsadressen zugreifen. Die im System-Modus laufenden Betriebssystemprogramme dürfen hier auch speichern und springen (einfach schraffiert).

Die doppelt schraffierten Speicherbereiche der Länge LBV und $64K - LBV$ sind beim Programmablauf grundsätzlich nicht erreichbar.

Im System-Modus ablaufende Betriebssystemprogramme können die vier erwähnten Parameter in der SVE einstellen. CBA und CBE werden nur beim Systemstart eingestellt. LBV und LBE werden jeweils beim Erststart oder Fortsetzungsstart eines Anwenderprogramms eingestellt.

Bei der Ein- und Ausgabe werden die betreffenden Speicherbereiche durch Adressen mit Halbwortwertigkeit bezeichnet. Daher genügen 16 Bits zur direkten Adressierung des gesamten Hauptspeichers von 128 KBytes = 64 K Halbwörtern.

2.3 Datenstruktur

In der AEG 80-20/2 werden die Datenformate

- Wort zu 32 Bits
- Halbwort zu 16 Bits
- Byte zu 8 Bits
- Bitfeld zu 1 bis 16 Bits

verwendet.

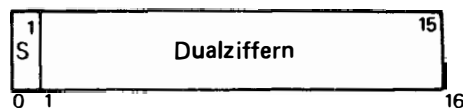
Wörter beginnen im Hauptspeicher auf Halbwortgrenzen, im Zentralprozessor in einem Arbeitsregister mit gerader Adresse.

Halbwörter beginnen im Hauptspeicher auf Halbwortgrenzen, im Zentralprozessor können sie in jedem Arbeitsregister stehen.

Bytes können im Hauptspeicher die linke oder die rechte Hälfte eines Halbworts belegen. Im Zentralprozessor werden Bytes in der rechten Hälfte des jeweiligen Arbeitsregisters verarbeitet.

Bitfelder können im Hauptspeicher an beliebiger Stelle liegen und dabei Byte- und Halbwortgrenzen überschreiten. Im Zentralprozessor werden Bitfelder rechtsbündig innerhalb des jeweiligen Arbeitsregisters verarbeitet.

Festpunktzahlen werden vorwiegend in Halbwörtern dargestellt:

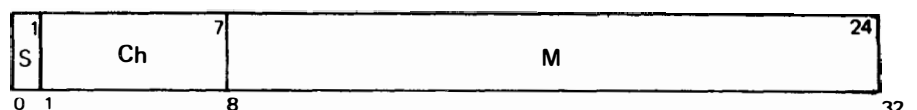


S = Vorzeichenbit: 0 = +; 1 = -

Negative Zahlen erscheinen im B-Komplement. Bei rechtsbündiger Interpretation steht damit der Zahlbereich von -32768 bis +32767 zur Verfügung, der Wert -32768 aus Symmetriegründen jedoch nur eingeschränkt.

Die Stellenwertzuordnung ist beliebig. Bei Bedarf können Festpunktzahlen durch Verarbeitung mit entsprechenden Befehlen als vorzeichenlos im Bereich von 0 bis 65535 behandelt werden. Mehrere solcher Festpunktzahlen können zu einer Zahl höherer Genauigkeit zusammengeschlossen und durch spezielle Befehle entsprechend verarbeitet werden.

Gleitpunktzahlen werden in Wörtern dargestellt:



S = Vorzeichenbit: 0 = +; 1 = -

Die Charakteristik Ch liegt im Bereich $0 \leq Ch \leq 127$. Die Mantisse M ist entweder 0 oder sie liegt bei normalisierten Gleitpunktzahlen im Bereich $1/16 \leq M \leq 1 - 2^{-24}$. Der Wert W einer Gleitpunktzahl ist $W = (-1)^S \cdot 16^{Ch-64} \cdot M$.

2 Zentraleinheit

Die Gleitpunktbefehle setzen normalisierte Gleitpunktoperanden voraus und liefern normalisierte Gleitpunktzahlen als Ergebnisse.

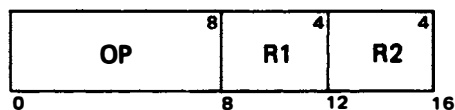
Die ersten 16 Halbwörter des Hauptspeichers fungieren als 8 Gleitpunktregister.

Zur Verarbeitung logischer und alphanumerischer Daten werden vorzugsweise die Datenformate Halbwort oder Byte verwendet, erforderlichenfalls das Bitfeldformat.

2.4 Befehlsformate

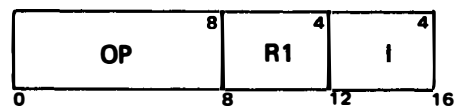
Befehle bestehen aus 1, 2 oder 3 Halbwörtern und sind an die Halbwortstruktur des Hauptspeichers gebunden. Das erste Byte jedes Befehls ist der Operationscode OP. Er legt die Struktur des Befehls selbst fest; außerdem bestimmt er die Herkunft und Struktur der Operanden, deren Verarbeitung sowie Verbleib und Struktur der Ergebnisse. Je nach der Länge der Befehle und der Bedeutung der einzelnen Felder werden folgende Befehlsformate unterschieden:

16R: Register-Register-Befehle



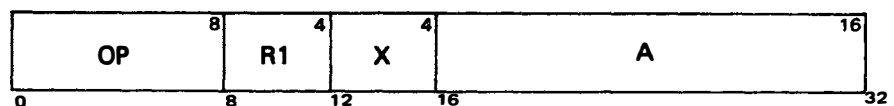
Diese Befehle werden bevorzugt verwendet, weil sie wenig Speicher belegen und besonders schnell ablaufen. Sie dienen vor allem zur arithmetischen oder logischen Verknüpfung des Inhalts zweier Arbeitsregister; meist wird das Ergebnis im Arbeitsregister R1 bereitgestellt (33 Befehle).

16I: Register-Konstantenbefehle



Diese Befehle verarbeiten den Inhalt des Arbeitsregisters R1 mit Hilfe der 4-Bit-Konstanten I (10 Befehle).

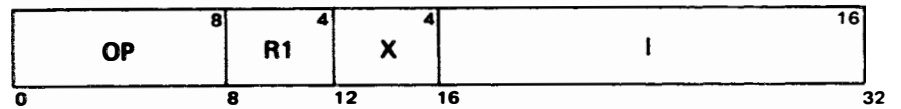
32M: Indizierbare Register-Speicher-Befehle



Diese Befehle dienen zum Transport oder zur rechnerischen Verknüpfung von Daten zwischen einem Arbeitsregister R1 und einem Halbwort, Wort oder Byte im Hauptspeicher. Der Adreßteil A zur Bestimmung des Speicheroperanden kann mit dem Inhalt eines in X bezeichneten Indexregisters indiziert werden (43 Befehle).

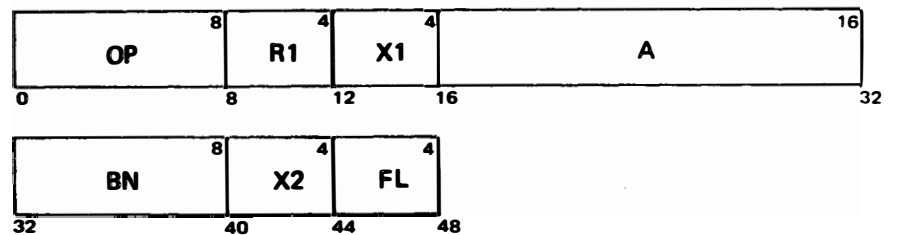
2 Zentraleinheit

32I: Indizierbare Register-Konstanten-Befehle



Bei diesen Befehlen wird die indizierbare Halbwortkonstante I entweder mit dem Inhalt eines Arbeitsregisters R1 rechnerisch verknüpft, oder sie legt z.B. die Anzahl von Shiftschritten fest (25 Befehle).

48MBF: Bitfeldbefehle



Diese Befehle dienen zum Transport von Bitfeldern zwischen einem Arbeitsregister R1 und dem Hauptspeicher. Nach der Auswertung der ersten beiden Halbwörter des Befehls genau so wie bei den 32M-Befehlen legt die mit X2 indizierbare Bitnummer BN und die Feldlänge FL die Lage und Länge des Bitfeldes im Hauptspeicher genau fest (3 Befehle).

2 Zentraleinheit

2.5 Befehlsliste

Nach ihren Funktionen werden die Befehle gegliedert in

- 16 Transportbefehle
- 4 Listenbefehle
- 24 Festpunktbefehle
- 13 Gleitpunktbefehle
- 9 logische Befehle
- 12 Shiftbefehle
- 11 Vergleichs-, Test- und Zählbefehle
- 12 Sprungbefehle
- 4 Systembefehle
- 9 EA-Befehle

114 belegte Befehlscodes insgesamt

Aus den 8 Maskensprungbefehlen werden durch Festlegung bestimmter Masken in zusätzlichen Assembler-Mnemocodes 35 weitere bedingte Sprungbefehle, unbedingte Sprungbefehle und Leerbefehle abgeleitet.

Zahlreiche unbelegte Befehlscodes können als Quasibefehle zur Erweiterung der Befehlsliste um benutzerspezifische Befehle verwendet werden. Sie sind durch entsprechende Programme im Hauptspeicher zu realisieren.

Auch mit dem Systembefehl YEMP können benutzerspezifische Befehle zusätzlich eingeführt werden. Diese müssen durch entsprechende Mikroprogramme in einer Erweiterung des Mikroprogrammspeicher realisiert werden.

Anwenderprogramme rufen mit dem Systembefehl SVC (Supervisor Call) die Systemdienste des Betriebssystems MARTOS—K auf. Alle übrigen Systembefehle und alle EA-Befehle sind nur im Betriebssystem ausführbar, weil es im System-Modus abläuft. In den im Normalmodus ablaufenden Anwenderprogrammen sind außer SVC die Systembefehle und die EA-Befehle nicht ausführbar.

2.6 Eingabe- und Ausgabesystem

Ein im Rechner ablaufendes Anwenderprogramm veranlaßt die Eingabe oder Ausgabe von Daten, indem es mittels eines SVC-Systemdienstaufrufs das Betriebssystem MARTOS—K damit beauftragt und ihm dabei die erforderlichen Parameter übergibt.

Die von einem Peripheriegerät veranlaßte spontane Eingabe von Daten beginnt mit einem Unterbrechungssignal des Geräts an den Rechner. Das Betriebssystem MARTOS—K wickelt diese Eingabe dann genau so ab wie die von einem Anwenderprogramm in Auftrag gegebene Dateneingabe; es sorgt dann noch für den Ablauf des Programms, das diese Daten zu verarbeiten hat.

Jede Dateneingabe oder -ausgabe besteht aus einem Abschnitt oder aus mehreren Abschnitten. Jeder Abschnitt besteht im allgemeinen aus mehreren Datenzyklen. Jeder Datenzyklus überträgt ein Zeichen von 16 Bits oder von 8 Bits zwischen Peripheriegerät und Hauptspeicher.

Das Betriebssystem MARTOS—K stellt im Hauptspeicher Versorgungsblöcke von je 6 Halbwörtern als "Kanalbefehle" bereit, die je einen Abschnitt einer Dateneingabe bzw. -ausgabe bestimmen.

Die Kanalbefehle bestehen aus den Parametern, die

- das Gerät
- seinen Betriebszustand
- die Betriebsart der Ein- bzw. Ausgabe
- die Lage und Länge der EA-Daten im Hauptspeicher und
- die Meldungen eventueller Störungen

betreffen. Diese Parameter bleiben während der Ein- bzw. Ausgabe zum Teil unverändert; einige Parameter werden auch vom Integrierten Multiplexkanal dem Ablauf der Ein- bzw. Ausgabe entsprechend verändert.

Der Integrierte Multiplexkanal IMX besteht aus Mikroprogrammfunktionen des Zentralprozessors zur automatischen Abwicklung der Dateneingabe und -ausgabe über den parallelen Ein- und Ausgabe-Bus (EA-Bus), an den bis zu 254 EA-Geräte angeschlossen werden können. Die maximal 16 Bits eines ein- bzw. auszugebenden Zeichens werden in einem Datenzyklus durch den EA-Bus parallel übertragen.

Ein Datenzyklus wird meist von einem EA-Gerät eingeleitet, das durch ein Unterbrechungssignal über den EA-Bus an den IMX meldet, daß es ein Zeichen zur Eingabe zu übergeben hat bzw. zur Ausgabe übernehmen kann.

Der IMX ermittelt dann durch ein Signal über den EA-Bus das betreffende Gerät, um zur Übertragung des Zeichens eine Verbindung zwischen dem Gerät und dem Hauptspeicher herzustellen.

Der IMX kann ebenfalls Datenzyklen zur Übertragung von Steuerdaten an die Geräte einleiten.

Die Verbindung zwischen einem EA-Gerät und dem Hauptspeicher über den EA-Bus wird für jeden Datenzyklus neu hergestellt und dann wieder gelöst. Zwischen zwei Datenzyklen eines EA-Geräts steht der EA-Bus für die Datenzyklen anderer EA-Geräte zur Verfügung. Daher können mehrere Eingabe- und Ausgabevorgänge quasi parallel ablaufen. Ihre maximale Zahl hängt von der Datenrate der EA-Geräte und von der Arbeitsgeschwindigkeit des IMX und des EA-Bus ab.

Die Unterbrechungssignale zur Einleitung eines Datenzyklus bewirken im allgemeinen keine Unterbrechung des gerade ablaufenden Programms, wenn nur zwischen dem Ablauf zweier Befehle ein Eingabe- bzw. Ausgabezeichen zwischen EA-Gerät und Hauptspeicher durch den Zentralprozessor geschleust wird. Bei einigen länger ablaufenden Befehlen kann dies auch an dafür bestimmten Unterbrechungspunkten ihres Ablaufs geschehen.

Bei Beendigung oder Störung eines Eingabe- oder Ausgabevorgangs wird dagegen das gerade ablaufende Programm nach Beendigung eines Befehlsablaufs unterbrochen. Ein Programm des Betriebssystems MARTOS—K erledigt verschiedene Verwaltungsaufgaben:

- das Programm, für das die Dateneingabe oder -ausgabe erledigt wurde, kann nun wieder weiterarbeiten.
- eventuell vorliegende Ein- bzw. Ausgabeaufträge an das betreffende EA-Gerät müssen erledigt werden.
- eventuell aufgetretene Störungen müssen gemeldet und vom Bedienungspersonal behoben werden.

2 Zentraleinheit

Bei der Dateneingabe oder -abgabe über Selektorkanäle wird die Verbindung zwischen Selektorkanal und Gerät wegen der hohen Übertragungsgeschwindigkeit der peripheren Speicher bzw. der schnellen Rechnerkopplung während eines Abschnitts ständig aufrecht erhalten. Daher kann an einem Selektorkanal immer nur ein Gerät arbeiten. Währenddessen eingehende weitere Aufträge an den Selektorkanal werden vom Betriebssystem für jeden der vier Unterkanäle getrennt gepuffert.

2.7 Unterbrechungssystem

Das Unterbrechungssystem der AEG 80—20/4 entspricht den Anforderungen der Prozeßanwendungen, daß der Rechner auf alle seine Aufgaben betreffenden inneren und äußeren Ereignisse unverzüglich reagiert. Die dem Zentralprozessor zugehenden Unterbrechungssignale bewirken entweder eine Unterbrechung des gerade ablaufenden Programms (Unterbrechungssignale 2. Art), damit ein Programm des Betriebssystems MARTOS—K die bei dem verursachenden Ereignis erforderlichen Reaktionen einleitet, oder sie bewirken ohne Programmunterbrechung die Erledigung eines Datenzyklus durch den IMX (Unterbrechungssignale 1. Art). Die Transporte von Zeichen zwischen Selektorkanal und Hauptspeicher berühren den Zentralprozessor nur insofern, als die Speicherzyklen für den Selektorkanal vorrangig erledigt werden; sie benötigen daher keine Unterbrechungssignale.

Programmunterbrechungen laufen folgendermaßen ab:

- Das Unterbrechungsmikroprogramm legt das Programmstatuswort an einer Stelle im Hauptspeicher ab, die der Unterbrechungsursache zugeordnet ist. Dann lädt es das Programmstatusregister mit einem neuen Programmstatuswort, das ebenfalls aus einer der Unterbrechungsursache zugeordneten Stelle des Hauptspeichers entnommen wird.
- Das so gestartete Betriebssystemprogramm legt den Inhalt der 16 Arbeitsregister im Registerrettbereich des unterbrochenen Programms im Hauptspeicher ab und beginnt mit der Erledigung seiner Aufgabe.

Das neue Programmstatuswort eines Unterbrechungsprogramms enthält meist eine Sperre gegen Unterbrechungen derselben Ursache, die zu seinem Start führte; dadurch wird vermieden, daß endlose Schleifen entstehen. Die im Hauptspeicher bereitliegenden Programmstatuswörter, die je nach der Unterbrechungsursache in das Programmstatusregister geladen werden, bleiben unverändert. Dadurch reagieren die Unterbrechungsprogramme auf die Unterbrechungsursachen stets in derselben Weise. Bei zeitlichem Zusammentreffen von Unterbrechungssignalen auf Grund von verschiedenen Ursachen wird die Reihenfolge ihrer Erledigung durch ihre Priorität geregelt.

Nach dem Ursachenbereich werden interne und externe Unterbrechungssignale voneinander unterschieden.

2.7.1 Interne Unterbrechungen

Die internen Unterbrechungssignale bewirken Programmunterbrechungen; bei einigen Unterbrechungsursachen geringerer Priorität kann die Unterbrechung durch entsprechende Besetzung des Programmstatusworts abgeschaltet werden. Nach den Ursachen werden unterschieden:

- **Arithmetikalarne (abschaltbar):**
Fehler bei der Festpunkt-Division, weil der Divisor einen zu kleinen Betrag oder den Wert 0 hat.
Die Charakteristik eines Gleitpunktergebnisses überschreitet den größten zulässigen Wert 128 oder der Divisor einer Gleitpunktdivision ist 0.
- **Maschinenalarne (abschaltbar):**
Paritätsfehler beim Transport von Daten zwischen Zentralprozessor und Hauptspeicher.
Speicherfehler
Spannungsausfall
Spannungswiederkehr
Diese Alarmursachen werden durch die Besetzung des Bedingungsfeldes im Programmstatuswort voneinander unterschieden. Bei Spannungsausfall steht 1 ms zur Verfügung, um den Wiederstart bei Spannungswiederkehr vorzubereiten und den Zentralprozessor in den Wartezustand zu versetzen.
- **System-Modus-Alarm (abschaltbar):**
Unzulässige Benutzung eines nur im System-Modus ausführbaren System- oder EA-Befehls im Normalmodus. Der betreffende Befehl wird nicht ausgeführt, aber der Befehlszähler ist bereits erhöht.
- **Warteschlangen-Alarm (nicht abschaltbar):**
Unterbrechungen 2. Art, die die Beendigung oder eine Störung eines EA-Vorgangs oder eine spontane Eingabe melden, führen zur Eintragung eines "Unterbrechungszeigers" in eine Warteschlange. Ist der Speicherbereich für die Warteschlange zu klein, und gelingt es daher nicht, sie schnell genug abzuarbeiten, so läuft sie über. Der überzählige Unterbrechungszeiger kann zwar noch gespeichert werden, aber dann wird das entsprechende Unterbrechungssignal erzeugt.
- **Quasibefehls-Unterbrechung (nicht abschaltbar):**
Das entsprechende Unterbrechungssignal wird erzeugt, wenn ein nicht belegter Befehlscode entschlüsselt wird. Das Unterbrechungsprogramm prüft, ob unter diesem Befehlscode ein zusätzlicher benutzerspezifischer Befehl registriert ist, der dann als Unterprogramm durchlaufen wird.
- **Systemaufruf-Unterbrechung (nicht abschaltbar):**
Der Befehl SVC erzeugt das entsprechende Unterbrechungssignal. Dadurch wird der Übergang von einem im Normalmodus ablaufenden Anwenderprogramm zu einem im System-Modus ablaufenden Programm des Betriebssystems MARTOS-K ermöglicht.
Der Befehl SVC wirkt im übrigen wie ein Unterprogrammsprungbefehl; das Feld R1 in seinem 32M-Befehlsformat dient zur mikroprogrammierten Verzweigung in den gewünschten Systemdienst; der indizierbare Adreßteil wird teils als Parameter, teils als Hauptspeicheradresse des Parameterblocks für den Systemdienst verwendet.

2 Zentraleinheit

2.7.2

Externe Unterbrechungen

Externe Unterbrechungssignale 2. Art melden

- die Beendigung eines Ein- oder Ausgabevorgangs
- die Störung eines Ein- oder Ausgabevorgangs
- eine spontane Eingabe;
dazu gehört z.B. auch das Taktsignal zur Führung der Systemuhr.

Ein externes Unterbrechungssignal 1. Art meldet

- den Beginn eines Datenzyklus zur Eingabe oder Ausgabe eines Zeichens innerhalb eines Abschnitts eines Ein- bzw. Ausgabevorgangs.

Externe Unterbrechungssignale 1. Art bewirken im allgemeinen keine Programmunterbrechung, sondern nur die Abwicklung eines Datenzyklus durch den IMX; sie können jedoch Unterbrechungssignale 2. Art auslösen.

Externe Unterbrechungssignale 2. Art bewirken eine Programmunterbrechung zum Start eines Betriebssystemsprogramms.

Externe Unterbrechungen 1. Art und 2. Art sind abschaltbar.

Externe Unterbrechungen 1. Art laufen ohne Mitwirkung von MARTOS—K im wesentlichen folgendermaßen ab:

- Der IMX ermittelt nach Eintreffen des Unterbrechungssignals 1. Art über den EA-Bus die Adresse des Geräts, von dem das Unterbrechungssignal ausging.
- Auf Grund der Geräteadresse wird aus dem "Unterbrechungsvektor" im Hauptspeicher der zu dem Gerät gehörige "Unterbrechungszeiger" entnommen.
- An Hand des Unterbrechungszeigers wird erkannt, ob eine "direkte Unterbrechung" z.B. zur Meldung eines Zeit-Takts oder einer spontanen Eingabe vorliegt. In diesem Fall wird der Unterbrechungszeiger in die Warteschlange eingetragen, ein Unterbrechungssignal 2. Art erzeugt und die externe Unterbrechung 1. Art beendet.
- Im andern Fall wird die Verbindung mit dem Gerät über den parallelen EA-Bus hergestellt. Mit Hilfe des Unterbrechungszeigers wird auf den zum Gerät gehörigen Kanalbefehl zugegriffen.
- Meldet das Gerät eine Störung, so wird die Meldung darüber in den Kanalbefehl sowie der Unterbrechungszeiger in die Warteschlange eingetragen. Nach Erzeugung eines Unterbrechungssignals 2. Art wird die Unterbrechung 1. Art beendet.
- Im Normalfall wird im Kanalbefehl die Länge des EA-Blocks heruntergezählt und die Hauptspeicheradresse für das ein- bzw. auszugebende Zeichen ermittelt und hochgezählt. Das Zeichen wird in der entsprechenden Richtung zwischen Gerät und Hauptspeicherzelle über den parallelen EA-Bus transportiert.
- Ist die Blocklänge auf 0 heruntergezählt oder wird das Endezeichen der Ein- bzw. Ausgabe erkannt, so wird zur Beendigung des EA-Abschnitts der Unterbrechungszeiger in die Warteschlange eingetragen und ein Unterbrechungssignal 2. Art erzeugt.
- Die Verbindung mit dem Gerät über den EA-Bus wird gelöst. Die Unterbrechung 1. Art ist damit beendet.

Die an alle AEG 80 – Rechner anschließbaren peripheren Einheiten ermöglichen

- die Verbindung mit einem technischen Prozeß außerhalb des Rechners, um dessen Verlauf zu messen und zu steuern
- die Korrespondenz zwischen Bedienpersonal und Rechner
- die vom Verlauf eines technischen Prozesses unabhängige Dateneingabe und -ausgabe
- die Speicherung von Daten auf peripheren Speichern
- die Datenfernübertragung einschließlich Rechnerkopplung.

Die meisten dieser Geräte werden an den parallelen EA-Bus angeschlossen; der EA-Bus läßt insgesamt eine Übertragungsrate von etwa 35000 Datenzyklen/s zu. Bei jedem Datenzyklus wird ein Zeichen von 16 oder 8 Bits übertragen. Periphere Speicher mit hoher Übertragungsrate und schnelle Rechnerkopplungen werden an die maximal zwei Selektorkanäle mit je vier Unterkanälen angeschlossen; ein Selektorkanal kann bis zu 700 kbyte/s übertragen.

Die Peripheriegeräte werden unterteilt in

- Prozeßperipherie,
- Datenperipherie und
- Datenübertragungsperipherie.

3.1 Prozeßperipherie

Die Prozeßperipherie dient

- zum Messen technisch-physikalischer Größen eines externen technischen Prozesses, und
- zum Steuern des technischen Prozesses auf Grund der gemessenen und rechnerisch verarbeiteten Werte.

Sowohl bei der Eingabe als auch bei der Ausgabe werden Digital- und Analogbaugruppen verwendet. Folgende Prozeßperipheriebaugruppen sind anschließbar:

- Digitalausgabe
- Digitaleingabe für statische Vorgänge
- Digitaleingabe für dynamische Vorgänge:
Spontane Digitaleingabe und Unterbrechungseingabe
- Digital-Ein- und -Ausgabe
- Byte-Ein- und -Ausgabe
- Analogausgabe
- Digitalpotentiometer
- Analogeingabe
- Meßstellenumschalter
- Differenzzähler
- Realzeituhr

3 Periphere Einheiten

- Single CAMAC-Crate-Controller
- Vorschalteinheit für Analogeingabe
- Thermoelemente-Bruchüberwachung
- Spannungsteiler
- Strommeßwiderstand.

Einige von ihnen können mit bis zu acht Varianten verschiedener Leistungsmerkmale den Anforderungen der technischen Prozesse angepaßt werden.

3.2 Datenperipherie

Zur Korrespondenz zwischen Bedienpersonal und Rechner, zur prozeßunabhängigen Datenein- und -ausgabe sowie zur Datenspeicherung zusätzlich zum Hauptspeicher können folgende EA-Geräte über den parallelen EA-Bus an den IMX des Zentralprozessors angeschlossen werden:

- Fernschreiber
- Drucker
- Sichtgeräte
- Lochstreifenstanzer
- Lochstreifenleser
- Lochkartenstanzer
- Lochkartenleser
- Trommelplotter
- Magnetbandkassettenpeicher
- Magnetfolienspeicher.

Einige dieser Geräte können mit bis zu vier Varianten oder Ausbaustufen den verschiedenen Betriebsanforderungen angepaßt werden.

An die maximal zwei Selektorkanäle können jeweils bis zu vier der folgenden peripheren Speicher angeschlossen werden:

- Magnetbandspeicher
- Kassettenplattenspeicher
- Wechselplattenspeicher
- Festkopfspeicher.

3.3 Datenfernübertragung

Zur Datenfernübertragung zwischen Rechner und entfernt aufgestellten EA-Geräten oder zur Rechnerkopplung dienen die folgenden Geräte. Sie unterscheiden sich nach

- der zu überbrückenden Entfernung,
- der zu verwendenden Übertragungsprozedur
- dem verfügbaren Übertragungsweg
- der Übertragungsrate
- dem Anteil der Übertragungszeit zur Gesamtzeit und
- der Zeit, nach der eine angeforderte Datenübertragung beginnen muß.

Diese sehr unterschiedlichen Anforderungen werden durch folgende Datenübertragungseinrichtungen mit entsprechendem technischen Konzept erfüllt:

- Asynchrone Fernbetriebseinheiten
- Synchrone Fernbetriebseinheiten
- Universelle Fernbetriebseinheiten

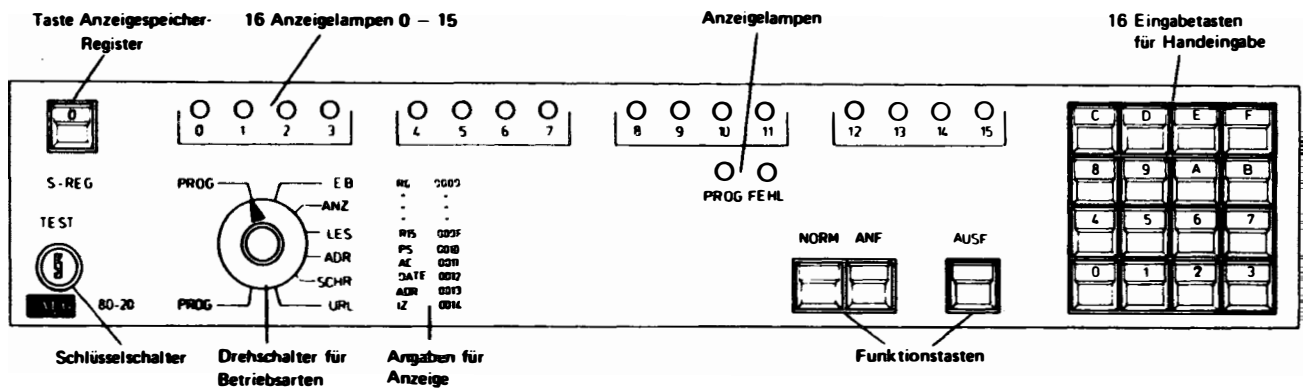
3 Periphere Einheiten

- Telex-Fernbetriebseinheiten
- Datenmodems
- Gleichstrom-Datenübertragungseinrichtungen
- Telegrafie-Umsetzer
- Fernlade-Einrichtungen
- Datenkonzentratoren.

Auch diese Geräte sind mit bis zu fünf verschiedenen Varianten einsetzbar, um die unterschiedlichen Anforderungen der Datenfernübertragung zu erfüllen.

Über die maximal zwei Selektorkanalwerke lassen sich außerdem schnelle Rechnerkopplungen realisieren.

Zur Bedienung der AEG 80-20/4 wird das Bedienfeld



über den parallelen EA-Bus angeschlossen.

Das Bedienfeld bietet viele Anzeige-, Eingabe- und Eingriffsmöglichkeiten; dies bedingt, daß es wie ein Datenperipheriegerät angeschlossen werden muß und außerdem über spezielle Steuerleitungen direkt mit dem Zentralprozessor verbunden ist.

Das Betriebssystem MARTOS-K (Multi Access Real-Time Operating System/ Kompakt) baut auf den Hardware-Eigenschaften der AEG 80-20/4 auf und vervollständigt sie durch die Funktionen

- Unterbrechungs- und EA-Verwaltung
- Programmverwaltung
- Laufbereichs- und Transferverwaltung
- Zeitverwaltung
- Initialisierung
- Systemdienste und
- Konvertierungsroutinen.

Das folgende Bild läßt das Zusammenwirken einiger zentraler MARTOS-K-Funktionen erkennen:

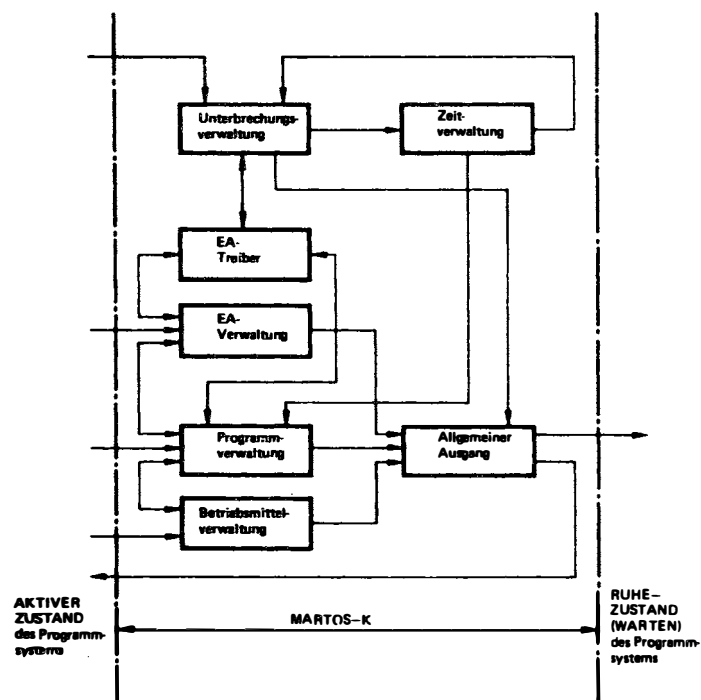


Bild 5.1 Funktionen des MARTOS-K

MARTOS-K ist das für alle AEG 80-20 - Systeme ausgelegte stark modularisierte Betriebssystem. Es ist als Hauptspeichersystem oder als Peripheriespeichersystem mit Umlaufspeichern als Peripheriespeicher verfügbar. Je nach der Konfiguration aus Peripheriegeräten, EA-System und Hauptspeicherausbau sowie nach der Auswahl von tatsächlich benötigten Einzelfunktionen, der Speicherein-

5 Betriebssystem MARTOS-K

teilung und sonstigen anwendungsbezogenen Parametern wird eine entsprechende Auswahl von MARTOS-K-Moduln bzw. ihrer Varianten zu einem funktionsfähigen Betriebssystem gebunden. Die vorliegende Beschreibung betrifft die Gesamtheit der für die AEG 80-20/4 in Frage kommenden Betriebssystem-Moduln, ohne sie ausdrücklich einzeln zu erwähnen.

5.1 Unterbrechungs- und EA-Verwaltung

Das Unterbrechungswerk der Zentraleinheit AEG 80-20/4 besitzt

- 7 interne Unterbrechungseingänge und
- maximal 4096 externe Unterbrechungseingänge.

In der Beschreibung des Unterbrechungssystems sind die sieben internen Unterbrechungsursachen bereits erläutert. Die Unterbrechungsverwaltung des Betriebssystems MARTOS-K behandelt sie sinngemäß verschieden:

- Ein fehlerhaft ablaufendes Programm wird abgebrochen, und auf dem Bedienungsfernseher wird eine entsprechende Alarm-Meldung ausgegeben bei
 - Speicherschutzverletzung
 - fehlerhafter Festpunkt-Division
 - Überlauf der Charakteristik von Gleitpunktergebnissen
 - System-Modus-Alarm wegen unzulässigen Aufrufs eines nur im System-Modus ausführbaren Befehls im Normalmodus
 - Quasibefehls-Unterbrechung, wenn der Quasibefehlscode nicht durch einen zusätzlichen anwendungsbezogenen Befehl belegt ist; sonst wird der Befehl von außen nicht erkennbar wie ein Unterprogramm ausgeführt.
- Alle Programme werden angehalten, und dieser "Systemhalt" wird auf dem Bedienfeld durch eine Kennung angezeigt bei
 - Paritätsfehler, Speicherfehler, Spannungsausfall und Spannungswiederkehr
 - Warteschlangen-Alarm
- Bei Systemdienstaufruf mit SVC wird der Systemdienst von außen nicht erkennbar wie ein Unterprogramm durchlaufen.

In der Beschreibung des Unterbrechungssystems ist erläutert, wie der IMX die externen Unterbrechungen 1. Art autonom ohne Mitwirkung von MARTOS-K bearbeitet.

Ein dabei erzeugtes Unterbrechungssignal 2. Art führt zu einer Programmunterbrechung und zum Start eines Programms der EA-Verwaltung. Dieses entnimmt den Unterbrechungszeiger aus der Warteschlange, ermittelt daraus die Ursache der Unterbrechung zweiter Art und handelt sie dementsprechend verschieden ab:

- Eine "direkte Unterbrechung" zur Meldung eines Zeit-Takts führt zum Start eines Programms, das zu dem betreffenden Zeitpunkt ablaufen soll.
- Eine "direkte Unterbrechung" zur Meldung einer spontanen Eingabe führt zu deren Erledigung wie eine durch Systemdienstaufruf in Auftrag gegebene Eingabe.
- Eine Gerätestörung wird zu beheben versucht oder dem Bedienpersonal zur Behebung gemeldet. Das Gerät bleibt gesperrt, solange die Störung nicht behoben ist.
- Bei Beendigung eines EA-Abschnitts wird ein daran eventuell geketteter weiterer EA-Abschnitt unmittelbar eingeleitet. Sonst wird das Gerät freigegeben, um für einen weiteren EA-Auftrag zur Verfügung zu stehen.

Eingegebene Daten werden dem Anwenderprogramm zur Verarbeitung gemeldet; ein geräumter Ausgabebereich wird dem Anwenderprogramm zur erneuten Belegung mit Ausgabedaten gemeldet.

5.2 Programmverwaltung

Objekte der Programmverwaltung sind

- Programme als im Speicher befindliche Befehlsfolgen, und
- (Rechen)prozesse als Abläufe dieser Programme.

Mehrere zeitlich aufeinanderfolgende Abläufe desselben Programms sind verschiedene Prozesse.

Ein Programm bzw. ein Prozeß über dem Programm befindet sich stets in genau einem der folgenden Zustände:

- **laufbereit:**
Dem Prozeß sind alle benötigten Betriebsmittel zugeteilt, und er kann laufen, wenn er unter allen laufbereiten Prozessen die höchste Priorität hat.
- **laufend:**
Dem Prozeß sind alle benötigten Betriebsmittel einschließlich des Zentralprozessors zugeteilt, weil er unter allen laufbereiten Prozessen die höchste Priorität hat.
- **wartend:**
Dieser Prozeßzustand ist vielfach zu unterscheiden nach Wartegründen, d.h. der Prozeß wartet auf
 - Betriebsmittel, d.h. Speicherbereiche oder EA-Geräte
 - Zeit, von der ab er laufen soll
 - Ereignis, z.B. Zeit-Takt oder spontane Dateneingabe
 - Freigabe durch einen anderen Prozeß.
- **ruhend:**
Für das Programm liegt kein Auftrag vor, d.h. es existiert kein Prozeß. Durch einen Auftrag an das Programm könnte aber jederzeit ein Prozeß beginnen.
- **gesperrt:**
Wegen eines Fehlers, z.B. bei der Festpunktdivision, wurde ein Prozeß abgebrochen. Für das ihm zugrunde liegende Programm wird kein Auftrag angenommen.

Zur Verwaltung der Programme bzw. Prozesse wird jedes Programm durch eine Programmnummer identifiziert. Die maximal zulässige Anzahl von Programmen und damit die maximale Programmnummer ≤ 127 ist eine vom Anwender zu bestimmende Systemgröße. Die Dringlichkeit der Programme wird durch ihre Priorität bestimmt. Je kleiner deren Zahlwert ≤ 127 ist, umso größeren Vorrang besitzt das betreffende Programm gegenüber anderen. Verschiedene Programme können dieselbe Priorität haben.

Für jedes dem Betriebssystem MARTOS-K gemeldete Programm wird ein Speicherbereich zur Verwaltung eingerichtet, in dem u.a. die Programmnummer und die Priorität verzeichnet sind. Die Programmverwaltung vermerkt darin z.B. auch, ob ein Programm ruht oder gesperrt ist bzw. ob darüber gerade ein Prozeß existiert.

5 Betriebssystem MARTOS-K

Außerdem trägt die Programmverwaltung die Prozesse in "Ketten" ein. Diese Ketten sind geordnete Listen von Programmnummern; für jeden Prozeßzustand bzw. für jeden Wartegrund existiert eine besondere Kette und jeder Prozeß befindet sich immer nur in genau einer Kette. Ändert sich der Zustand eines Prozesses, so hängt die Programmverwaltung die Programmnummer des ihm zugrundeliegenden Programms aus einer Kette in eine andere Kette um.

Die Kette der laufbereiten Prozesse ist nach Priorität geordnet, bei gleicher Priorität nach der Reihenfolge der Eintragung. Die Wartezeitkette ist nach Wartezeiten geordnet; alle übrigen Ketten sind nach der Reihenfolge der Eintragung geordnet.

Mit der Programmverwaltung eng verbunden sind

- **die Laufbereichsverwaltung:**
In einem Peripheriespeichersystem befinden sich "residente" Programme ständig in ihrem Hauptspeicherbereich; "transiente" Programme befinden sich dagegen im allgemeinen auf dem Peripheriespeicher und werden zum Ablauf in den ihnen zugeordneten Hauptspeicherbereich transferiert. Ein solcher Hauptspeicherbereich ist meist mehreren transienten Programmen zugeordnet, kann aber jeweils nur einem dieser Programme als Laufbereich zur Verfügung stehen. Aufgabe der Laufbereichsverwaltung ist die Zuteilung, Belegung und Freigabe dieser Laufbereiche.
- **die Transferverwaltung:**
In einem Peripheriespeichersystem sind Programme und Objektdaten zwischen Peripheriespeicher und Hauptspeicher in beiden Richtungen zu transferieren. Die Transferverwaltung koordiniert Programmtransfers nach Priorität und in zweiter Linie nach Auftragsreihenfolge. Die der Transferverwaltung nicht unterstehenden Datentransfers werden nach den Programmtransfers erledigt.
- **die Zeitverwaltung:**
Die Zeitverwaltung wird von einem netzfrequenzabhängigen Zeit-Takt abgeleitet, wenn kein Differenzzähler DZR und keine Realzeituhr RZU zur Verfügung steht. Ihre Aufgaben sind
 - Führung der Systemzeit, z.B. zur Angabe der jeweiligen Uhrzeit bei der Protokollierung von Ereignissen oder zur Kalenderführung
 - Bearbeitung der Wartezeit von Prozessen
 - Zeitüberwachung der EA-Geräte.

5.3 Systeminitialisierung

Nach dem erstmaligen Start oder Wiederstart des Betriebssystems müssen dessen Verwaltungslisten im Hauptspeicher und alle angeschlossenen EA-Geräte in einen definierten Grundzustand versetzt werden.

Dazu dient das Programm zur Systeminitialisierung; nach seinem Ablauf befindet sich die Rechenanlage im prozeßgekoppelten Betrieb.

An die Systeminitialisierung sind weitere Initialisierungsroutinen anschließbar, um auch Anwenderprogramme in einen definierten Grundzustand zu versetzen.

5.4 Systemdienste

Von den im Normalmodus ablaufenden Anwenderprogrammen aus können Funktionen des Betriebssystems MARTOS-K, die ganz oder teilweise im System-Modus ablaufen, nur mit dem Systemdienstauf Ruf SVC (Supervisor Call) aufgerufen werden. Ein 4-Bit-Feld des SVC-Befehls enthält die Dienstnummer zur Spezifizierung eines von maximal 16 Systemdiensten. Der indizierbare 16-Bit-Adreßteil des SVC-Befehls wird bei manchen Systemdiensten direkt als Parameter verwendet; bei den anderen Systemdiensten bildet er die Anfangsadresse des Parameterblocks im Hauptspeicher.

Zehn Dienstnummern sind fest belegt; weitere Dienstnummern sind für optionale Systemdienste reserviert. Nicht belegte Dienstnummern können mit gewissen Einschränkungen zum Aufruf zusätzlicher anwendungsorientierter Systemdienste benutzt werden. Die meisten Systemdienste stehen allen Anwenderprogrammen zur Verfügung; einige Systemdienste dürfen nur von privilegierten Programmen benutzt werden. Mit der Anmeldung beim Betriebssystem wird ein Programm als "privilegiert" bzw. "nicht privilegiert" deklariert.

Fest belegt sind die Dienstnummern:

- 0 Änderung der Priorität und Erlaubnis bzw. Verbot sekundärer Unterbrechungen
- 1 Ein- und Ausgabe über Datenperipherie
- 2 Belegung und Freigabe von Pufferbereichen im Hauptspeicher
- 3 Aktivierung, Beendigung und Versetzung in den Wartezustand von Prozessen
- 4 Ausgabe einer Fehlermeldung
- 6 Be- und Entlastung einer Synchronisationsvariablen
- 7 Rechnerkopplung: Senden und Empfangen von Daten
- 11 Reservieren von Datenperipheriegeräten für Prozesse bzw. Freigeben
- 13 Weckerdienst zum Festlegen von Wartezeiten für Prozesse
- 15 Ein- und Ausgabe über Prozeßperipherie

5 Betriebssystem MARTOS-K

5.5 Laufzeitorganisation

In FORTRAN geschriebene Anwenderprogramme benötigen während ihres Ablaufs Standardprogramme, die jeweils bestimmte FORTRAN-Sprachelemente realisieren. Diese Standardprogramme bilden die FORTRAN-Laufzeitorganisation.

Fast jedes FORTRAN-Programm benötigt folgende Standardprogramme, die daher zum Ablauf von FORTRAN-Programmen stets geladen werden:

- FORTRAN IV – Initialisierungsroutine
- FORTRAN IV – Passivierungsroutine
- FORTRAN IV – formatierte EA
- FORTRAN IV – unformatierte EA
- MARTOS–K, Umwandlung Kartencode nach ASCII
- MARTOS–K, Konvertierungspaket
- FORTRAN IV – STOP-Routine
- FORTRAN IV – PAUSE-Routine
- FORTRAN IV – Erweiterung Peripheriespeichertransfer

Weitere Standardprogramme werden beim Laden von FORTRAN einzeln dazugebunden, soweit sie benötigt werden. Dies sind Bibliotheksprogramme zur Funktionsberechnung für die Variablentypen

- REAL und INTEGER
- DOUBLE
- COMPLEX

Die dritte Gruppe der FORTRAN-Standardprogramme realisiert den Sprachumfang von Prozeß-FORTRAN und wird gegebenenfalls geschlossen mit dem Anwenderprogramm geladen und gebunden:

- Task-Organisation
- Datei-Verarbeitung
- Shift-Operationen
- logische Operationen
- Bit-Operationen

Die Standardprogramme der FORTRAN-Laufzeitorganisation sind mehrfach benutzbar.

5.6 Konvertierungsroutinen

MARTOS–K enthält auch Programme zur Umwandlung von Daten zwischen verschiedenen Darstellungen. Diese Konvertierungsroutinen laufen im Normalmodus ab und werden daher mit normalen Unterprogramm-Sprungbefehlen aufgerufen. Sie sind mehrfach benutzbar und bewirken paarweise die Umwandlung zwischen zwei Darstellungen in je einer Richtung:

- duale Festpunktzahl/dezimale Zahl in ASCII-Zeichen
- duale Gleitpunktzahl/dezimale Zahl in ASCII-Zeichen
- duale Festpunktzahl/sedezimale Zahl in ASCII-Zeichen
- duale Festpunktzahl/duale Gleitpunktzahl
- Lochkartencode KC4K/ASCII-Code
- Lochkartenzeichen in 16 Bits / in 12 Bits
- EBCDIC-Code/ASCII-Code

“ASCII-Zeichen” bzw. “-Code” bedeutet Darstellung von Zeichen im 7-Bit-Code nach DIN 66003 in je einem Byte.

Auf der AEG 80–20/4 stehen zwei Programmiersysteme zur Verfügung:

- Das Programmiersystem PRODOS läuft in Peripheriespeichersystemen unter MARTOS–K mit niederer Priorität im Hintergrundbetrieb.
- Das autonome Programmiersystem PROGA existiert in einer Hauptspeicher- und einer Peripheriespeichervariante. Es läuft ohne MARTOS–K und dient vor allem zum Systemaufbau.

Beide Programmiersysteme sind miteinander kompatibel, z.B. in der Assemblersprache und in der Struktur der lade- und bindefähigen Darstellung der übersetzten Programme auf Datenträgern.

6.1 Programmierersystem PRODOS

Das Programmiersystem PRODOS läuft im Hintergrund des prozeßgekoppelten Betriebs unter MARTOS–K. PRODOS enthält einige Funktionen, die auch im integrierten Dienstsistem PROGI enthalten sind. Soweit PRODOS im Peripheriespeicher verfügbar ist, werden die betreffenden PROGI-Funktionen weggelassen.

PRODOS besteht aus

- PRODOS-Katalog (ersetzt PROGI-Katalog):
Mit dieser Funktion werden Anwenderprogramme bei der Programmverwaltung von MARTOS–K an- und abgemeldet. Für angemeldete Programme wird ein Speicherbereich reserviert, um sie zu laden. Ist ein Programm abgemeldet worden, so wird der von ihm belegte Speicherbereich freigegeben und kann dann für andere Programme reserviert werden.
- PRODOS-Lader (ersetzt PROGI-Lader):
Der PRODOS-Lader dient zum
 - Laden
 - Binden und
 - Vergleichenvon Programmen in der vom PRODOS- oder vom PROGA-Assembler gelieferten Form. Im Normalmodus werden alle auf einem Datenträger vorhandenen Moduln ohne Eingriffsmöglichkeit hintereinander geladen; im Selektivmodus wird nach dem Laden jedes Moduls angehalten, um durch Operateur-Eingriff darüber entscheiden zu lassen, ob der nächste Modul geladen oder übersprungen wird.
Die zum Binden geladener Moduln erforderlichen Angaben werden in einer Symboltabelle eingetragen. Diese Eintragungen können durch Operateur-Eingriff ergänzt oder geändert werden.
Mit dem PRODOS-Lader läßt sich auch ein Programm auf einem Datenträger mit einem geladenen Programm vergleichen, um z.B. die Fehlerfreiheit beim Ausstanzen oder Einlesen zu prüfen.

6 Programmiersystem

- **PRODOS-Prozeß-FORTRAN-Compiler:**
Dieser Compiler übersetzt Programme, die in FORTRAN IV einschließlich der Prozeß-FORTRAN-Sprachelemente geschrieben sind. Er liefert das Übersetzungsergebnis in der Assemblersprache und startet vor seiner Beendigung den PRODOS-Assembler zur weiteren Übersetzung in die lade- und bindefähige Binärform.
- **PRODOS-Assembler:**
Der PRODOS-Assembler läuft in zwei Gängen ab und bietet verschiedene Nebenleistungen (z.B. unterschiedliche Protokollierung und variable Anzahl der erzeugten Datenträger mit dem übersetzten Programm) je nach den eingegebenen Parametern.
Die Quellmoduln in der AEG 80-20-Assemblersprache können von Lochkarten oder Lochstreifen eingelesen werden. Der Assembler kann auch Quellmoduln verarbeiten, die der Prozeß-FORTRAN-Compiler auf einem peripheren Speicher erzeugt hat.
- **PRODOS-Testfunktion:**
Die PRODOS-Testfunktion schützt den prozeßgekoppelten Betrieb umfassend gegen eventuelle Fehler bei Speicher-, Sprung- und EA-Operationen. Während des Testlaufs kann der Operateur mit der Testfunktion korrespondieren. Stopstellen im Testling können eingebaut, dokumentiert und wieder gelöscht werden. An vorgebbaren Stellen des Testlings können Speicher- und Registerinhalte ausgedruckt werden. Beim Anhalten des Testlaufs können Speicher- und Registerinhalte auch geändert werden; außerdem läßt sich der Testlauf an beliebiger Stelle im Testling starten und fortsetzen.
- **PRODOS-Editor:**
Mit dem PRODOS-Editor können beliebige Texte, vorzugsweise Programm-Moduln in Prozeß-FORTRAN oder in der AEG 80-20-Assemblersprache erzeugt, geändert oder ergänzt werden. Er wird im Dialog mit einem Dialog-Gerät bedient. Bei der Editierung wird der Quelltext zeilenweise in einem Pufferbereich im Hauptspeicher gelesen und als Zieltext wieder ausgegeben. Dabei lassen sich Zeilen löschen, einfügen und überschreiben.
- **PRODOS-Bindemodulverwaltung**
Die Bindemodulverwaltung nimmt die vom Assembler übersetzten binde- und ladbaren Moduln auf und stellt sie dem Lader auf Anforderung zur Verfügung.

6.2 Programmiersystem PROGA

Das autonome Programmiersystem PROGA läuft prozeßentkoppelt ohne MARTOS—K. Es besteht aus:

— PROGA-Editor:

Der Editor wird mit Hilfe des PROGA-Laders geladen. Er dient zur Korrektur, Änderung oder Ergänzung von Texten beliebiger Syntax, vor allem von Programmen in Prozeß-FORTRAN oder in der Assemblersprache. Das zu bearbeitende Quellprogramm wird von Lochstreifen oder Lochkarten satzweise in einen Textpuffer im Hauptspeicher eingelesen. Über ein Dialog-Gerät erhält der Editor Anweisungen zum Ändern, Löschen, Einfügen, Protokollieren bzw. Ausgeben von Text. Das bearbeitete Programm wird auf Lochstreifen ausgegeben und kann dann übersetzt werden.

Speziell wird der Editor zur Anpassung des Systemdefinitor-Moduls benutzt. Der Systemdefinitor dient zum Erzeugen eines lauffähigen Betriebssystems MARTOS—K, das durch die Auswahl der Programm-Moduln und die Festlegung zahlreicher Systemparameter auf die Konfiguration des Systems und auf die Anforderungen an das System abgestimmt ist.

— PROGA-Assembler:

Der Assembler wird mit Hilfe des PROGA-Laders geladen. Er übersetzt Programm-Moduln aus der AEG 80—20—Assemblersprache in binde- und ladbare Moduln.

Die Quellmoduln werden von Lochstreifen oder Lochkarten eingelesen. Normale Ausgabe der übersetzten Moduln auf Lochstreifen, der Meldungen auf einem Bedienungs-Dialog-Gerät und der Übersetzungsprotokolle auf einem Drucker.

Der Assembler läuft wahlweise in ein bis drei Gängen mit unterschiedlichen Nebenleistungen ab und bietet Variationsmöglichkeiten bezüglich der Ein- und Ausgabegeräte.

— PROGA-Lader:

Der Lader ist Grundlage des Programmiersystems PROGA, da er zum Laden der übrigen PROGA-Funktionen dient. Eine weitere Schlüsselfunktion des PROGA-Laders ist das Laden und Binden des Betriebssystems MARTOS—K aus den zuvor assemblierten Programm-Moduln einschließlich des Systemdefinitors. Ferner dient der Lader zum Laden und Binden von Anwenderprogrammen und zum Laden von Daten.

Der PROGA-Lader existiert in einer Hauptspeichervariante und einer Peripheriespeichervariante.

Die Peripheriespeichervariante läßt sich in verschiedenen Betriebsmodi benutzen:

- Im kontinuierlichen Peripheriespeicher-Modus wird ein Hauptspeicher-Abbild ohne Berücksichtigung von Sektorgrenzen auf den Peripheriespeicher geladen. Beim Generieren des Betriebssystems MARTOS—K wird so der permanente Hauptspeicherinhalt auf dem Peripheriespeicher erzeugt.
- Im sektorweisen Peripheriespeicher-Modus werden Programme und Daten jeweils bei den Sektorgrenzen beginnend auf den Peripheriespeicher geladen. Programme sind dann auf den ihnen zugeordneten Laufbereichen im Hauptspeicher lauffähig.
- Im Hauptspeicher-Modus werden Programme und Daten unmittelbar in den Hauptspeicher geladen.

— PROGA-Testfunktion:

Die Testfunktion und der Testling werden mit Hilfe des PROGA-Laders geladen. Auch die Testfunktion existiert als Haupt- und Peripherenspeicher-variante.

Der Benutzer korrespondiert mit der Testfunktion über ein Bediengerät zur Eingabe von Anweisungen und zur Ausgabe von Testergebnissen.

Durch die Vorgabe einer Hauptspeicher-Bezugsadresse sind die relativen Adressen aus dem Übersetzungsprotokoll des Testlings unmittelbar in der Korrespondenz mit der Testfunktion zu verwenden.

Register-, Hauptspeicher- oder Peripherenspeicherinhalte werden nach Anweisung angezeigt und geändert. In den Testling lassen sich Stopstellen einbauen und ausbauen; zwischen den Stopstellen läuft der Testling nach Anweisung frei ab.

Die Testfunktion umfaßt als weitere Hilfsfunktionen:

- Absuchen eines Haupt- oder Peripherenspeicherbereichs mit Suchwert und Maske,
- Ausdrucken solcher Speicherbereiche,
- Ausstanzen solcher Speicherbereiche,
- Rückassemblieren von Programmen.

Zu den Dienstprogrammen gehören Funktionen zur Generierung eines betriebsfähigen Betriebssystems und das integrierte Dienstsysteem PROG1.

7.1 Systemgenerierung

Das Betriebssystem MARTOS—K besteht aus zahlreichen Moduln, die zunächst in der Assemblersprache vorliegen. Soweit diese Moduln z.B. für einen Hauptspeicher- und ein Peripheralspeichersystem in verschiedenen Varianten benötigt werden, können sie mit Hilfe des Editors aus einer Grundvariante erzeugt werden.

Der Assembler erzeugt aus den Quellmoduln invariante Lademoduln, in denen alle modulinternen Adreßbeziehungen und sonstigen Parameter bereits festliegen. Globale Größen, z.B. Adreßbeziehungen zwischen verschiedenen Moduln werden dagegen erst beim Laden und Binden festgelegt.

Zum Generieren des lauffähigen Betriebssystems werden die dafür benötigten Moduln ausgewählt. Außerdem werden Angaben über die Konfiguration (Art und Anschluß von Geräten) und Systemparameter (z.B. maximale Zahl gleichzeitig zu ladender Programme) in den Systemdefinitor eingetragen.

Der Systemdefinitor ist ein spezieller Modul, der in Tabellenform zentral alle systemspezifischen Parameter als globale Größen enthält. Nach der Eintragung dieser Parameter mit Hilfe des PROGA-Editors wird er assembliert und ist dann ebenfalls ein invarianter Lademodul. Der PROGA-Lader lädt und bindet den Systemdefinitor zusammen mit den übrigen ausgewählten Moduln und generiert so ein auf der betreffenden Konfiguration lauffähiges Betriebssystem mit den gewünschten Systemeigenschaften.

Dieses Verfahren zur Generierung von modular aufgebauten und in verschiedenen Varianten zu betreibenden Programmsystemen wird auch bei der Standard-Anwendungssoftware benutzt. Es kann auch ohne Mehraufwand zur Generierung benutzereigener Software dieser Art eingesetzt werden.

7 Dienstprogramme

7.2

Integriertes Dienstsysteem PROGI

PROGI ist Bestandteil von MARTOS—K. Es existiert ebenfalls in einer Hauptspeicher- und einer Peripheriespeichervariante. Das Bedienpersonal steuert die Rechenanlage mit PROGI über ein Dialog-Gerät. Einige PROGI-Funktionen sind auch Bestandteil des Programmiersystems PRODOS. Sie werden weggelassen, wenn PRODOS geladen ist.

Nach Betätigen der Wagenrücklauftaste meldet sich das zentrale PROGI-Programm SPNB zum Aktivieren und Passivieren von Programmen.

Mit SPNB werden Anwenderprogramme und die anderen PROGI-Programme gestartet. SPNB dient auch zur Übermittlung der Nachricht an Anwenderprogramme, sich zu beenden. SPNB beendet sich selbst jeweils nach Erledigung eines Auftrags.

Werden mit SPNB andere PROGI-Programme gestartet, so fragen diese über das Dialog-Gerät, was zu tun ist. Nach der meist aus mehreren Parametern bestehenden Eingabe wird die angeforderte Operation ausgeführt. Die Folge aus Anfrage, Eingabe von Parametern und Ausführung der dadurch spezifizierten Operation wiederholt sich meist zyklisch, bis das betreffende PROGI-Programm durch Eingabe eines Endezeichens beendet wird.

Weitere PROGI-Funktionen sind:

- **PROGI-Katalog:**
Damit werden Anwenderprogramme beim Betriebssystem MARTOS—K an- und abgemeldet. Die Anmeldung ist Voraussetzung für das Laden des Programms; sie wird nur angenommen, wenn der benötigte Speicherbereich verfügbar ist.
Ist ein Programm abgemeldet, so kann der von ihm bisher belegte Speicherbereich neu belegt werden; das Programm kann nicht mehr gestartet werden.
- **PROGI-Lader:**
Er ladet Programm-Moduln nach vorangehender Anmeldung mit dem PROGI-Katalog von Lochstreifen oder Lochkarten in den Hauptspeicher bzw. Peripheriespeicher. Absolute Moduln werden in bzw. für einen festen Hauptspeicherbereich geladen; relative Moduln erlauben noch die Vorgabe der Hauptspeicheradresse, bei der der Laufbereich des Moduln beginnt; sie wird beim Laden zu allen relativen Adressen addiert.
Binden, d.h. Absättigen globaler Größen zwischen mehreren Moduln, ist nicht möglich, sondern ist Aufgabe des PRODOS- bzw. PROGA-Laders. In den Systembereich wird nicht geladen. Ein auf Lochstreifen bzw. Lochkarten vorliegender Modul kann mit einem geladenen Modul verglichen werden.

Die übrigen PROGI-Programme ermöglichen

- Anzeige und Änderung von Halbwörtern im Haupt- bzw. Peripheriespeicher
- Ausdrucken von Speicherbereichen
- Ausstanzen von Speicherbereichen
- Bedienung der Protokollausgabe mit Pufferung
- Bedienung von Umlaufspeichern
- Eingabe von Datum und Uhrzeit
- Bibliotheksaufbau
- Dokumentation aller Systemgrößen

Beim Generieren des Betriebssystems wird ausgewählt, welche von diesen Funktionen zu laden sind. Nicht benötigte Funktionen können weggelassen werden, um den dadurch eingesparten Speicherbereich für andere Zwecke zu verwenden.

Die Überwachung und Führung eines technischen Prozesses mit einem Prozeßrechner erfordert

- Einlesen analoger oder digitaler Meßwerte entweder vom Programm aus oder durch spontane Eingabe.
- Rechnerische Verarbeitung der eingelesenen Werte durch logische oder arithmetische Operationen, Funktionsberechnungen oder Vergleichsoperationen entsprechend dem logischen oder mathematischen Steuerungs- bzw. Regelungsmodell des technischen Prozesses.
- Ausgabe analoger oder digitaler Daten, die den technischen Prozeß zweckentsprechend beeinflussen.
- Protokollierung des Ablaufs des technischen Prozesses.
- Alarmgebung bei Prozeßabläufen, die Eingriffe des Bedienpersonals erfordern.
- Dateneingabe durch das Bedienpersonal.

Diese Vorgänge werden gestartet

- zu bestimmten Zeitpunkten, oder
- auf Grund externer Ereignisse, oder
- auf Grund der Ergebnisse der rechnerischen Verarbeitung.

Trotz der Vielfalt technischer Prozesse sind diese Aufgaben auf eine überschaubare Menge vorprogrammierter Grundfunktionen zurückführbar. Mit den beiden Anwendungs-Softwarepaketen ARS180 und AS80 werden bestimmte Gruppen von Prozeßsteuerungsaufgaben durch Zusammenstellung von Programm-Moduln gelöst. Dazu sind nach den spezifischen Anforderungen des zu steuernden technischen Prozesses die Programm-Moduln auszuwählen und durch Parameterwerte (z.B. Anzahl von Geräten, Sollwerte und Wertbereiche von Prozeßvariablen oder Zeitintervalle) zu vervollständigen.

Dann wird das Programmsystem zur Prozeßsteuerung nach demselben Verfahren generiert wie das Betriebssystem MARTOS—K.

Die Auswahl der Programm-Moduln und die Festlegung der Parameter für den zu steuernden technischen Prozeß ist bei ARS180 und AS80 jeweils in einer speziellen Eingabesprache systematisiert. Mit dieser Sprache können Prozeßsteuerungs-Fachkräfte ohne spezielle Programmierungkenntnisse das Programmsystem für die Prozeßsteuerung weitgehend selbständig generieren, korrigieren und ändern.

ARS180 und AS80 unterscheiden sich im Typus der Prozeßvariablen und deren Verarbeitung.

8 Standard-Anwendungssoftware

8.1 ARSI80

Das Programmpaket ARSI80 ist für Prozeßsteuerungs- und -regelungsaufgaben mit stetig veränderlichen Prozeßgrößen bestimmt, z.B. Durchflußmengen, Drücke oder Temperaturen. Daher werden die Prozeßvariablen zum großen Teil als Analogwerte ein- bzw. ausgegeben.

Das Prozeßsteuerungsprogramm verbessert die eingegebenen Analogwerte durch selbsttätige Nullpunktkorrektur und optimale Einstellung des Verstärkungsfaktors. Die Zeitpunkte für Eingabe, Verarbeitung und Ausgabe werden weitgehend durch die im Prozeßsteuerungsprogramm festgelegten Zeitzyklen bestimmt.

Die rechnerische Verarbeitung der Eingabewerte nach regelungstechnischen Prozeßmodellen besteht vorwiegend aus arithmetischen Operationen, Funktionsberechnungen, Summen- und Mittelwertbildungen. Daneben sind auch logische Operationen möglich.

8.2 AS80

Das Programmpaket AS80 ist für Prozeßsteuerungsaufgaben mit diskreten Variablen, z.B. Kontaktstellung "ein" oder "aus", bestimmt. Daher werden bei AS80 Digitalwerte ein- und ausgegeben.

Die Zeitpunkte für Eingabe, Verarbeitung und Ausgabe werden vorwiegend durch Änderungen von Prozeßvariablen bestimmt.

Die rechnerische Verarbeitung der diskreten Eingabewerte nach logischen Prozeßmodellen besteht aus logischen Operationen (Konjunktion, Disjunktion, Negation usw.). Die diskreten Ergebnisse bestimmen jeweils den weiteren Ablauf des Prozeßsteuerungsprogramms.

9.1

Charakteristische Daten

Verkehrsstruktur:

Verarbeitungsbreite
Register

16 Bits
16 Arbeitsregister je 16 Bits
1 Programmstatusregister 16 Bits
1 Befehlszähler 16 Bits
1 Befehlsregister 16 Bits
1 look-ahead-Befehlsregister 16 Bits

Befehle:

Befehlslänge
Anzahl der Befehle

16, 32 und 48 Bits
114
57 Befehle für Transport, Festpunkt-arithmetik, logische und Vergleichsoperationen
3 Bitfeldbefehle
4 Listenbefehle
12 Sprungbefehle
12 Shiftbefehle
4 Systembefehle
9 Ein- und Ausgabebefehle
13 Gleitpunktbefehle

Adressierung:

Umfang
Modifikationen allgem.
Modifikationen beim Sprung

64 K Bytes
Indizierung (15 Arbeitsregister)
Relativierung, Indizierung

Speicher:

Mikroprogrammspeicher
Hauptspeicher
Ausbau des Hauptspeichers

Festwertspeicher
Kernspeicher
in Stufen von 16 oder 32 K Bytes
max. 128 K Bytes

Zykluszeit Lesen/Schreiben:

ca. 0,70 μ s

Zeiten:

Addition Register/Register
Addition mit Hauptspeicher

ca. 0,7 μ s
ca. 2,7 μ s

Datenkanäle:

Selektorkanalwerk: max. 700 kbyte/s
Autom. Ein- und Ausgabe
Verkehr: ca. 35 kDatenzyklen/s

Schnittstellen:

paralleler EA-Bus
KE-Anschluß (optional)
über Selektorkanalwerk

9 Anhang

Transportbefehle

Operation	Mnemo-Code	Format
Lade Byte	LB	32M
Lade Byte aus Register	LBR	16R
Lade mit Bytevertauschung aus Register	EBR	16R
Lade Halbwort	LH	32M
Lade aus Register	LHR	16R
Lade Halbwortkonstante	LIH	32I
Lade Kurzkonstante	LIS	16I
Lade Kurzkonstante negativ	LNIS	16I
Lade mehrfach	LM	32M
Speichere Byte	STB	32M
Speichere Byte in Register	STBR	16R
Speichere Halbwort	STH	32M
Speichere mehrfach	STM	32M

Bitfeldbefehle (optional)

Operation	Mnemo-Code	Format
Lade Teilwort	LP	48MBF
Lade Teilwort arithmetisch	LAP	48MBF
Speichere Teilwort	STP	48MBF

Listenbefehle

Operation	Mnemo-Code	Format
Bringe Halbwort an Listenanfang	ATLH	32M
Bringe Halbwort an Listeneende	ABLH	32M
Entferne Halbwort von Listenanfang	RTLH	32M
Entferne Halbwort von Listeneende	RBLH	32M

Festpunktbefehle

Operation	Mnemo-Code	Format
Addiere Halbwort	AH	32M
Addiere Register	AHR	16R
Addiere Halbwortkonstante	AIH	32I
Addiere Kurzkonstante	AIS	16I
Addiere und Speichere Halbwort	ASTH	32M
Addiere Halbwort mit Übertrag	AYH	32M
Addiere Register mit Übertrag	AYHR	16R
Addiere Halbwortkonstante mit Übertrag	AYIH	32I
Subtrahiere Halbwort	SH	32M
Subtrahiere Register	SHR	16R
Subtrahiere Halbwortkonstante	SIH	32I
Subtrahiere Kurzkonstante	SIS	16I
Subtrahiere Halbwort mit Übertrag	SYH	32M
Subtrahiere Register mit Übertrag	SYHR	16R
Subtrahiere Halbwortkonstante mit Übertrag	SYIH	32I
Multipliziere Halbwort	MH	32M
Multipliziere Register	MHR	16R
Multipliziere Halbwortkonstante	MIH	32I
Multipliziere Halbwort vorzeichenlos	MUH	32M
Multipliziere Register vorzeichenlos	MUHR	16R
Multipliziere Halbwortkonstante vorzeichenlos	MUIH	32I
Dividiere Halbwort	DH	32M
Dividiere Register	DHR	16R
Dividiere Halbwortkonstante	DIH	32I

Gleitpunktbefehle (optional)

Operation	Mnemo-Code	Format
Lade Gleitpunkt-Wort	LE	32M
Lade Gleitpunkt-Register	LER	16R
Speichere Gleitpunkt-Wort	STE	32M
Addiere Gleitpunkt-Wort	AE	32M
Addiere Gleitpunkt-Register	AER	16R
Subtrahiere Gleitpunkt-Wort	SE	32M
Subtrahiere Gleitpunkt-Register	SER	16R
Multipliziere Gleitpunkt-Wort	ME	32M
Multipliziere Gleitpunkt-Register	MER	16R
Dividiere Gleitpunkt-Wort	DE	32M
Dividiere Gleitpunkt-Register	DER	16R
Vergleiche Gleitpunkt-Wort	CE	32M
Vergleiche Gleitpunkt-Register	CER	16R

Logische Befehle

Operation	Mnemo-Code	Format
UND Halbwort	NH	32M
UND Register	NHR	16R
UND Halbwortkonstante	NIH	32I
ODER Halbwort	OH	32M
ODER Register	OHR	16R
ODER Halbwortkonstante	OIH	32I
Exklusiv-ODER Halbwort	XH	32M
Exklusiv-ODER Register	XHR	16R
Exklusiv-ODER Halbwortkonstante	XIH	32I

Shiftbefehle

Operation	Mnemo-Code	Format
Schiebe links logisch Registerpaar	HL	32I
Schiebe links logisch Register (indizierbar)	HLH	32I
Schiebe links logisch Register	HLHS	16I
Schiebe rechts logisch Registerpaar	HR	32I
Schiebe rechts logisch Register (indizierbar)	HRH	32I
Schiebe rechts logisch Register	HRHS	16I
Schiebe links zyklisch Registerpaar	HLC	32I
Schiebe rechts zyklisch Registerpaar	HRC	32I
Schiebe links arithmetisch Registerpaar	HLA	32I
Schiebe links arithmetisch Register	HLAH	32I
Schiebe rechts arithmetisch Registerpaar	HRA	32I
Schiebe rechts arithmetisch Register	HRAH	32I

Vergleichs-, Test- und Zählbefehle

Operation	Mnemo-Code	Format
Vergleiche Halbwort	CH	32M
Vergleiche Register	CHR	16R
Vergleiche Halbwortkonstante	CIH	32I
Vergleiche Byte logisch	CB	32M
Vergleiche Halbwort logisch	CLH	32M
Vergleiche Register logisch	CLHR	16R
Vergleiche Halbwortkonstante logisch	CLIH	32I
Teste Halbwort	TH	32M
Teste Register	THR	16R
Teste Halbwortkonstante	TIH	32I
Zähle Nullen im Register von links(optional)	CTLZ	16R

Sprung-Befehle

Operation	Mnemo-Code	Format
Springe, wenn Maskenbedingung Null	BZ	32M
Springe unbedingt	B*	
Springe, wenn kein Übertrag	BNY*	
Springe, wenn kein Überlauf	BNO*	
Springe, wenn gleich	BE*	
Springe, wenn größer gleich	BGE*	
Springe, wenn kleiner gleich	BLE*	
Springe, wenn Maskenbedingung Null nach Registerinhalt	BZR	16R
Springe unbedingt, nach Registerinhalt	BR*	
Springe, wenn kein Übertrag, nach Registerinh.	BNYR*	
Springe, wenn kein Überlauf, nach Registerinh.	BNOR*	
Springe, wenn gleich, nach Registerinhalt	BER*	
Springe, wenn größer gleich, nach Registerinh.	BGER*	
Springe, wenn kleiner gleich, nach Registerinh.	BLER*	
Springe, wenn Maskenbedingung Null relativ vorwärts	JZF	16I
Springe, wenn Maskenbedingung Null, relativ rückwärts	JZB	16I
Springe unbedingt relativ	J*	
Springe, wenn kein Übertrag, relativ	JNY*	
Springe, wenn keine Überlauf, relativ	JNO*	
Springe, wenn gleich, relativ	JE*	
Springe, wenn größer gleich, relativ	JGE*	
Springe, wenn kleiner gleich, relativ	JLE*	
Springe, wenn Maskenbedingung nicht Null	BNZ	32M
Springe, wenn Übertrag	BY*	
Springe, wenn Überlauf	BO*	
Springe, wenn größer	BG*	
Springe, wenn kleiner	BL*	
Springe, wenn nicht gleich	BNE*	
Führe keine Operation aus	NOP*	
Springe, wenn Maskenbedingung nicht Null, nach Registerinhalt	BNZR	16R
Springe, wenn kein Übertrag, nach Registerinh.	BYR*	
Springe, wenn kein Überlauf, nach Registerinh.	BOR*	
Springe, wenn größer, nach Registerinhalt	BGR*	
Springe, wenn kleiner, nach Registerinhalt	BLR*	
Springe, wenn nicht gleich, nach Registerinh.	BNER*	
Führe keine Operation aus	NOPR*	
Springe, wenn Maskenbedingung nicht Null, relativ vorwärts	JNZF	16I
Springe, wenn Maskenbedingung nicht Null, relativ rückwärts	JNZB	16I
Springe, wenn Übertrag, relativ	JY*	
Springe, wenn Überlauf, relativ	JO*	
Springe, wenn größer, relativ	JG*	
Springe, wenn kleiner, relativ	JL*	
Springe, wenn nicht gleich, relativ	JNE*	
Springe, solange Wert größer als Endwert	BXH	32M
Springe, solange Wert kleiner oder gleich Endwert	BXLE	32M
Springe in Unterprogramm	BU	32M
Springe in Unterprogramm nach Registerinhalt	BUR	16R

* Diese Befehle sind durch Vorgabe einer festen Maske aus den vorstehenden Befehlen abgeleitet.

System-Befehle

Operation	Mnemo-Code	Format
Vom Normal-Modus in den System-Modus führend:		
Beauftrage Betriebssystem	SVC	32M
Nur im System-Modus ausführbar:		
Lade Programmstatuswort	LPSW	32M
Vertausche Programmstatuswort	EPSR	16R
Führe Mikroprogramm aus	YEMP	32M

Ein-Ausgabe-Befehle

Nur im System-Modus ausführbar

Operation	Mnemo-Code	Format
Hole Status	YSS	32M
Hole Status in Register	YSSR	16R
Sende primäres Steuerzeichen	YOCR	16R
Gib in Halbwort ein	YRH	32M
Gib in Register ein	YRHR	16R
Gib aus Halbwort aus	YWH	32M
Gib aus Register aus	YWHR	16R
Normiere EA-Bus	YNP	16R
Simuliere Unterbrechung (optional)	YII	32I

AEG-TELEFUNKEN

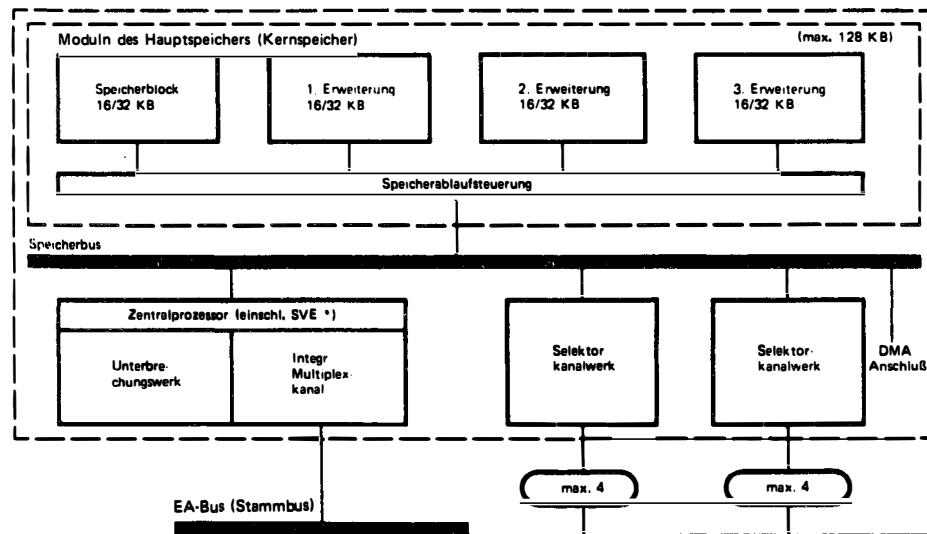
Prozeß-Datenverarbeitung

Prozeßrechensystem AEG 80-20

Moduln des Gerätesystems

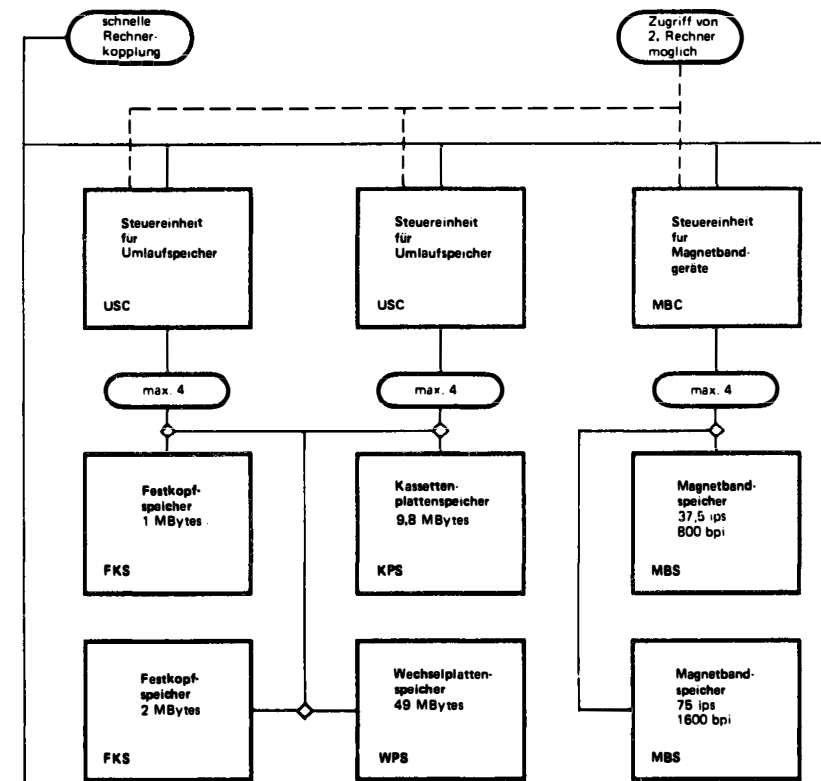
Systemübersicht – Blatt 1 –

Zentraleinheit AEG 80-20/4



* Speicherverwaltungseinheit SVE, erforderlich für mehr als 64 KB

Periphere Speicher am Selektorkanal



AEG-TELEFUNKEN

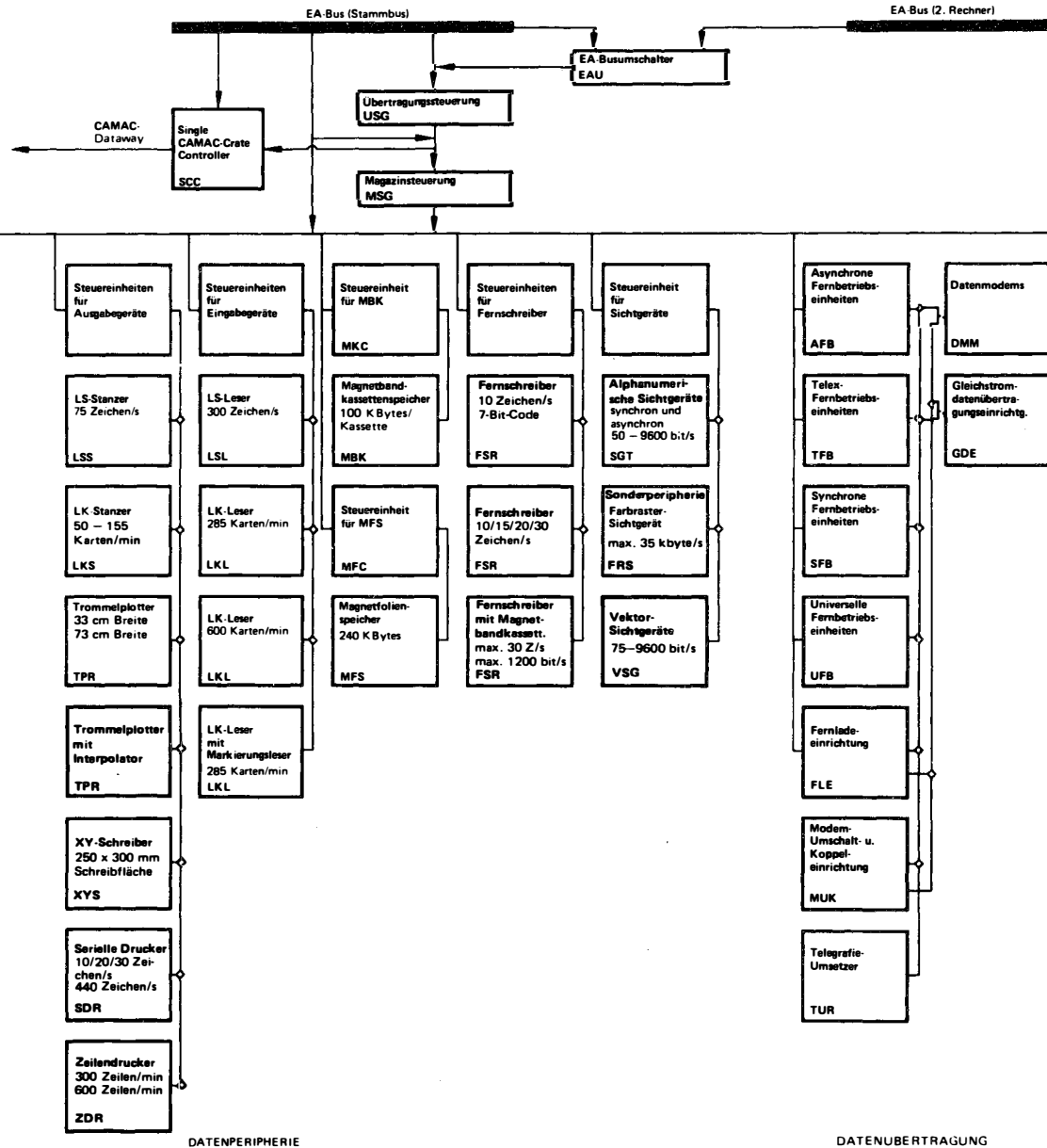
Prozeß-Datenverarbeitung

Prozeßrechensystem AEG 80

Moduln des Gerätesystems

Systemübersicht – Blatt 2 –

Peripheriegeräte und Baugruppen am EA-Bus

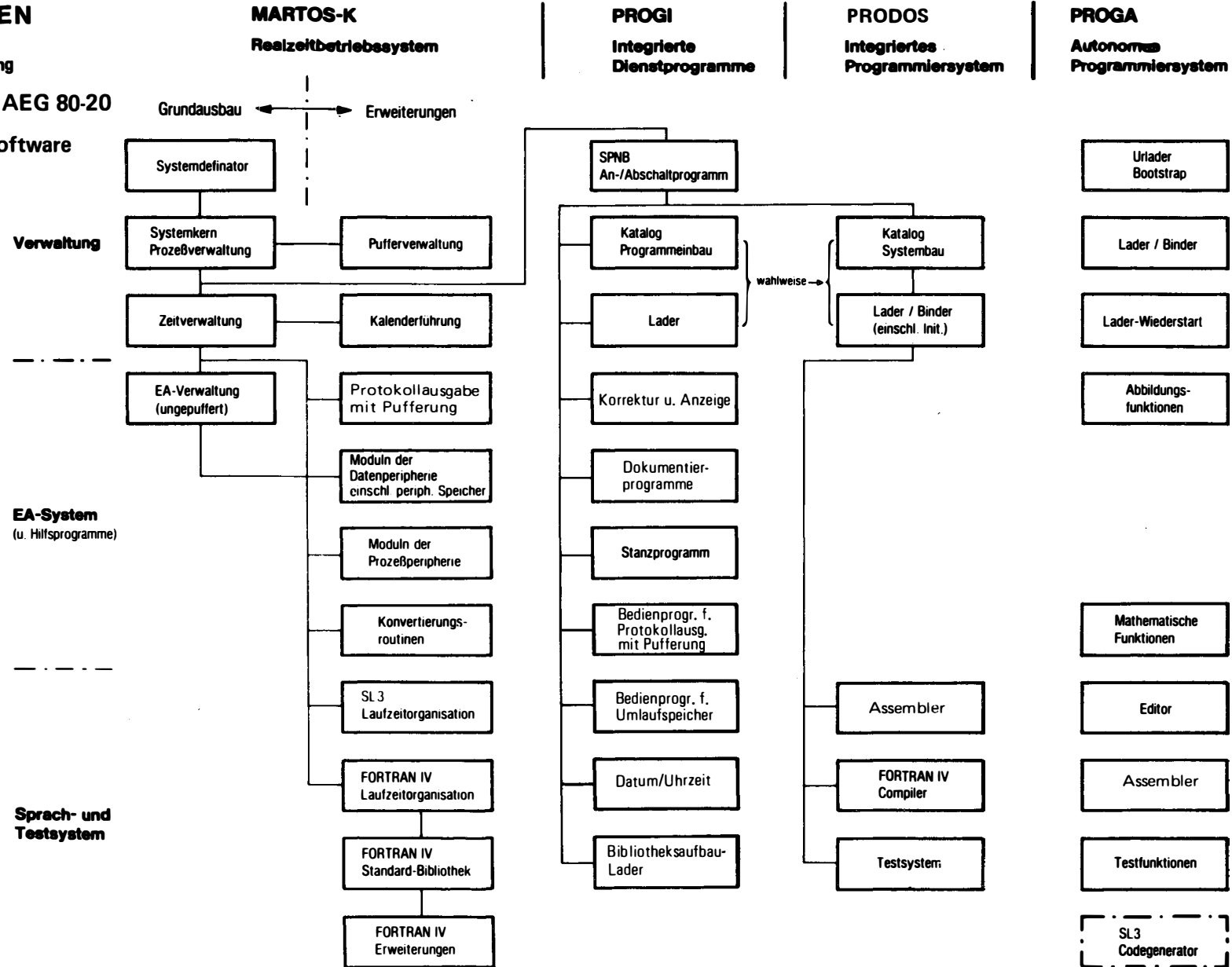


AEG-TELEFUNKEN

Prozeß-Datenverarbeitung

Prozeßrechensystem AEG 80-20

Moduln der Systemsoftware
Systemübersicht



Absender:

in Firma:

Straße:

Ort:

Wir bemühen uns zwar, finden aber
auch nicht jeden Fehler. Daher bitten
wir Sie um

**Verbesserungs-Vorschläge
Sachliche Berichtigungen
Druckfehler-Korrekturen**

Firma

AEG-TELEFUNKEN

Abt. E511V7

Bücklestraße 1 - 5

7750 Konstanz

Western Germany

zur vorliegenden Schrift:

AEG 80-20

mit Zentraleinheit AEG 80-20/4

Kurzbeschreibung

E511V.6.230/1078

bzw. zur Schrift:

.

.

.

.

vorbereitet für Fenster-Umschlag

AEG-TELEFUNKEN

GESCHÄFTSBEREICH PROZESSTECHNIK

D-6000 Frankfurt 71, Lyoner Straße 19, Tel. (06 11) 66 98-1
Prozeß-Datenverarbeitung: Telefon (06 11) 66 98-314

D-7750 Konstanz, Bücklestraße 1-5, Telefon (0 75 31) 86-1
Prozeß-Datenverarbeitung: Telefon (0 75 31) 86-2427

Printed in Western Germany