Mikrocomputer-Gerätegeneration

M C 80.3x

-B E T R I E B S S Y S T E M-

(EGOS 30.1 - Teil 1)

Systembeschreibung

Z.-Nr. 50300- 4012.00 BA

VEB Elektronik Gera

Ausgabe 09/86

Inhaltsverzeichnis

1. Speicherplatzverteilung aus Anwendersicht

0000-0FFF	Betriebssystemkern und VIS-Treiber (Tastatur-routinen, Sprungverteiler, Anwenderschnitt-stellen, Zeichengenerator, BSYS-Initialisie-rungsprogramm)
1000-1FFF	Magnetbandsoftware (Treiber, Handler)
2000-23FF	Sprungprogramme und Ladeprogramme zur Segment-umschaltung, Relativlader Verschiebeadresstabellenerstellprogramm
2400-27FF	Geräteverwaltungskommandos und -Programme, Ausschrift Initialisierungtext auf Display
4000-BFFF	RAM zur freien Verfügbarkeit; segmentierbar in bis zu 8 32k-Segmenten, wobei immer nur 1 Segment eingeschaltet ist Bei Verwendung von weiteren Leiterplatten SPE sind ab Segment 4 (Nr. 3) auch ROM-Segmente zu wahlweise 16k-Byte (2716) oder 8k-Byte (2708) möglich.
C000-F000	ZVE-RAM zur freien Verfügbarkeit; nicht seg-mentierbar (Vorzugsplatz für BASIC-Interpreter)
F001-FFFF	ZVE-RAM zur bedingten freien Verfügbarkeit; von Adr. F600H nach oben als lokaler Stack, gezeigert durch IY-Register, nutzbar, ab F800H CPU-Stack (bis F6FF) F701-FFFF Arbeitszellen BSYS, Megnetband FC00: 1k-Byte-Schreibschutz

2. Speicherplatzverwaltung ZVE-RAM ab F600

....-F600H	lokaler Stack, durch IY-Register gezeigert
F601-F800H	CPU-Stack
F880-F8DBH	BZPU-BASIC -Zwischencode -Zielpuffer
F8DC-F8FFH	Bereich zum Ausführen der Segmentumschaltung
F900-FB3FH	Arbeitszellen Magnetband
FB40-FB7FH	Arbeitszellen VIS und Tastatur
FB80-FBDFH	Eingabepuffer für Tastatur (Kommandoeingabe)

FBEØ-FBFAH Systemzellen für BASIC
FBFB-FBFFH Arbeitszellen für Segmentumschaltung
FCØØ-FC6FH schreibgeschützte Zellen für active device table
FC7Ø-FECFH schreibgeschützte Zellen für Anwender
FEDØ-FEE7H schreibgeschützte Aussprungstellen für Sondertasten
FEE8-FEFFH schreibgeschützte NMI- und RST-Aussprünge
FFØØ-FFAFH für Erweiterungen vorgesehen
FFBØ-FFFFH INT-Vektoren

3. INT-Vektoren und I-O-Adressen

FFØØ-FF7FH frei für Anwender INT-Vektoren für zusätzliche E/A-Platinen
FF8Ø-FFAFH frei für Systemerweiterungen
FFBØ-FFBBH AKB für 52ØØ (nur MC 8Ø.31)
FFBC-FFD7H reserviert für AKM (MC 8Ø.3Ø/31)
FFD8-FFDFH CTC ZVE
FFEØ-FFE3H PIO ZVE
FFE4-FFE7H PIO ASP
FFE8-FFEFH CTC ASP
FFFØ-FFFFH SIO ASP

Adr. ØØ-2FH frei für Anwender
     3Ø-3FH AKB, für Massenspeicher
     4Ø-7FH frei für Anwender
     8Ø-83H CTC auf ZVE (Kanal Ø-3 frei für Anwender)
     84-87H PIO ZVE: 84H Datenwort Kanal A
                      85H Datenwort Kanal B
                      86H Steuerwort Kanal A
                      87H Steuerwort Kanal B
     88-8FH intern verwendet
     9Ø-BFH reserviert für Weiterentwicklung
     CØ-CFH VIS-Adressen für Displaysteuerung
     DØ-D3H SIO ASP: DØH Datenwort Kanal A(Tastatur)
                      D1H Steuerwort Kanal A
                      D2H Datenwort Kanal B (IFSS)
                      D3H Steuerwort Kanal B

D4-D7H CTC ASP: D4H Kanal Ø, als Takt für SIO Kanal A
                 D5H Kanal 1, als Takt für SIO Kanal B, KMBG
                 D6H Kanal 2, für KMBG (nur bei MC 80.30)
                 D7H Kanal 3, für Tastatur
D8-DBH PIO ASP: D8H Datenwort Kanal A (frei, auf Koppelbus)
                 D9H Datenwort Kanal B (IFSP)
                 DAH Steuerwort Kanal A
                 DBH Steuerwort Kanal B
DC-DFH reserviert für Weiterentwicklung
EØ-FFH EPR-Adressen
FØ-FFH reserviert für Weiterentwicklung

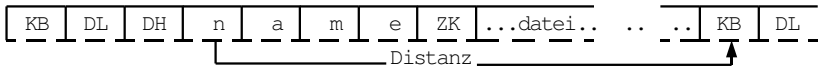
4. VIS-Arbeitszellen und Tastaturzellen

FB4ØH-41H	Zeichengenerator (bei RESET ØCA4H)
FB42H	internes Statusbyte (Bit 4 = Ø: einfache Zeichenhöhe. Bit 4 = 1: doppelte Zeichenhöhe)
FB43H	Zeilenhöhe (ØAH=1Ø PIXEL normal, 14H=2Ø PIXEL doppel)
FB44H	Adresse CTC Kanal 3 auf ASP
FB45H	letzter Tastencode (unausgewertet)
FB46H/47H	Bildfenster und Bildfensterhintergrund
FB48H/49H	z. Z. frei
FB4AH	Rollzähler
FB4BH	Bildhintergrund
FB4CH/4DH	Zeiger auf lokaler Stack, Grundzustand
FB4EH/4FH	IY-Adresse für RIO-Vektor, Grundzustand
FB5ØH/51H	Textpufferanfang
FB52H/53H	maximales Textpufferende (Folgebyte)
FB54H/55H	Adresse Textpuffer Kursor
FB56H/57H	aktuelles Textpufferende
FB58H-5BH	aktueller Kursor in Pixelkoordinaten
FB5CH/5DH	Bildfensteranfang/Bildfensterende
FB5EH/5FH	aktuelles Bildfensterende
FB6ØH	Statusbyte Tastatur
FB61H	z. Z. frei
FB62H	letztes Zeichen von der Tastatur
FB63H/64	aktueller Kursor (Spalte/Zeile)
FB65H/7BH	Tabulatorzellen in Pixelkoordinaten
FB7CH/7FH	Tastaturarbeitszellen

5. Dateiverwaltung durch RAM-Kettung

Das Betriebssystem EGOS 3Ø ermöglicht ein namenorientiertes Suchen und Aufrufen von unterschiedlichsten Dateien. Eine Datei ist durch ihren Dateityp (Kennbyte) und ihren Namen charakterisiert. Diese Information ist für jede Datei, die in die Dateiverwaltung einbezogen werden soll, in einem Dateikopf abzuspeichern. Außerdem steht in diesem Dateikopf in Form einer Distanz (a,b) oder einer Absolutadresse / Segmentnummer (c) eine Information, wo die nächste Datei steht. Diese Art des Zeigerns von einer Datei auf die nächste beinhaltet der Begriff RAM-Kettung. Diese beginnt fest auf der Adresse ØAFEH bei dem Kopf der Treiberroutine CON mit dem Kennbyte Ø2 und wird über alle Betriebssystemkommandos auf die Adresse 4ØØØH geführt. Für die Weiterführung der Kettung ist der Anwender verantwortlich, da jedes vom Magnetband eingelesene Programm die Kettung beeinflußt, denn der Programmkopf ist mit auf Magnetband aufgezeichnet. Gleichzeitig verhindert diese Dateiverwaltung aber auch, daß man versehentlich andere Dateien überschreiben kann. Sowohl das Vergrößern eines Arbeitsbereiches mit dem Objektcodeeditor oder BASIC-Editor, als auch das Einlesen von Kassette kann nur solange erfolgen, wie freier RAM mit dem Kennbyte ØØ in der Kettung steht. Mit Hilfe des Kommandos RKET können Dateien überkettet oder die RAM-Kettung korrigiert werden, wie es z. B. meist nach einem Einlesen einer Datei von Kassette notwendig ist, um die Directory-Datei wieder in die Kettung einzubeziehen. Eine überkettete Datei steht zwar physisch noch im Speicher, jedoch kann über die Speicherverwaltung des Betriebssystems nicht mehr auf diese zugegriffen werden. Der für die RAM-Kettung erforderliche Dateikopf hat folgenden Aufbau:

a) Datei mit Namen



b) Datei ohne Namen (z. B. RESET-Routinen)

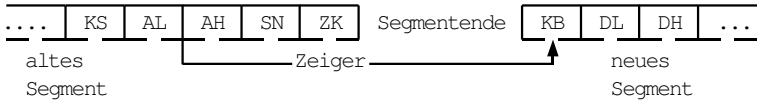
KB: Kennbyte (siehe Pkt. 6)
DL: Distanz Low-Teil
DH: Distanz High-Teil
ZK: Zusatzkennbyte (Bedingung:Bit 7 gesetzt)

Die Distanz wird ab dem ersten Byte des Namens bzw. bei Dateien ohne Namen ab dem Zusatzkennbyte gerechnet und zeigt auf das Kennbyte der folgenden Datei. Das heißt vom tatsächlichen Abstand zwischen 2 Dateien, gerechnet von Kennbyte zu Kennbyte, müssen zur Ermittlung der Distanz für die RAM-Kettung 3 Byte subtrahiert werden.

ACHTUNG:

Wenn die Summe aus Distanz und Adresse des ersten Namensbytes bzw. bei namenlosen Dateien der Adresse des Zusatzkennbytes eine Uberschreitung des 64k-Adreßraumes der CPU ergibt, erfolgt ein Zurückketten. Ein solches Zurückketten auf ein anderes schon in der RAM-Kettung stehendes Kennbyte ist unbedingt zu vermeiden, da sonst ein Abschalten des Rechners notwendig wird (Kreiskettung).

c) Dateikopf zur Segmentumschaltung



KS: Kennbyte zur Segmentumschaltung. ØFEH

AL: Adresse Lowteil, dort steht im neuen Segment nächstes KB

AH: Adresse Highteil

SN: Segmentnummer des neuen Segmentes

Die Softwareschnittstellen RSU,RSA,RSF und RLL tragen diesem Dateiverwaltungsprinzip Rechnung. Durch die RAM-Kettung ist besonders in der Assemblersprache, ein modulares Programmieren mit untereinander sehr variablen aber in sich völlig abgeschlossenen Teilmodulen möglich.

Beispiel: Ein Programm schreibt Bilder aus die als Binärdateien abgespeichert sind. Ihre Anzahl soll im Laufe der Zeit vergrößert werden, ohne das Programm das mit den Bildern arbeitet jedesmal mitzuändern. Deshalb sollen die Anfangsadressen der Bilder nicht fest sein. Die Ermittlung der Anfangsadressen der Bilder. die die Namen BILD1, BILD2 ... BILDn haben, kann nach folgendem Muster erfolgen:

INPUT: (BINR)= Nummer des gerade auszugebenen Bildes (hexa.dezimal) zwischen ØØ und ØFFH)

```
BIAN:  LD     HL,BNAS          ;Adresse für Bildnummer in ASCII
       LD     A,(BINR)         ;Bildnummer hexa
       CALL   HBS              ;Bildnummer in ASCII schreiben
       LD     DE,TEPU          ;Textpuffer für Namen der Datei
       LD     A,(SEGC)         ;aktuelle Segmentnummer
       LD     (SEGE),A         ;aktuelle Segmentnr., in der Name steht
       LD     A,6              ;Kennbyte
       CALL   RSU
       INC    HL               ;HL zeigt auf 1. Byte des Bildes
       .........
TEPU:  DM     'BILD'           ;Name
BNAS:  DM     '    '           ;Platz für Bildnummer in ASCII
       DB     ØFFH             ;Abschlußbyte Name
```

Auf die selbe Art und Weise können auch andere Programme (Kennbyte (Ø1) gesucht und mit JMP (HL) abgearbeitet werden.

6. Tabelle der möglichen Kennbytes in der RAM-Kettung

ØØ freier RAM
Ø1 Assemblerprogramm ohne Parameterliste, über Namen aus dem Kommandoeingabemodus des BSYS startbar
Ø2 Treiberroutine mit IY-Schnittstelle
Ø3 namenlose Routine, die bei jedem RESET abgearbeitet wird
Ø4 Directory
Ø5 reserviert für BSE
Ø6 Binärdatei
Ø7 ASCII-Datei (z.B. Quelltext)
Ø8 BASIC-Prozedur, Zwischencode-Parameterliste oder Assemblerprogramm mit Parameterliste
Ø9 wie Ø8 aber als INTEGER-Prozedur
ØAH)
ØBH)
ØCH) für Weiterentwicklung des BASIC
ØDH)
ØEH)
ØFH)
1ØH Verschiebeadresstabelle
11H RAM-Bereich für lokalen Stack
12H Markentabelle
13H Arbeitsbereich für Objektcodeeditor
14H numerisches Feld
15H boolean Feld
16H)
17H) für Weiterentwicklung des BASIC
18H-1AH zur Zeit frei
1BH gesperrter Bereich für feste Programme
1CH BASIC-Bereich
1DH-1FH zur Zeit frei
2ØH-ØFDH nicht erlaubt (RAM-Kettungsfehler)
ØFEH Segmentumschaltung
ØFFH Ende RAM-Bereich

7. Betriebssystemkommandos

Das Betriebssystem meldet sich im Kommandoeingabemodus mit

OK
>

in der letzten und vorletzten Displayzeile. Die Eingabe folgender Kommandos ist möglich:

INIT

Syntax: INIT ENTER / INIT "letztes zu init. Segment" ENTER

Aktion: löscht den gesamten RAM, schaltet alle Segmentmerkzellen auf Ø, kettet den RAM mit KBØ durch, wobei die Kettung vom letzten zu initialisierenden RAM-Segment auf Adresse CØØØH geführt wird; wenn nur 1 Segment vorhanden ist, dann sind keine Parameter erforderlich. Die Active Device Table wird angelegt. Nach INIT wird außerdem eine Initialisierung, wie nach RESET durchgeführt.

CAT

Syntax: "CAT" ENTER Anzeige aller Dateien mit Kennbyte Ø1 in der RAM-Kettung
"CATA" ENTER oder "CAT Ø" ENTER Anzeige aller Dateien in der RAM-Kettung
"CAT" "Kennbyte" ENTER Anzeige aller Dateien mit angegebenen Kennbyte, die im Speicher stehen

Aktion: Die CAT-Kommandos besitzen rein informativen Charakter. Es erfolgt eine tabellarische Auflistung aller Dateien in der Reihenfolge ihrer Kettung. (Dies muß nicht mit der adressmäßigen Reihenfolge übereinstimmen, da eine Distanz auch scheinbar zurückführen kann.)
Sind mehr als 23 Dateien mit entsprechenden KB im Speicher enthalten, muß ENTER betätigt werden, um die nächsten Dateien zur Anzeige zu bringen. Eine Tabellenzeile enthält folgende Daten:

"Name" ....% "Kennbyte" % "Segment" . "Adr. des Kennbytes"
Bei"CATA"und "CAT Ø" werden auch die Kennbytes ØØ und ØFFH mit zur Anzeige gebracht.

RELAD

Syntax: "RELAD" "Name" ENTER

Aktion: Nachdem ein auf Ø gebundenes Programm mit dazugehöriger Verschiebeadresstabelle auf die gewünschte Adresse geladen wurde, werden bei Aufruf dieses Kommandos alle nicht verschieblichen Adressen (Direktsprünge, Unterprogrammaufrufe) gemäß der Verschiebeadresstabelle auf der geladenen Adresse lauffähig gemacht.

LADT

Syntax: "LADT" ENTER

Aktion: Das Kommando gibt in tabellarischer Form Auskunft über die in der Active Device Table stehenden Treiberroutinen. Eine Zeile enthält dabei folgende Informationen:

"Name" ...%"Segment" %"Adresse""log. Nummer"

Die Informationen Segment, Adresse und Name sind identisch mit den in CAT verwendeten Begriffen. Die logische Gerätenummer kann durch DEFINE zugeordnet werden. Nach INIT stehen folgende Treiber in der ADT:

CON Konsolentreiber logische Nummern 1, 2, 3
KLOG Kassettentreiber logische Nummer 4

DEFINE

Syntax: "DEFINE" ENTER

Aktion: Rechner meldet sich "Name" ....> "alte log. Gerätenummer"
Eingabe der neu vereinbarten log. Gerätenummer, Bestätigung der alten log. Nummer mit ENTER oder Verlassen des Programmes mit OFF möglich. Nach jeder Eingabe wird die jeweils nächste Treiber-routine aus der ADT angeboten.

ACTIVATE

Syntax: "ACT" "Name 1", "Name 2", ... ENTER

Aktion: Die jeweils gewählten Treiber mit Kennbyte Ø2 in der RAM-Kettung werden in die ADT eingetragen und erhalten dabei die log.Nummer Ø. Ein Treiber kann auch mehrmals aktiviert werden. Anschließend wird LADT angesprungen.

DEACTIVATE

Syntax: "DEACT" "Name 1", '"Name 2", ... ENTER oder
"DEACT" ENTER

Aktion: Ein Treiber mit dem angegebenen Namen wird aus der ADT gestrichen und das Tabellenende verschoben. Wird kein Name angegeben, bleiben nur die ersten 4 bei INIT eingetragenen Treiber erhalten.
Danach erfolgt eine Anzeige der ADT.

DIRECTORY

Syntax: "DIR" ENTER

Aktion: Anzeige des Directorys (Kennbyte Ø4); bei mehr als 17 Positionen erneute Betätigung von ENTER.
Das Directory wird bei INIT auf die Adresse ØBAF8H gelegt.
Eine Zeile enthält folgende Informationen:
Name ... "Dateinummer auf Kassette"

RKET

Syntax: "RKET" "Segmentnummer", "Ausgangsadresse", "Zieladresse"

Aktion: Die RAM-Kettung wird von der Ausgangsadresse, die in der Kettung liegen muß, auf die Zieladresse geführt. Hat die Ausgangsadresse das Kennbyte ØFFH, wird dort freier RAM geschaffen. Das Programm ist nur innerhalb eines Segmentes anwendbar.

Nach Einschalten des Gerätes oder Absturz und einem damit verursachten Speicherlöschen ist das Kommando INIT aufzurufen, nur dann ist eine Weiterarbeit mit dem Magnetband und der Console möglich (Kommandos READ, WRITE, KINIT, CAT, CATA, DIR und LADT).
Mit dem Kommando INIT wird auf der Adresse ØBAF8H eine Directory-Datei mit dem Kennbyte Ø4 und der Länge 5Ø5H angelegt. Wenn eine solche Datei nicht in der RAM-Kettung steht, ist keine Magnetbandarbeit und keine Directory-Anzeige möglich. (ERROR %C7 bei den Kommandos READ, WRITE, KINIT und DIR)
Die Adresse ØBAF8H stellt lediglich eine Vorzugsadresse für das Directory dar. Ein Directory kann auch auf jeder anderen RAM-Adresse angelegt werden, dabei muß lediglich die Distanz in der RAM-Kettung dieser Ø4-Datei ohne Namen groß genug sein.
Nach dem Einlesen einer Datei vom Magnetband zeigt die RAM-Kettung nicht zwangsläufig auf das Directory, da sich die Kettung der eingelesenen Datei in dem Zustand befindet, wie die Datei beim Auslagern bis zur letzten ausgelagerten Adresse gekettet war. Deshalb ist es meist notwendig, nach dem Einlesen von Kassette mit Hilfe des Kommandos RKET vom aktuellen Ende der RAM-Kettung auf die Directoryadresse zu ketten. Mittels des Kommandos CAT 4 kann man sich jederzeit überzeugen, ob sich ein Directory in der Kettung befindet.
Beispiel:

1. Einschalten des Rechners
2. Eingabe: INIT (jede Eingabe ist mit der Taste ENTER abzuschließen)
   Der gesamte RAM wird gelöscht und die ADT angelegt. Die RAM-Kettung wird von der Adresse 4ØØØ über BAF8 - BFFC auf die Adresse CØØØH geführt. Von 4ØØØ bis BAF7H befindet sich freier RAM.
3. Einlegen der Systemkassette
4. Eingabe: READ undefinierte Datei
   Diese Datei gibt es natürlich nicht auf der Systemkassette, das Directory wird eingelesen und der Rechner meldet sich mit ERROR %D4 und bietet die Eingabe zur Korrektur noch einmal an.

5. Betätigung der OFF-Taste (Sie haben diesen Fehler absichtlich begangen. Sie wollten nur das Directory sehen, aber noch keine neue Datei in den Speicher einlesen).
6. Eingabe DIR (Es erfolgt eine Anzeige aller auf der Kassette stehenden Dateien und es kann die einzulesende Datei gewählt werden.)
7. Eingabe READ RAM.V (Es soll als Beispiel der verschiebliche RAM-Modus eingelesen werden. Der Zusatz ".V" bedeutet, daß es sich um eine verschiebliche Datei handelt, die vom Hersteller schon als Arbeitsbereich (mit leerer Datei Namens E am Anfang und leerer Datei Namens F am Ende) auf Ihre Kassette aufgezeichnet wurde. Dadurch werden diese Dateien auf die erste Adresse gelesen, auf der das Kennbyte ØØ für freien RAM steht und dessen Länge groß genug für diese Datei ist. Nach Ausführung von INIT ist das die Adresse 4ØØØ, wovon Sie sich durch CATA nach Ausführung von INIT leicht überzeugen können.
8. Eingabe RELAD RAM (Der RAM-Modus ist eine verschiebliche Datei, die zu diesem Zweck vom Hersteller vor dem Aufzeichnen auf Ihre Systemkassette auf die Adresse ØØØØ gebunden wurde. Jetzt soll der RAM-Modus aber auf der Adresse 4ØØØH laufen. Das Kommando RELAD bindet das angegebene Programm mit Hilfe einer Verschiebeadreßtabelle um. Sie dürfen RELAD RAM nicht noch einmal aufrufen, da dieses Programm jetzt nicht mehr auf die Adresse ØØØØ gebunden ist!)
9. Eingabe: CATA (Sie können sich damit einen Überblick über alle Dateien verschaffen, die zur Zeit im Speicher stehen. Die letzte Datei in der Kettung ist das Kommando FILL des RAM-Modus und die Kettung endet auf der Adresse 45ØØH. Eine Directory-Datei mit dem Kennbyte Ø4 finden Sie nicht in der Kettung.)
10. Eingabe: RAM BAF8 (Sie haben die Bedienungsanleitung aufmerksam gelesen und können sich nicht erklären, wodurch das alte Directory überschrieben worden sein könnte. Deshalb möchten Sie nachsehen, was jetzt auf dieser Adresse steht. Der RAM-Block ab der Adresse ØBAF8H wird angezeigt. Sie können den Aufbau der RAM-Kettung und des Directorys erkennen. Vergleichen Sie mit der Betriebssystembeschreibung, Punkt 5! Rechnen Sie sich aus, auf welcher Adresse das nächste Kennbyte steht und sehen Sie sich auch einmal den RAM ab dieser Adresse an! Mit CTR OFF können Sie den RAM-Modus verlassen. Sie kommen dann zurück in den Kommandoeingabemodus des Betriebssystems und können dort den RAM-Modus erneut aufrufen.)

11. Eingabe RKET Ø1,45ØØ,BAF8 (Sie haben erkannt, daß die Programme, die nach der Adresse 45ØØH stehen, zwar noch vorhanden sind, aber von der Speicherverwaltung nicht mehr gefunden werden können, da die Kettung durch das Einlesen des RAM-Modus unterbrochen wurde. Mit Hilfe des Kommandos RKET können Sie diese Lücke wieder schließen, ohne daß Sie die Distanz auszurechnen brauchen.)
12. Eingabe CATA (Sie sehen, daß jetzt das Directory wieder in der Kettung steht. Außerdem hat das Kommando RKET bewirkt, daß jetzt hinter dem eingelesenen RAM-Modus ab der Adresse 45ØØH, auf der nach Punkt 9 die Kettung abgebrochen war, jetzt das Kennbyte für freien RAM steht, der bis an die Directory-Datei reicht.)
    Jetzt können Sie mit dem Magnetband weiterarbeiten und z.B. andere verschiebliche Dateien einlesen. Wenn Sie Dateien einlesen wollen, die auf festen Adressen stehen, so ist das nur möglich, wenn auf dieser Adresse das Kennbyte ØØ für freien RAM steht. Anderenfalls meldet sich das Kommando READ mit ERROR %CD. Damit wird ein versehentliches Überschreiben von Dateien verhindert. Wenn Sie absichtlich eine Datei überschreiben möchten, müssen Sie dem Kommando READ dateiname den Zusatz ",R" anhängen.

Folgende Dateien befinden sich auf Ihrer Systemkassette (max.Lieferumfang)

Name im Directory	Adressen	Bemerkung
OCED.RAM	4ØØØ-63FF	Objectcodeeditor und RAM-Modus
EPROG.V	verschieblich	EPROM-Programmierung (nur MC 8Ø.3Ø)
RAM.V	verschieblich	RAM-Modus
BEDIT	4ØØØ-5FFF	BASIC-Editor und RAM-Modus (BASIC-Editor ohne Interpreter nicht nutzbar)
BINTER	CØØØ-D4FF	BASIC-Interpreter (Nach Einlesen kein RKET auf Directory nötig)
GRAB	9ØØØ-D4FF	Graphic-BASIC und BASIC-Interpreter und neuer Directory Kopf
BASIC	4ØØØ-D4FF	BASIC-Editor,RAM-Modus,Graphik-BASIC und BASIC-Interpreter mit neuem Directory-Kopf.

ØBBBH RLL Löschen eines RAM-Objektes mit FF und Anlegen von freiem RAM mit KB ØØ auf dieser Adresse

INPUT: HL= Zeiger auf das Kennbyte der zu löschenden Datei, (SEGD)= deren Segmentnummer
OUTPUT: HL,DE unverändert, BC zerstört
CY=1 und A=48H: HL zeigte auf unzulässiges KB

ØBBEH OUT Ausgabe von ASCII-Zeichen an die VIS innerhalb einer Zeile ab Position DE, Abbruch bei ØDH oder ØFFH oder BC=Ø

INPUT: BC= maximale Zeichenzahl
DE= Zeilen-/Spaltenposition auf Display
HL= Textpufferanfang
OUTPUT: BC= Ø (alle Zeichen geschrieben) oder tatsächlich geschriebene Zeichenzahl
HL= Zeiger auf Position nach letztem geschriebenen Zeichen, DE bleibt erhalten, A wird zerstört
CONTROL:19H Rest der Zeile löschen
Ø7H Piep an Tastatur

ØBC1H WBN Ausgabe von ASCII-Zeichen an die VIS ab aktuellem Pixel-Kursor (ØFB58H), auch über eine Zeile hinaus, Pixel-Kursor wird aktualisiert, Abbruch bei ØFFH oder BC=Ø

INPUT: BC= maximale Zeichenzahl
HL= Textpufferanfang
OUTPUT: BC= Ø (alle Zeichen geschrieben) oder tatsächlich geschriebene Zeichenzahl
HL= Zeiger auf Postion nach letztem geschriebenen Zeichen, DE bleibt erhalten, A wird zerstört
CONTROL:19H Rest der Zeile löschen
Ø7H Piep an Tastatur
ØDH CARRIGE RETURN LINE FEED

ØBC4H RSF Freibereich mit Länge BC in der RAM-Kettung suchen und auf die angegebene Länge führen, wobei mehrere Freibereiche hintereinander zusammengefaßt werden. Soll auf diesem freien RAM eine Datei angelegt werden, so ist zur benötigten Dateilänge noch die Länge des Dateinamens mit Zusatzkennbyte zu addieren.

INPUT: BC= Länge Freibereich +3 Byte für KB und Distanz

OUTPUT: Freiraum gefunden (CY=Ø)
HL= Zeiger auf Kennbyte Ø
A und (SEGD)= Segmentnummern des Freiraumes
BC= unverändert, DE zerstört

OUTPUT: Freiraum nicht gefunden (CY=1)
HL= Zeiger auf Fehlerstelle (Adresse, auf der ein unlässiges Kennbyte, das Kennbyte für Kettungsende oder das Kennbyte für freien RAM, der aber zu klein ist, steht), (SEGD)=Numner des Segmentes, in dem die Suche abgebrochen wurde
A= Fehlercode (%C7 oder %48)
BC= bleibt erhalten, DE zerstört

ØBC7H RSU RAM-Objektsuche im gesamten verwalteten Speicher

INPUT: DE= Ø (Datei ohne Namen suchen) oder Zeiger auf den Namen der zu suchenden Datei
(SEGE)=Nummer des Segmentes, in dem der Name steht
A= Kennbyte der zu suchenden Datei

OUTPUT: Datei gefunden (CY=Ø)
BC= Zeiger auf Kennbyte
HL= Zeiger auf Zusatzkennbyte
DE= Zeiger auf 1 Byte nach dem vorgegebenen Namen
A= (SEGD)= Nummer des Segmentes, in dem die gefundene Datei steht, (SEGE) unverändert

OUTPUT: Datei nicht gefunden (CY=1)
HL= Zeiger auf Fehlerstelle (Adresse, auf der ein unzulässiges Kennbyte oder das Kennbyte für Kettungsende steht, (SEGD)= Nummer des Segmentes,in dem die Suche abgebrochen wurde, (SEGE) erhalten,
A= Fehlercode (%C7 oder %48), BC zerstört
DE= Zeiger auf 1 Byte nach dem vorgegebenen Namen

ØBCAH RSA RAM-Objektsuche ab einer vorgegebenen Adresse, die auf ein Kennbyte zeigen muß. Die Suche erfolgt ab einschließlich dieser Adresse im vorgegebenen Segment.

INPUT: DE= Ø Datei ohne Namen suchen) oder Zeiger auf den Namen der zu suchenden Datei
(SEGE)=Nummer des Segmentes, in dem der Name steht
A= Kennbyte der zu suchenden Datei
HL= Adresse, ab der gesucht werden soll, (SEGD)= Nummer des Segmentes, in dem die Suche beginnt

OUTPUT: Datei gefunden (CY=Ø)
BC= Zeiger auf Kennbyte
HL= Zeiger auf Zusatzkennbyte
DE= Zeiger auf 1 Byte nach dem vorgegebenen Namen
A= (SEGD)= Nummer des Segmentes, in dem die gefundene Datei steht, (SEGE) unverändert

OUTPUT: Datei nicht gefunden (CY=1)
HL= Zeiger auf Fehlerstelle (Adresse, auf der ein unzulässiges Kennbyte oder das Kennbyte für Kettungsende steht, (SEGD)= Nummer des Segmentes in dem die Suche abgebrochen wurde, (SEGE) erhalten,
A= Fehlercode (%C7 oder %48), BC zerstört
DE= Zeiger auf 1 Byte nach dem vorgegebenen Namen

ØBCDH TST Tastaturabfrage (Abfrage der Arbeitszelle ØFB62H, nicht in Schleife gelegt (muß Anwenderseitig realisiert werden)

INPUT: keiner

OUTPUT: A= Tastencode (A=Ø: keine Taste gedrückt)
CY=Ø, Z=1:keine Taste gedrückt
CY=1, Z=Ø: Taste neu gedrückt
CY=1, Z=1: Taste repetierend betätigt
(ØFB62H) wird nach Abfrage mit ØØ gelöscht

ØBDØH ZLE Zahl dezimal oder hexadezimal von einem ASCII-Puffer lesen
Anzahl der zu lesenden Stellen ist variabel
Das Unterprogramm ist nicht segmentfähig!

INPUT: DE= Textpufferadresse der zu lesenden Zahl
A= Steuerwort (Bit 7=1: Zahl als Hexazahl lesen, Bit 7=Ø: Zahl als Dezimalzahl lesen und in Hexazahl umwandeln, Bit 2 bis Bit Ø: Anzahl der zu lesenden Stellen, aber maximal 4 bzw. 5).

OUTPUT: DE= Textpufferzeiger nach letzter gelesener Ziffer
HL= gelesene Zahl (hexadezimal), A= Anzahl der gelesenen Ziffern, CY=1: vorzeitiger Abbruch durch nicht als Ziffer interpretierbares Zeichen,
CY=Ø: alle Stellen gelesen, Z=1: Zahl war Ø
BC bleibt erhalten

ØBD3H WLN Ausgabe von ASCII-Zeichen an die VIS innerhalb einer Zeile ab Pixelkursorposition (ØFB58H), Abbruch bei ØDH ØFFH oder wenn BC=Ø, Pixelkursor wird aktualisiert

INPUT: BC= maximale Zeichenzahl
HL= Textpufferanfang

OUTPUT: BC= Ø (alle Zeichen geschrieben) oder tatsächlich geschriebene Zeichenzahl
HL= Zeiger auf Position nach letztem geschriebenen Zeichen, DE bleibt erhalten, A wird zerstört

CONTROL: 19H Rest der Zeile löschen
Ø7H Piep an Tastatur

ØBD6H BFE Umwandlung der Zeilen-/Spaltenposition für den Kursor in Pixelkoordinaten und Ablegen dieser in der Arbeitszelle ØFB58H

INPUT: D= Zeilenposition (Ø-18H)
E= Spaltenposition (Ø-47H / Ø-4FH)

OUTPUT: Register unverändert, Pixelkursor in ØFB58/59H abgelegt

ØBD9H CLB Löschen des Displays und des Rollzählers, Pixelkursor steht in linker oberer Ecke

INPUT/OUTPUT: keiner, alle Register gerettet

ØBDCH ROL Display um 1Ø Pixel nach oben rollen (Rollzähler wird um 1Ø decrementiert), bei Großschrift ist zweimaliges Rollen erforderlich

INPUT/OUTPUT: keiner, alle Register gerettet

ØBDFH ROU Display um 1Ø Pixel nach unten rollen (Rollzähler wird um 1Ø incrementiert), bei Großschrift ist zweimaliges Rollen erforderlich

INPUT/OUTPUT: keiner, alle Register gerettet

ØBE8H LIY Laden eines minimalen IY-Vektors (ohne Rückkehradressen, logische Gerätenummer, Fertigstellungscode und logische Gerätenummer)

INPUT: A= Anforderungscode, BC= Datenlänge
HL= Datenstartadresse, IY= Zeiger auf den zu ladenden RIO-Vektor

OUTPUT: (IY+1) = Anforderungscode
(IY+2/3)= Datenstartadresse
(IY+4/5)= Datenlänge

ØBEBH HBS Hexabyte in Textpuffer als ASCII-Zahl schreiben
(Unterprogramm ist nicht segmentfähig!)

INPUT: HL= Zeiger auf Textpuffer,
A= zu schreibendes Byte

OUTPUT: HL= Zeiger auf Byte nach letzter geschriebener Ziffer, alle Register bis auf A gerettet

ØBEEH CON Consolentreiber mit IY-Schnittstelle

INPUT: (IY+1) = Anforderungscode ((ØØ,ØAH,ØCH,ØEH,1ØH,4ØH,42H,48H)
(IY+2/3)= Datenstartadresse
(IY+4/5)= Datenlänge
(IY+ØAH)= Ø (nach Aufruf steht dort Fehlercode)
(IY+ØBH)= Textpufferkursor (nur bei 1ØH und 48H)

OUTPUT: (IY+3/4)= tatsächliche übertragene Datenlänge
(IY+ØAH)= Fehler- oder Fertigstellungscode
(IY+ØBH)= Textpufferkursor (nur bei 1ØH und 48H)
(IY+ØCH)= letztes Zeichen von der Tastatur
(nur bei Requests 1ØH und 48H)

ØBF1H NIXE Zentraler Geräteverteiler mit IY-Schnittstelle
Der IY-Vektor wird in Abhängigkeit von (IY+Ø) unter Berücksichtigung der Treiberzuordnung in der ACTIVE DEVICE TABLE auf den entsprechenden Treiber weitergeleitet.

INPUT: vollständiger IY-Vektor mit logischer Gerätenummer in (IY+Ø)

OUTPUT: Der Treiber, dem in der ADT die logische Gerätenummer entspricht, wird abgearbeitet und in NIXE zurückgekehrt, Beeinflussung des IY-Vektors durch angesprochenen Treiber alle Register zerstört

ØBF4H LOUT Ausgabe einer Textzeile auf das logische Gerät Nr. 3 über die Schnittstelle NIXE mit Requestcode ØEH (WRITE BINARY), Text muß mit ØDH/ØFFH enden

INPUT: HL= Zeiger auf Textpufferanfang

OUTPUT: alle Register zerstört

ØBF7H LZOU Ausgabe eines CARRIGE RETURN / LINE FEED an das logische Gerät Nr.3 über die Schnittstelle NIXE

INPUT/OUTPUT: keiner, alle Register unverändert

9. Aufbau des IY-Vektors

Der IY-Vektor stellt die einheitliche Schnittstelle zu allen Treibern dar. Damit wird die Geräteverwaltung RIO-kompatibel.
Der RIO-Vektor hat folgenden Aufbau:

IY	logische Gerätenummer (von 1 bis 14 H)
IY+1	Request-Code
IY+2/3	Datenstartadresse
IY+4/5	Datenlänge
IY+6/7	Rückkehradresse (vom Treiber verwendet)
IY+8/9	ERROR-RETURN-ADRESS (vom Treiber verwendet)
IY+ØAH	Fehler- oder Fertigstellungscode
IY+ØBH/ØCH	Folge-Vektor-Adresse bei benötigter Zusatzinformation

1Ø. Die Arbeit mit dem zentralen Treiberverwaltungsprogramm "NIXE"

Über das Unterprogramm NIXE wird die Kommunikation der logischen Treiber mit den logischen Geräten über die ACTIVE DEVICE TABLE (ADT) realisiert. Dadurch ist es mögich, durch Umweisung der Zuordnung von logischen Gerätenummern zu logischen Treibern mit einem logischen Treiber wahlweise verschiedene logische Geräte,wie z.B. Drucker, Display oder Lochbandperipherie zu bedienen. Dieses wird durch das Mitführen der logischen Gerätenummer im IY-Vektor vom Unterprogramm "NIXE" unter Zuhilfenahme der ADT ausgewertet. Zur Ausnutzung der vom Betriebssystem EGOS angebotenen Möglichkeiten wird daher empfohlen, alle vom Anwender selbst erstellten Programme zum Ansteuern von peripheren Geräten so zu schreiben, daß sie in Bedienprogramm, logische Treiber und physische Treiber unterteilt sind und als einheitliche Schnittstelle den IY-Vektor benutzen.
Vom Handler oder vom Bedienprogramm ist dann dieser IY-Vektor nicht direkt an den anzusprechenden logischen Treiber zu übergeben, sondern statt dessen die Softwareschnittstelle "NIXE" aufzurufen. Als logische Gerätenummer in (IY+Ø) des IY-Vektors ist eine Vorzugsnummer zwischen Ø3 und 14H einzutragen. Das Unterprogramm "NIXE" sucht dann in der ADT die Startadresse des logischen Treibers, die der in (IY+Ø) angegebenen logischen Gerätenummer entspricht. Die Zuordnung der logischen Treiber zu den logischen Gerätenummern kann über das Kommando "DEFINE" (Betriebssystembeschreibung, Punkt 7) geändert werden, so daß der Anwender sehr variabel, betreffend des Ansprechens unterschiedlicher externer Geräte,über ein und dasselbe Bedienprogramm ist. Die möglichen Request-Codes in (IY+Ø1) werden an den Treiber weitergegeben, unabhängig davon, ob der entsprechende Request für diesen Treiber sinnvoll ist oder nicht. Diese Auswertung muß der entsprechende Treiber übernehmen und einen unsinnigen Request mit einer Fehlermeldung ablehnen (ØC1H in IY+ØAH). Nach Abarbeitung des gewünschten Requestes im Treiber kehrt das Programm in NIXE zurück und von dort in das aufrufende Programm. Dort wird das Rückkehrsegment wieder eingeschaltet.
Eine ähnliche Geräteverwaltung wird in fast allen namenhaften Betriebssystemen durchgeführt. In CP/M beispielsweise übernimmt ein soqenanntes IO-Byte die Funktion der ADT, was aber nur eine sehr begrenzte Anzahl von externen Geräten erlaubt. Das Prinzip der ADT ist im Betriebssystem EGOS 3Ø an RIO/ ZDOS (UDOS)angelehnt,wobei in EGOS die ADT um die Information über die Segmentnummer des Treibers ergänzt werden mußte.
Im Betriebssystem des MC 8Ø.3Ø/31 sind folgende Grundzuordnungen vorhanden, die bei INIT in die ADT eingetragen werden: KLOG (logischer Magnetbandtreiber) Ø4
CON (Console Ausgabe auf Display) Ø3

Der logische Treiber Ø4 wird vom Magnetbandhandler (READ, WRITE, KINIT) aufgerufen. Die Kommandos CAT, CATA, DIR und LADT arbeiten mit den Unterprogrammen LOUT und LZOU, die über NIXE den logischen Treiber mit der Nummer Ø3 bedienen . Mit den Unterprogrammen LOUT und LZOU sind dem Nutzer Schnittstellen gegeben, in denen schon intern der IY-Vektor aufgebaut wird und das Gerät Ø3 über "NIXE" in der ADT gesucht wird. Diese beiden Unterprogramme arbeiten mit dem Request ØEH (WRITE BINARY).

Die Fehlerauswertung kann nach folgendem Muster erfolgen:

```
BSP1:  ......
       CALL      NIXE
       LD        A,(IY+ØAH)    ;Fehlercode
       CMP       8ØH           ;fehlerfrei ?
       JRZ       WEIT          ;ja: weiter im Programm
       SCF
       RET                     ;Rückkehr in BSYS mit CY=l
WEIT:    ......
```

Folgende Requestcodes werden im BSYS "EGOS 3Ø" verwendet:

ØØ	INITIALIZE	Initialisierung des Treibers	(CON,KLOG)
Ø4	OPEN	siehe Magnetbandbeschreibung	(KLOG)
Ø6	CLOSE	"	(KLOG)
ØAH	READ BINARY	Daten eingeben	(CON,KLOG)
ØCH	READ LINE	Daten eingeben bis ØDH	(CON)
ØEH	WRITE BINARY	Daten ausgeben	(CON,KLOG)
1ØH	WRITE LINE	Daten ausgeben bis ØDH	(CON)
4ØH	READ STATUS	Status einlesen	(CON)
42H	WRITE STATUS	Status ausgeben	(CON)
48H	READ CORRECTURE	wie READ LINE, aber mit Korrekturmöglichkeit	(CON)

Aufbau der ACTIVE DEVICE TABLE:

Die ADT ist im Betriebssystem EGOS eine 4-Byte-Tabelle und steht im ZVE-RAM ab der Adresse ØFCØØH. Die Tabelle endet mit ØØ. Die Kommandos INIT, ACT, DEACT und DEFINE beeinflussen die ADT in der gewünschten Weise, mit DEFINE kann man die Zuordnung Treiber <-->log. Gerätenummer verändern.

```
Byte 1:   Bit 7= Akivbit (immer gesetzt), Bit 5.6 frei
          Bit 4 bis Bit Ø= logische Gerätenummer
Byte 2:   Segmentnummer des Treibers
Byte 3/4: Adresse des Kennbytes des Treibers
```

11. Die Arbeit mit den Schnittstellen zur Segmentschaltunq

Das Betriebssystem des MC 8Ø.3x untersützt die Möglichkeit, auch Speicher über den 64 kByte adressierbaren Adressraum des Prozessors zu nutzen und zu verwalten. Dazu ist es erforderlich, die 8 Ausgänge des PIO-Port's B die auf den Koppelbus geführt sind. mit den entsprechen-

den MEMDI-Eingängen der verwendeten Speicherkarten zu verbinden. Alle zusätzlichen Speicherkarten müssen auf die Adresse 4ØØØH gewickelt sein und dürfen pro Segment nicht über 32 kByte Speicherplatz hinausgehen. Alle Segmente haben demzufolge die Adressen 4ØØØH bis maximal BFFFH und sind erst durch ihre Segmentnummer adressmäßig eindeutig charakterisiert. Es ist darauf zu achten, daß nicht mehrere Segmente gleichzeig eingeschaltet sind. Deshalb darf im auszugebenden Datenwort zur Segmentumschaltung nur 1 Bit gesetzt sein *)Mit einer Ausgabe auf die Kanaladresse 85H wird die Segmentumschaltung realisiert. Mögliche Segmentnummern sind ØØ (alle Segmente gesperrt), Ø1, Ø2, Ø4, Ø8 , 1ØH. 2ØH, 4ØH und 8ØH. Es ist darauf zu achten, daß alle Programme, die eine Segmentumschaltung durchführen, nicht im segmentierbaren Speicher stehen, da sie sich mit Ausgabe des Datenwortes zur Segmentumschaltung selbst wegschalten würden. Aus diesem Grunde werden ab der Adresse 2ØØØH einige Softwareschnittstellen zur Segmentumschaltung angeboten, die mit CALL aufgerufen werden können.

2ØØ4H JHL : simuliert Befehl JMP (HL) in das Segment, dessen Nummer in A steht

2ØØ7H JHD : simuliert Befehl JMP (HL) in das Segment, dessen Nummer in (SEGD) steht

2ØØDH CHL : simuliert Befehl CALL (HL) in das Segment, dessen Nummer in A steht, nach Abarbeitung erfolgt Rückkehr in das aufrufende Programm. A zerstört

2Ø1ØH CHD : simuliert Befehl CALL (HL) in das Segment, dessen Nummer in (SEGD) steht, nach Abarbeitung erfolgt Rückkehr in das aufrufende Programm. A zerstört

2Ø1CH SYS : überträgt 3 Byte Programm ab Adresse nach SEGP, arbeitet das Programmstück bezüglich der Zielsegmentnummer, die in (SEGD) steht ab und kehrt auf die Adresse nach dem Aufruf im Segment (SEGC) zurück

2Ø1FH S1C : wie SYS. aber Anzahl der zu übertragenden Bytes im Register C

2Ø22H S1A : wie SYS, aber Anzahl der zu übertragenden Bytes im Register A

2Ø25H S2C : wie SYS, aber bezüglich (SEGE) als Zielsegmentnummer und Anzahl der zu übertragenden Bytes im Register C

2Ø28H S2A : wie SYS, aber bezüglich (SEGE) als Zielsegmentnummer und Anzahl der zu übertragenden Bytes im Register A.

*) Vom Anwender ist durch geeignete Testmöglichkeiten zu garantieren, daß nur 1 Bit im Datenwort gesetzt ist, da bei mehreren gesetzten Bits auch gleichzeitig mehrere Segmente parallel angesprochen werden. Dies kann zur Zerstörung der Treiber Speichersteckeinheiten führen.

2Ø37H LDIR: simuliert Befehl LDIR, Quelladresse im Segment, dessen Nummer in (SEGD) steht, Zieladresse im Segment (SEGE)

2Ø3AH LDDR: simuliert Befehl LDDR, Quelladresse im Segment, dessen Nummer in (SEGD) steht, Zieladresse im Segment (SEGE)

2Ø3DH LØL : simuliert Befehl LD A,M aus dem Segment, dessen Nummer in A steht

2Ø4ØH LØE : simuliert Befehl LD A,(DE) aus dem Segment, dessen Nummer in A steht

2Ø43H LØX : simuliert Befehl LD A,(IX) aus dem Segment, dessen Nummer in A steht

2Ø46H LØY : simuliert Befehl LD A,(IY) aus dem Segment, dessen Nummer in A steht

2Ø49H L1L : simuliert Befehl LD A,M aus dem Segment, dessen Nummer in (SEGD) steht

2Ø4CH L1E : simuliert Befehl LD A,(DE) aus dem Segment, dessen Nummer in (SEGD) steht

2Ø4FH L1X : simuliert Befehl LD A,(IX) aus dem Segment, dessen Nummer in (SEGD) steht

2Ø52H L1Y : simuliert Befehl LD A,(IY) aus dem Segment, dessen Nummer in (SEGD) steht

2Ø55H L2L : simuliert Befehl LD A,M aus dem Segment, dessen Nummer in (SEGE) steht

2Ø58H L2E : simuliert Befehl LD A,(DE) aus dem Segment, dessen Nummer in (SEGE) steht

2Ø5BH L2X : simuliert Befehl LD A,(IX) aus dem Segment, dessen Nummer in (SEGE) steht

2Ø5EH L2Y : simuliert Befehl LD A,(IY) aus dem Segment, dessen Nummer in (SEGE) steht

2Ø61H S1L : simuliert Befehl LD M,A in das Segment, dessen Nummer in (SEGD) steht

2Ø64H S1X . simuliert Befehl LD (IX),A in das Segment, dessen Nummer in (SEGD) steht

2Ø6AH S2L . simuliert Befehl LD M,A in das Segment, dessen Nummer in (SEGE) steht

2Ø6DH S2X : simuliert Befehl LD (IX),A in das Segment dessen Nummer in (SEGE) steht

Die Segmentmerkzellen liegen auf folgenden Adressen:

SEGC: ØFBFBH als Rückkehrsegmentnummer zu verwenden
SEGD: ØFBFDH Zielsegmentnummer 1
SEGE: ØFBFFH Zielsegmentnummer 2
SEGP: ØFBDCH Puffer zur Programmabarbeitung (3Ø Bytes lang) von allgemeinen Segmentumschaltprogrammem des Nutzers im nichtsegmentierten ZVE-RAM

12. Verschiebeadresstabellenerstellung und Relativlader

Ein mit üblichen Programmierhilfsmitteln (Editor-Assembler-Linker oder Objektcodeeditor)erstelltes Assemblerprogramm ist meist nur auf einer einzigen Adresse lauffähig, da der Befehlssatz des Prozessors U 88Ø außer 6 Sprungbefehlen keine weiteren relativ adressierbaren Befehle aufweist. Um die Vorteile der Speicherverwaltung des Betriebssystemes EGOS 3Ø aber voll nutzen zu können und eine variable Speicheraufteilung erzielen zu können, bietet es sich an, möglichst alle Programme mittels einer sogenannten Verschiebeadresstabelle mit wenig Aufwand und geringem zusätzlichen Speicherbedarf auf jeder beliebigen RAM-Adresse lauffähig zu machen. Aus diesem Grund bietet das Betriebssystem EGOS 3Ø ein Programm zum Erstellen von Verschiebeadresstabellen und einen Relativlader an. Auf der Systemkassette befinden sich nur 2 verschiebliche Programme: RAM.V und EPROG.V. Diese Programme sind auf die Adresse Ø gebunden und besitzen eine Verschiebeadresstabelle mit dem Kennbyte 1ØH. Mittels des Kommandos

```
RELAD programmname
```

können diese Programme auf die Adresse, auf der sie physisch stehen, gebunden werden. Im Vergleich zu einer Markentabelle ist die Verschiebeadresstabelle sehr kurz, da sie nur die

auf den Programmanfang bezogenen und auf Ø gebundenen Adressen enthält. Das auf Ø gebundene Programm unterscheidet sich von dem auf eine andere Adresse gebundenen Programm durch (z. B. CALL-Adressen, JMP-Adressen, Lade-Adressen von Doppelbyteregistern).

Das Kommando RELAD darf nur einmal auf ein Programm angewendet werden, da nach Aufruf von RELAD das Programm nicht mehr auf Ø, sondern auf die Ladeadresse gebunden ist und der Relativlader eine falsche Bezugsadresse hat.
Das Erstellen von Verschiebeadresstabellen ist relativ aufwendig und sollte nur dem erfahrenen Programmierer vorbehalten bleiben.
Bedingung ist, daß das Programm zweimal im Speicher vorliegt, einmal auf die Adresse Ø gebunden und einmal auf eine andere, beliebige Adresse gebunden:

Programm 1:	auf Adresse Ø gebunden und auf Adresse X stehend
Programm 2:	auf Adresse Y gebunden und auf Adresse Y stehend
Tabelle:	Adresse Z (Zieladresse der Verschiebeadresstabelle)

Weiterhin wird die Programmlänge benötigt. Alle Angaben schließen den Programmkopf für die RAM-Kettung aus. diese müssen nachträglich mit Hilfe des RAM-Modus erstellt werden. Damit können aber auch für andere U 88Ø-Rechner verschiebliche Programme erzeugt werden.

```
22A8H  VERA  Verschiebeadresstabellenerstellprogramm

             INPUT:  IX,HL,DE = Adresse X
                     BC = Programmlänge ohne Kopf
                     IY = Adresse Y
                     im Stack: Zieladresse, auf der die Ver-
                     schiebeadresstabelle angelegt werden
                     soll (ohne Kopf!)
             OUTPUT: alle Register verändert
```

Das Programm VERA ist nach folgendem Muster aufzurufen:

```
BSP2:  PUSH  IY              ;localer Stack muß gerettet
                             ;werden
       LD    IY,Adresse Y
       LD    HL,Adresse Z
       PUSH  HL              ;Adresse Z im Stack übergeben
       LD    HL.Adresse X    ;HL,DE und IX mit Adresse X
                             ;laden
       PUSH  HL
       PUSH  HL
       POP   DE
       POP   IX
       LD    BC,Länge
       CALL  VERA            ;Tabellenerstellung
       POP   HL              ;Stack wieder sauber
       POP   IY              ;Bereich für localen Stack und IY-
       RET                   ;Vektor wieder da
```

Ausgangspunkt für das Kommando RELAD ist das auf Ø gebundene Programm und die Verschiebeadresstabelle. Beide Dateien müssen den selben Namen erhalten und zum Zwecke eines verschieblichen Auslagerns mit Magnetband in einem E/F-Arbeitsbereich mit Kennbyte 13H eingebettet liegen. Zwischen dem Programm mit dem Kennbyte Ø1 und der Verschiebeadresstabelle mit dem Kennbyte 1ØH sollten mindestens noch 6 Byte ØFFH liegen, da sonst bei RELAD die RAM-Kettung zerstört wird. Die Programme müssen auf folgende Weise zusammengeschoben und gekettet werden:

```
    13H   E         Arbeitsbereichsanfang
     |
    Ø1H   name      auf Ø gebundenes Programm mit 6 Byte
     |              ØFFH dahinter
     |
    1ØH   name      Verschiebeadresstabelle
     |
    13H   F         Arbeitsbereichsende
```

Beim Auslagern des erstellten, nun verschieblichen Programmes ist darauf zu achten, daß keine Adressen angegeben werden, sonst kann das Programm nur auf die angegebene Andresse wieder eingelesen werden und die ganze Mühe wäre umsonst gewesen! Es darf sich beim Auslagern auch kein anderer Arbeitsbereich in der RAM-Kettung befinden, notfalls muß dieser einfach überkettet werden.

13. Die RESET-Initialisierung mit Kennbyte Ø3

Bei Betätigung von RESET arbeitet das Betriebssystem nacheinander alle Routinen mit Kennbyte Ø3, beginnend am Anfang der RAM-Kettung, ab. Die RESET-ROUTINEN müssen als Unterprogramm geschrieben sein. IY muß unbedingt gerettet sein. Die Rückkehr muß mit gelöschtem CY-Flag erfolgen. RESET-Routinen können Anwendung finden für die Initialisierung von zusätzlichen Geräten oder für die Ausschrift eines eigenen Initialisierungstextes auf dem Display. Sie stehen in der RAM-Kettung mit Kennbyte Ø3 und dürfen keinen Namen erhalten. Augenmerk ist darauf zu richten, daß keine falsch gebundenen oder fehlerhaften Ø3-Routinen in der RAM-Kettung stehen. In diesem Falle wäre nur durch Ausschalten des Rechners und erneutes Einschalten eine Fortsetzung der Arbeit möglich, da durch Betätigung von RESET diese fehlerhafte Routine immer wieder abgearbeitet würde. Dieser Effekt würde auch bei einem RST Ø oder JMP Ø am Ende der Routine auftreten.

14. Die Arbeit mit den Sondertasten (8-ter-Feld)

Das Tastaturprogramm ist so ausgelegt, daß der Anwender diese Sondertasten selbst mit Funktionen belegen kann. Die Tasten liefern die Codes zwischen Ø und 7. Bei Betätigung einer dieser Tasten wird von der SIO auf der ASP ein INT ausgelöst. In der INT-Routine wird getestet. ob auf der Aussprungstelle für Sondertasten ein C3 (JMP nn) steht. Ist das nicht der Fall, wird die Betätigung übergangen, anderenfalls die nachfolgend eingetragene Adresse angesprungen. Der Anwender hat die Fortführung der Tastaturinterruptroutine als Unterprogramm zu schreiben d.h. mit RET abzuschließen. Die INT-Routine muß kurz (<lØms) sein, da während dieser Zeit alle weiteren INT gesperrt sind. Somit ist während der Abarbeitung dieser Routinen auch keine Tastaturbetätigung möglich. Für längere Anwenderprogramme ist es empfehlenswert, mittels dieser kurzen INT-Roztinen niedriger priorisierte INT-Routinen (z.B. CTC) anzustoßen.

Tastencode	Adresse für Anwender
ØØ	ØFEDØH C3 nn nn
Ø1	ØFED3H C3 nn nn
Ø2	ØFED6H C3 nn nn
Ø3	ØFED9H C3 nn nn
Ø4	ØFEDCH C3 nn nn
Ø5	ØFEDFH C3 nn nn
Ø6	ØFEE2H C3 nn nn
Ø7	ØFEE5H C3 nn nn

Die Sondertasten im Achterfeld haben folgende Tastencodes:

Ø6	Ø5
Ø4	Ø3
Ø2	Ø1
ØØ	Ø7

Bei Arbeiten mit dem Objektcodeeditor dient die Sondertaste mit dem Tastencode ØØ als Break-Taste. Sie ist deshalb nicht als Sondertaste verwendbar.