Syst. av. [redacted] Gera. [redacted] b.20.9.07

1. [illegible].V
2. SB1152S.V
3. AG [illegible]
4. OOS[illegible]
5. GRAB
6. BEDIEP
7. BEITERR
8. BASIOP
9. BÜDTERP
10. SGOS. [illegible]
11. PASPAYER

KILL.V
BAS Quelle
SB1152S.Quelle

Koll. [redacted] 2404

VEB Elektronik Gera
- [redacted] -

6. Jan. 1987

N u t z u n g s h i n w e i s e
zur Nachnutzung von Software für MC 80/30, 31 (V.1.0Y 1,2,3)

Das Angebot besteht in einer Verbesserung der Systemkassette Stand G3

1. Basic

1.1. Veränderungen im Interpreter

1.1.1. LEN → (→ Stringvariname → (→ von Zeichen → , → Anzahl Zeichen →)→
vorher: Länge des Textes, der durch Textausdruck erzeugt wird, normalerweise im Programm vorgegeben

jetzt: Position des 1. Leerzeichens von links an, damit Ermittlung der Länge einzelner Worte in Texten

1.1.2. INKEY$
jetzt: repetierende Abfrage der Tastatur

1.1.3. Booleanausdruck: Vergleich von Texten
vorher: max. 31 Zeichen in einem Text

jetzt: max. 127 Zeichen

1.1.4. INSTR
vorher: Text links max. 20 Zeichen

jetzt: Text links und rechts je max. 127 Zeichen

1.1.5. PRINT
vorher: max. 70 Zeichen, Fehlercode kann von Basic nicht getestet werden

jetzt: max. 127 Zeichen
PRINT ist eine RIO-kompatible Ausgabeschnittstelle auf Log. Gerät Nr. 3 mit Request-code 0E#h
Nach Ausgabe kann Fehler- oder Fertigstellungscode mit FNE$ ausgewertet werden (z. B. Test auf Papierende vom Drucker)

1.1.6. INPUT, INPUT
vorher: keine allgemeine Schnittstelle, keine Auswertungen auf letzte Taste und Fehlercode möglich

jetzt: - RIO-kompatible Eingabeschnittstelle vom Log. Gerät Nr. 3 mit Request-Code 48 h (INPUT) bzw. 49 h (INPUT)

- kam die Eingabe von Systemtreiber "CON", dann kann der Tastencode, der die Eingabe beendete, mittels KINPØ abgefragt werden (=IY+ØCh) z. B. Tabelleneingabe

- Fehler- bzw. Fertigstellungscode kann nach Eingabe mit FMEØ (=IY+ØAh) ermittelt werden, z. B. Paritätsfehler bei Lochstreifenlesen.

Anforderungen an den Treiber:

INP: IY + 4/5: angeforderte bzw. vorgegebene maximale Datenlänge
IY + 2/3: Datenstartadresse, bei INPUT (Text) Var steht ab dort der vorgegebene Text

OUP: entsprechend RIO-Vektor

zusätzlich bei INPUT: - Treiber rettet Informationen aus IY-Vektor
- Anstoßen der interruptgesteuerten Eingabe
- mit RET zurückkehren
- Eingaberoutine trägt die Zeichen in den Puffer ein
- nach Eingabeende BIT Ø in ØFBØFh für INPRDY auf 1 setzen und zur Adresse aus (IY+6/7) zurückkehren

1.1.7. Verlängerung des nutzbaren Basicstacks
vorher: von F6ØØ bis FØØØ: damit Begrenzung von Feldern

jetzt: in BINTERR: F6ØØ bis E6ØØ
in RINTERP: F6ØØ bis E9ØØ
Der Stack wird bis maximal 2ØØ h tiefer belegt!

1.1.8. Abarbeiten von Maschinenprozeduren
vorher: Parameterübergabe nicht mit Betriebssystem kompatibel

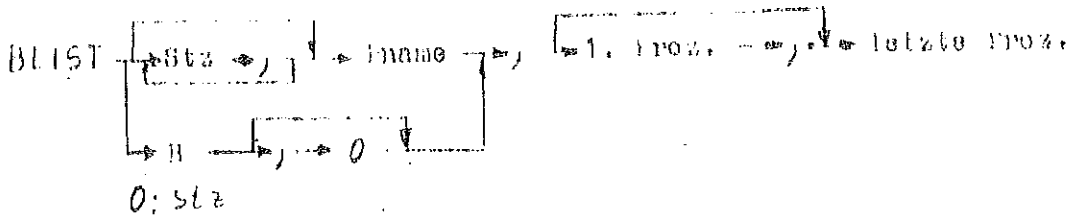
jetzt:
Syntax: dname␣→Textausdruck→
dname: Dateiname mit Kennbyte Ø1
INP: DE zeigt auf den im Textausdruck erzeugten Text;
Textende: ØFFh
IY: zeigt auf nutzbaren Stack, maximal bis IY+8Øh nutzbar
Die Syntax im Text muß den Forderungen der Prozedur entsprechen
OUP: wenn Fehler, dann CY=1, A: Fehlercode
ein Fehler führt nicht zum Programmabbruch
Fehlercode kann mit FMEØ gelesen werden
ist kein Text vorhanden, darf der Stack oberhalb IY nicht genutzt werden und DE muß gerettet werden!

K o r r e k t u r b l a t t

Zu Nutzungshinweise von Software für PC 80/30.31

S. 3 Punkt 1.2.2. Letzte Zeile: 'A' bis '^'

S. 5 Punkt 1.3.1.



S.6 Punkt 1.3.2.

- Zeile 10: Die Bedingungsabfrage ...
- Die globalen Vereinbarungen werden auch strukturiert ausgegeben.

Punkt 1.3.4.

- Zeile 9
  14 Ersatz '{ 144 }'
- Zeile 15: BLIST ...

S. 7 Punkt 2.

Beachten: alle Programme arbeiten wie OCED nur im 1. Segment!

Punkt 1.5.

Hinweis: Die Prozeduren von GRAB können nur von Prozeduren im gleichen Segment aufgerufen werden!

Zeile 7: BINTERP mit BINTER, Reset, Directory und ACTIVATE

S. 9 Punkt 2.4. Zeile 10: auf RAM-Adresse >= Endadresse

Punkt 2.7. c) Zeile 2: das 3. Zeichen ...

S. 11 Punkt 3.1.

Hinweis: Der Text soll im nichtsegmentierten Bereich oder im gleichen Segment wie ACTIVATE stehen. Ist vom Basic und Kommandoebene gewährleistet.

S. 12 Punkt 4.1.

- Zeile 12: ist letztes Zeichen ...

Hinweise für alle Treiber: Da über den RTO-Vektor (IY) mit der Pufferadresse nicht die Segment-Nummer übermittelt wird, muß der Text im nichtsegmentierten Bereich (>= C000h) bzw. im Segment des Treibers stehen. Von Basic gewährleistet.

S.14 Punkt 5.1,
Syntaxdiagramm: vor name muß ","

Punkt 5.11, Eingabemodus:

← → Kursorbewegung

S. 15 Punkt 5.3, Zeile 4
Kopieren von name bis einschließlich name im Segment Sgnr 1 nach name 2 im Segment Sgnr 2

Punkt 6,
Hinweis: Bei den in den letzten Monaten ausgelieferten Geräten ist das Tastatur-EPROM eingelötet!

S. A 4
Hinweis zum Arbeiten mit Datendateien: In Vorbereitung sind Erweiterungen, die das Schreiben und Lesen von Datenfiles im RAM ohne Verwendung globaler Variabler ermöglicht (etwa ab 07/87).

S. A 6
```
POKE  DP' 10, &H1528
      DP' 13, &H1523

PEEK  DP' 13, PEEK'
PACK  DP' 'ACTIVATE'
```

S. A 6 Punkt 8,

Beachte entfällt!

z. B.: Es können alle Prozeduren des Betriebssystems genutzt werden, die Parameter über den Text vermittelt bekommen.

z. B: Nutzung der Magnetbandbefehle KINIT, WRITE, READ, Aktivierungsbefehl ACTIVATE

z. B.: LET A = 2: D$(Ø,4) = 'NAME'
READ FORMAT!1.Ø;!,!; D$(Ø,4)
führt aus: Lesen der Daten "NAME" vom Laufwerk 2

READ FORMAT! 1.Ø; A; ', '; D$(Ø,4)

1.2. Zusätzliche Basicbefehle

1.2.1. Realfunktion

VAL → (→ Textausdruck →) → (Value)
Lesen einer Zahl in ASCII-Zeichen unter Übergehen von Leerzeichen am Anfang und Leerzeichen zwischen Vorzeichen und Zahl.

z. B. Analyse einer Eingabe, die aus Zahl und Dimensionszeichen besteht

z. B. Lesen einer formatierten Zahl bei Tabelleneingaben

z. B. INPUT (FORMAT !3.2.1;A) E$(Ø,7)
LET A = VAL (E$(Ø,7))

CODE → (→ Textausdruck →) →

Ausgabe des Wertes des Binärcodes des 1. Zeichens im Text. Ist Umkehrung zu CHR$

1.2.2. Stringfunktion

- RDIR → (→ arithm. Ausdruck →) → (read directory)
Ausgabe des Dateinamens aus dem Directory; max. 327 Zeichen
arithm. Ausdruck: Nr. der Datei; Ø entspricht 2 in DIR
ist 1. Zeichen ein Leerzeichen, dann keine Datei mit dieser Nr. vorhanden
z. B. Menügesteuertes Einlesen von Datendateien

- HEX → (→ arithm. Ausdruck →) →
Ausgabe 4 Zeichen des in Hexadezimalform gewandelten arithm. Ausdrucks; korrekte Wandlung nur im Bereich ± 65535
z. B. Adressenberechnung für ~~SAVE~~ WRITE

- ERR$ (Fehlermeldung)
Ausgabe 2 Zeichen vom RIO-Vektor IY + ØAh in Hexadezimalform von PRINT, INPUT, XINPUT und Maschinenprozeduren, nur gültig sofort nach der Aktion. Maschinenprot: Inhalt von A, wenn CY=1; sonst 80h

- KINP$
Ausgabe des Codes von RIO-Vektor IY+ØDh nach INPUT
  - Zeichen rechts vom eingegebenen Text (Überlauf)
  - LF, ENTER, OVF (ØAh, 11h, 13h)
  - alle Zeichen der Control-ebene ("@")
    Steuerzeichen bis ';': Code + 3Øh
    'A' bis '↑': Code + AØh

Folgende Steuerzeichen sind konstant vordefiniert als globale Stringvariable für INPUT: ENTER; cENTER; OFF; cOFF; LF; cLF; cBS; cVT; cFF; cCR;

für Display: BEL (Piepton)
EM (Rest der Zeile löschen)

für Drucker etc: LF, VT, FF, HT, BS, ESC, DEL, NUL, CR

auf die Steuerzeichen darf nur lesend zugegriffen werden!

1.2.3. Prozeduren

- EXC → (→ Var 1 → , → Var 2 →) → (exchange)
schneller Austausch der Werte zweier Variablen
Var: REAL, INTEGER, STRING

Fehler: unterschiedliche Typen, Strings unterschiedlich lang
z. B. schnelles Sortieren von Feldern

- FORMATV␣→Textausdruck → (format variable)
Setzen des Ausgabeformates von Zahlen, variabel programmierbar. Die Textausgabe muß in der gleichen Prozedur bzw. Unterprogrammebene erfolgen

Textausdruck: 5 Zeichen mit Syntax ähnlich FORMAT
→ Zeichen → Ziffer → , → Ziffer → Zeichen →

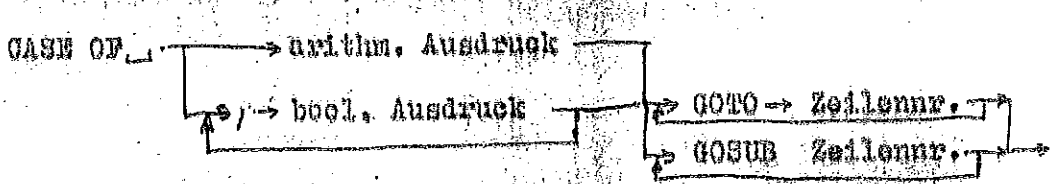
Zeichen '!'; ' ' : setzen bzw. löschen von '!'
'?' : vorhandene Information bleibt

Ziffer 'Ø,...7' : setzen der Ziffer
'8' : vorhandene Information bleibt

- ROLO (ohne Parameter)
Bild um eine Zeile nach oben schieben

- ROLU (ohne Parameter)
Bild um eine Zeile nach unten schieben
z. B. Für die Eingabe in langen Tabellen ist beim Schieben der Tabelle auf dem Bildschirm nur die Ausgabe der ersten oder letzten Zeile vom Basic aus notwendig.

1.2.5. Fallabhängige Verzweigung mit GOTO/GOSUB-Liste



- Ermitteln der Nr. des Zweiges

a) arithm. Ausdruck: gleich Integer-Wert
< = Ø und > Anzahl der Zweige: übergehen

b) bool. Ausdruck: gleich Nr. des letzten wahren Ausdruckes
keiner wahr: übergehen

- Programm wird im berechneten Zweig fortgesetzt

- RETURN bei GOSUB-Liste: Rücksprung zur Zeile nach der Liste

- Ende der Liste: Befehl, der verschieden vom 1. Befehl der Liste ist

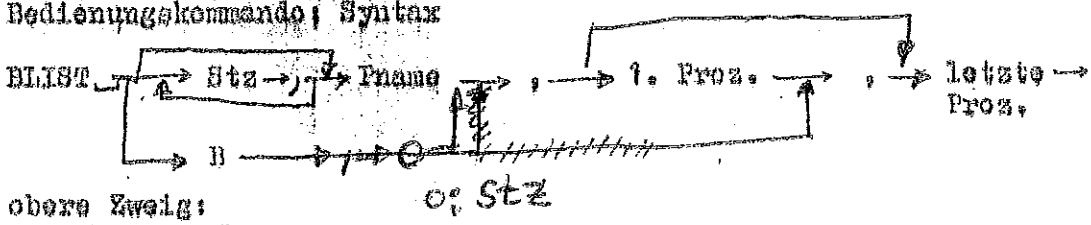
z. B. bessere Strukturierung des Programmes bei der Reaktion auf eine Eingabe, wenn mehr als 2 Reaktionsfälle auftreten.

z. B. Tabelleneingabe

```
BEGIN LOOP

  BEGIN LOOP                                   ; Eingabeschleife
     DPL Z, S;
     INPUT  (E$(Ø,4)) E$(Ø,4)
     LET T$ (Ø)=KINP$
   IF INSTR (T$(Ø) = ENTER; cVT;cLF; OFF)=ØØØ DO
  LOOP

  IF T$(Ø) <> OFF DO                           ; Verlassen

  CASE OF INSTR (T$(Ø) = ENTER; cVT; cLF); Reaktion auf
  Tastencode
     GOSUB 1ØØØ                                ; ENTER
     GOSUB 2ØØØ                                ; cVT (c ↑ )
     GOSUB 3ØØØ                                ; cLF (c ↓ )

LOOP
```

1.3. Strukturgerechtes Listen und Kommentieren von Basicprogrammen

1.3.1. Bedienungskommando; Syntax



obere Zweig:
Druckerausgabe:
kein Stz: mit Kommentar, kurze Zeile, Endlospapier
mit Stz: O : ohne Kommentar
L : lange Zeile
P : Stop der Ausgabe nach Seitenwechsel z. B. für Drucken auf Einzelblättern

untere Zweig: Ausgabe nur auf Bildschirm

Pname: Name des Basicprogrammes
1. Proz.: Dateiname der 1. Basicprozedur
globale Vereinbarungen: V
Hauptprogramm: ohne Namen

letzte Prozedur: Es werden alle Basicprozeduren von der ersten bis letzten in der Reihenfolge wie im RAM ausgegeben.

Die globalen Vereinbarungen werden auch strukturiert ausgegeben.

1.3.2. Ausgabeform

Das Gesamtprogramm wird fortlaufend mit 8 Zeilen weniger als Zeilen pro Format seitenweise ausgegeben. Jede Seite beginnt mit einem Kopf aus Seitenzahl, Programmname und Name der laufenden Datei. Die Ausgabe wird beendet mit dem Verzeichnis aller Prozeduren und ihrer Seitennummer;

Die Basic-Strukturelemente: BEGIN LOOP-IF ... DO - LOOP WHILE ... DO - LOOP; IF ... DO - ELSE DO - DOEND;
IF ... THEN - Zeilen-Nr.; FOR ... TO - NEXT werden durch Einrücken der Zeilen innerhalb der Struktur gekennzeichnet.
Die Bedienungsabfrage in BEGIN LOOP und ELSEDO werden durch Linksrücken gekennzeichnet.
Um die Übersichtlichkeit bei langen Strukturen zu erhalten, werden die Einrückungen mit Punkten gekennzeichnet; Damit wird eine maximale Tranzparenz des Listungs gewährleistet.
Die Konstruktion BEGINLOOP - IF ... DO - ELSED - LOOP ist zu vermeiden!

1.3.3. Kommentarsteuerung

Zu jeder Zeile kann ein Kommentar eingegeben werden, der mit ausgedruckt wird, aber nicht gespeichert werden kann.

Steuerung: ('c': Control-Ebene)

c ENTER:	Kommentarausgabe und nächste Zeile
c ↓ :	Kommentarausgabe und weitere Kommentarzeile möglich
c OFF :	Kommentar übergehen und nächste Zeile
c N:	Kein Kommentar bis nächste ZeilenNr.
c L :	Kein Kommentar bis Prozedurende
OFF :	Ausgabestop
ENTER :	Ausgabe weiter nach Fehler (BEL), Stop, Seitenwechsel

Während Eingabe ist die Tastatur auf Schreibmaschinenmodus!

1.3.4. Anpassung an unterschiedliche Ausgabetreiber

Adresse: Adresse des Konnbytes Ø 1 von BLIST + Anzahl (dezimal) Bytes

Anzahl	Inhalt	Bedeutung
11	48 h	Anzahl Zeilen pro Format: hier 72
12	48 h	Anzahl Zeichen pro kurze Zeile: hier 72 (A4-Format)
13	84 h	Anzahl Zeichen pro lange Zeile; hier 132 (A4-Format quer)
14	1Bh'[144 }'	Steuerfolge für SD 1152 für 72 Zeilen pro Format; kann durch ØØh für alle ergetzt werden
30 bis 40	'SD1152S'	Treibername für Drucker SD 1152/251 mit IFSS-Schnittstelle. Steuerzeichen für den Treiber: CR, LF, FF

BLIST benötigt noch Activates, LIT, LIST.
Beachten: - Ausgabe ist während BLIST auf Treiber LIT geschaltet!
- Ausgabe nur Y für Programme im Speicher per Name

1.4. Speichern und Laden der globalen Vereinbarungen

Der globale Vereinbarungsteil befindet sich in der Datei "G" (Kennbyte: ØG)

Aufzeichnen: WRITE Programmname, Ø1, 5626, 99CC

Lesen: READ Programmname, H

1.5. Dateien für Basic

Editor: BEDITØ mit BEDIT, G, LLT, BLIST
Adressen: 4ØØØ bis 5FFF

Interpreter: BINTERH mit BINTERF, ACTIVATE, SOUND
und RAND
Adressen: CØØØ bis E352

BINTERF mit BINTER, Reset, Directory und ACTIVATE
Adressen: CØØØ bis DC69

Gesamt-Basic: BASICØ mit BEDITF, BINTERH
Adressen: 4ØØØ bis E352

Grafikergänzung: GRAB
Adressen: 9ØØØ bis BFFF

Basic im kleinen Segment!

2. Objektcodeeditor

* Alle Programme arbeiten wie OCED nur im 1 Segment

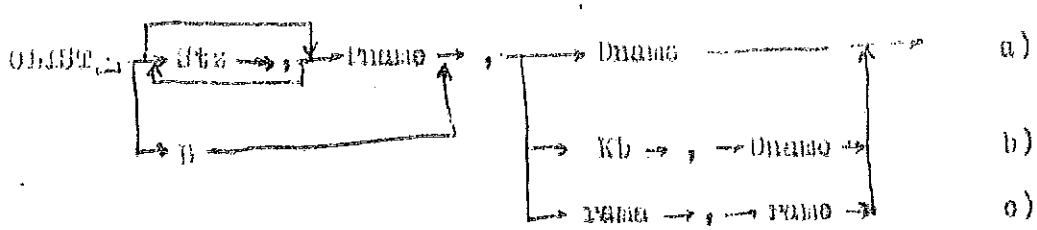
Beachten: alle Programme arbeiten nur im RAM-Segment von OBDIT

2.1. Veränderungen in OBDIT

Wegen der Einheitlichkeit der Darstellung wird das Register r immer als (HL) dargestellt, M kann aber zur Eingabeverkürzung benutzt werden.

2.2. Ausgabemöglichkeit für Objektcodeeditor

2.2.1. Bedienkommando



obere Zweig, untere Zweig, Stz, B, Pname: siehe 1.3.1.

a): Listen von Programmen, die mit OBDIT erarbeitet wurden und Markenbereich vorhanden ist. Ausgegeben werden Lfd. Adresse, Marken, Befehle, Kommentare.

b): Listen von Maschinenprogrammen zwecks Analyse
Kb.: Kennbyte mit 2 Ziffern
Dname: Dateiname
Ausgegeben werden lfd. Adresse, die Hex-Codierung, die ASCI-Übersetzung der Codes und die Befehle

c): wie b) ramA: Anfangsadresse
ramE: Endadresse

2.2.2. Kommentarsteuerung
siehe 1.3.3.
oN: ohne Kommentar bis nächste Marke

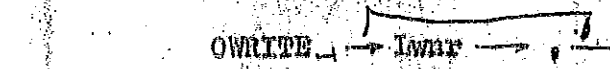
2.2.3. Anpassung
siehe 1.3.4.

Anmerkung: Wird zur Analyse von Betriebssystemprogrammen vorher die Datei "EGOS.MARKEN" eingelesen, so werden alle Zugriffe auf die Software-Schnittstellen mit dem Namen gekennzeichnet (siehe Systembeschreibung)

Beachten: Es können nur Programme im Segment des Objektcode-editors gelistet werden.

2.3. Speichern von Programmen auf Kassette

Bedienkommando:



Pname: Name, der ins Directory eingetragen werden soll
Dname: Name, der mit ODEF festgelegt wurde

Leistung:

Die Dateien Programm (Ø1) und Markentabelle (12) von Dname werden unabhängig davon, wie sie im RAM angeordnet sind, so aufgezeichnet:

13 "E" Ø1 Dname 12 Dname 13 "F" mit ØØØh-Byte frei

Anwendung:

- Herauslösen einer Datei aus Assemblerbereich mit mehreren Dateien

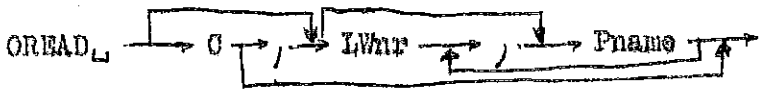
- Aufzeichnen eines z. B. 2. Assemblerbereiches; da WRITE␣ Pname , A nur 1. Bereich aufgezeichnet.

Fehler?
41: Datei zu nahe an Directory
44: Syntaxfehler
C7: Dname Ø1 oder Ø12 nicht vorhanden
sonst wie WRITE

Beachte: Programmablauf niemals mit Reset unterbrechen!

2.4. Einlesen von Programmen mit Erhalten der RAM-Kettung

Bedienkommando:



Pname: Dateiname im Direktory (nur für Assemblerbereiche

Leistung: Einlesen aller angegebenen Programme;

jedes Programm wird in den freien RAM eingelesen und der RAM vom Programmende bis zum Directory (Ø4) gekettet.

Das Kommando C bewirkt das Löschen des RAM's ab Anfangsadresse und Anlegen der Directory-Datei auf RAM-Adresse >= Endadresse

Anpassungen:
Anfangsadresse: Adr. Kennbyte Ø1 + 11 Byte (dez); hier 73ØØh
Endadresse: Adr. Kennbyte Ø1 + 13 Byte (dez); hier CØØØh

Anwendung:

- nach Einlesen des ODEDP : OREAD
  damit Kettung des RAM; RKET fällt weg

- Einlesen mehrerer Assemblerbereiche ohne RKET

Fehler:

4A kein RAM für Anlegen der Directory-Datei
sonst wie READ

Beachten: Programme, die hinter einem freien RAM-Gebiet angeordnet sind, sollen erst ab Endadresse stehen, sonst werden sie mit gelöscht.

2.5. Binden von Assemblerprogrammen

Bedienkommando:

Dname: Name, der mit ODEF festgelegt wurde
Adresse: die Adresse des Kennbytes Ø1 der Datei Dname, die das Programm lauffähig haben soll

Leistung/Anwendung
Das Binden mit OEDIT benötigt das Berechnen der Zuweisungsadresse, was erhöhte Fehleranfälligkeit bedeutet. BIND vermeidet diese Fehler.
Fehler: C7: Ø1- oder 12-Datei nicht gefunden
Ø7,Ø8: Bindefehler, zurück zu OEDIT

Erstellen von relativ ladbaren Programmen

Bedienkommando:

VERA␣→ Dname →

Dname: Name, der mit ODEF festgelegt wurde

Leistung/Anwendung

Die Markentabelle wird in eine Verschiebeadressentabelle (1Ø) umgewandelt; Das Programm kann anschließend mit WRITE Pname, A gespeichert werden. Nach dem Wiedereinlesen in freien RAM wird das Programm mit RELAD Dname lauffähig gebunden.

Fehler: ~~wie bei~~ Ø1 : 1-Byte-Zugriff auf Marke nicht gesetzt ~~oder EQU-Marke~~
Ø7,Ø8: Bindefehler (lokale)
ØØ: kein freier RAM

Beachten: Es wird freier RAM von etwa 1/4 der Länge der Markentabelle benötigt.

Verknüpfen zweier Assemblerprogramme

Bedienkommando:

LINK␣→ HDname → , → NDname →

HDname Hauptprogramm
NDname Nebenprogramm, im gleichen oder anderen Assemblerbereich
beide Namen, die mit ODEF festgelegt worden.
Im RAM muß HDname vor NDname stehen!

Leistung

Das Nebenprogramm wird an das Hauptprogramm gekoppelt und die Markentabelle so erweitert, daß das neue Hauptprogramm mit ODBUT weiter bearbeitet werden kann.

Dabei werden

- alle nichtzugewiesenen Marken des Nebenprogrammes gelöscht

- alle Markennamen des Haupt- und Nebenprogrammes auf Namensgleichheit getestet (EQU)
  a) 2 gleiche, mit EQU zugewiesene Marken
  es wird der Wert vom Hauptprogramm übernommen
  b) 2 gleiche Marken, wovon eine lokal und die andere mit EQU auf einen beliebigen von Ø verschiedenen Wert zugewiesen ist,
  es wird der lokal zugewiesene Wert übernommen
  c) 2 gleiche lokal zugewiesene Marken
  das 3 Zeichen des Markennamens vom Nebenprogramm wird mit "$" markiert

Das Nebenprogramm ist nach dem Linken als Datei nicht mehr im RAM vorhanden.

Anwendung:

Aufbau einer Assemblerprogrammbibliothek, die das Erstellen von Maschinenprogrammen wesentlich erleichtert, da viele Unterprogramme aus der Bibliothek entnommen werden können (mit OREAD) und an das Hauptprogramm gekoppelt werden können.

Fehler:

07 Datei nicht gefunden
44 NDname liegt vor HDname oder ist identisch

Beachten: Es sollen nur getestete Programme verwendet werden!

2.8. Dateien für OCED

OCEDP mit OCMD, OLIST, OWRITE, OREAD,
BIND, VERA, LINK, RAMP,
Reset, ACTIVATE, SD1152

Adressen von 4000 bis 7300
EGOS, MARKEN relativ ladbar

3. Prozedur zur Aktivierung, Deaktivierung und Initialisierung von Treiberprogrammen

3.1. ACTIVATE
von Bedienkommandoebene:

ACTIVATE␣ → Log. Ger. Nr. → , → Treibername →

Log. Ger. Nr.: 1 bis 2 Zeichen Hexadezimal

von Basic:

ACTIVATE␣→ Textausdruck → ; Syntax wie oben

Funktionen:

- ist Log-Gerät Nr. = Ø, dann wird der Treiber mit Request ØØh initialisiert (Treiber: Kennbyte: Ø2)

- ist Log.Ger. Nr. >= 1 und <= 14, wird dieser Treiber mit dieser Nr. in die ADT eingetragen und aktiviert

- war dieser Treiber ~~mit gleicher Nr. bzw. mit Ø~~ in der ADT, so wird er auf die Log.Ger.Nr. aktiviert

- andere Treiber mit der gleichen Nr. werden mit Ø deaktiviert

- Fehlermeldungen können vom Basic aus über FNM$ geholen werden

Hinweis: Der Text sollte ins aufeinanderfolgenden Bereich oder in gleichen Segment wie ACTIVATE stehen, sonst BASIC sind Kommandoebene gewährleistet.

- Fehler: C 7: Treiber nicht vorhanden; 6 D: ADT ist voll; bzw. bei Initialisierung Code von IY + ØAh; bzw. 44: Syntaxfehler

- ACTIVATE ersetzt die Bedienkommandos ACT und DEACT und DEFINE des Betriebssystems und ist von Basic und Maschinenprogrammen aufrufbar

- Aufruf von Maschinenprogrammen aus:

  INP: DE zeigt auf Text
  Textabschluß: ØF F h

  z. B. Initialisieren Druckerschnittstelle und Drucken

Die Prozedur ist auf beliebigen Adressen lauffähig

Beachten: ACTIVATE arbeitet nur, wenn der Bereich der ADT nicht schreibgeschützt ist.

3.2. Reset-Routine

Es wird bei Reset die ADT wieder neu initialisiert, so daß ein Ausschalten des Gerätes nach Absturz vermieden werden kann.

3.3. Datei:

ACTIVATE.V mit ACTIVATE und Reset
wird in freies RAM geladen und ist sofort lauffähig

4. <u>Druckertreiber</u>

Für Drucker SD 1152 Typ 251 mit IFSS-Schnittstelle

4.1. Eintritt
Eintritt über Softwareschnittstelle NIXE

- Request: ØØh: Initialisieren
  ØEh: Ausgabe

- INP: IY+1: Request
  IY+2/3: Pufferadresse
  IY+4: Datenlänge (max. 255 Zeichen)

- OUP: IY+4: Anzahl übertragene Zeichen
  IY+1Ø: Fehlercode

- ist letztes Zeichen im Puffer CR, dann wird vom Treiber noch LF gesendet

- Ausgabeende:
  . alle Zeichen gesendet
  . nicht zulässige Zeichen: vom Betriebssystem verwendet: 19h (EM); Bit 7=1 (z. B. ØFFh)

  Zulässige Steuerzeichen: NuL, ESC, BS, FF, LF, CR, HT, FBU, DEL

* Hinweis für alle Treiber: Da über den RW-Vektor (IY) [illegible] nicht die Segm.-Nr. übermittelt wird, [illegible] Text im [illegible] (> COOOH) bzw. im Segm. des Treibers stehen [illegible]

4.2. Fehlermeldungen:

01	falsche Request
30	Drucker arbeitet nicht (nach Initialisierung)
31	falscher Druckertyp (nach Initialisierung); Übertragungsfehler
32	Papierende, Farbbandende
33	Havarie
34	Operationsfehler

4.3. Anpassung an SD 1152 Typ 25 2

Druckertyp: Byte auf Adresse Kennbyte + 13 (dez) auf 32h ändern

Tabelle der zulässigen Steuerzeichen um zusätzliche Steuerzeichen erweitern:
Tabelle ab Adressekennbyte + 14 (dez), 12 Byte lang
2 8 LF2, FF2

4.4. Laden des Treibers:

READ SD1152S.V
RELAD SD1152S

mit RAM-Programm Kennbyte 01 zu 02 ändern. Der Treiber benutzt den IFSS-Anschluß des MC 80.

4.5. Datei

SD1152S.V relativ ladbar

4.6. Anwendungsbeispiel

Von Console aus:
```
ACTIVATE 0, SD1152S,3,SD1152S
CATA
ACTIVATE3,CON
```
Druckt den Katalog aus

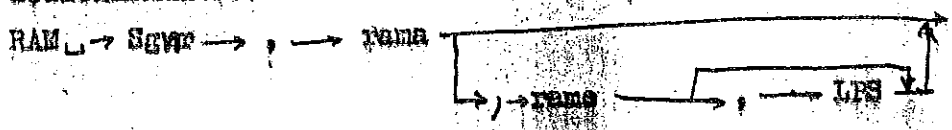
von Basic:
```
ACTIVATE'0,SD1152S,3,SD1152S'
PRINT'Drucker'
ACTIVATE'3,CON'
```

5. Neue RAM-Programm

Vorteile:
- Kursor läuft immer auf der aktuellen Position auch im ASCII-Modus und während der Eingabe.
- TRAN arbeitet auch zwischen verschiedenen Segmenten
- konkrete Syntaxprüfung verhindert Abstürze

5.1. RAM

Bedienkommando:



Sgnr: Segmentnummer
rama: RAM-Anfangsadresse
rame: RAM-Endadresse
LPS: Zeilen pro Seite

5.1.1. RAM-Anzeige (obere Zweig)

Anzeige: . 24 Zeilen Hex und ASCII
. laufende Adresse inverse gedruckt
. Kursor: aktuelle Byte bzw. Zeichen inverse

Steuertasten: ("c" = Control)

←→↑↓	Kursorbewegung
c↑ c↓	Seitensteuerung
cA	ASCII-Modus
cB	Byte-Modus
cE	Eingabe-Modus
cOFF	Verlassen RAM

ASCII

Eingabe-Modus:

Wird angezeigt durch "e" neben laufender Adresse
Tasten:

←→	Kursorbewegung
0...9,A..F	im Byte-Modus
Space bis Tilde	im ASCII-Modus
ENTER	Eingabebestätigung
OFF	Verlassen der Eingabe ohne Bestätigung
cOFF	Verlassen RAM ohne Bestätigung

Es können innerhalb einer Zeile beliebig viele Bytes geändert werden. In den RAM werden sie nur durch ENTER übernommen.

5.1.2. RAM-Ausgabe (untere Zweig)

Ausgabe des RAM-Inhaltes zum log.Gerät Nr. 3 von rama bis rame mit LPS Zeilen pro Seite. Wird LPS nicht eingegeben, dann 64 Zeilen
Stop der Ausgabe: OFF

Weiter: ENTER

Anwendung: Ausdrucken von größeren RAM-Bereichen

5.1.3. Fehler:

44: Syntaxfehler
nicht zulässige Tasten: Beep

5.2. Füllen eines RAM-Bereiches

Bedienkommando:

FILL␣ → Sgnr → , → rama → , → rame → , → Byte →

Wirkung: Das Byte wird an allen Adressen von rama bis einschließlich rame in Segment Sgnr eingetragen

Fehler: 44: Syntaxfehler

5.3. Kopieren eines RAM-Bereiches in einem anderen Bereich

Bedienkommando:

TRAN␣ → Sgnr1 → , → rama1 → , → rame1 → , → Sgnr2 → , → rama2 →

Kopieren von rama 1 bis einschließlich rame 1 im Segment Sgnr 1 nach rame 2 im Segment Sgnr 2

TRAD␣ → Kb → , → Dname → , → Sgnr 2 → , → rama 2 →

Kopieren der Datei Dname mit Kennbyte kb nach rama 2 im Segment Sgnr 2

Fehler: 44: Syntax
67: Datei nicht vorhanden

5.4. Datei

RAMP.V relativ ladbares Programm, mit RELAD RAM binden

6. Veränderten Tastenfeld

6.1. Begründung

Das Hauptproblem der vom VEB Elektronik Gera gelieferten Tastatur ist das Fehlen von "." im Numerikfeld. Für die Eingabe von Zahlen, die normalerweise durch Basisprozeduren erfolgt, ist dieses Zeichen aber unbedingt notwendig.
Das Operationszeichen "*", "/" können gar nicht genutzt werden, da kein Formelinterpreter vorhanden ist.
Der Austausch von Z und Y (amerikanische Tastatur) ist hinderlich beim Übergang von Schreibmaschine oder MC 80/22 zum MC 80/30.

6.2. Aufbau des veränderten Tastenfeldes

- Z und Y entsprechen deutschen Tastaturen

- im Numerikfeld sind alle Zeichen für eine schnelle Eingabe von Gleitkomma- und Hexadezimalzahlen und ENTER vorhanden.

A	B	C	D
7	8	9	E
4	5	6	F
1	2	3	,
Ø	.	-	Enter

- Soll die amerikanische Version erhalten bleiben, sind die Codes für Z und Y der EPROM-Adressen Ø329 und Ø32A, Ø369 und Ø36A auszutauschen

6.3. Durchführung der Änderung

Die Binärdatei "TASTATUR" wird in einem freien RAM eingelesen und ein EPROM 2716 wird im Adressbereich ØØØØ bis 3FFF ab RAM-Adresse; Adresse Kennbyte "Ø6" +12d beschrieben und mit dem EPROM in der Tastatur ausgetauscht.
Die Tasten sind auszutauschen bzw. neu zu beschriften.

6.4. Datei

TASTATUR relativ ladbar

7. Kontaktadresse

Bei inhaltlichen Fragen bitte telefonisch Gera 395 App. 2680
Kollege Preiß

Anlage zu: Nutzungshinweise

Beispiele

1. PRINT, INPUT, ACTIVATE in Basic

1.1. Initialisierung eines Treiberprogrammes (Kennbyte Ø2)
Treibername steht in Tr$ (Ø,1Ø); z. B.: Tr$ (Ø,1Ø)='SD1152S';

```
ACTIVATE 'Ø,'Tr$(Ø,1Ø)   ; Initialisieren mit Log.Gr.Nr.Ø
LET F$ (Ø,2) = FME$      ; 2 Zeichen Fehlermeldung lesen
IF F$ (Ø,2) <> 'ØØ' DO   ; wenn Fehler, dann reagieren
   IF F$ (Ø,2)='Ø7' DO   ; Treiber nicht gefunden
      PRINT 'Treiber'Tr$(Ø,1Ø)' nicht vorhanden!'BEL
   DOEND
   .                     ; weitere Fehlerreaktionen
   .
   .
   .
DOEND
```

1.2. Ausgaben zum Drucker

```
Treiber schon initialisiert; das Integerfeld Tb enthält 10 Ta-
bulatorpositionen
ACTIVATE '3,SD1152S'      ; Drucker als log. Gerät Nr. Ø
PRINT DEL;                ; Löschen des Druckers
PRINT ESC'[ 144 }';       ; setzen Drucker auf 72 Zeilen
                          ; pro Format
                          ; setzen
FOR I = Ø TO 9            ; setzen der 10 Tabulator-
                          ; positionen
    PRINT ESC'[ 'FORMAT!3,ØTb(I) ' ' ESC'H';
NEXT I
PRINT 'Drucker' ;         ; Ausdrucken ohne Zeilenwechsel
PRINT 'bereit!'           ; Ausdrucken mit Zeilenwechsel
A TIVATE '3 ,CON'         ; Drucker wieder inaktiv
```

1.3. Lesen eines Lochstreifens mit (angenommenen) Treiber PAT

```
ACTIVATE '0,PAT,3,PAT'          ; Treiber initialisieren und aktivieren   /
IF FMES$<>'80' DO               ; Reaktion auf Fehlermeldung
    .
    .
    .
DoEND
INPUT (L$(0,N)  L$(0,N)         ; bei der Notation INPUT (Text) Var
    .                             wird dem Treiber in IY+4/5 die
    .                             Länge des Textes und in IY + 2/3
    .                             die Adresse des Textpuffers übergeben.
    .                             Der Treiber soll jetzt N Zeichen
                                  in diesen Puffer einlesen. N muß
                                  wieder in 14+2/3 eingetragen werden.   IY /
INPUT P$ (0,M)                  ; bei dieser Notation ohne Text werden
    .                             in IY+4/5 70 als Zeichenanzahl und in
    .                             IY+2/3 die Adresse eines leeren Puffers
    .                             übergeben. Von den gelesenen Zeichen
                                  werden aber nur M nach P$ gelesen.
IF FMES$<>'80' DO               ; Reaktion auf Fehler
    .
    .
    .
DOEND
ACTIVATE '3,CON'                ; immer als Abschluß durchführen
```

2. KINIT, WRITE, READ, LEN, HEX, RDIR in Basic

Speichern und Lesen von Daten in der Kassette
Die Daten sollen sich in einem reservierten Globalbereich, der z. B. im RAM als Binärdatei mit Kennbyte 06 an der Adresse (dezimal) Adrkb und der Länge bis zum nächsten Kennbyte Dlen. Diese Datei wird vor Programmierbeginn mit dem RAM-Programm und RKET angelegt. Die Datei muß sich im gleichen RAM-Segment befinden wie die Basicprozeduren, die auf diese Daten zugreifen!

2.1. Globale Definition der Datenvariablen

von Kommandoebene: VEDIT; ENTER; OK

```
GLOBAL % Hexadr                 ; Hexadr. ist umgerech    = Adrkb + Nal;
                                ; Nal: Anzahl Byte für den Namen der
                                ; Binärdatei im RAM + 4 Byte
```

```
REAL     ....          ; Vereinbarung der Variablen;
STRING   ...           ; Anzahl Byte für alle Var.  Dlen-Mal
```

2.2. Initialisieren einer neuen Datenkassette

```
a) KINIT  FORMAT ! 1.Ø; Lwnr  ; wenn Doppellaufwerk: Lwnr = 1;2

b) KINIT  '   '               ; MC 80/3Ø; mit leeren Text um das
                                Retten aller Register zu garan-
                                tieren und FME$ zu erhalten
   IF FME$< 'C2' DO           ; reagieren auf Fehler
      PRINT 'Gerät ist nicht bereit!' BEL
   DOEND
   WRITE 'Daten.'Dat$(Ø,6)',Ø1,ØØØØ, ØØFF'
                              ; markieren der Kassette mit einer
                                leeren Datei
                                Im Directory steht jetzt der
                                Kassettenname mit dem Datum aus Dat$
```

2.3. Speichern der Datei auf Band, Dateiname soll in Nad$ stehen, max. 32 Zeichen

```
WRITE Nad$(Ø,LEN(Nad$(Ø,32)))',Ø1,'HEX (Adrkb)','HEX (adrkb+Dlen)
                              ; die Syntax in WRITE entspricht WRITE
                                Pname, Sgnr, zana, rane
```

Beachten: wird, wie bei Maschinenprozeduren und FORMATV, ein Text einer bestimmten Syntax benötigt, so ist dieser in der Testphase mit PRINT '*' Text '*' auf dem Bildschirm zu kontrollieren.

Die häufigsten Fehler sind

- Leerzeichen mitten im Text sind bei WRITE, READ, Textendekennzeichen. Deshalb hier Länge des Namens mit LEN ermitteln, damit keine Leerzeichen
- falsch formatierte Zahlen haben z. B. Leerzeichen vor der Zahl oder zu viele Ziffern

2.4. Einlesen der Daten vom Band

```
READ 'A'                      ; Es wird nur Directory eingelesen
IF FME$ < > 'D4' DO           ; D4 ist korrekte Fehlermeldung
     .
     .
     .
DOEND
LET N$(Ø,32) = RDIR (Ø)       ; Name der Kassette kontrollieren
IF N$ (Ø,6)< > 'Daten.'  DO    ;
     .
     .
     .                        ; Reaktion, wenn falsche Kassette
DOEND
```

```
LET DM = 1                          ; Ausgabe der Datendateinamen
                                      als Menü zum Aussuchen

BEGIN LOOP
  LET N$(Ø,32) = RDIR (DM)
                                    ; Lesen der Dateinamen
IF N$(Ø)<>'    ' DO                 ; bis Ende des Directory
  IF N$(Ø,3) = 'Te.' DO             ; z. B.: nur die Datendateien ausgeben,
                                      die mit 'Te.' beginnen
     PRINT FORMAT !2.Ø! DM, N$ (Ø,32)
  DOEND
  LET DM = DM + 1                   ; Zählen der Dateien
LOOP
PRINT
PRINT 'Eingabe der Nummer der gesuchten Datei'
BEGIN LOOP
  INPUT ('ØØ') N                    ; Eingabe 2 Ziffern
  LET N$(Ø,32) = RDIR (N)           ;
IF KINP$<> ENTER OR N$(Ø,3)<>'Te.' DO
LOOP                                ; Schleifen nur beenden, wenn Taste ENTER
                                      und korrekte Dateinummer

READ N$ (Ø,LEN (N$(Ø,32)))',R'      ; Syntax entspricht READ Pname, R
```

Beachten:
Es muß softwaremäßig abgesichert sein, daß nur die dem Programm zugehörigen Dateien gelesen werden, da sonst bei fälschlichem Lesen die RAM-Kettung oder Programme zerstört werden!

→ in Verbindung: Anweisungen die das Schreiben und Lesen von Datenfiles im RAM ohne Verkettung globaler Variablen ermöglicht (07/87)

3. EXC

Sortieren einer Namenstabelle nach dem Alphabet
Die Namen stehen im Stringfeld Na$, jeder Name mit maximal L Zeichen, insgesamt N Namen; b sei Booleanvariable

```
BEGIN LOOP                          ; Sortierschleife
   LET b = Ø                        ; b = false
   FOR I = Ø TO N-2                 ; Schleife mit Test aller Nachbarnamen
      IF Na$(L*I,L) > Na$(L*I+L,L) DO
         EXC (Na$(L*I,L), Na$(L*I+L,L))
                                    ; wenn Vorgänger- hinter Nachfolgernamen
                                      gehört, dann austauschen
         LET b = 1                  ; und b = tru
      DOEND
   NEXT I
IF b DO                             ; Sortierschleife so lange fortsetzen, bis
LOOP                                  b = false bleibt
```

4. FORMATV

```
a) LET V = 32/10            ; z. B
   FORMATV '!'FORMAT! 1.1 V '!'
                            ; erzeugt FORMAT !3.2!

   LET Z$ (Ø,1Ø) = 1ØØØ * PI;
                            ; Zahl formatiert in String lesen

b) LET F$(Ø,5) = '12.1!'    ; Formatstring löschen
   BEGIN LOOP
      INPUT (F$(Ø,5)) F$ (Ø,5)
                            ; Eingabe Testformat

      FORMATV F$(Ø,5)       ; Format setzen

      PRINT F$(Ø,5),PI/1ØØ,,PI,,PI * 1ØØØ
                            ; Beispiele ausgeben

      IF A=A DO             ; Leere Bedingung, um
   LOOP                     ; in der Schleife zu bleiben
```

5. CASE OF
Reaktion in Abhängigkeit des Wertebereiches zweier Zahlen A u. B

```
CASE OF, A<Ø AND  B<Ø, A >=Ø AND  B<Ø , A<Ø  AND  B >= Ø

GOSUB 1ØØ                   ; Fall A < Ø  AND  B <Ø
GOSUB 2ØØ                   ; Fall A >=Ø  AND  B <Ø
GOSUB 3ØØ                   ; Fall A < Ø  AND  B >=Ø

GOTO ...                    ; Ende der Gosub-Liste; Rücksprung aus den
                              Subroutinen, bzw. Fall A >=ØAND  B >= Ø
```

6. ACTIVATE von Maschinenprogrammen aus, Drucken

```
DINI LD DE, PDRU            ; Initialisieren und Aktivieren des Druckers
RC   CALL ACTI              ; Return mit CY = 1 : Fehler
     RC
     .
     .
     .
DRUP LD HL, DBUF            ; Drucken des Druckpuffers über Log. Gerät Nr. 3
     CALL LOUT
     .
     .
     .
ENDE LD DE, PCON            ; vor Abschluß wieder CON als Log.-Gerät Nr. 3
```

```
ACTI PUSH DE            ; UP: Übergabe des Puffers (DE) an
     LD DE,PACT           ACTIVATE
     LD A,1
     CALL RSU           ; suchen von ACTIVATE
     POP DE
     RC                 ; nicht gefunden
     INC HL             ; HL  Programmbeginn von ACTIV.
     CALL CHD           ; Softwareschnittstelle:
                        ; führt aus: CALL (HL) zu Segm. (SEGD)
                        ; SEGD wird von RSU geladen
     RET                ; RET mit evt. Fehlermeldungen

PDRU DM '0,SD1152S,'
     DM '3,SD1152S'
PCON DM '3,CON'
PACT DM 'ACTIVATE'
```

7. Anmerkung zu Testmodus in OEDIT
Da in einem Teil der Bedienungsanleitungen falsche Informationen gegeben wurden

Funktionstasten:

cT:	Einzelschritt
cN:	Schleifenabarbeitung
cL:	Lauf
c\:	Befehl übergehen
cR:	eingegebene Registerwerte werden übernommen
cOFF:	Verlassen Beachten: wurde mit cT in ein Programm außerhalb des eigenen gesprungen, so ist nicht mit cOFF, sondern mit Reset zu verlassen, auch wenn das eigene Programm wieder erreicht wurde!
Break:	Steuertaste unten links; wirkt im Basic und im Testlauf als Abbruch

8. Anmerkung zu VERA

Ist ein Programm schon an sich verschiebbar, wird keine Verschiebeadressentabelle angelegt, das Programm nur als Assemblerbereich markiert. RELAD ist nicht notwendig. z. B. ACTIVATE.V

Beachten: Beim Magnetbandschreiben werden Blöcke zu 256 Byte aufgezeichnet. Deshalb nach Vera die 00-Datei hinter 13'F' so ketten, daß ab 13'8' volle Blöcke erreicht werden, sonst werden zu lange Kettungen mit gespeichert.

9. Beispiel für das Linken

Anlegen der Bereiche: ONEW 3ØØ
ODEF .HAUPT
ODEF NEBEN

Eingeben Hauptprogramm: OEDIT HAUPT

```
EXT  EQU  Ø1234H          ; konstanter Wert
NEB  EQU  1               ; Aufruf Nebenprogramm,
                            Wert beliebig > Ø
HAUP LD HL,EXT            ; Programmbeispiel
GLEI CALL NEB
     JR HAUP
```

Eingeben Nebenprogramm

```
  NULL EQU Ø              ; nicht zugewiesene Marke
  EXT  EQU ØABCD H        ; gleiche Name wie in HAUPT
  HAUP EQU ØFFFF H        ; Aufruf Hauptprogramm

  NEB LD HL,EXT           ; Programmbeispiel
  GLEI CALL HAUP          ; gleiche lokale Markenname
       JR NEB
```

Linken durchführen: LINK HAUPT, NEBEN

OEDIT HAUP zeigt an:

```
EXT EQU Ø1234H            Anzeige mit oG; ENTER
                          ; nur noch eine Konstante
HAUP LD HL,EXT            ; Hauptprogrammteil
GLEI CALL NEB
     JR HAUP

NEB LD HL,EXT             ; Nebenprogrammteil
GLEI CALL HAUP            ; gleiche lokale Marke verändert
     JR NEB
```

OEDIT NEBEN zeigt an: ERROR C7;
Nebenprogramm-Datei existiert nicht mehr.

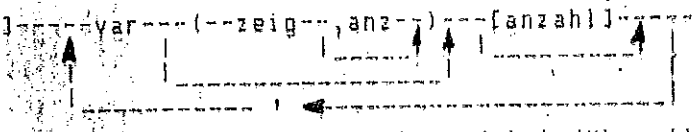
Kurzinformationen zu Kassette "Syst.sw.EKD.Ger[redacted].20.5.87"

1, Erweiterungen gegenueber der Version 25.3.87

1,1 neue Basicbefehle

- HEX$ ist gleich HEX
- HEX# Umkehrung zu HEX$; wandelt eine Hexadezimalzahl in einem Textausdruck in eine Realzahl.

- DATOUT, DATIN: Transport von Daten vom Basic zu einer Binaerdatei und umgekehrt.
  Syntax: DAT.. text-----[zeiger]---var---(--zeig--,anz--)---[anzahl]-----



  text : der Textausdruck muss den Dateinamen einer Binaerdatei (Kb,: 6) ergeben.
  zeiger : arithmetischer Integerausdruck im Bereich 0 .., 32000; er zeigt auf die Datenzeile in der Datei.
  var : Variablenname vom Typ Real, Integer, String; keine Globalvariable in der Variablenliste duerfen unterschiedliche Typen vertreten sein.
  zeig : Zeiger in einem Feld
  anz : Anzahl Zeichen bei Strings
  anzahl : arithmetischer Integerausdruck >=1; damit werden die Anzahl der Variablenwerte aus dem Feld der Variable var festgelegt. Wenn nicht angegeben, dann gilt 1.
  Fehlermeldungen, die nach der Aktion mit FME$ gelesen werden kann:
  80: Operation ist gueltig
  C7: Datei nicht vorhanden
  CB: Datei ist ohne Daten
  CD: Datei ist zu Ende
  D2: Zeiger ist groesser als Anzahl der enthaltenen Datenzeilen
  B3: die Variablenliste ist laenger als eine Datenzeile (Anzahl der Bytes)

Eine Datei wird mit DATCLR text (text: Dateiname) geloescht. Die Datei wird mit dem ersten Beschreiben mittels DATOUT und zeiger=0 initialisiert. Damit wird in der Datei die Anzahl Bytes einer Datenzeile und die Anzahl der enthaltenen Datenzeilen gespeichert. Beim naechsten Beschreiben muss der zeiger immer <=(enthalt. Datenzeilen +1) sein (sonst Fehler D2) und die Anzahl Bytes in der Variablenliste <= der Anzahl beim ersten Beschreiben.

Beim Lesen der Datei mittels DATIN muss der Zeiger immer kleiner gleich der Anzahl der enthaltenen Datenzeilen sein, sonst Fehler D2. Damit die Variablen der Liste nach DATIN sinnvolle Werte enthalten, muss die Variablenliste beim Beschreiben und beim Lesen uebereinstimmen. Beim Lesen duerfen weniger Variablen angegeben werden als beim Beschreiben.

Hinweis zu den Interpretern BINTERR, BINTERP, BASICP: wenn nicht benoetigt, koennen alle Dateien nach RUN geloescht werden. An Adresse C1BC steht der hoeherwertige Teil der Adresse der untersten Stackgrenze. Diese Adresse soll mindestens 200H Bytes groesser sein als das Ende der letzten Datei vom Interpreter.

1,2 Kommandos zur Dateiarbeit

Dateiname auf Kassette: KILL.V; nach Laden: RELAD KILL, die Datei 10H,KILL kann dann geloescht werden.

KILL ---- Kb, dname ----

Loeschen der angegebenen Datei, Leere Bereiche werden vernichtet und freier RAM vereinigt. Fehler: 48: RAM-Kettung

CREATE --be,--Kb,--dname,laenge--,sg#--

be : Bereichkennung A oder B (Assembler, Basic)
Kb : Kennbyte hex (1 bis 1F)
laenge: Laenge der Datei in 4 Zeichen hex
sg# : datei muss in dieses Segment
Erzeugen einer Datei im freien RAM mit angegebenen Kennbyte und Dateiname; wenn gefordert eingrenzen als Bereich; die Laenge wird auf volle 100H Bloecke aufgerundet. Ohne dname: nur Kb 1.
Fehler: 44: Syntax; 48: RAM-Kettung; 01: kein freier RAM; D0: Datei schon da

BWRITE --lw#,--pname,be,Kb,dname ---- (Bereich-Schreiben)

lw# : Laufwerknr.
pname: Name der Datei in dem Directory der Kassette
Schreiben des mit be angegebenen Bereiches auf Band, der die Datei Kb,dname enthaelt. BWRITE ist fuer das gezielte Schreiben notwendig, da WRITE pname,be nur den ersten Bereich schreibt.

Fehler: 44: Syntax; C7: Datei nicht vorhanden; Fehlermeldungen von WRITE

BREAD --lw#,--pname (Bereich-Lesen)

Lesen einer Datei von Kassette in freien RAN mit anschliessender Korrektur der RAM-Kettung. Vorzugsweise fuer Assembler- oder Basicbereiche. Ist notwendig, da READ die RAM-Kettung zerstoert.
Fehler: 48: RAM-Kettung kann nicht korrigiert werden; Fehler von READ

2. Beispiel
Eine APSK-Datei soll erstellt und archiviert werden. Mit dem Basic muss KILL.V gelesen werden.

```
        DEF APSKOUT
        REAL I,T(3)                          ;I; Zeiger, T; Feld mit 3 Arbeitszeiten
        INTEGER NR                           ;Nr. des Arbeitsganges
        STRING AG$(30)                       ;Bezeichnung des Arbeitsganges
10      KILL '6,APSK'                        ;Loeschen der evt. vorh. Datei
        CREATE 'A,6,APSK,'HEX$(1000)         ;Datei "APSK" mit 1000 Bytes als Ass.b,
        IF FME$='01' DO                      ;Fehlerbehandlung "kein freier RAM"
          ....
        DOEND
20      DATCLR 'APSK'                        ;nach CREATE nicht notwendig
        I=0                                  ;Zeiger setzen
        BEGIN LOOP                           ;Eingabeschleife
        .  Eingabe(NR,T,AG$)                 ;Eingabe der Daten in einer Prozedur
        .  DATOUT 'APSK'[I]NR,T[3],AG$(0,30) ;Schreiben der Zeile in die Datei
        .IF FME$='80' DO                     ;Eingabe bis Dateiende
        .  LET I=I+1                         ;Zeiger incrementieren
        LOOP
30      BWRITE 'APSK.Kondensator,1,A,6,APSK' ;schreiben der Datei auf Band,
                                             ;vorher musste Kassettenname getestet
                                             ;werden und Bedienerfuehrung.
          ......

        DEF Eingabe(%Nr,#Zeit,$Bez$)         ;Parameteruebergabe fuer die
                                             ;Eingabeprozedur.
          ......

        DEF APSKIN                           ;Prozedur zur Ausgabe der Daten
        REAL J,Z1,Z2,Z3                      ;J:Zeiger, Z1,Z2,Z3: Arbeitszeiten
        STRING AGN$(30)                      ;Arbeitsgangname
        INTEGER AGNR                         ;Arbeitsgangnr.
10      KILL '6,APSK'                        ;Loeschen evt. vorh. Datei
        BREAD 'APSK.Kondensator.1'           ;Lesen der Datei von Band; besser
                                             ;Auswahl der Datei ueber Menue
20      IF FME$='80' DO                      ;Ausgabe nur, wenn Datei gelesen
        .  CLEAR
        .  PRINT 'APSK 'Kondensatoren'
        .  PRINT
        .  PRINT 'AG-Nr','Zeit 1','Zeit 2','Zeit 3','Arbeitsgang'
        .  PRINT
        .  LET J=0
30      .  BEGIN LOOP                        ;Ausgabeschleife
        .  .  DATIN 'APSK'[J]AGNR,Z1,Z2,Z3,AGN$(0,20) ;Lesen der Datei mit
                                             ;identischem Variablenaufbau
        .  .IF FME$='80' DO                  ;Ausgabe bis Datenende
40      .  .  PRINT FORMAT'(3,0)' AGNR,FORMAT'(3,2)' Z1,Z2,Z3,AGN$(0,30)
        .  .  LET J=J+1
        .  LOOP
50      .  PRINT
        .  PRINT 'Dateiende';BEL             ;Abschluss der Ausgabe
60      .ELSEDO
        .  ....                              ;Fehlerbehandlung Kassettenlesen
        DOEND
```

3. Quellen fuer Analysezwecke oder eigenen Veraenderungen

- BAS: Quelle der Basicerweiterungen; FME# = Adr. D4D9
- MPRO: Abarbeitung von Maschinenprozeduren
- BNIX: fuer INPUT
- BNPR: fuer PRINT
- SD1152S: Druckertreiber