MOPPEL
System-Software

Monitor-Handhabung
und
System-Handbuch

Modulares Prozessor-Programm zum Entwickeln und Lernen

® Eingetragenes Warenzeichen

hms

Allgemeine Bedienungshinweise

Das ROM-residente Monitor-Programm ist in konsequenter Menü-Technik aufgebaut; in den Abschnitten 1...4 ist die Monitor-Handhabung ausführlich beschrieben und anhand von Beispielen erläutert.

Die Eingabe der Kommando-Buchstaben kann wahlweise in Klein- oder Großschreibung erfolgen; die danach eingegebenen Parameter können, müssen aber nicht durch einen Leerschritt ('Blank') vom Kommando-Buchstaben getrennt sein. Mehrere Parameter sind wahlweise durch Komma oder Punkt voneinander zu trennen, führende Nullen brauchen nicht eingegeben zu werden. Aus Gründen der Übersichtlichkeit ist das Kommando in den folgenden Beschreibungen stets ein Großbuchstabe, der von den Parametern durch einen Leerschritt abgesetzt ist.

Zur Deutlichmachung ist an alle Hexadezimalzahlen ein 'h' angehängt; soweit Mißverständnisse möglich sind, kennzeichnet ein nachgestelltes 'd' Dezimalzahlen. Die Abkürzung 'LSB/LSD' steht für 'Least Significant Bit/Digit' (niedrigstwertiges Bit/Digit), und 'MSB/MSD' bedeutet entsprechend 'Most Significant Bit/Digit' (höchstwertiges Bit/Digit).

Bei Falscheingaben in der Kommandozeile werden folgende Fehler sofort erkannt und gemeldet:

Error # ØØ: Eingabe fehlt oder entstammt nicht dem Menü
Error # Ø2: Parameter-Fehler (kein HEX-Zeichen)
Error # Ø3: Parameter-Feld zu lang

Das Gerät zeigt seine Eingabe-Bereitschaft durch das Blinken des Cursors an; bei ausgeschaltetem Cursor erfolgen interne Verarbeitungen, die keine Eingaben des Anwenders zulassen.

Für den praktischen Einsatz ist es sehr vorteilhaft, daß der Drucker jederzeit (auch bei laufenden Listings!) zu- oder abgeschaltet werden kann; zur Drucker-Ansteuerung ist das Universal-Interface erforderlich.- Auch die frei belegbare Funktions-Ebene der Tastatur bietet dem Anwender vielseitige Unterstützung bei der Arbeit (vgl. Abschnitt 6.1 'Monitor-Eingabe-Routinen CI und CSTS').

Sämtliche ROM-residenten Programme unterstützen zwei Bildformate, die abhängig sind von der Einstellung des Video-Interfaces: 24 Zeilen mit je 8Ø Zeichen oder 2Ø Zeilen mit je 4Ø Zeichen. Darüber hinaus läßt sich der Video-Zeichengenerator auf ein internes RAM umschalten, dessen Zeichen -auch während des Betriebes- frei ladbar sind und sich so unterschiedlichen Anwendungsfällen anpassen lassen (z.B. für nationale Zeichensätze oder Grafik-Symbole; vgl. Abschnitt 7.6 'Modifikation des Video-Zeichengenerators').

Die vorliegende Beschreibung legt die aktuelle Software-Version V.6 zugrunde, die für die 89er- und 87er-Hardware in gleichem Maße gilt; soweit Hardware-Komponenten angesprochen sind, bezieht sich dies auf die 89er-Baureihe. Für die 87er-Baugruppen gibt eine Kreuz-Referenzliste die jeweiligen Abweichungen gegenüber dem 89er-Äquivalent an.- Für die weitergehende Beschäftigung mit dem Gerät sind im Abschnitt 7 die wesentlichen System-Parameter zusammengestellt und erläutert (u.a. Bus- und Speicherbelegung, Monitor-Unterprogramme).

Inhaltsverzeichnis

Ø. Funktionen des HEX-Monitors

1. MEMORY-Menü
1.1 Copy (Speicherbereich umkopieren)
1.2 Fill (Speicherbereich auffüllen)
1.3 Kill (Speicherinhalt ersetzen)
1.4 List (HEX-Listing des Speicherinhalts)
1.5 Memory (Speicherinhalt inspizieren und modifizieren)
1.6 Revise (Speicherbereiche vergleichen)
1.7 Text (ASCII-Listing des Speicherinhalts)
1.8 Exchange (Speicherbereich umkopieren mit Adreßanpassung)

2. OUTWARD-Menü
2.1 Assembler (Umsetzen von Quellprogrammen in den Objektcode)
2.2 BASIC (Programmerstellung in symbolischer Hochsprache)
2.3 Disassembler (Umsetzen von Objektcode in Assemblersprache)
2.4 Editor (Erstellen von Quellprogrammen in Assemblersprache)
2.5 Floppy (Programmpaket für die Disketten-Handhabung)
2.6 Prommer (Programmpaket für die EPROM-Handhabung)
2.7 Extension 2ØØØh (Anwender-Festprogramm ab 2ØØØh)

3. PERIPHERAL-Menü
3.1 Cass-In (Cassetten-Eingabe)
3.2 Cass-Out (Cassetten-Ausgabe)
3.3 Terminal (Vollduplex-Modem- und DFÜ-Betrieb)
3.4 Uhr (Echtzeit-Uhr auslesen bzw. stellen)
3.5 V.24-In (V.24-Eingabe)
3.6 V.24-Out (V.24-Ausgabe)

4. START-Menü
4.1 Go (Programm im Echtzeit-Betrieb starten)
4.2 Step (Programm im Einzelschritt-Betrieb starten)
4.3 User FØØØh (Programm bei der User-Adresse FØØØh starten)
4.4 Byte (Programm im Einzel-Byte-Betrieb starten)
4.5 Break (Programm bis Breakpoint im Echtzeit-Betrieb starten)

5. Peripherie-Anschluß
5.1 Monitor (Datensichtgerät), BAS- und TTL-Ansteuerung
5.2 Drucker, RS232C (seriell) und Centronics (parallel)
5.3 Magnetbandgerät (fernsteuerbar)
5.4 Terminal und Modem (Daten-Fernübertragung DFÜ)
5.5 Floppy-Disk-Laufwerke

6. Ein- und Ausgaben
6.1 Monitor-Eingabe-Routinen CI und CSTS
6.2 Codierung der ASCII-Tastatur
6.3 Monitor-Ausgabe-Routine CO
6.4 Verarbeitung der Bildschirm-Codes
6.5 Escape-Sequenzen
6.6 Einsatz der Echtzeit-Uhr
6.7 Drucker-Aktivierung

7. System-Parameter
7.1 Belegung des Adreßraums (Memory-Map)
7.2 HEXMO-Festadressen und -Unterprogramme
7.3 MOVID-Festadressen und -Unterprogramme
7.4 Belegung des Anwender- und Monitor-RAMs ab 28ØØh
7.5 Disketten-Datensicherung
7.6 Modifikation des Video-Zeichengenerators
7.7 System-Konstanten im EPROM
7.8 Busbelegung
7.9 Belegung der Portadressen

ø. HEX-Monitor

Für Ausbildungs- oder Prüfzwecke, aber auch in kleineren Anwendungsfällen, kann eine Minimalkonfiguration des MOPPEL eingesetzt werden, die außer der Zentraleinheit nur eine hexadezimale Tastatur mit hexadezimaler Siebensegment-Anzeige verwendet.

Die Bedienungs- und Anzeige-Funktionen spielen sich dabei auf unterster Ebene ab, da der Komfort von Text-Ein- und -Ausgaben über ASCII-Tastatur bzw. Video-Interface nicht zur Verfügung steht.

Prinzipiell benutzt auch diese Ausbaustufe dieselben Monitor-Routinen wie der anschließend beschriebene Video-Monitor, nur eben mit eingeschränkten Eingabe- und Anzeige-Funktionen:

DAT: Daten-Mode einstellen
Der Dezimalpunkt im Datenfeld leuchtet, und Eingaben von der HEX-Tastatur gelangen ins Datenfeld.

ADR: Adreß-Mode einstellen
Der Dezimalpunkt im Adreßfeld leuchtet, und Eingaben von der HEX-Tastatur gelangen ins Adreßfeld; Mode-Anzeige: Eine LED.

REG: Register-Mode einstellen
Anstelle einer Speicheradresse wird eine Register-Bezeichnung angegeben; Mode-Anzeige: Zwei LEDs.

FCT: Funktions-Eingabe vorbereiten
Dient zur Vorbereitung der auf dem folgenden Blatt genannten Funktionen; Mode-Anzeige: Drei LEDs.

NXT: Weiterschaltung
Angezeigte Adresse oder Register-Bezeichnung erhöhen.

BST: Rückschaltung
Angezeigte Adresse oder Register-Bezeichnung erniedrigen.

RUN: Ausführung starten
Programm oder Funktionsaufruf starten.

RES: Rücksetzen
Hardware-RESET mit Neu-Initialisierung aller Parameter; anschließend ist der Daten-Mode eingestellt; Mode-Anzeige: Eine LED.

Ø. Funktionen des HEX-Monitors

FCT Ø: Cass-Out (Cassetten-Ausgabe); vgl. Abschnitt 3.2
Format: aaaa<NXT> eeee<RUN>

FCT 1: Cass-In (Cassetten-Eingabe); vgl. Abschnitt 3.1
Format: aaaa<NXT> eeee<RUN>

FCT 2: CPU-Out (Ausgabe über SOD des 8085-Mikroprozessors)
Format: aaaa<NXT> eeee<RUN>

FCT 3: CPU-In (Eingabe über SID des 8085-Mikroprozessors)
Format: aaaa<NXT> eeee<RUN>

FCT 4: List (HEX-Listing über den Thermodrucker)
Format: aaaa<NXT> eeee<RUN>

FCT 5: Text (ASCII-Listing über den Thermodrucker)
Format: aaaa<NXT> nn<RUN>; nn: Zeilenzahl (hexadezimal)

FCT 6: Exchange (Umkopieren m.AdreBanpassung); vgl. Abschnitt 1.8
Format: aaaa<NXT> eeee<RUN>

FCT 7: Insert (ab Adresse aaaa NOPs (=ØØh) einfügen)
Format: aaaa<NXT> nn<RUN>; nn: Anzahl (hexadezimal)

FCT 8: Uhr (Echtzeit-Uhr stellen); vgl. Abschnitt 3.4
Format: ssmm<NXT> WKK<NXT> MMJJ<RUN>; Std/Min;Tag;Mon/Jahr

FCT 9: Uhr (Echtzeit-Uhr auslesen); vgl. Abschnitt 3.4
Format: aaaa<RUN>; aaaa: BCD-Uhrzeit-Buffer

FCT A: Fill (Speicherbereich auffüllen); vgl. Abschnitt 1.2
Format: aaaa<NXT> eeee<NXT> kk<RUN>; kk: Konstante

FCT B: Break (Programmstart bis Breakpoint); vgl. Abschnitt 4.5
Format: aaaa<MXT> eeee<RUN>; ab 'eeee' im Einzelschritt

FCT C: Copy (Speicherbereich umkopieren); vgl. Abschnitt 1.1
Format: aaaa<NXT> eeee<NXT> zzzz<RUN>

FCT D: Delete (ab Adresse aaaa Speicherinhalt nachrücken)
Format: aaaa<NXT> nn<RUN>; nn: Anzahl zu löschender Bytes

FCT E: Step (Einzelschritt-Betrieb), vgl. Abschnitt 4.2
Format: aaaa<RUN>; Anzeige:'run' <RUN>

1. MEMORY-Menü

Aus dem Monitor-Grundmenü erreichen Sie das MEMORY-Menü, indem Sie einfach 'm' eingeben (ohne 'Return').

Das MEMORY-Menü bietet die folgenden acht Betriebsarten an, um auf den Speicherinhalt zuzugreifen, d.h. ihn auszugeben oder zu verändern:

1.1 Copy (Speicherbereich umkopieren)
1.2 Fill (Speicherbereich auffüllen)
1.3 Kill (Speicherinhalt ersetzen)
1.4 List (HEX-Listing des Speicherinhalts)
1.5 Memory (Speicherinhalt inspizieren und modifizieren)
1.6 Revise (Speicherbereiche vergleichen)
1.7 Text (ASCII-Listing des Speicherinhalts)
1.8 Exchange (Speicherbereich umkopieren mit Adreßanpassung)

Aus dem MEMORY-Menü kommen Sie wie folgt zurück ins Monitor-Grundmenü:

- durch 'TAB' unmittelbar zum Monitor-Kaltstart
  (sofortige Ausgabe des Monitor-Grundmenüs)
- durch 'Escape' bzw. CTL+C jeweils eine Ebene höher im Menü
  (max. zweimalige Betätigung bis zum Monitor-Kaltstart)

hms

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>c 9000,a000,b000                    o.k.

Mm>c 9000,8000,b000                    Error # 16

Mm>c 9000,a000,1000                    Error # 19
9001=D0  1001:4B

Mm>
```

1.1 C: Copy (Speicherbereich umkopieren)

Format: Mm>C aaaa,eeee,zzzz<Ret>

Funktion: Umkopieren des Speicherbereichs von Anfangsadresse aaaa bis einschließlich Endadresse eeee in den bei Zieladresse zzzz beginnenden RAM-Bereich.

Zulässig: Quell- und Zielbereich können sich überlappen, d.h. die Zieladresse zzzz kann ohne weiteres im Bereich aaaa....eeee liegen.

Quittung: o.k.

Error # 16: Adresse eeee ist kleiner als Adresse aaaa.
Error # 19: Einschreiben fehlerhaft (mit Abbruch der Operation und Fehler-Listing).

Hinweis: Um das überlappende Kopieren zu ermöglichen, verläuft der Kopiervorgang aufwärts (von aaaa -> eeee), wenn aaaa>eeee ist und abwärts (eeee -> aaaa), wenn gilt aaaa<eeee.

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>f 9000,9fff,e5                    o.k.

Mm>f 9000,8000,e5                    Error # 17

Mm>f 9000,9000,e5                    Error # 18

Mm>f 1000,1fff,e5                    Error # 19
1000=C3  1001:48

Mm>
```

1.2 F: Fill (Speicherbereich auffüllen)

Format: Mm>F aaaa,eeee,kk<Ret>

Funktion: Füllen des Speicherbereichs von Anfangsadresse aaaa bis einschließlich Endadresse eeee mit der Konstanten kk; bei kk=ØØ kann die letzte Eingabe entfallen.

Zulässig: Für die Konstante kk kann jeder hexadezimale Wert von ØØ...FFh eingesetzt werden.

Quittung: o.k.

Error # 17: Adresse eeee ist kleiner als Adresse aaaa.
Error # 18: Adresse eeee ist gleich Adresse aaaa.
Error # 19: Einschreiben fehlerhaft (mit Abbruch der Operation und Fehler-Listing).

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>k 8010,802f,e500                    o.k.

Mm>k 1000,1fff,e500                    Error # 19
1146=E5  1145:11

Mm>
```

1.3 K: Kill (Speicherinhalt ersetzen)

Format: Mm>K aaaa,eeee,xxyy<Ret>

Funktion: Im Speicherbereich von Anfangsadresse aaaa bis Endadresse eeee alle Bytes xx durch yy ersetzen.

Zulässig: Für die Konstanten xx und yy kann jeder hexadezimale Wert von ØØ...FFh eingesetzt werden. Ist Adresse eeee kleiner als aaaa, erfolgt das Absuchen des Speichers über das Speicherende FFFFh hinaus.

Quittung: o.k.

Error # 19: Einschreiben fehlerhaft (mit Abbruch der Operation und Fehler-Listing).

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>l 8000,805a
8000   E5 E5 E5 E5   E5 E5 E5 E5     E5 E5 E5 E5   E5 E5 E5 E5
8010   00 00 00 00   00 00 00 00     00 00 00 00   00 00 00 00
8020   00 00 00 00   00 00 00 00     00 00 00 00   00 00 00 00
8030   E5 E5 E5 E5   E5 E5 E5 E5     E5 E5 E5 E5   E5 E5 E5 E5
8040   E5 E5 E5 E5   E5 E5 E5 E5     E5 E5 E5 E5   E5 E5 E5 E5
8050   E5 E5 E5 E5   E5 E5 E5 E5     E5 E5 E5

Mm>
```

1.4 L: List (HEX-Listing des Speicherinhalts)

Format: Mm>L aaaa,eeee<Ret>

Funktion: Hexadezimales Listing des Speicherinhalts ab Anfangsadresse aaaØ (LSD auf Ø abgerundet) bis Endadresse eeee.

Zusatzfunktionen: Das Anhalten bzw. Fortsetzen des Listings ist jederzeit durch Betätigen der Blank-Taste möglich. Beim Stoppen kann der Paralleldruck ein- oder wieder ausgeschaltet werden (durch CTL+7... CTL+4); 'Escape' bzw. CTL+C veranlassen einen Abbruch des Listings mit anschließendem Rücksprung ins MEMORY-Menü.

Zulässig: Die Adressen aaaa und eeee können im gesamten Adreßbereich von ØØØØ...FFFFh liegen; ist Adresse eeee kleiner als aaaa, erfolgt das Listen über das Speicherende FFFFh hinaus.

```
M>m
.
C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>m

Adr.Inh.

2800 cd
2801 48
2802 10
2803 c3
2804 03
2805 10
2806 FF                                  Error # 02

Mm>m 1000

Adr.Inh.

1000 cd
1000 C3                                  Error # 19

Mm>
```

1.5 M: Memory (Speicherinhalt inspizieren und modifizieren)

Format: Mm>M aaaa<Ret> oder Mm>M<Ret>

Funktion: Speicheradresse aaaa und deren Inhalt ii listen; Inhalt ii modifizieren (sofern die Adresse aaaa im RAM-Bereich liegt) sowie Adresse aaaa um Eins erhöhen bzw. erniedrigen.

Zusatzfunktion: Wenn beim Aufruf die Angabe der Adresse aaaa fehlt, setzt der Monitor dafür die unterste RAM-Adresse 2800h ein.

Zulässig: Für die Adresse aaaa kann grundsätzlich auch ein Wert aus dem ROM-Bereich eingesetzt werden, sofern nur eine Inspektion und keine Modifikation des Inhalts erfolgt.

Bedienung: Zur Modifikation des Speicherinhalts ii sind die HEX-Tasten Ø...9 und A...F zu betätigen; der blinkende Cursor gibt die augenblickliche Schreibposition an, die sich mit jeder Eingabe um eine Stelle verändert. Der Cursor läßt sich auch ohne HEX-Eingabe mit der Tasten 'links' (=CTL+S) bzw. 'rechts' (=CTL+D) bewegen.

Das Einschreiben des modifizierten Speicherinhalts ins RAM erfolgt erst nach dem Weiterschalten zur nächsten Adresse aaaa+1 (durch 'Return' oder 'Cursor abwärts' =CTL+X). Das Zurückschalten zur vorigen Adresse aaaa-1 ist durch die Taste 'Minus' möglich (oder 'Cursor aufwärts' =CTL+E); der Inhalt der Adresse aaaa wird dabei nicht verändert.

Der Rücksprung ins MEMORY-Menü kann nur durch Betätigen der Taste 'm' erfolgen; in dieser Betriebsart ist kein Warmstart über 'Escape' oder CTL+C möglich.

Quittung: Implizit gegeben (sonst erfolgt Fehlermeldung).

Error # Ø2: Unzulässige Eingabe (nur HEX, Return, Cursor oder 'm').
Error # 19: Einschreiben fehlerhaft.

hms

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>r 9000,a000,b000                    o.k.


Mm>f 8000,ffff,49                      o.k.

Mm>r 8000,ffff,8001
FFFF=49  0000:C3

Mm>
```

1.6 R: Revise (Speicherbereiche vergleichen)

Format: Mm>R aaaa,eeee,zzzz<Ret>

Funktion: Vergleich des Speicherinhalts ab Anfangsadresse aaaa bis Endadresse eeee mit dem bei Zieladresse zzzz beginnenden Speicherinhalt und Listen aller Abweichungen:

SSSS=ss DDDD:dd (S/D:Source/Dest-Adresse;
s/d:Source/Dest-Inhalt)

Zusatzfunktionen: Das Anhalten bzw. Fortsetzen des Listings ist jederzeit durch Betätigen der Blank-Taste möglich. Beim Stoppen kann der Paralleldruck ein- oder wieder ausgeschaltet werden (durch CTL+7...CTL+4); 'Escape' bzw. CTL+C veranlassen einen Abbruch des Listings mit anschließendem Rücksprung ins MEMORY-Menü.

Zulässig: Die Adressen aaaa, eeee und zzzz können im gesamten Adreßbereich von 0000...FFFFh liegen; ist Adresse eeee kleiner als aaaa, erfolgt der Vergleich über das Speicherende FFFFh hinaus.

Quittung: o.k. (wenn keine Abweichungen vorlagen).

Hinweis: Das REVISE-Kommando bietet eine einfache Möglichkeit für einen Speichertest. Man braucht dazu nur den gewünschten RAM-Bereich von Adresse aaaa...eeee mit einer Konstanten zu füllen und REVISE anzuwenden auf Mm>r aaaa,eeee,aaaa+1. In diesem Fall darf nur eine einzige Abweichung gelistet werden (bei Adresse eeee/eeee+1).

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>t9000,24


Dieser Beispieltext wurde mit dem Editor erstellt
(ab Adresse 9000h im RAM) und mit dem Dollar-Zeichen
(24h) abgeschlossen (als End-Kennung für 'TEXT').

Mm>
```

1.7 T: Text (ASCII-Listing des Speicherinhalts)

Format: Mm>T aaaa,ee<Ret> oder Mm>T aaaa,lee<Ret>

Funktion: Listen des Speicherinhalts ab Anfangsadresse aaaa als ASCII-Zeichenfolge, bis das Endzeichen ee erreicht ist (bei fehlender Eingabe von ee: Bis das Endzeichen ØØh erreicht ist). Bei jedem Wagenrücklauf (CR=Carriage Return, ØDh) wird automatisch ein Zeilenvorschub (LF=Line Feed, ØAh) eingefügt. Ist vor 'ee' der Line Feed Operator 'l' eingesetzt (HEX-Zeichen ungleich Null), wird der automatische Zeilenvorschub unterdrückt.

Zusatzfunktionen: Das Anhalten bzw. Fortsetzen des Listings ist jederzeit durch Betätigen der Blank-Taste möglich. Beim Stoppen kann der Paralleldruck ein- oder wieder ausgeschaltet werden (durch CTL+7...CTL+4); 'Escape' bzw. CTL+C veranlassen einen Abbruch des Listings mit anschließendem Rücksprung ins MEMORY-Menü.

Zulässig: Es kann von jeder Adresse aaaa aus dem Adreßbereich ØØØØ...FFFFh aus gestartet werden, und für das das Endzeichen ee kann jeder hexadezimale Wert ØØ...FFh eingesetzt werden.

```
M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>x 4af,4be,2800                    o.k.

Mm>x 4af,3be,2800                    Error # 16

Mm>x 4af,4be,800                     Error # 19
04AF=F5  080F:51
04B0=3A  0810:C2
04B1=06  0811:3F
...

M>
```

1.8 X: Exchange (Speicherbereich umkopieren mit AdreBanpassung)

Format: Mm>X aaaa,eeee,zzzz<Ret>

Funktion: Wie beim Umkopieren (C: Copy), jedoch mit Anpassung aller Absolutadressen, die im AdreBbereich aaaa...eeee liegen, d.h. nach der Anpassung sind die übertragenen Programme im neuen Zielbereich lauffähig!

Zulässig: Es können ohne weiteres auch Programme aus dem ROM ins RAM umkopiert und dort lauffähig gemacht werden!

Quittung: o.k.

Error # 16: Adresse eeee ist kleiner als Adresse aaaa.
Error # 19: Einschreiben fehlerhaft (mit Fehler-Listing).

hms

2. OUTWARD-Menü

Aus dem Monitor-Grundmenü erreichen Sie das OUTWARD-Menü, indem Sie einfach 'o' eingeben (ohne 'Return').

Das OUTWARD-Menü bietet die folgenden sieben Möglichkeit an, zu externen Programmteilen zu verzweigen:

2.1 Assembler (Umsetzen von Quellprogrammen in den Objektcode)
2.2 BASIC (Programmerstellung in symbolischer Hochsprache)
2.3 Disassembler (Umsetzen von Objektcode in Assemblersprache)
2.4 Editor (Erstellen von Quellprogrammen in Assemblersprache)
2.5 Floppy (Programmpaket für die Disketten-Handhabung)
2.6 Prommer (Programmpaket für die EPROM-Handhabung)
2.7 Extension 2000h (Anwender-Festprogramm ab 2000h)

Aus den externen Programmteilen kommen Sie wie folgt zurück ins Monitor-Grundmenü:

- durch 'm' zum Monitor-Warmstart und von dort durch 'Escape' bzw. CTL+C zum Monitor-Kaltstart (Ausgabe des Monitor-Grundmenüs) (in BASIC durch 'bye' direkt zum Monitor-Kaltstart)
- Anwender-spezifisch im Extension-Programmteil

```
 0/1/2/3: Pass #
 I: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>1

A>2
o.k.; Code End: 2C6Fh

A>s
CHRLIN 0050   CHRLOP 2C4F   CO     0049   CRLF   2C63   HCOPY  2C40
IOFLAG 2FC8   LINLOP 2C4A   LINPAG 0018   LO     004F   LODEV  0080

A>3

0001 0000             *
0002 0000             ORG 2C40H
0003 2C40             OFS 3C40H
0004 2C40             *
0005 2C40             LO     4FH
0006 2C40             LODEV  80H            *I/O-FLAG für Druck
0007 2C40             CO     49H
0008 2C40             IOFLAG 2FC8H
0009 2C40             *
0010 2C40             CHRLIN   50H          *80d Zeichen pro Zeile
0011 2C40             LINPAG   18H          *24d Zeilen pro Seite
0012 2C40             *
0013 2C40             *
0014 2C40  21 00 F8 HCOPY: LXI  H,0F800H  *Video-Buffer ausdrucken
0015 2C43  06 18           MVI  B,LINPAG
0016 2C45  3E 80           MVI  A,LODEV
0017 2C47  32 C8 2F        STA  IOFLAG
0018 2C4A  CD 63 2C LINLOP:CALL CRLF
0019 2C4D  0E 50           MVI  C,CHRLIN
0020 2C4F  C5       CHRLOP:PUSH B
0021 2C50  4E              MOV  C,M
0022 2C51  CD 49 00        CALL CO
0023 2C54  23              INX  H
0024 2C55  C1              POP  B
0025 2C56  0D              DCR  C
0026 2C57  C2 4F 2C        JNZ  CHRLOP
0027 2C5A  05              DCR  B
0028 2C5B  C2 4A 2C        JNZ  LINLOP
0029 2C5E  AF              XRA  A
0030 2C5F  32 C8 2F        STA  IOFLAG
0031 2C62  C9              RET
0032 2C63             *
0033 2C63  C5       CRLF:  PUSH B
0034 2C64  0E 0D           MVI  C,0DH
0035 2C66  CD 4F 00        CALL LO
0036 2C69  0E 0A           MVI  C,0AH
0037 2C6B  CD 4F 00        CALL LO
0038 2C6E  C1              POP  B
0039 2C6F  C9              RET
0040 2C70             *
0041 2C70             END

A>
```

2.1 A: Assembler (Umsetzen von Quellprogrammen in den Objektcode)

Aufruf: Mo>a oder Mo>A (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul 'Assembler', für das eine eigene, ausführliche Beschreibung existiert.

Kurzbeschreibung:

Ø: **Pass # Ø:** Vorhandene Symboltabelle erweitern.
Ein neues Quellprogramm verwendet die symbolischen Namen (Label) einer Symboltabelle, die zuvor beim Assemblieren eines anderen Programms angelegt wurde (ersetzt Pass # 1).

1: **Pass # 1:** Symboltabelle anlegen.
Ersten Assembler-Durchlauf eines Quellprogramms, bei dem für alle symbolischen Namen (Label) die zugehörigen Absolutadressen ermittelt und in eine Symboltabelle eingetragen werden.

2: **Pass # 2:** Objektcode erzeugen.
Zweiter Assembler-Durchlauf mit Umsetzung eines in Assemblersprache vorliegenden Quellprogramms in die Maschinensprache (Objektcode; Standard-Beginn bei F0ØØh), wobei die Absolutadressen der zuvor angelegten Symboltabelle verwendet werden.

3: **Pass # 3:** Komplett-Listing erstellen.
Dritter Assembler-Durchlauf mit paralleler Ausgabe von Quellprogramm und Objektcode für Dokumentationszwecke.

D: **Disassembler:** Direkte Verzweigung zum Disassembler.

E: **Editor:** Direkte Verzweigung zum Editor.

F: **Floppy:** Direkte Verzweigung zu den Floppy-Routinen.

H: **HEX-Mode:** Objektcode im HEX-Format ausgeben (im Pass # 3).

I: **In/Out on/off:** Anwender-Ein/Ausgabe (de)aktivieren.

M: **Monitor:** Direkter Rücksprung zum Monitor-Kaltstart.

O: **OCT-Mode:** Objektcode im OCT-Format ausgeben (im Pass # 3).

S: **Symbol Table:** Symboltabelle alphabetisch listen.

hms

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->x

o.k.

B>10 rem Dieses BASIC-Programm wurde mit dem Editor erstellt, in BASIC
B>11 rem per X-Einsprung aus dem RAM eingelesen und automatisch gestar-
B>12 rem tet ("run" ist der letzte Befehl des Editor-Listings)
B>13 rem
B>100 print " " : rem Leerzeile einfügen
B>110 crspos 10,24 : rem Cursor auf Spalte 10, (unterste) Zeile 24
B>120 porint hex("1b"),hex("4d") : rem Nur an Drucker: Engschrift select
B>130 lprint "Es ist ": ti$ : rem Uhrzeit/Datum anzeigen und eng drucken
B>run

          Es ist Mi,25.03.87;12:24:09h


o.k.

B>
```

2.2 B: BASIC (Programmerstellung in symbolischer Hochsprache)

Aufruf: Mo>b oder Mo>B (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul 'BASIC-Interpreter', für das eine eigene, ausführliche Beschreibung existiert.

Kurzbeschreibung:

C: Convert: Version-V-3.3-Programme umsetzen auf Version V 3.6.
K: Kalt: BASIC-Kaltstart mit Initialisierung aller Parameter.
M: Monitor: Direkter Rücksprung zum Monitor-Kaltstart.
W: Warm: BASIC-Warmstart mit Übernahme der vorhandenen Parameter.
X: Extern: Externen Eingabe-Kanal aktivieren
(Eingaben als ASCII-String fortlaufend aus dem RAM holen, nicht mehr von der Tastatur).

Befehlsumfang:

Programmausführung:	RUN <i>	:STOP	:CONT
Verzweigung:	GOTO i	:IF...THEN	:ON...GOTO
Schleife:	FOR i=m TO n	:STEP s	:NEXT
Unterprogrammaufruf:	GOSUB i	:RETURN	:CALL i
Definitionen:	CLEAR i	:DIM x(i)	:LET x=1
Rand/Kommentar/Ende:	NULL (i)	:REM	:END
Exponentialfunktionen:	SQR (x)	:LOG (x)	:EXP (x)
Trigonometrische Fkt.:	SIN (x)	:COS (x)	:TAN (x) :ATN (x)
Übrige Funktionen:	DEF FN f(x)	:USR (i)	:TI$
Variablen-Bewertung:	ABS (x)	:INT (x)	:SGN (x)
Logische Operationen:	NOT x	:x AND y	:x OR y
Daten-Direktzugriff:	DATA x	:READ x	:RESTORE
Zahlenumsetzung:	HEX ("h")	:ASC ("a")	:CHR$(d)
String-Umsetzung:	LEN(a$)	:STR$(x)	:VAL(a$)
String-Verarbeitung:	LEFT$(a$,i)	:MID$(a$,i,j)	:RIGHT$(a$,i)
RAM-In/Out:	PEEK (i)	:POKE i,x	
Port-In/Out:	INP (i)	:OUT i,x	:WAIT i,x,y
Console In/Out:	INPUT x	:PRINT x	:LIST <i>
Cassette In/Out:	CLOAD	:CSAVE	
Drucker-Ansteuerung:	LPRINT x	:PPRINT x,y	:LLIST <i>
Bildschirm-Verwaltung:	CLRSCN	:CLRLIN	:CRSPOS i,j
	SPC (i)	:TAB	:POS (x)
Speicher-Verwaltung:	FRE (x)	:NEW	:BYE

```
8080-Disassembler V 0.6
Copyright (C) hms'86

Start,End:

D>4af,4be

  04AF  F5          ...... PUSH PSW
  04B0  3A 06 00    ...... LDA  0006h
  04B3  87          ...... ADD  A
  04B4  87          ...... ADD  A
  04B5  3C          ...... INR  A
  04B6  87          ...... ADD  A
  04B7  87          ...... ADD  A
  04B8  87          ...... ADD  A
  04B9  3D          ...... DCR  A
  04BA  C2 B9 04    ...... JNZ  04B9h
  04BD  F1          ...... POP  PSW
  04BE  C9          ...... RET

D>
```

Hinweis: Bei 4AFh beginnt 'DELY1' (Einsprung bei 0006h)

2.3 D: Disassembler (Umsetzen von Objektcode in Assemblersprache)

Aufruf: Mo>d oder Mo>D (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul 'Disassembler', das Bestandteil des Assemblers ist und im Zusammenhang mit diesem ausführlich beschrieben wird.

Kurzbeschreibung:

Nach Angabe der Start- und Endadresse führt der Disassembler eine Rückübersetzung von Maschinenprogrammen in die Assemblersprache durch.

Es können sowohl Programme im RAM- als auch im ROM-Bereich disassembliert werden, nur ist bei Angabe der Startadresse darauf zu achten, daß diese bei Mehrwort-Befehlen auf den OpCode des Befehls zeigt. Diese Bedingung gilt für die Endadresse nicht, weil der Disassembler-Durchlauf bei Erreichen oder bei Überschreiten der Endadresse abgebrochen wird.

Das Listen kann jederzeit durch die Blank-Taste unterbrochen bzw. wieder fortgesetzt werden; bei angehaltenem Listen erfolgt der Rücksprung ins MEMORY-Menü durch 'Escape' bzw. CTL+C.

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend                                  Error # 79

E>s 3.2

E>s:0015/9140

      0001  ************************************************
      0002  *                                              *
      0003  *                                              *
      0004  *                                              *
      0005  ************************************************
      0006  *
      0007  *   Stand:
      0008  *
      0009  *   Ende:    Mi,25.03.87:15:54:14h
      0010  *
      0011  org 0f000h
      0012  dfs 0h
      0013  *
      0014  *
      0015  end
*

E>l 11

      0001  org 0f030h
      0002  dfs 0h
      0003  *
      0004  test:  lxi  h,1234h   *Beginn des Quellprogramms.....
      0005         nop  ... .....................................
      0006  ..................................................
      0007  Anzeige des freien Platzes pro Zeile..............
      0008  ..................................................


E>x 0,0,80

E>l 7

*
Anzeige des freien Platzes pro Zeile
*
.....................................................................
und maximal 79d Anschläge pro Zeile ohne Zeilen-Nummer für reine Text-Eingaben.


E>
```

2.4 E: Editor (Erstellen von Quellprogrammen in Assemblersprache)

Aufruf: Mo>e oder Mo>A (ohne 'Return')

Funktion: Verzweigung in das ROM-residente Software-Modul 'Editor', für das eine eigene, ausführliche Beschreibung existiert.

Kurzbeschreibung:

A: Assembler: Direkte Verzweigung zum Assembler.

C: Clear: Zeile(n) löschen.
Eine oder mehrere Zeile(n) mit der/den angegebenen Nummer(n) löschen, die verbleibenden Zeilen nachrücken und neu numerieren.

F: Floppy: Direkte Verzweigung zu den Floppy-Routinen.

G: Go to: Zeichen-Einfügen vorbereiten.
Sprung zur angegebenen Zeilennummer und das Einfügen von Zeichen vorbereiten, d.h. bei Neueingaben werden die übrigen Zeichen einer Zeile nach rechts verschoben (halbzeiliger Cursor). Umschalten zum Zeilen-Einfügen mit CTL+A, zurück zum Editor-Grundmenü mit 'Escape' bzw. CTL+C.

I: Insert: Zeilen-Einfügen vorbereiten.
Vor der angegebenen Zeilennummer eine neue Zeile einfügen, die übrigen Zeilen des Textes nach hinten verschieben und neu numerieren (ganzzeiliger Cursor). Fortsetzen des Zeilen-Einfügens mit 'Return', Umschalten zum Zeichen-Einfügen mit CTL+F, und zurück zum Editor-Grundmenü mit 'Escape' bzw. CTL+C.

L: List: Zeilen listen.
Listen des erstellten Textes von der angegebenen Zeilennummer an; Stoppen und Fortsetzen des Listens durch die Blank-Taste möglich, Abbruch und Rücksprung zum Editor-Grundmenü durch 'Escape' bzw. CTL+C.

M: Monitor: Direkter Rücksprung zum Monitor-Kaltstart.

S: Setup RAM: Text-Buffer initialisieren.
Vorbereiten des Textbuffers ab Adresse 9000h für die Neueingabe von Texten, d.h. Auffüllen mit 00h, Einschreiben der Standard-Kopfzeilen für Assembler-Programme und Einblenden der Echtzeit-Uhr.

X: Extend: Zeilennummer ein/aus, Console einstellen.
Listen der Zeilennummer ein- oder ausschalten, linken Rand und Zeilenlänge einstellen (für den Einsatz als Text-Editor).

Hinweis: Es ist möglich, in den Text hexadezimale Steuerzeichen einzugeben, um beispielsweise Bildschirm- oder Drucker-Kommandos zu verarbeiten (Umschalten der Schriftart o.ä.).

```
Drv.A:xx     Stp/Sid:xx       Dens:xD
Drv.B:xx     Tracks: xxd      Secs:xxd
Drv.C:xx     RAM-Beg:xxh      Byts:xxh
Drv.D:xx     RAM-End:xxh      Comd:xxh
-------------------------------------------------------------------------------

RAMb,RAMe,Tracks>90,EF,40

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor




Drv.A:07     Stp/Sid:30       Dens:DD
Drv.B:xx     Tracks: 40d      Secs:16d
Drv.C:xx     RAM-Beg:90h      Byts:01h
Drv.D:xx     RAM-End:EFh      Comd:A2h
-------------------------------------------------------------------------------

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor




F>w
Dst-Drv,Trk,Sec> A, 2, 1
o.k.

F>
```

2.5 F: Floppy (Programmpaket für die Disketten-Handhabung)

Aufruf: Mo>f oder Mo>F (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul 'Disketten-Handhabung', für das eine eigene, ausführliche Beschreibung existiert.

Kurzbeschreibung:

A: Assembler: Direkte Verzweigung zum Assembler.

B: Batch Out: Vorspann für Selbst-Start ausgeben.
Ausgabe eines numerierten Vorspanns (Nr.Ø...31), der sämtliche Angaben zum Einlesen und Starten eines Programms von Diskette enthält.

C: Copy: Spuren kopieren.
Die unter 'Tracks' angegebene Anzahl von Spuren kopieren. Quelle: Wie unter 'Source' (Seite/Laufwerk, Spur) spezifiziert; Ziel: Wie unter 'Destination' (Seite/Laufwerk, Spur) spezifiziert.

D: Disk in: Batch-Vorspann einlesen und Programm starten.
Batch-Vorspann mit der angegebenen Nummer von der Diskette holen, den darin bezeichneten Datenblock einlesen, und das Programm bei der im Vorspann enthaltenen Adresse automatisch starten.

E: Editor: Direkte Verzweigung zum Editor.

F: Format: Diskette formatieren.
Die unter 'Tracks' spezifizierte Anzahl von Spuren einschließlich Spur- und Sektor-Kennung auf die Diskette schreiben; im Datenbereich jeder neu formatierten Spur stehen nur Bytes mit Inhalt 'E5h', die bei CP/M-Programmen als Leer-Information interpretiert werden.

M: Monitor: Direkter Rücksprung zum Monitor-Kaltstart.

R: Read: Diskette lesen.
Daten von Diskette ins RAM einlesen, beginnend bei Anfangsadresse 'RAM-Beg' (abgerundet auf xxØØh) bis einschließlich Endadresse 'RAM-End' (aufgerundet auf xxFFh); Blockanfang: Seite/Laufwerk, Spur, Sektor wie spezifiziert.

W: Write: Diskette beschreiben.
Daten aus dem Speicher auf Diskette schreiben, beginnend bei Anfangsadresse 'RAM-Beg' (abgerundet auf xxØØh) bis einschließlich Endadresse 'RAM-End' (aufgerundet auf xxFFh); Blockanfang: Seite/Laufwerk, Spur, Sektor wie spezifiziert.

```
Typ: 2764          SB:8000                                    Mi,25.03.87:16:1e:22h
Vpp: 21 V          SE:9FFF
Tim: 0'43          DB:0000
-------------------------------------------------------------------------------

MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>p8
Vpp: 2í V?
o.k.
o.k.

P>
```

2.6 P: Prommer (Programmpaket für die EPROM-Handhabung)

Aufruf: Mo>p oder Mo>P (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul 'EPROM-Handhabung', für das eine eigene, ausführliche Beschreibung existiert.

Kurzbeschreibung:

A: **Adjust:** Programmierspannungen abgleichen.
Bildschirmführung zur Einstellung der drei auf dem Programmier-Modul vorhandener Potentiometer, die den Feinabgleich der verschiedenen Programmierspannungen vornehmen.

B: **Blank Check:** EPROM auf Leer-Inhalt prüfen.
Auslesen des gesamten EPROM-Inhalts und Überprüfen, ob alle Speicherplätze gelöscht sind (Inhalt FFh haben); abweichende Inhalte werden mit den zugehörigen Adressen gelistet.

C: **Compare:** EPROM- mit Speicher-Inhalt vergleichen.
Den EPROM-Inhalt von der angegebenen Anfangs- bis einschließlich Endadresse vergleichen mit dem Inhalt eines Speicherbereichs, der bei der ebenfalls angegebenen Zieladresse beginnt; Abweichungen zwischen Quell- und Zielbereich werden gelistet.

M: **Monitor:** Direkter Rücksprung zum Monitor-Kaltstart.

P: **Program:** EPROM programmieren.
Den Speicherinhalt von der angegebenen Anfangs- bis einschließlich Endadresse ins EPROM überschreiben ('programmieren'), dort beginnend bei der ebenfalls angegebenen Zieladresse. Die verstrichene Zeit wird im Feld 'Tim' der Kopfzeilen eingeblendet. Beim anschließenden automatischen Prüflesen festgestellte Fehler werden gelistet.

R: **Read:** EPROM auslesen.
Den EPROM-Inhalt von der angegebenen Anfangs- bis einschließlich Endadresse ins RAM übertragen (auslesen), dort beginnend bei der ebenfalls angegebenen Zieladresse; Fehler beim Einschreiben ins RAM werden gelistet.

T: **Typ:** EPROM-Typenangabe ändern.
Ändern des EPROM-Typs, der im Feld 'Typ' der Kopfzeilen eingeblendet wird.

V: **Vpp:** Programmierspannung ändern.
Ändern der beim Programmieren verwendeten Spannung, deren Wert im Feld 'Vpp' der Kopfzeilen eingeblendet wird.

```
**************************************************
*                                                *
*      FUNKUR    DCF77-Auswertung  Ver.5         *
*                                                *
**************************************************

       Portadresse für DCF-Signal: C8

Sekundenzähler:  00
Schieberegister: 40       Fehlermeldung: --  ----
-------------------------------------------------PAR-
Minuten:              1  0  1    0  1  1  0      0

Stunden:                 0  0    1  0  0  1      0

Kalendertag              1  0    0  1  1  1

Wochentag:                          1  0  1

Monat:                      0    0  0  1  1

Jahr:              1  0  0  0    0  1  1  1      0

Uhrzeit-Übernahme:    Mm> Fr,27.03.87;09:56:00h
```

2.7 X: Extension 2000h (Anwender-Festprogramm ab 2000h)

Aufruf: Mo>x oder Mo>X (ohne 'Return')

Funktion: Verzweigung zum ROM-residenten Software-Modul ab Adresse 2000h (Hilfsbank 2, vgl. Abschnitt 7.1 'Belegung des Adreßraums (Memory Map)').

Beschreibung: Auf dem 8-K-EPROM des Floppy-Disk-Controllers (FDC) ist ein Speicherbereich von 2 KBytes frei, den der Anwender für eigene, ROM-residente Programme nutzen kann; mit der X-Anweisung erfolgt der direkte Sprung zu diesem Programmteil; die Möglichkeit des Monitor-Rücksprungs ist vom Anwender bereitzustellen.

3. PERIPHERAL-Menü

Aus dem Monitor-Grundmenü erreichen Sie das PERIPHERAL-Menü, indem Sie einfach 'p' eingeben (ohne 'Return').

Das PERIPHERAL-Menü bietet die folgenden sechs Möglichkeiten an, um auf periphere Baugruppen zuzugreifen:

3.1 Cass-In (Cassetten-Eingabe)
3.2 Cass-Out (Cassetten-Ausgabe)
3.3 Terminal (Vollduplex-Modem- und DFÜ-Betrieb)
3.4 Uhr (Echtzeit-Uhr auslesen bzw. stellen)
3.5 V.24-In (V.24-Eingabe)
3.6 V.24-Out (V.24-Ausgabe)

Aus dem PERIPHERAL-Menü kommen Sie wie folgt zurück ins Monitor-Grundmenü:

- durch 'TAB' unmittelbar zum Monitor-Kaltstart
  (sofortige Ausgabe des Monitor-Grundmenüs)
- durch 'Escape' bzw. CTL+C jeweils eine Ebene höher im Menü
  (max. zweimalige Betätigung bis zum Monitor-Kaltstart)

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>p

I: Cass-In
O: Cass-Out
T: Terminal
U: Uhr
V: V.24-In
W: V.24-Out

Mp>i 8000,8fff

o.k.

Mp>
```

3.1 I: Cass-In (Cassetten-Eingabe)

Format: Mp>I aaaa,eeee<Ret>

Funktion: Daten über das Cassetten-Interface ins RAM einlesen, beginnend bei Anfangsadresse aaaa bis einschließlich Endadresse eeee.

Zulässig: Für die Adressen aaaa und eeee können alle Werte aus dem RAM-Bereich eingesetzt werden.

Bedienung: Das Magnetband muß an den Anfang des ca.5 s langen Vorspanns gebracht werden, der bei der Magnetband-Aufzeichnung vor dem eigentlichen Datenblock ausgegeben worden ist; innerhalb dieses Vorspanns sind erst das Bandgerät (Stellung 'Wiedergabe') und dann das Einlese-Programm zu starten (durch Betätigen der Return-Taste; vgl. Abschnitt 3.2 'O: Cass-Out').

Voraussetzung: Die Schaltung des Casssetten-Interfaces muß entsprechend der Abgleichanweisung justiert sein; die Vorgehensweise hierfür und die Angaben über Datenformat, Übertragungsrate und Anschlußbelegung sind im Abschnitt 5.3 zusammengefaßt ('Magnetbandgerät (fernsteuerbar)').

Quittung: o.k.

Error # 50: Vorspannfehler; beim Einlesen ist innerhalb von 100 Bytes kein Block von 10 aufeinanderfolgenden 00-Bytes erkannt worden.
Error # 51: Startzeichen-Fehler; die bei der Aufzeichnung ausgegebenen drei Bytes 'D3h' sind nicht erkannt worden.
Error # 52: Das Trennzeichen (00-Byte) nach den drei Startbytes 'D3h' ist nicht erkannt worden.
Error # 55: Einlesefehler; ein Datenbyte ist unvollständig empfangen worden.

```
MOPPEL-Video-Monitor V 7.5
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M'p

I: Cass-In
O: Cass-Out
T: Terminal
U: Uhr
V: V.24-In
W: V.24-Out

Mp>o 9000,9fff

o.k.

Mp>
```

3.2 O: Cass Out (Cassetten-Ausgabe)

Format: Mp>O aaaa,eeee<Ret>

Funktion: Daten über das Cassetten-Interface ausgeben, beginnend bei Anfangsadresse aaaa bis einschließlich Endadresse eeee.

Zulässig: Für die Adressen aaaa und eeee können Werte aus dem gesamten Adreßbereich ØØØØ...FFFFh eingesetzt werden.

Bedienung: Das Magnetband ist an die gewünschte Position zu bringen (Zählwerkstand notieren!) und in Stellung 'Aufnahme' zu starten; erst bei laufendem Band wird das Ausgabe-Programm gestartet (durch Betätigen der Return-Taste).

Ausgabeschema: Vor Beginn der eigentlichen Datenausgabe erzeugt das Programm einen ca.5 s langen Vorspann, bestehend aus einer Serie von ØØ-Bytes, gefolgt von drei Bytes 'D3h' und abgeschlossen mit wiederum einem ØØ-Byte. Dieser Vorspann ermöglicht es, das Einlesen und Zusamensetzen der empfangenen Daten definiert zu beginnen (vgl. Abschnitt 3.1 'I: Cass-In'). Die Angaben über Datenformat, Übertragungsrate und Anschlußbelegung sind im Abschnitt 5.3 zusammengefaßt ('Magnetbandgerät (fernsteuerbar)').

Quittung: o.k.

hms

```
MOPPEL-Terminal  V 15.6  (C) hms'86
Transmit:  CTS  ---  ---
Receive:   DSR  811A  ---
--------------------------------------------------------------------------------

Bitte Return druecken:
XXX XXX XX  XXX XXX
 X  X   X X X X X
 X  XXX X X X X XXX
 X  X   X X XXX   X
 X  XXX XX  X X XXX

Sie sind der 156593. Anrufer!

t = 17
  1 Infos
  2 Briefkasten
  3 mc-Inhalt
  4 Software-Service
  5 Firmen-Adr.
  6 Param.aend.
  7 Ende
Ihre Eingabe:
```

3.3 T: Terminal (Vollduplex-Modem- und DFÜ-Betrieb)

Format: Mp>T iiii,oooo<Ret> oder Mp>T<Ret>

Funktion: Daten-Ein- und -Ausgabe über die V.24-Schnittstelle im Vollduplex-Betrieb vorbereiten. Bei Adresse iiii beginnt der Einlese-Buffer im RAM, und Adresse oooo legt den Beginn des Ausgabe-Buffers fest.

Zusatzfunktion: Wenn die Angabe der Adresse iiii fehlt, setzt der Monitor dafür die RAM-Adresse 8ØØØh ein.

Zulässig: Für die Adresse iiii kann jeder Wert aus dem RAM-Bereich, und für die Adresse oooo kann jeder beliebige Wert aus dem Adreßbereich eingesetzt werden.

Bedienung: An die V.24-Schnittstelle ist entweder ein anderer Computer oder ein Modem für die Datenfernübertragung (DFÜ) anzuschließen (z.B. mit Telefon-Kopplung; vgl. linke Seite: Dialog mit 'TEDAS' vom Franzis-Verlag, München).

Im **Empfangskanal** (Receive) kann mit der Tastenkombination CTL+I der Einlese-Buffer zugeschaltet werden. Sämtliche Eingaben werden dann parallel zur Bildschirm-Anzeige fortlaufend ins RAM geschrieben, von wo aus man sie beispielsweise ausdrucken kann; CTL+U beendet diese Parallelschaltung. Die Zuschaltung beginnt automatisch beim Empfang des Zeichens STX (Start of Text, Ø2h), und sie endet automatisch beim Empfang des Zeichens ETX (End of Text, Ø3h). Bei parallelem Einlese-Buffer erscheint in der Kopfzeile der aktuelle Stand des Eingabe-Buffer-Pointers.

Die Tastenkombination CTL+S veranlaßt die Ausgabe des Zeichens DC3 (=XON Übertragung stoppen, 13h), und bei CTL+Q wird DC1 ausgegeben (=XOF Übertragung fortsetzen, 11h). Beide Signale erscheinen auch in der Kopfzeile.

Im **Sendekanal** (Transmit) kann mit der Tastenkombination CTL+O der Ausgabe-Buffer aktiviert werden. Sämtliche Ausgaben kommen dann nicht mehr von der Tastatur, sondern fortlaufend aus dem RAM, wo sie zuvor zweckmäßigerweise mit dem Editor erstellt worden sind. Das Programm fügt bei jedem Wagenrücklauf (=CR Carriage Return, ØDh) einen Zeilenvorschub ein (=LF Line Feed, ØAh). Die Tastenkombination CTL+U beendet diese automatische Ausgabe. Bei Auftreten des Zeichens ETX (End of Text, Ø3h) geht die Ausgabe selbsttätig wieder auf die Tastatur über. Bei aktiviertem Ausgabe-Buffer erscheint in der Kopfzeile der aktuelle Stand des Ausgabe-Buffer-Pointers.

Beim Empfang des Zeichens DC3 (=XOF, 13h) wird der Ausgabekanal gesperrt, und beim Empfang von DC1 (=XON, 11h) wieder freigegeben. Beide Signale erscheinen auch in der Kopfzeile.

Der Rücksprung ins PERIPHERAL-Menü kann nur durch Betätigen der Tastenkombination CTL+Z erfolgen; in dieser Betriebsart ist kein Warmstart über 'Escape' oder CTL+C möglich.

Übertragungsformat: Die Angaben über Datenformat, Übertragungsrate und Anschlußbelegung sind im Abschnitt 5.4 zusammengefaßt ('Terminal und Modem (Daten-Fernübertragung DFÜ)').

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>p

I: Cass-In
O: Cass-Out
T: Terminal
U: Uhr
V: V.24-In
W: V.24-Out

Mp>u 1715,325,0387
Mi,25.03.87;17:15:00h

Mp>
```

3.4 U: Uhr (Echtzeit-Uhr auslesen bzw. stellen)

Format: Mp>u<Ret> zum Auslesen bzw.
Mp>U ssmm,WKK,MMJJ<Ret> zum Stellen

Funktion Auslesen: Auslesen der von der internen Echtzeit-Uhr gelieferten Uhrzeit- und Datums-Information und Darstellung in folgendem Format: Fr,Ø6.Ø3.87;11:35:31h

Funktion Stellen: Die interne Echtzeit-Uhr wird mit den eingegebenen Daten wie folgt gestellt: ss=Stunden, mm=Minuten (Sekunden automatisch ØØ), W=Wochentag (1=Montag usf.), KK=Kalendertag, MM=Monat, JJ=Jahres-Zehner und -Einer.

Zulässig: Die Stunden werden selbstverständlich im 24-h-Format eingegeben.

Quittung: Nach erfolgtem Stellen wird die Uhrzeit ständig oben rechts in den Bildschirm eingeblendet; diese Einblendung wird erst dadurch wieder gelöscht, daß die dafür benötigte Interrupt-Freigabe verändert wird (durch einen Hardware-RESET oder eine andere Interrupt-Belegung, z.B. durch Aktivierung des Floppy-Disk-Interfaces oder Terminals).

Hinweis: Die Echtzeit-Uhr kann auch in allen anderen Programmen (z.B. im Editor oder unter CP/M) ausgelesen werden, indem man die vorprogrammierte Tastenkombination FCT+U aufruft (vgl. Abschnitt 6.6 'Einsatz der Echtzeit-Uhr').

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>p

I: Cass-In
O: Cass-Out
T: Terminal
U: Uhr
V: V.24-In
W: V.24-Out

Mp>v 8000,8fff

o.k.

Mp>
```

3.5 V: V.24-In (V.24-Eingabe)

Format: Mp>V aaaa,eeee<Ret>

Funktion: Daten über die V.24- oder Stromtreiber-Schnittstelle ins RAM einlesen, beginnend bei Anfangsadresse aaaa bis einschließlich Endadresse eeee.

Zulässig: Für die Adressen aaaa und eeee können alle Werte aus dem RAM-Bereich eingesetzt werden.

Übertragungsformat: Die Angaben über Datenformat, Übertragungsrate und Anschlußbelegung sind im Abschnitt 5.4 zusammengefaßt ('Terminal und Modem (Daten-Fernübertragung DFÜ)').

Quittung: o.k.

```
MOPSEL-Video-Monitor V 2.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>p

I: Cass-In
O: Cass-Out
T: Terminal
U: Uhr
V: V.24-In
W: V.24-Out

Mp>w 9000,9fff

o.k.

Mp>
```

3.6 W: V.24-Out (V.24-Ausgabe)

Format: Mp>W aaaa,eeee<Ret>

Funktion: Daten über die V.24- oder Stromtreiber-Schnittstelle ausgeben, beginnend bei Anfangsadresse aaaa bis einschließlich Endadresse eeee.

Zulässig: Für die Adressen aaaa und eeee können Werte aus dem gesamten Adreßbereich ØØØØ...FFFFh eingesetzt werden.

Übertragungsformat: Die Angaben über Datenformat, Übertragungsrate und Anschlußbelegung sind im Abschnitt 5.4 zusammengefaßt ('Terminal und Modem (Daten-Fernübertragung DFÜ)').

Quittung: o.k.

4. START-Menü

Aus dem Monitor-Grundmenü erreichen Sie das START-Menü, indem Sie einfach 's' eingeben (ohne 'Return').

Das START-Menü bietet die folgenden fünf Möglichkeiten an, um Programme zu starten und ihre Ausführung zu verfolgen:

4.1 Go (Programm im Echtzeit-Betrieb starten)
4.2 Step (Programm im Einzelschritt-Betrieb starten)
4.3 User FØØØh (Programm bei der User-Adresse FØØØh starten)
4.4 Byte (Programm im Einzel-Byte-Betrieb starten)
4.5 Break (Programm bis Breakpoint im Echtzeit-Betrieb starten)

Aus dem START-Menü kommen Sie wie folgt zurück ins Monitor-Grundmenü:

- durch 'TAB' unmittelbar zum Monitor-Kaltstart
  (sofortige Ausgabe des Monitor-Grundmenüs)
- durch 'Escape' bzw. CTL+C jeweils eine Ebene höher im Menü
  (max. zweimalige Betätigung bis zum Monitor-Kaltstart)

```
0001 00C0            ORG 0F000H
0002 0000            *
0003 F000  21 A8 32 DEMO:  LXI  H,32A8H   *Video-Adresse (Bildmitte)
0004 F003  3E 41           MVI  A,41H     *Acc: ASCII-Code für "A"
0005 F005  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0006 F006  23              INX  H
0007 F007  3C              INR  A         *Acc: ASCII-Code für "B"
0008 F008  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0009 F009  23              INX  H
0010 F00A  3C              INR  A         *Acc: ASCII-Code für "C"
0011 F00B  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0012 F00C  C3 03 10        JMP  1003H     *Monitor-Warmstart
```

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>s                                        ABC

G: Go
S: Step
U: User F000h
Y: Byte
Z: Dreal

Ms>g f000

M>
```

4.1 G: Go (Programm im Echtzeit-Betrieb starten)

Format: Ms>G aaaa<Ret> oder Ms>G<Ret>

Funktion: Die CPU-Register einschließlich Stack-Pointer mit den Werten aus dem Register-Buffer im RAM laden, den Cursor ausschalten und die Programmausführung bei Adresse aaaa beginnen.

Zusatzfunktion: Wenn beim Aufruf die Angabe der Adresse aaaa fehlt, setzt der Monitor dafür die unterste RAM-Adresse 2800h ein.

Zulässig: Für die Adresse aaaa kann jeder beliebige Wert aus dem gesamten Adreßbereich 0000...FFFFh eingesetzt werden, allerdings muß an dieser Stelle ein OpCode stehen (erstes Byte eines Mehrwort-Befehls).

Hinweis: Wenn das aufgerufene Programm nicht definiert abgeschlossen ist (z.B. durch einen Sprung zum Monitor-Warmstart), kann die Rückkehr zum Monitor nur durch einen Hardware-RESET erreicht werden. Nach dem RESET werden die Inhalte aller CPU-Register einschließlich Stack-Pointer in den Register-Buffer im RAM übertragen (vgl. Abschnitt 7.4 'Belegung des Anwender- und Monitor-RAMs ab 2800h').

hms

```
0001 0000            ORG 0F000H
0002 0000            *
0003 F000  21 A8 32 DEMO:  LXI  H,32A8H   *Video-Adresse (Bildmitte)
0004 F003  3E 41           MVI  A,41H     *Acc: ASCII-Code für "A"
0005 F005  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0006 F006  23              INX  H
0007 F007  3C              INR  A         *Acc: ASCII-Code für "B"
0008 F008  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0009 F009  23              INX  H
0010 F00A  3C              INR  A         *Acc: ASCII-Code für "C"
0011 F00B  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0012 F00C  C3 03 10        JMP  1003H     *Monitor-Warmstart
```

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>s

G: Go
S: Step
U: User F000h
Y: Byte
Z: Break

Ms>s f000

Adr.Inh.A  B&C  D&E  H&L   SP  SZxcxPxC

F000 21 FF FFFF FFFF FFFF 2F70 00000000
F003 3E FF FFFF FFFF 32A8 2F70 00000000
F005 77 41 FFFF FFFF 32A8 2F70 00000000
F006 23 41 FFFF FFFF 32A8 2F70 00000000
F007 3C 41 FFFF FFFF 32A9 2F70 00000000
F008 77 42 FFFF FFFF 32A9 2F70 00000000 ABC
F009 23 42 FFFF FFFF 32A9 2F70 00000000
F00A 3C 42 FFFF FFFF 32AA 2F70 00000000
F00B 77 43 FFFF FFFF 32AA 2F70 00000000
F00C C3 43 FFFF FFFF 32AA 2F70 00000000


M>
```

4.2 S: Step (Programm im Einzelschritt-Betrieb starten)

Format: Ms>S aaaa<Ret> oder Ms>S<Ret>

Funktion: Die CPU-Register einschließlich Stack-Pointer mit den Werten aus dem Register-Buffer im RAM laden, und die Einzelschritt-Ausführung bei Adresse aaaa beginnen.

Zusatzfunktion: Wenn beim Aufruf die Angabe der Adresse aaaa fehlt, setzt der Monitor dafür die unterste RAM-Adresse 2800h ein.

Zulässig: Für die Adresse aaaa kann jeder beliebige Wert aus dem RAM-Bereich eingesetzt werden, allerdings muß an dieser Stelle ein OpCode stehen (erstes Byte eines Mehrwort-Befehls).

Bedienung: Nach der Darstellung der Register-Inhalte können alle Register einschließlich FLAGs und Stack-Pointer modifiziert werden (Betätigen der HEX-Tasten 0...9 und A...F); der blinkende Cursor gibt die augenblickliche Schreibposition an, die sich mit jeder Eingabe um eine Stelle nach rechts verschiebt. Der Cursor läßt sich auch ohne HEX-Eingabe mit den Tasten 'links' (=CTL+S), 'rechts' (=CTL+D) oder 'TAB' (=2x 'rechts') bewegen.

An der acht Bit-Positionen des FLAG-Registers wird bei Eingaben größer '1' nur das niedrigstwertige Bit des entsprechenden ASCII-Codes berücksichtigt.

Das Ausführen des aktuellen Befehls unter Einbeziehung der angezeigten Register-Inhalte erfolgt durch 'Return' (oder 'Cursor abwärts' =CTL+X) mit automatischem Stop beim nächsten Befehl und erneuter Anzeige der Register-Inhalte.

Der Rücksprung ins MEMORY-Menü (nicht START-Menü!) kann nur durch Betätigen der Taste 'm' erfolgen; in dieser Betriebsart ist kein Warmstart über 'Escape' oder CTL+C möglich.

Einschränkung: Das im RAM stehende Anwender-Programm darf nicht die Befehle 'RST0', 'RST1' oder 'RST2' enthalten, weil diese für die Einzelschritt-Behandlung des Monitors reserviert sind; ferner dürfen im Anwender-Programm keine Sprünge bzw. Unterprogramm-Aufrufe zur jeweils nächstfolgenden Adresse vorkommen, weil der Monitor dort einen Stop einfügt (und beim Weiterschalten natürlich wieder ersetzt).

Hinweis: In der BREAK-Betriebsart kann vor Beginn des Einzelschritt-Betriebs ein bestimmter Programmteil unter Echtzeit-Bedingungen durchlaufen werden (vgl. Abschnitt 4.5).

Error # 01: Cursor aufwärts in dieser Betriebsart gesperrt.
Error # 02: Unzulässige Eingabe (nur HEX, Return oder Cursor).

```
0001 0000            ORG 0F000H
0002 0000            *
0003 F000  21 A9 32 DEMO:  LXI  H,32A9H   *Video-Adresse (Bildmitte)
0004 F003  3E 41           MVI  A,41H     *Acc: ASCII-Code für "A"
0005 F005  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0006 F006  23              INX  H
0007 F007  3C              INR  A         *Acc: ASCII-Code für "B"
0008 F008  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0009 F009  23              INX  H
0010 F00A  3C              INR  A         *Acc: ASCII-Code für "C"
0011 F00B  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0012 F00C  C3 03 10        JMP  1003H     *Monitor-Warmstart
```

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>s                                    ABC

G: Go
S: Step
U: User F000h
Y: Byte
Z: Break

M>u

M>
```

4.3 U: User F000h (Programm bei der User-Adresse F000h starten)

Format: Ms>U ssss,eeee,dddd<Ret>

Funktion: Die auf dem Bildschirm eingegebenen Werte ssss, eeee und dddd in den Parameter-Buffer im RAM laden (vgl. Abschnitt 7.4 'Belegung des Anwender- und Monitor-RAMs ab 2800h'), den Cursor ausschalten und die Programmausführung bei Adresse F000h beginnen (Standard-Beginn eines assemblierten Programms).

Zulässig: Für die Parameter ssss, eeee und dddd kann jeder hexadezimale Wert von 0000...FFFFh eingesetzt werden. Zur Abkürzung kann die Parameter-Eingabe auch ganz entfallen, sie werden dann beim Programmstart auf Null gesetzt.

```
0001 0000            ORG 0F000H
0002 0000            *
0003 F000 21 A8 32 DEMO:  LXI  H,32A8H   *Video-Adresse (Bildmitte)
0004 F003 3E 41           MVI  A,41H     *Acc: ASCII-Code für "A"
0005 F005 77              MOV  M,A       *Acc-Inhalt -> (H&L)
0006 F006 23              INX  H
0007 F007 3C              INR  A         *Acc: ASCII-Code für "B"
0008 F008 77              MOV  M,A       *Acc-Inhalt -> (H&L)
0009 F009 23              INX  H
0010 F00A 3C              INR  A         *Acc: ASCII-Code für "C"
0011 F00B 77              MOV  M,A       *Acc-Inhalt -> (H&L)
0012 F00C C3 03 10        JMP  1003H     *Monitor-Warmstart
```

AdreBbus	Datenbus	OpCode	Maschinen-Zyklus	Bedeutung
1203h	E9h	PCHL	- - -	BYTE-Einsprung
F000h	21h	LXI H	OpCode Fetch	H&L mit Bytes 2&3 laden
F001h	A8h	- -	Memory Read	Zieladresse (lower)
F002h	32h	- -	Memory Read	Zieladresse (upper)
F003h	3Eh	MVI A	OpCode Fetch	Acc mit Byte 2 laden
F004h	41h	- -	Memory Read	Datenbyte "A"
F005h	77h	MOV m	OpCode Fetch	Acc-Inhalt -> ((H&L))
32A8h	41h	- -	Memory Write	Datenbyte ins RAM (32A8h)
F006h	23h	INX H	OpCode Fetch	Reg-Paar H&L plus 1
F007h	3Ch	INR A	OpCode Fetch	Acc plus 1
F008h	77h	MOV m	OpCode Fetch	Acc-Inhalt -> ((H&L))
32A9h	42h	- -	Memory Write	Datenbyte ins RAM (32A9h)
F009h	23h	INX H	...	

4.4 Y: Byte (Programm im Einzel-Byte-Betrieb starten)

Format: Ms>Y aaaa<Ret> oder Ms>Y<Ret>

Funktion: Den Cursor ausschalten und die Programmausführung im Einzel-Byte-Betrieb bei Adresse aaaa beginnen.

Zusatzfunktion: Wenn beim Aufruf die Angabe der Adresse aaaa fehlt, setzt der Monitor dafür die unterste RAM-Adresse 2800h ein.

Zulässig: Für die Adresse aaaa kann jeder beliebige Wert aus dem gesamten Adreßbereich 0000...FFFFh eingesetzt werden, allerdings muß an dieser Stelle ein OpCode stehen (erstes Byte eines Mehrwort-Befehls).

Bedienung: In der BYTE-Betriebsart sind Tastatur und Bildschirm inaktiv. Das Weiterschalten zum nächsten Befehlszyklus kann nur über die STEP-Taste erfolgen (auf der 89er-CPU bzw. auf dem 87er-Einzelschritt-Modul); die Zustandsanzeige von Adreß- und Datenbus sowie Prozessor-Zyklus erfolgt auf der 89er-LED-Einheit bzw. auf dem 87er-Einzelschritt-Modul.

Hinweis: Das Beenden der BYTE-Betriebsart ist ausschließlich über einen Hardware-RESET möglich, weil die Steuerung des Computers vollständig auf die Hardware übergegangen ist. Nach dem RESET werden die Inhalte aller CPU-Register einschließlich Stack-Pointer in den Register-Buffer im RAM übertragen (vgl. Abschnitt 7.4 'Belegung des Anwender- und Monitor-RAMs ab 2800h').

```
0001 0000            ORG 0F000H
0002 0000            *
0003 F000  21 49 32 DEMO:  LXI  H,3249H   *Video-Adresse (Bildmitte)
0004 F003  3E 41           MVI  A,41H     *Acc: ASCII-Code für "A"
0005 F005  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0006 F006  23              INX  H
0007 F007  3C              INR  A         *Acc: ASCII-Code für "B"
0008 F008  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0009 F009  23              INX  H
0010 F00A  3C              INR  A         *Acc: ASCII-Code für "C"
0011 F00B  77              MOV  M,A       *Acc-Inhalt -> (H&L)
0012 F00C  C3 03 10        JMP  1003H     *Monitor-Warmstart
```

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>s

G: Go
S: Step
U: User F000h
Y: Byte
Z: Break

M>z f000,f009

Adr,Inh.A  B&C  D&E  H&L   SP  SZxcxPxC

F009 23 42 000B 00F0 32A9 2F70 00000000
```

AB

4.5 Z: Break (Programm bis Breakpoint im Echtzeit-Betrieb starten)

Format: Ms>Z aaaa,eeee<Ret> oder Ms>Z,eeee<Ret>

Funktion: Programmausführung ab Anfangsadresse aaaa starten, wie unter 'G: Go' beschrieben (vgl. Abschnitt 4.1); ab Adresse eeee in den Einzelschritt-Betrieb übergehen, wie unter 'S: Step' beschrieben (vgl. Abschnitt 4.2).

Zusatzfunktion: Wenn beim Aufruf die Angabe der Adresse aaaa fehlt, setzt der Monitor dafür die unterste RAM-Adresse 2800h ein.

Bedienung und **Einschränkung** wie unter 'G: Go' bzw. 'S: Step' beschrieben (vgl. Abschnitte 4.1 bzw. 4.2), d.h. die Anfangsadresse aaaa kann wahlweise im ROM oder RAM-Bereich liegen, während die Stopadresse eeee zwingend im RAM liegen muß.

5. Peripherie-Anschluß

Die Geräte-Konzeption sieht eine ganze Reihe von Schnittstellen zum Anschluß von Peripheriegeräten vor. Diese Anschlußmöglichkeiten sind vielseitig ausgelegt, um den unterschiedlichsten Anforderungen gerecht zu werden.

Der vorliegende Abschnitt beschreibt die Anschluß- und Betriebsbedingungen für folgende Peripheriegeräte:

5.1 Monitor (Datensichtgerät), BAS- und TTL-Ansteuerung
5.2 Drucker, RS232 (seriell) und Centronics (parallel)
5.3 Magnetbandgerät (fernsteuerbar)
5.4 Terminal und Modem (Daten-Fernübertragung DFÜ)
5.5 Floppy-Disk-Laufwerke

5.1 Monitor (Datensichtgerät), BAS- und TTL-Ansteuerung

Zur Bildschirm-Ausgabe lassen sich handelsübliche Sichtgeräte (Monitore) mit beliebiger Größe und farblicher Darstellung anschließen; hierfür stehen zwei Schnittstellen zur Verfügung, die alternativ, aber auch gleichzeitig belegt werden können (Ansteuerung über das Video-Interface):

Der Anschluß eines Standard-Datensichtgerätes **mit BAS-Eingang** erfolgt an der Cinch-Buchse 'MON' auf der Bus-Platine; hier steht das zusammengesetzte Bild-, Austast- und Synchron-Signal (BAS) mit einem Standard-Pegel von ca.1,4...1,6 Vss zur Verfügung.

Datensichtgeräte **mit reiner TTL-Ansteuerung** werden an der vierpoligen Stiftleiste 'VID' auf der Bus-Platine angeschlossen, die wie folgt beschaltet ist:

Pin	Signal	Bus	Bedeutung	Pegel
1	VID	c11	Video-Signal	(TTL true)
2	VS	c13	Vertikaler Synchron-Impuls	(TTL invers, 125 us)
3	HS	c12	Horizontaler Synchron-Impuls	(TTL true, 5 us)
4	GND	x32	Bezugspotential ('Masse')	

Die 12-V-Stromversorgung für einen TTL-Monitor kann -unter Beachtung der Netzteil-Spezifikationen- in der Regel noch vom Schaltnetzteil übernommen werden.

5.2 Drucker, RS232C (seriell) und Centronics (parallel)

Der Anschluß eines Druckers **mit serieller RS232C-Schnittstelle** erfolgt über die 9polige D-Sub-Buchse 'PRT' auf der Bus-Platine, die wie folgt belegt ist (Ansteuerung über das Universal-Interface; vgl. Abschnitt 6.7 'Drucker-Aktivierung'):

Pin	Signal	Bus	Bedeutung	Drucker-Signal	Pin
2	PRTIN	a19	(Drucker-Bereitmeldung)	Data Terminal Ready	20
3	PRTOT	c16	(Drucker-Ansteuerung)	Received Data RxD	3
5	Ground	b14	(Bezugspotential'Masse')	Signal Ground SG	7

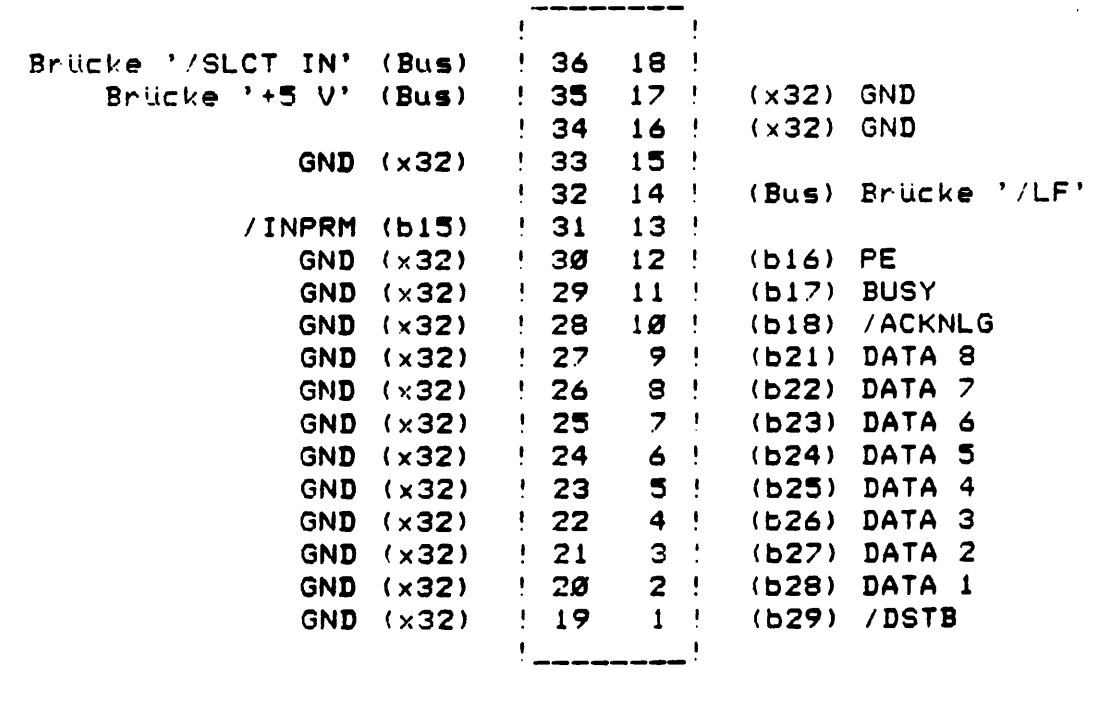
Die Datenübertragung erfolgt mit folgenden Randbedingungen, die im Monitor fest programmiert sind (in Klammern sind die jeweiligen EPROM-Adressen angegeben; identisch mit V.24/20-mA-Ansteuerung! Vgl. Abschnitt 7.7 'System-Konstanten im EPROM'):

Übertragungsformat (005Eh): 1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung, Taktfrequenz 1:1 (Mode: 14h)

Datenrate (000E/000Fh): 4800 Baud

Übertragungsverfahren: Mark (logisch 1): -12 V
Space (logisch 0): +12 V

Der Anschluß eines Druckers mit **paralleler Centronics-Schnittstelle** erfolgt über die 36polige Buchse 'CEN' auf der Bus-Platine, die wie folgt belegt ist ('/': aktiv LOW; Ansteuerung über das Universal-Interface):



5.3 Magnetbandgerät (fernsteuerbar)

Bevor der Datentransfer zwischen Computer und Magnetband-Gerät erfolgen kann, muß das Cassetten-Interface (einmalig) abgeglichen werden; bei fertig gelieferten Baugruppen ist dieser Abgleich bereits herstellerseitig duchgeführt worden.

Für die Dauer des Abgleichs sind auf dem Universal-Interface die Brücken TPØ/GND sowie TP1/GØ zu schließen. Die Software-Abgleich-Routine 'Adjust' ist Bestandteil der Cassetten-Ausgabe-Routine 'O' im PERIPHERAL-Menü (vgl. Abschnitt 3.2). Aus dem Monitor-Grundmenü wird sie wie folgt aufgerufen: 'p' (für 'PERIPHERAL-Menü'), gefolgt von 'O ad' (für 'Cass-Out/Adjust').

Das Potentiometer P2 ist nun so einzustellen, daß sich der angezeigte Meßwert auf ca.3125d stabilisiert; ist dies nicht möglich, ist das zeitbestimmende RC-Glied R23/C4 zu verändern (durch schrittweises Vergrößern bzw. Verkleinern des Widerstandes R23).

Der Bandgeräte-Anschluß erfolgt über die 5polige DIN-Buchse 'CAS' auf der Bus-Platine, die wie folgt belegt ist (Ansteuerung über das Universal-Interface):

Pin	Signal	Bus	Bedeutung
1	COT	a27	Cassette-Out
2	GND	x32	Bezugspotential ('Masse')
3	CIN	c25	Cassette-In
4	REL1	b31	Relais 1
5	REL0	b30	Relais Ø

Die potentialfrei an die Anschlüsse 4 und 5 geführten Relais-Kontakte können bei fernbedienbaren Geräten zur automatischen Motor-Steuerung verwendet werden; beim Starten der Cassetten-Ein- bzw. -Ausgabe-Routine ('I' bzw. 'O' im PERIPHERAL-Menü) werden diese Relais-Kontakte für die Dauer der Übertragung geschlossen.

Die Datenübertragung erfolgt mit folgenden Randbedingungen, die im Monitor fest programmiert sind (in Klammern sind die jeweiligen EPROM-Adressen angegeben):

Übertragungsformat (ØØ5Fh): 1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung, Taktfrequenz 1:1 (Mode: 14h)

Datenrate (ØØ16/ØØ17h): 12ØØ Baud

Übertragungsverfahren: Phasen-Codierung, d.h. Aufzeichnung einer konstanten Frequenz mit Halbwellen-Umtastung.

5.4 Terminal und Modem (Daten-Fernübertragung DFÜ)

Voraussetzung für den Terminal- bzw. Modem-Betrieb ist eine Hardware-Verbindung von Pin 7 des ACIA-Bausteins (6850) zum Anschluß a23 der 64poligen VG-Leiste (/RST75; ggfls. nachverdrahten).

Beim Betrieb des Computers als Terminal (z.B. bei Anschluß eines Telefon-Modems zur Daten-Fernübertragung, DFÜ) oder beim Datentransfer über die V.24- bzw. die 20-mA-Stromtreiber-Schnittstelle (vgl. Abschnitt 4 'PERIPHERAL-Menü') erfolgt die Verbindung zur Peripherie über die 25polige D-Sub-Buchse 'V24' auf der Bus-Platine, die wie folgt belegt ist (Ansteuerung über das Universal-Interface):

Pin	Signal	Bus	Term/DFÜ	V.24-Out	V.24-In	20 mA
1	T[illegible]	b3				* Tx (+)
2	TxD (D1/103)	b5	* TxD	* RxD		
3	RxD (D2/104)	b6	* RxD		* TxD	
4	RTS (S2/105)	b7	* RTS	* CTS		
5	CTS (M2/106)	b8	* CTS		* RTS	
6	DSR (M1/107)	b9	* DSR/CD	* DTR		
7	SG (E2/102)	b10	* SG	* SG	* SG	
8	Rx	b12				* Rx (+)
9	+12 V	a13	xx)			
14	Tx Ret	b4				* Tx Ret
20	DTR (S1)	b11			* DSR	
21	Rx Ret	b13				* Rx Ret

xx) Die 12-V-Verbindung zum Anschluß 9 kann bei Bedarf hergestellt werden, um darüber z.B. ein Modem zu speisen.

Die **V.24/20-mA-Datenübertragung** erfolgt mit folgenden Randbedingungen, die im Monitor fest programmiert sind (in Klammern sind die jeweiligen EPROM-Adressen angegeben; identisch mit Drucker!):

Übertragungsformat (005Eh): 1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung, Taktfrequenz 1:1 (Mode: 14h)

Datenrate (000E/000Fh): 4800 Baud

Übertragungsverfahren: RxD/TxD: 0(=Space)+12 V; 1(=Mark)-12 V
sonstige:0(=Aus) -12 V; 1(=An) +12 V

Die **Datenübertragung beim Modem/DFÜ-Betrieb** erfolgt mit folgenden Randbedingungen, die im Monitor fest programmiert sind (in Klammern sind die jeweiligen EPROM-Adressen angegeben):

Übertragungsformat (0060h): 1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung, Taktfrequenz 1:16, DTR aktiv (Mode: 95h)

Datenrate (001E/001Fh): 4800 (mit CLK/16=300 Baud)

Übertragungsverfahren: RxD/TxD: 0(=Space)+12 V; 1(=Mark)-12 V
sonstige:0(=Aus) -12 V; 1(=An) +12 V

5.5 Floppy-Disk-Laufwerke

Der Floppy-Disk-Controller kann maximal vier Laufwerke ansteuern, die mit einer 34poligen Standard-Schnittstelle ausgestattet sind. Es lassen sich 3-Zoll-, 3.5-Zoll- oder 5.25-Zoll-Laufwerke verwenden, einseitig oder zweiseitig beschreibbar, auch im gemischten Betrieb.

Beim Anschluß mehrerer Laufwerke sollen die Abschlußwiderstände auf der Laufwerk-Elektronik nur in demjenigen Gerät vorhanden sein, das am Ende des Flachband-Kabels liegt; in den dazwischenliegenden Geräten sind diese Widerstände (in der Regel Widerstands-Netzwerke) zu entfernen. Maßgeblich hierfür sind jedoch letztlich die Hersteller-Angaben.

Die Stromversorgung von maximal zwei Mikro-Laufwerken (3 Zoll oder 3.5 Zoll) kann -unter Beachtung der Netzteil-Spezifikationen- in der Regel noch vom Schaltnetzteil übernommen werden.

Die Belegung des vierpoligen **Stromversorgungs-Anschlusses** ist wie folgt genormt:

1: +12 V (c16 am Netzteil-Steckplatz)
2: GND für +12 V
3: GND für +5 V
4: +5 V (a/c20/22/24 am Netzteil-Steckplatz)

Die **Verbindung zu den Laufwerken** wird über die 34polige Stiftleiste auf der Platine des Floppy-Disk-Controllers hergestellt; sie ist -der Norm entsprechend- wie folgt belegt:

GND (x32)	1	2	
GND (x32)	3	4	
GND (x32)	5	6	Drive 3 Select
GND (x32)	7	8	Index Puls
GND (x32)	9	10	Drive 0 Select
GND (x32)	11	12	Drive 1 Select
GND (x32)	13	14	Drive 2 Select
GND (x32)	15	16	Motor On
GND (x32)	17	18	Direction Select
GND (x32)	19	20	Step
GND (x32)	21	22	Write Data
GND (x32)	23	24	Write Gate
GND (x32)	25	26	Track 00
GND (x32)	27	28	Write Protect
GND (x32)	29	30	Read Data
GND (x32)	31	32	Side One
GND (x32)	33	34	

6. Ein- und Ausgaben

Die Kommunikation zwischen Anwender und Computer läuft im Normalfall über Tastatur und Bildschirm ab; dieser Dialog vollzieht sich über die dafür vorgesehenen Unterprogramme CI und CO, deren Eigenschaften und weitergehenden Leistungsmerkmale eingehend beschrieben werden.

Wegen der vielfältigen Bedeutung sind in diesem Zusammenhang der Echtzeit-Uhr-Aufruf und die Drucker-Aktivierung zusammenfassend dargestellt.

6.1 Monitor-Eingabe-Routinen CI und CSTS
6.2 Codierung der ASCII-Tastatur
6.2.1 Codierung der 89er-Tastatur-Erweiterung
6.2.2 Codierung der 87er-Tastatur-Erweiterung

6.3 Monitor-Ausgabe-Routine CO
6.4 Verarbeitung der Bildschirm-Codes
6.5 Escape-Sequenzen
6.6 Einsatz der Echtzeit-Uhr
6.7 Drucker-Aktivierung

6.1 Monitor-Eingabe-Routinen CI und CSTS

Der normale Weg von Eingaben auf der Anwender-Ebene läuft über die Monitor-Unterprogramme 'CI' und 'CSTS' ab (Console Input, Festadresse 0043h und Console Status, Festadresse 0052h; vgl. Abschnitt 7.3 'MOVID-Festadressen und -Unterprogramme'). Das Unterprogramm CI wartet auf das Betätigen einer ASCII-Taste und liefert bei deren Betätigung den ihr zugeordneten ASCII-Code im Register A; Das Unterprogramm CSTS ermittelt nur, ob überhaupt eine Taste betätigt ist oder nicht; im ersten Fall ist beim Rücksprung der Inhalt von Register A=FFh, und das ZERO-FLAG ist off; im anderen Fall (keine Taste aktiv) ist Register A=00h und ZERO ist gesetzt.

Die Tastatur liefert sämtliche Codes im Bereich 00...7Fh; darüber hinaus ermöglicht sie die Aktivierung verschiedener Funktionen. Dazu ist sie funktionell in vier Ebenen unterteilt:

Normal-Ebene: Codes der Kleinbuchstaben bzw. unteren Tastensymbole
SHIFT-Ebene: Codes der Großbuchstaben bzw. oberen Tastensymbole
Control-Ebene: Codes 00...1Fh und fehlende ASCII-Codes 27h/60h
Funktions-Ebene: Pro Taste vier frei programmierbare Codes

Im folgenden Abschnitt ist die Zuordnung der ASCII-Codes zu den einzelnen Tasten tabellarisch zusammengestellt. Hierzu sind folgende Erläuterungen angebracht: Die Taste 'SHIFT LOCK' schaltet die gesamte Tastatur auf die SHIFT-Ebene um bzw. von dort zurück auf die Normal-Ebene; ihre Betätigung wird mit der internen Schnarre quittiert. Unabhängig von der jeweiligen Ebene ist die Escape-Taste (links neben der überbreiten Leertaste 'Blank'); sie liefert immer den Steuercode '1Bh' (er wird auch bei der Tasten-Kombination CTL+DEL erzeugt). Die mit CTL+7...CTL+4 zu- bzw. abschaltbare Drucker-Aktivierung ist im Abschnitt 6.7 beschrieben.

In der Funktions-Ebene sind zwei Funktionen fest programmiert: Bei Betätigung von FCT+U erfolgt das Auslesen der Echtzeit-Uhr, und bei FCT+V wird der gesamte Bildspeicher ins RAM kopiert (beginnend bei Adresse F800h); er kann von dort beispielsweise für einen Ausdruck abgerufen werden ('Hard-Copy' der Bild-Darstellung). Bei den übrigen Kombinationen von Funktions- plus ASCII-Taste werden aus dem RAM-Bereich 2B00...2C3Fh jeweils vier aufeinanderfolgende Codes abgerufen, die vom Anwender dort abgelegt werden müssen (vgl. Tabelle). Eine Sicherungsmöglichkeit für diese Codes ist im Abschnitt 7.5 beschrieben ('Disketten-Datensicherung').

Bei diesen Codes besteht hinsichtlich der Belegung völlige Freiheit. Es kann sich dabei also um 1...4 beliebige ASCII-Zeichen handeln oder auch um eine mit '1Bh' eingeleitete Escape-Sequenz; über diesen Weg ist eine ganze Reihe weiterer Funktionen aktivierbar, u.a. das Ein- und Ausschalten der Invers-Darstellung, die Cursor-Positionierung oder die direkte Drucker-Ansteuerung (vgl. Abschnitt 6.5 'Escape-Sequenzen').

Wenn die Code-Folge mit 'C3h' beginnt, erfolgt in der Funktions-Ebene ein direkter Programmsprung zu derjenigen Adresse, die als zweiter und dritter Code nach 'C3h' folgt (erst untere, dann obere Adreßhälfte). Auf diese Weise kann einer Funktionstaste beispielsweise auch ein Unterprogramm-Aufruf zugeordnet werden, der wiederum eine Reihe weiterer Abläufe veranlaßt.

6.2 Codierung der ASCII-Tastatur

norm.	SHIFT	CTL = Abk.	FCT-Buffer	FCT-Codes Ø;1;2;3 =Bedeutung
a 61h	A 41h	Ø1h = SOH	2B4Ø...43h	.. h;.. h;.. h;.. h=.....
b 62h	B 42h	Ø2h = STX	2B3Ø...33h	.. h;.. h;.. h;.. h=.....
c 63h	C 43h	Ø3h = ETX	2B1C...1Fh	.. h;.. h;.. h;.. h=.....
d 64h	D 44h	Ø4h = EOT	2B18...1Bh	.. h;.. h;.. h;.. h=.....
e 65h	E 45h	Ø5h = ENQ	2B2Ø...23h	.. h;.. h;.. h;.. h=.....
f 66h	F 46h	Ø6h = ACK	2B7C...7Fh	.. h;.. h;.. h;.. h=.....
g 67h	G 47h	Ø7h = BEL	2B2C...2Fh	.. h;.. h;.. h;.. h=.....
h 68h	H 48h	Ø8h = BS	2B54...57h	.. h;.. h;.. h;.. h=.....
i 69h	I 49h	Ø9h = HT	2C24...27h	.. h;.. h;.. h;.. h=.....
j 6Ah	J 4Ah	ØAh = LF	2BØ4...Ø7h	.. h;.. h;.. h;.. h=.....
k 6Bh	K 4Bh	ØBh = VT	2C1C...1Fh	.. h;.. h;.. h;.. h=.....
l 6Ch	L 4Ch	ØCh = FF	2CØ8...ØBh	.. h;.. h;.. h;.. h=.....
m 6Dh	M 4Dh	ØDh = CR	2BØ8...ØBh	.. h;.. h;.. h;.. h=.....
n 6Eh	N 4Eh	ØEh = SO	2B58...5Bh	.. h;.. h;.. h;.. h=.....
o 6Fh	O 4Fh	ØFh = SI	2C1Ø...13h	.. h;.. h;.. h;.. h=.....
p 7Øh	P 5Øh	1Øh = DLE	2BFC...FFh	.. h;.. h;.. h;.. h=.....
q 71h	Q 51h	11h = DC1	2B48...4Bh	.. h;.. h;.. h;.. h=.....
r 72h	R 52h	12h = DC2	2B84...87h	.. h;.. h;.. h;.. h=.....
s 73h	S 53h	13h = DC3	2B68...6Bh	.. h;.. h;.. h;.. h=.....
t 74h	T 54h	14h = DC4	2B34...37h	.. h;.. h;.. h;.. h=.....
u 75h	U 55h	15h = NAK	Fest-Fkt.:	Echtzeit-Uhr auslesen
v 76h	V 56h	16h = SYN	Fest-Fkt.:	Video-RAM kopieren: F8ØØh
w 77h	W 57h	17h = ETB	2B7Ø...73h	.. h;.. h;.. h;.. h=.....
x 78h	X 58h	18h = CAN	2B6C...6Fh	.. h;.. h;.. h;.. h=.....
y 79h	Y 59h	19h = EM	2B44...47h	.. h;.. h;.. h;.. h=.....
z 7Ah	Z 5Ah	1Ah = SUB	2B5C...5Fh	.. h;.. h;.. h;.. h=.....
ä 7Bh	Ä 5Bh	ØØh = NUL	2BEØ...E3h	.. h;.. h;.. h;.. h=.....
ö 7Ch	Ö 5Ch	1Dh = GS	2BF4...F7h	.. h;.. h;.. h;.. h=.....
ü 7Dh	Ü 5Dh	1Eh = RS	2BE8...EBh	.. h;.. h;.. h;.. h=.....
ß 7Eh	? 3Fh	1Fh = US	2BDC...DFh	.. h;.. h;.. h;.. h=.....
20h	20h	ØØh = NUL	2BE4...E7h	.. h;.. h;.. h;.. h=.....
Ø 3Øh	= 3Dh	ØØh = NUL	2BFØ...F3h	.. h;.. h;.. h;.. h=.....
1 32h	! 21h	I/O-FLAG1	2B3C...3Fh	.. h;.. h;.. h;.. h=.....
2 32h	" 22h	I/O-FLAG2	2B64...67h	.. h;.. h;.. h;.. h=.....
3 33h	§ 40h	I/O-FLAG3	2B14...17h	.. h;.. h;.. h;.. h=.....
4 34h	$ 24h	I/O-FLAG4	2B78...7Bh	.. h;.. h;.. h;.. h=.....
5 35h	% 25h	I/O-FLAG5	2B28...2Bh	.. h;.. h;.. h;.. h=.....
6 36h	& 26h	I/O-FLAG6	2B5Ø...53h	.. h;.. h;.. h;.. h=.....
7 37h	/ 2Fh	I/O-FLAG7	2BØØ...Ø3h	.. h;.. h;.. h;.. h=.....
8 38h	(28h	ØØh = NUL	2C18...1Bh	.. h;.. h;.. h;.. h=.....
9 39h	) 29h	ØØh = NUL	2CØ4...Ø7h	.. h;.. h;.. h;.. h=.....
# 23h	^ 5Eh	ØØh = NUL	2BCC...CFh	.. h;.. h;.. h;.. h=.....
+ 2Bh	* 2Ah	ØØh = NUL	2BC8...CBh	.. h;.. h;.. h;.. h=.....
- 2Dh	_ 5Fh	6Øh = `	2BF8...FFh	.. h;.. h;.. h;.. h=.....
. 2Eh	: 3Ah	1Ch = FS	2CØC...ØFh	.. h;.. h;.. h;.. h=.....
, 2Ch	; 3Bh	27h = '	2C2Ø...23h	.. h;.. h;.. h;.. h=.....
< 3Ch	> 3Eh	ØØh = NUL	2BD4...D7h	.. h;.. h;.. h;.. h=.....
DEL Ø8h	7Fh	1Bh = ESC	2BB4...B7h	.. h;.. h;.. h;.. h=.....
TAB Ø9h	Ø9h	Ø3h = ETX	2BCØ...C3h	.. h;.. h;.. h;.. h=.....
RET ØDh	ØDh	ØDh = CR	2BB8...BBh	.. h;.. h;.. h;.. h=.....
ESC 1Bh	1Bh	1Bh = ESC	2BDØ...D3h	.. h;.. h;.. h;.. h=.....

6.2.1 Codierung der 89er-Tastatur-Erweiterung

norm.	= SHIFT	CTL = Abk.	FCT-Buffer	FCT-Codes Ø;1;2;3	=Bedeutung
Curs.re.	Ø4h	ØØh = NUL	2C14...17h	.. h;.. h;.. h;.. h=	....
Curs.auf.	Ø5h	ØØh = NUL	2C28...2Bh	.. h;.. h;.. h;.. h=	....
Curs hom.	11h	ØØh = NUL	2B1Ø...13h	.. h;.. h;.. h;.. h=	....
Curs.li.	13h	ØØh = NUL	2BC4...C7h	.. h;.. h;.. h;.. h=	....
Curs.abw.	18h	ØØh = NUL	2C ØØ...Ø3h	.. h;.. h;.. h;.. h=	....
* (MULT)	2Ah	ØØh = NUL	2B38...3Bh	.. h;.. h;.. h;.. h=	....
+ (PLUS)	2Bh	ØØh = NUL	2B24...27h	.. h;.. h;.. h;.. h=	....
- (MINUS)	2Dh	ØØh = NUL	2B88...8Bh	.. h;.. h;.. h;.. h=	....
. (Komma)	2Eh	ØØh = NUL	2B4C...4Fh	.. h;.. h;.. h;.. h=	....
/ (DIVI)	2Fh	ØØh = NUL	2B74...77h	.. h;.. h;.. h;.. h=	....
? (PRINT)	3Fh	ØØh = NUL	2B6Ø...63h	.. h;.. h;.. h;.. h=	....
Ret.x	ØDh	ØDh = CR	2BB8...BBh	.. h;.. h;.. h;.. h=	....
Ø x	3Øh	ØØh = NUL	2B4Ø...43h	.. h;.. h;.. h;.. h=	....
1 x	31h	ØØh = NUL	2B3Ø...33h	.. h;.. h;.. h;.. h=	....
2 x	32h	ØØh = NUL	2B1C...1Fh	.. h;.. h;.. h;.. h=	....
3 x	33h	ØØh = NUL	2B18...1Bh	.. h;.. h;.. h;.. h=	....
4 x	34h	ØØh = NUL	2B2Ø...23h	.. h;.. h;.. h;.. h=	....
5 x	35h	ØØh = NUL	2B7C...7Fh	.. h;.. h;.. h;.. h=	....
6 x	36h	ØØh = NUL	2B2C...2Fh	.. h;.. h;.. h;.. h=	....
7 x	37h	ØØh = NUL	2B54...57h	.. h;.. h;.. h;.. h=	....
8 x	38h	ØØh = NUL	2C24...27h	.. h;.. h;.. h;.. h=	....
9 x	39h	ØØh = NUL	2B04...07h	.. h;.. h;.. h;.. h=	....

x : Belegung identisch mit der 89er-ASCII-Tastatur

6.2.2 Codierung der 87er-Tastatur-Erweiterung

norm.	= SHIFT	CTL = Abk.	FCT-Buffer	FCT-Codes Ø;1;2;3	=Bedeutung
Curs.re.	Ø4h	ØØh = NUL	2BD8...DBh	.. h;.. h;.. h;.. h=	....
Curs.auf.	Ø5h	ØØh = NUL	2C3C...3Fh	.. h;.. h;.. h;.. h=	....
Curs.hom.	11h	ØØh = NUL	2B9C...9Fh	.. h;.. h;.. h;.. h=	....
Curs.li.	13h	ØØh = NUL	2BC4...C7h	.. h;.. h;.. h;.. h=	....
Curs.abw.	18h	ØØh = NUL	2BBØ...B3h	.. h;.. h;.. h;.. h=	....
.	2Eh	ØØh = NUL	2C34...37h	.. h;.. h;.. h;.. h=	....
Ret.	ØDh	ØDh = CR	2B94...97h	.. h;.. h;.. h;.. h=	....
Ø	3Øh	ØØh = NUL	2BA8...ABh	.. h;.. h;.. h;.. h=	....
1	31h	ØØh = NUL	2BA4...A7h	.. h;.. h;.. h;.. h=	....
2	32h	ØØh = NUL	2C3Ø...33h	.. h;.. h;.. h;.. h=	....
3	33h	ØØh = NUL	2B9Ø...93h	.. h;.. h;.. h;.. h=	....
4	34h	ØØh = NUL	2BAC...AFh	.. h;.. h;.. h;.. h=	....
5	35h	ØØh = NUL	2C38...3Bh	.. h;.. h;.. h;.. h=	....
6	36h	ØØh = NUL	2B98...9Bh	.. h;.. h;.. h;.. h=	....
7	37h	ØØh = NUL	2BAØ...A3h	.. h;.. h;.. h;.. h=	....
8	38h	ØØh = NUL	2C2C...2Fh	.. h;.. h;.. h;.. h=	....
9	39h	ØØh = NUL	2B8C...8Fh	.. h;.. h;.. h;.. h=	....

6.3 Monitor-Ausgabe-Routine CO

Der normale Weg von Ausgaben auf der Anwender-Ebene läuft über das Monitor-Unterprogramm 'CO' ab (Console Out, Festadresse 0049h; vgl. Abschnitt 7.3 'MOVID-Festadressen und -Unterprogramme'). Dabei wird der Code des auszugebenden Zeichens im CPU-Register C an das Unterprogramm CO übergeben, und es erfolgt dann die Darstellung des betreffenden Zeichens auf dem Bildschirm. Die jeweilige Bildposition bestimmt der blinkende Cursor, dessen Bildschirm-Absolutadresse in den RAM-Zellen UPDATE steht (vgl. Abschnitt 7.4 'Belegung des Anwender- und Monitor-RAMs ab 2300h').

Auf diese Weise passieren nur die ASCII-Codes 20...7Eh die Ausgabe-Routine CO, d.h. nur diese Codes führen direkt zu einer Darstellung auf dem Bildschirm; für die übrigen Codes (<20h und >7Eh) gelten folgende Einschränkungen:

Beim Auftreten des Codes '1Bh' leitet die CO-Routine eine Steuersequenz ein, d.h. sie prüft, ob die unmittelbar folgenden Zeichen einer der im Abschnitt 6.5 beschriebenen Escape-Sequenzen entsprechen; ist dies der Fall, wird die betreffende Funktion ausgeführt, ohne daß die beteiligten Zeichen auf dem Bildschirm erscheinen.

Die Codes 04h, 05h, 11h, 12h und 13h sind bei den ROM-residenten Programmen für die Cursor-Steuerung vorgesehen (WordStar-kompatibel); sie entsprechen den Tasten-Kombinationen CTL+D (=rechts), CTL+E (=aufwärts), CTL+Q (=home), CTL+S (=links) und CTL+X (=abwärts). Der Code 09h (=TAB) führt zur Ausgabe von 8 Leerzeichen (Blanks), und 07h (=BELL) aktiviert die interne Schnarre.

Folgende fünf Steuercodes beeinflussen die Bild-Darstellung: 08h (=Back Space, BS) und 7Fh (=Delete, DEL) löschen das zuletzt eingegebene Zeichen, 0Dh (=Carriage Return, CR) bringt den Cursor zurück an den Zeilenanfang, 0Ah (=Line Feed, LF) veranlaßt einen Zeilensprung (in der untersten Zeile verbunden mit dem Hochschieben des gesamten Bildes) und 0Ch schließlich löscht den gesamten Bildschirm (=Form Feed, FF).

Alle übrigen Codes <20h werden ohne weitere Reaktion zurückgewiesen. Bei Codes im Bereich 80...FFh wird generell das oberste Bit (MSB) unterdrückt, d.h. sie werden zu 00...7Fh. Das MSB bleibt für die Bildschirm-Inversdarstellung vorbehalten: Bei der Ausgabe wird jedes Zeichen mit dem Inhalt der RAM-Zelle 'INVERS' ODER-verknüpft (Adresse 2FC9h; vgl. Abschnitt 7.4, 'Belegung des Anwender- und Monitor-RAMs ab 2300h'). Bei INVERS=80h erscheint das Zeichen invers, bei INVERS=00h normal dargestellt.

Nach jeder Bildschirm-Ausgabe testet die CO-Routine die I/O-FLAGs in RAM-Adresse 2FC8h; ist eins der vier oberen Bits auf HIGH, wird der Ausgabe-Code auch noch an die entsprechene Drucker-Schnittstelle übergeben (vgl. Abschnitt 6.7 'Drucker-Aktivierung').

Wirkung und Verarbeitung der Bildschirm-Codes sind im folgenden Abschnitt tabellarisch zusamengestellt.

6.4 Verarbeitung der Bildschirm-Codes

00h	CTL+	x		20h		40h	§	60h	' CTL -
01h	CTL+A	x		21h	!	41h	A	61h	a
02h	CTL+B	x		22h	"	42h	B	62h	b
03h	CTL+C	x		23h	#	43h	C	63h	c
04h	CTL+D	(V)	Cursor rechts	24h	$	44h	D	64h	d
05h	CTL+E	(V)	Cursor aufwärts	25h	%	45h	E	65h	e
06h	CTL+F	x		26h	&	46h	F	66h	f
07h	CTL+G	(C)	Bell	27h	' CTL ,	47h	G	67h	g
08h	CTL+H	(P)	Back Space	28h	(	48h	H	68h	h
09h	CTL+I	(V)	TAB	29h	)	49h	I	69h	i
0Ah	CTL+J	(P)	Line Feed	2Ah	*	4Ah	J	6Ah	j
0Bh	CTL+K	x		2Bh	+	4Bh	K	6Bh	k
0Ch	CTL+L	(P)	Form Feed	2Ch	,	4Ch	L	6Ch	l
0Dh	CTL+M	(P)	Return	2Dh	-	4Dh	M	6Dh	m
0Eh	CTL+N	x		2Eh	.	4Eh	N	6Eh	n
0Fh	CTL+O	x		2Fh	/	4Fh	O	6Fh	o
10h	CTL+P	x		30h	0	50h	P	70h	p
11h	CTL+Q	(V)	Cursor home	31h	1	51h	Q	71h	q
12h	CTL+R	x		32h	2	52h	R	72h	r
13h	CTL+S	(V)	Cursor links	33h	3	53h	S	73h	s
14h	CTL+T	x		34h	4	54h	T	74h	t
15h	CTL+U	x		35h	5	55h	U	75h	u
16h	CTL+V	x		36h	6	56h	V	76h	v
17h	CTL+W	x		37h	7	57h	W	77h	w
18h	CTL+X	(V)	Cursor abwärts	38h	8	58h	X	78h	x
19h	CTL+Y	x		39h	9	59h	Y	79h	y
1Ah	CTL+Z	x		3Ah	:	5Ah	Z	7Ah	z
1Bh	Escape	(E)		3Bh	;	5Bh	Ä	7Bh	ä
1Ch	CTL+'.'	x		3Ch	<	5Ch	Ö	7Ch	ö
1Dh	CTL+Ö	x		3Dh	=	5Dh	Ü	7Dh	ü
1Eh	CTL+Ü	x		3Eh	>	5Eh	^	7Eh	ß
1Fh	CTL+ß	x		3Fh	?	5Fh	_	7Fh	DEL

x: keine Ausgabe
(C): nur Steuerfunktion
(E): leitet Escape-Sequenz ein (vgl. Abschnitt 6.5)
(P): Bildschirm- und Drucker-Ausgabe (soweit Paralleldruck)
(V): nur Bildschirm-Ausgabe

6.5 Escape-Sequenzen

Vor jeder Zeichenausgabe über die Monitor-Routine CO (Console Out, Startadresse ØØ49h) wird zunächst geprüft, ob eine Escape-Sequenz eingeleitet werden soll, die mit dem Steuerzeichen 1Bh (=ESC) beginnt. Folgt bei der Ausgabe über CO auf das Steuerzeichen 1Bh unmittelbar einer der Codes ØE...1Dh, führt der Monitor anstelle einer Ausgabe die entsprechende, unten erläuterte Funktion durch.

Die Zeichenübergabe an die Routine CO erfolgt im CPU-Register C; je nach Anzahl der erforderlichen Steuercodes (2 bzw. 4) ist das Register C nacheinander mit den entsprechenen Codes zu laden, jeweils gefolgt vom anschließenden Aufruf der Routine CO.

Wenn zum Beispiel der Cursor in Zeile 1Ø (=ØAh) und Spalte 5 positioniert werden soll, sind nacheinander die Codes 1Bh, 16h, ØAh und Ø5h an CO zu übergeben, was in Maschinensprache folgendermaßen abläuft:

```
MVI  C,1Bh          *Escape-Sequenz einleiten
CALL 49h            *CO aufrufen
MVI  C,16h          *2.ESC-Code: Cursor positionieren
CALL 49h            *CO aufrufen
MVI  C,ØAh          *3.ESC-Code: Zeile 1Ød=ØAh
CALL 49h            *CO aufrufen
MVI  C,Ø5h          *4.ESC-Code: Spalte 5
CALL 49h            *CO aufr. und Funktion ausführen
```

Wenn nur zwei Codes für die betreffende Funktion erforderlich sind, erfolgt das Laden von Reg C und Aufrufen von CO dementsprechend nur zweimal.

Mögliche Escape-Sequenzen und ihre Funktionen:

```
1Bh ØEh          Inversdarstellung ausschalten
1Bh ØFh          Inversdarstellung einschalten
1Bh 1Øh          Bildschirm ganz löschen
1Bh 11h          Cursor in HOME-Position bringen (linke obere Ecke)
1Bh 12h          Cursor ausschalten
1Bh 13h          Cursor einschalten
1Bh 14h          Bild ab 5.Zeile um eine Zeile nach oben rollen
1Bh 15h          Bild ab 5.Zeile um eine Zeile nach unten rollen
1Bh 16h ZZh SSh  Cursor positionieren (ZZ:Zeile, SS:Spalte)
1Bh 17h          Bildschirm ab Cursor-Position löschen
1Bh 18h          Bildzeile ganz löschen (Zeile der Cursor-Position)
1Bh 19h          Bildzeile ab Cursor-Position löschen
1Bh 1Ah LLh UUh  Programmsprung zur Adresse UU(upper)/LL(lower)
1Bh 1Bh          Echtzeit-Uhr auslesen und darstellen
1Bh 1Ch iih IOh  Drucker init.(ii=ØØ:nein)/I/O-FLAG mit IO laden
1Bh 1Dh XXh YYh  'XX' und 'YY' direkt an den Drucker ausgeben
```

Hinweis: Weitere Angaben zum Einsatz der Echtzeit-Uhr finden Sie im Abschnitt 6.6 und zur Drucker-Aktivierung im Abschnitt 6.7.

6.6 Einsatz der Echtzeit-Uhr

Die auf der Zentraleinheit integrierte Echtzeit-Uhr bietet jederzeit die Möglichkeit, Uhrzeit und Datum abzurufen; wegen der Bedeutung dieser Funktion für die reine Anzeige, die Dokumentation und sogar Zeitmessung sind im Monitor-Programm verschiedene Routinen enthalten, die die Nutzung dieser Möglichkeiten unterstützen.

Aufruf per Funktions-Taste: Bei Betätigung der Tastenkombination FCT+U wird die Echtzeit-Uhr ausgelesen und als ASCII-Zeichenfolge an der aktuellen Cursor-Position dargestellt. Dieser Aufruf läßt sich in allen Programmen aktivieren (z.B. auch in BASIC oder unter CP/M) und genauso weiterverarbeiten, als handele es sich um eine Anwender-Eingabe von der Tastatur (vgl. Abschnitt 6.2 'Codierung der ASCII-Tastatur').

Aufruf per Monitor-Funktion: Im PERIPHERAL-Menü des Video-Monitors genügt die Eingabe von 'u', um die Echtzeit-Uhr auszulesen und darzustellen; anschließend wird die fortlaufende Einblendung von Uhrzeit und Datum in die rechte obere Ecke des Bildschirms aktiviert (vgl. Abschnitt 3.4 'U: Uhr').

Aufruf per BASIC-Befehl: Der ROM-residente BASIC-Interpreter bietet mit der Anweisung 'TI$' die Möglichkeit, auf die Echtzeit-Uhr zuzugreifen; die daraufhin gelieferte ASCII-Zeichenfolge kann mit allen Befehlen der String-Verarbeitung kombiniert werden (vgl. Abschnitt 2.2 'B: BASIC').

Aufruf per Escape-Sequenz: Wenn die beiden Steuerzeichen 1Bh, 1Bh nacheinander an die Ausgabe-Routine CO übergeben werden, wird das Auslesen der Echtzeit-Uhr veranlaßt. Dies geschieht auch, wenn in einer ASCII-Zeichenfolge, die per Unterprogramm STRING ausgegeben wird, die aufeinanderfolgenden Zeichen 1Bh, 1Bh auftauchen (vgl. Abschnitt 6.5 'Escape-Sequenzen').

Aufruf per Unterprogramm (1): Das Monitor-Unterprogramm RDEZU (Read EZU; Festadresse bei 1Ø1Bh) veranlaßt das Auslesen der Echtzeit-Uhr mit Bildschirm-Darstellung und aktiviert anschließend die fortlaufende Einblendung von Uhrzeit und Datum in die rechte obere Ecke des Bildschirms (vgl. Abschnitt 7.3 'MOVID-Festadressen und -Unterprogramme').

Aufruf per Unterprogramm (2): Das Monitor-Unterprogramm TIME (Festadresse bei ØØ61h) veranlaßt das Auslesen der Echtzeit-Uhr ohne Bildschirm-Darstellung. Die ASCII-Zeichenfolge von Uhrzeit und Datum steht anschließend im RAM ab Adresse 2FEAh, abgeschlossen mit der End-Kennzeichnung ØØh, und das Registerpaar H&L ist mit der String-Anfangsadresse 2FEAh geladen (vgl. Abschnitt 7.3 'MOVID-Festadressen und -Unterprogramme').

Hinweis: Während der Bildschirm-Einblendung der Uhrzeit werden die RAM-Zellen 'SECOND' im Sekundentakt hochgezählt (16-Bit-Sekundenzähler; Adresse 2FDA/DBh, vgl. Abschnitt 7.4). Eine Änderung der Interrupt-Verwaltung beendet diese Funktion wieder (z.B. beim Zugriff auf ein Disketten-Laufwerk).

6.7 Drucker-Aktivierung

Nachdem ein Zeichen auf dem Bildschirm erscheint, prüft die dafür vorgesehene Monitor-Ausgabe-Routine CO, ob dieses Zeichen auch an den Drucker ausgegeben werden soll (Drucker-Parallel-Schaltung). Maßgebend hierfür ist der **Zustand der I/O-FLAGs** (RAM-Adresse 2FC8h, vgl. Abschnitt 7.4 'Belegung des Anwender- und Monitor-RAMs ab 2800h'). Wenn eins der vier oberen Bits der I/O-FLAGs auf HIGH ist, wird parallel zur Bildschirm-Ausgabe auch der Drucker angesteuert. Dabei existiert folgende Zuordnung (vgl. Abschnitt 5.2 'Drucker-Anschluß'):

Bit 7 -> Serielle Drucker-Ansteuerung (Bus-Anschluß 'PRT')
Bit 6 -> Parallele Drucker-Ansteuerung (Bus-Anschluß 'CEN')
Bit 5 -> Parallele Drucker-Ansteuerung (Parallel-Interface 8255)
Bit 4 -> Thermodrucker-Ansteuerung (Bus-Anschluß 'PAD')

Die Umschaltung dieser Bits erfolgt über die Tastenkombination CTL+7 (für I/O-FLAG 7)...CTL+4 (für I/O-FLAG 4); gleichzeitig werden die Drucker-Schnittstellen automatisch initialisiert. Diese Umschaltung kann zu jedem Zeitpunkt erfolgen, also auch während laufender Listings (z.B. bei Textausgaben oder einem Assembler-Durchlauf). Dazu ist wie folgt zu verfahren: Stoppen der fortlaufenden Bildschirm-Ausgabe durch Betätigen der Leertaste ('Blank'), Betätigen der gewünschten Tastenkombination CTL+7...CTL+4 und Freigabe der weiteren Ausgabe durch erneute Betätigung der Leertaste. Nach diesem Schema ist sinngemäß zum Abschalten des Paralleldrucks zu verfahren. Die bei jeder Änderung der I/O-FLAGs ausgegebene Bildschirm-Meldung 'I/O-FLAGs: xx-0000' wird nicht mitgedruckt.

Um einen **Bildschirm-Abzug** anzufertigen, ist wie folgt zu verfahren: Kopieren des gesamten Bildspeichers ins RAM ab Adresse F800h (festprogrammierte Tastenkombination FCT+V; vgl. Abschnitt 6.2 'Codierung der ASCII-Tastatur'); Parallelschaltung des gewünschten Drucker-Kanals (CTL+7...CTL+4, s.o.); Aktivierung der Monitor-Betriebsart 'TEXT' und Ausgabe des RAM-Inhalts ab Adresse F800h (MEMORY-Menü, T f800<Return>).

Die separate **Initialisierung der Drucker-Schnittstellen** ist folgendermaßen möglich: Aufruf des Programms LOINIT (Festadresse 0058h; vgl. Abschnitt 7.3 'MOVID-Festadressen und -Unterprogramme') oder Ausgabe der Escape-Sequenz 1Bh, 1Ch, iih, IOh. Bei dieser Sequenz erfolgt die Initialisierung der Drucker-Schnittstellen, wenn das Byte 'iih' ungleich Null ist; gleichzeitig wird das Byte 'IOh' in die I/O-FLAGs übertragen (entspricht der Drucker-Parallelschaltung über die Tastenkombinationen CTL+7...CTL+4).

Mit der **Escape-Sequenz** 1Bh, 1Dh, XXh, YYh werden die beiden Bytes 'XXh' und 'YYh' direkt an den Drucker ausgegeben (ohne Bildschirm-Darstellung). Dies ist insbesondere für Drucker-Steuerzeichen von Bedeutung, die die normale Ausgabe-Routine CO umgehen sollen.

Ähnliches leistet der **BASIC-Befehl** PPRINT x,y. Er bewirkt die direkte Ausgabe der beiden Variablen 'x' und 'y' an den Drucker; bei y=255d (=FFh) wird die Ausgabe von y unterdrückt. Die Initialisierung der Drucker-Schnittstellen erfolgt bereits automatisch beim BASIC-Aufruf, braucht also vom Anwender nicht besorgt zu werden.

7. System-Parameter

Um die Transparenz der System-Konzeption zu erhöhen und dem Anwender die Möglichkeit zu geben, vorhandene Monitor-Unterprogramme zu verwenden, sind im vorliegenden Abschnitt Software- und Hardware-Belegungen zusammengefaßt.

Bevor in diesem Bereich Änderungen vorgenommen werden, sind mögliche Auswirkungen in allen Richtungen sorgfältig zu untersuchen.

7.1 Belegung des Adreßraums (Memory Map)
7.2 HEXMO-Festadressen und -Unterprogramme
7.3 MOVID-Festadressen und -Unterprogramme
7.4 Belegung des Anwender- und Monitor-RAMs ab 2800h
7.5 Disketten-Datensicherung
7.6 Modifikation des Video-Zeichengenerators
7.7 System-Konstanten im EPROM
7.8 Busbelegung
7.9 Belegung der Portadressen

7.1 Belegung des Adreßraums (Memory Map)

Nach dem Einschalten (und nach jedem Monitor-Kaltstart) sind in der unteren Hälfte des Mikroprozessor-Adreßraums (0000...7FFFh) die ROM-residenten Systemprogramme aktiv, während die gesamte obere Hälfte als RAM (8000...FFFFh) zur Verfügung steht. Beim Laden des Disketten-orientierten Betriebssystems CP/M werden die ROM-residenten Programme ausgeblendet, und der gesamte 64-KByte-Adreßraum ist als RAM-Arbeits-speicher verfügbar.

Im Normalbetrieb ist der Speicher wie folgt belegt (Warmstart: Aufruf des betreffenden Programms mit Übernahme aller vorherigen Einstellun-gen, keine Neu-Initialisierung):

0000h: HEXMO; Monitor für HEX-Tastatur und SSG-Anzeige
0003h: HEXMO-Warmstart
übrige HEXMO-Festadressen: siehe Abschnitt 7.2
1000h: MOVID; Monitor für ASCII-Tastatur und Video-Ausgabe
1003h: MOVID-Warmstart
übrige MOVID-Festadressen: siehe Abschnitt 7.3
2000h: FLOPPY; Floppy-Disk-Utilities (Hilfsbank 0, s.u.)
THERMO; Thermodrucker-Routinen (Hilfsbank 1, s.u.)
EXTERN; Anwender-Festprogramm (Hilfsbank 2, s.u.)
PROMER; EPROM-Utilities (Hilfsbank 3, s.u.)
2003h: FLOPPY/THERMO/PROMER-Warmstart
2006h: Block schreiben (bei FLOPPY)
2009h: Block lesen (bei FLOPPY)
200Ch: Disk-0-Aufruf und Programmstart (bei FLOPPY)
2800h: Anwender- und Monitor-RAM (Belegung siehe Abschnitt 7.4)
3000h: Video-RAM; Bildspeicher
3800h: Video-Zeichengenerator; ROM: lesen; RAM: schreiben
4000h: BASIC; BASIC-Interpreter
4003h: BASIC-Programmstart per 'run'
4006h: I/O-Byte für BASIC-Drucker-Ansteuerung
4007h: Externen Eingabe-Kanal aktivieren; H&L: Buffer-Pointer
6000h: ASSEMB; Assembler
6003h: ASSEMB-Warmstart
6C00h: DISASS; Disassembler
7000h: EDITOR; Zeilen-Editor
7003h: EDITOR-Warmstart
700Ah: Hexadezimal-Umsetzung und Ausgabe von H&L (dezimal)
700Dh: Dezimal-Umsetzung und Ausgabe von H&L (hexadezimal)
7AF0h: ASSORT; Assembler-Symboltabelle alphabetisch sortieren
7C90h: PROM16; Ansteuerung des 87er-EPROM-Programmiermoduls

Der 2-K-Adreßbereich 2000...27FFh ist vierfach belegt (8-K-EPROM); die Selektion des gewünschten Programmteils erfolgt über die beiden obersten Bits des Adreß-Latches auf dem Floppy-Disk-Controller (Port-Adresse $40h).

Hilfsbank 0 Select: 00h -> $40h (EPROM-Adresse 0000...07FFh)
Hilfsbank 1 Select: 40h -> $40h (EPROM-Adresse 0800...0FFFh)
Hilfsbank 2 Select: 80h -> $40h (EPROM-Adresse 1000...17FFh)
Hilfsbank 3 Select: C0h -> $40h (EPROM-Adresse 1800...1FFFh)

7.2 HEXMO-Festadressen und -Unterprogramme

Die Festadressen im Bereich 0000...009Fh sind mit definierten Sprungzielen belegt, deren Funktion unabhängig von der jeweiligen Monitor-Version stets erhalten bleibt. Bei Einsatz der beschriebenen Unterprogramm in Anwender-Programmen müssen eventuell benutzte Register-Inhalte vom Anwender gerettet werden.

0000h: HEXMO-Kaltstart
0003h: HEXMO-Warmstart
000Bh: DELY1; Laufzeit-Unterprogramm 1 ms
0013h: DLY100; Laufzeit-Unterprogramm 100 ms
0018h: RESTART-3-Einsprung (Sprung ins RAM zu Adresse 2F8Fh)
001Bh: ONESEC; Laufzeit-Unterprogramm 1 s
0020h: RESTART-4-Einsprung (Sprung ins RAM zur Adresse 2F92h)
0024h: TRAP-Einsprung (8085; von FLOPPY benutzt <PCHL>)
0028h: RESTART-5-Einsprung (Sprung ins RAM zur Adresse 2F95h)
002Ch: RESTART-5.5-Einsprung (von FLOPPY benutzt <JMP 2F98h>)
0030h: RESTART-6-Einsprung (Sprung ins RAM zur Adresse 2F9Bh)
0034h: RESTART-6.5-Einsprung (von EZU benutzt <JMP 2F9Eh>)
0038h: RESTART-7-Einsprung (Sprung ins RAM zur Adresse 2FA1h)
003Ch: RESTART-7.5-Einsprung (von TERMINAL benutzt <JMP 2FA4h>)
003Fh: HEXMO-Versions-Nr. 5x
0040h: ADRSET Adreß-Mode einstellen und Adresse laden
0066h: TRAP-Einsprung (NSC800; von FLOPPY benutzt <PCHL>)
0067h: CNVBYT Inhalt von Reg A in den Siebensegment-Code umsetzen
006Ah: RELES auf Loslassen einer HEX-Taste warten
006Dh: PRESS auf Betätigung einer HEX-Taste warten
0070h: HHKEY HEX-Tastatur einlesen
0073h: CCKEY CMD-Tasten einlesen
0076h: DATSET Daten-Mode einstellen
007Fh: SAVE CPU-Register in den Register-Buffer des RAMs retten
0082h: RESTOR CPU-Register laden und Programm starten
0094h: EZUDSP Uhrzeit in der SSG-Anzeige darstellen
0097h: DSPACC Inhalt von Register A in der SSG-Anzeige darstellen
009Ah: DSPH&L Inhalt von Reg-Paar H&L in der SSG-Anzeige darst.
009Dh: DSPERR Error-Anzeige

7.3 MOVID-Festadressen und -Unterprogramme

Die Festadressen im Bereich 0043...0063h und 1000...1047h sind mit definierten Sprungzielen belegt, deren Funktion unabhängig von der jeweiligen Monitor-Version stets erhalten bleibt. Bei Einsatz der beschriebenen Unterprogramm in Anwender-Programmen müssen eventuell benutzte Register-Inhalte vom Anwender gerettet werden.

0043h: CI; Console Input (ASCII-Tastatur abfragen)
Auf Tastendruck warten und Tasten-Code in Reg A übergeben
0049h: CO; Console Output (Video-Ausgabe)
Inhalt von Reg C ausgeben, bei IOFLAG=on Paralleldruck
004Fh: LO; Lister Output (Drucker-Ausgabe)
Inhalt von Reg C ausdrucken (keine Video-Ausgabe)
0052h: CSTS; Console Status (Status der ASCII-Tastatur ermitteln)
No Key: Reg A=00h, ZERO=on; Key: Reg A=FFh, ZERO=off
0058h: LOINIT; LO initialisieren
Drucker-Schnittstellen für die Ausgabe vorbereiten
0061h: TIME; Echtzeit-Uhr auslesen
ASCII-String ab 2FEAh ins RAM (mit 00h beenden); H&L=2FEAh
1000h: MOVID-Kaltstart
Neu-Initialisierung aller Parameter
1003h: MOVID-Warmstart
Übernahme aller vorgenommenen Einstellungen
100Fh: BREAK; Programmlauf stoppen, fortsetzen oder abbrechen
No Key: RETURN; Blank: Stop/Continue; Escape/CTL+C: Warmstart
1012h: PARIN; Parameter vom Bildschirm in Parameter-Buffer holen
H&L: auf Blank hinter Params positioniert; Reg C: Par-Länge
1015h: HEXTST; ASCII-Zeichen im Reg A umsetzen in den HEX-Code
Fehler: ZERO=on (ASCII-Zeichen war nicht 0...9/A...F)
101Bh: RDEZU; Echtzeit-Uhr aktivieren
Uhrzeit/Datum darstellen und ständig oben rechts einblenden
1021h: STRING; ASCII-Zeichenfolge ausgeben
Endzeichen: 00h (wird nicht mehr ausgegeben)
1027h: ERROR; Fehlermeldung ausgeben
Fehlercode in Reg A; Ausgabe: Zeilenanfang plus 40d Zeichen
102Ah: UNDRLN; ASCII-Zeichenfolge invers ausgeben
wie STRING, aber mit Ein/Ausschaltung des Invers-FLAGs
102Dh: CLRVID; Bildschirm löschen
identisch mit der Ausgabe von 0Ch an CO (=Form Feed)
1030h: CLRLIN; Zeile löschen
betrifft die Zeile, in der der Cursor augenblicklich steht
1033h: CRSPOS; Cursor positionieren
H&L nach UPDATE laden (Bildschirm-Absolutadresse)
1036h: CRSON; Cursor einschalten
Position beibehalten
1039h: CRSOFF; Cursor ausschalten
Position beibehalten
103Ch: ADROT; H&L hexadezimal ausgeben
vierstellige Bildschirmdarstellung von H&L
103Fh: BYTOT; Reg A hexadezimal ausgeben
zweistellige Bildschirmdarstellung von Reg A
1042h: BYTBIN; Reg A binär ausgeben
achtstellige Bildschirmdarstellung von Reg A
1045h: REGSOT; RAM-Register-Buffer darstellen
Darstellung der CPU-Register-Inhalte nach dem letzten RESET

7.4 Belegung des Anwender- und Monitor-RAMs ab 2800h

Das RAM im Adreßbereich 2800...2FFFh wird zum Teil vom Monitor sowie von anderen Programmen benutzt (vgl. Abschnitt xyz). Dies ist vom Anwender zu beachten, wenn er dort Daten ablegt.

Unabhängig von der Benutzung einiger Bereiche dieses RAMs sind die darin enthaltenen Daten Batterie-gepuffert, d.h. sie bleiben auch nach Abschalten der Stromversorgung mehrere Wochen lang erhalten (je nach Ladezustand des Akkus auf der Zentraleinheit).

2800...29FFh: Buffer-Bereich für den CP/M-Urloader

2A00...2AFFh: Buffer-Bereich für CP/M-System-Parameter

2B00...2C3Fh: Buffer für Funktions-Ebene der ASCII-Tastatur

2C40...2CFFh: Unbenutzter Bereich von 160d Bytes

2D00...2DFFh: Buffer für den Disketten-Batch-Betrieb

2E00...2E1Fh: Buffer für Thermodrucker

2E20...2FFFh: vom Monitor belegt

2E4Eh: HORDSP Zeichen pro Zeile (50h=80d oder 28h=40d)

2E4Fh: VERDSP Zeilenanzahl (18h=24d oder 14h=20d)

2E56h: UPDATE Cursor-Position (Bildschirm-Absolutadresse)

2F00h: STACK Beginn des Monitor-Stacks (Stack-Top)

2F70h: USRSTK Beginn des Anwender-Stacks (Stack-Top)

2F8Ch: INTBUF Interrupt-Buffer (EPROM-Sprungziele)

2FA7h: REGBUF Register-Buffer für CPU-Register nach RESET

2FB2h: SSGBUF Siebensegment-Buffer für SSG-Anzeige

2FBBh: ADRES Adresse für Programmstart/RAM-Modifikationen

2FBFh: SRCBEG Parameter-Buffer (erster Param.: Source Begin)

2FC1h: SRCEND Parameter-Buffer (zweiter Param.: Source End)

2FC3h: DSTBEG Parameter-Buffer (dritter Param.: Destin.-Begin)

2FC8h: IOFLAG Aktivierung des Paralleldrucks bei Ausgaben

2FC9h: INVERS Inversdarstellung bei INVERS=80h, normal bei 00h

2FDAh: SECOND Sekundenzähler; aktiv bei Uhrzeit-Einblendung

2FEAh: ASCBUF Uhrzeit-Buffer; ASCII-String nach dem EZU-Auslesen

7.5 Disketten-Datensicherung

Bei der Belegung der Funktionstasten oder beim Erstellen bzw. Ändern des Video-Zeichengenerators fallen Datenmengen an, die mit großem Zeitaufwand verbunden sind; dementsprechend kommt der Sicherung und schnellen Abrufbarkeit dieser Daten eine große Bedeutung zu, was von den Monitor-Programmen folgendermaßen unterstützt wird:

Die Floppy-Disk-Utilities besitzen bei Adresse 200Ch einen Einsprung, der unmittelbar den Batch-0-Vorspann einliest und das darin spezifizierte Programm startet; dieser Sprung zur Adresse 200Ch läßt sich beispielsweise auf eine Funktionstaste legen, indem man der gewünschten Funktionstaste die Code-Folgen C3h, 0Ch, 20h und 00h zuordnet ('C3h' für 'Sprung', 0Ch/20h für die Zieladresse und 00h als vierten Code ohne Auswirkung; vgl. Abschnitt 6.1 'Monitor-Eingabe-Routinen CI und CSTS'). Auf diese Weise erreicht man durch eine einzige Tasten-Kombination das Einlesen und Starten eines Programms über die Disk-Io-Funktion Nr.00.

Es ist vorteilhaft, auch den CP/M-Urloader (Eigen-Start-Programm für das Betriebssystem 'CP/M') auf diese Weise per Funktionsaufruf zu starten; im Zusammenhang mit diesem Urloader werden außerdem folgende Daten geladen:

Von Spur 0, Sektor 4: System-Parameter ins RAM 2A00...2AFFh
(werden immer geladen; 2AE0/E1h bestimmen weiteres Vorgehen)
Von Spur 0, Sektoren 5...6: Funktions-Codes ins RAM 2B00...2CFFh
(werden nur geladen, wenn RAM-Inhalt 2AE0h ungleich 00h ist)
Von Spur 0, Sektoren 9...16: Video-Zeichengenerator ins RAM 3800h
(wird nur geladen, wenn RAM-Inhalt 2AE1h ungleich 00h ist)

Aus diesen Zusammenhängen resultiert folgende Vorgehensweise für die oben angesprochene Datensicherung:

1. Jede Belegung oder Änderung der Funktions-Codes (im RAM-Bereich 2800...2C3Fh) sollte auf Spur 0, Sektor 5&6 einer CP/M-Diskette gesichert werden: RAMb,RAMe>2B,2C; Write Dst-Drv,Trk,Sec>0A,00,05.

2. Jede Belegung oder Änderung des Video-Zeichengenerators (vgl. Abschnitt 7.6 'Modifikation des Video-Zeichengenerators') sollte auf Spur 0, Sektoren 9...16 einer CP/M-Diskette gesichert werden: RAMb,RAMe>80,87; Write Dst-Drv,Trk,Sec>0A,00,09.

3. Die beiden RAM-Zellen 2AE0/E1h der System-Parameter bestimmen, ob beim späteren Auto-Start der Diskette auch die Funktions-Codes und/oder ein modifizierter Zeichengenerator geladen werden sollen. Dazu ist wie folgt vorzugehen: System-Parameter von Spur 0, Sektor 4 ins RAM laden (nach 2A00...2AFFh; RAMb,RAMe>2A,2A; Read Src-Drv,Trk,Sec>0A,00,04), die RAM-Zellen 2AE0/E1h wie gewünscht modifizieren und RAM-Inhalt 2A00...2AFFh zurück auf Spur 0, Sektor 4 derselben Diskette schreiben: RAMb,RAMe>2A,2A; Write Dst-Drv,Trk,Sec>0A,00,04.

7.6 Modifikation des Video-Zeichengenerators

Die Konzeption des Video-Interfaces ermöglicht es, einen im RAM definierbaren Zeichengenerator zu aktivieren; der kann ganz oder teilweise andere Zeichen enthalten als die im EPROM gelieferten Standard-Zeichen.

Um diese individuellen Änderungen vorzunehmen, ist wie folgt zu verfahren:

1. Einlesen des Standard-Video-Zeichengenerators ins RAM ab 8000h (Spur 0, Sektoren 9...16 jeder 60-K-CP/M-Diskette ab Version 6): RAMb,RAMe>80,87; Read Src-Drv,Trk,Sec>0A,00,09.
2. Modifikation des/der gewünschten Zeichen(s) im RAM und Rückladen des RAM-Bereichs 8000...87FFh auf Spur 0, ab Sektor 9: RAMb,RAMe>80,87; Write Dst-Drv,Trk,Sec>0A,00,09.
3. Aktivieren des neuen Zeichengenerators beim CP/M-Urloader (vgl. vorigen Abschnitt 7.5 'Disketten-Datensicherung')

Nach dem Laden des Standard-Zeichengenerators ins RAM geben das zweite und dritte Digit der RAM-Adresse den ASCII-Code desjenigen Zeichens an, dessen Video-Codierung bei dieser Adresse beginnt. Beispiel: Der ASCII-Code für das Zeichen 'G' ist 47h; der zugehörige Video-Code beginnt daher bei Adresse 8470h. Er umfaßt stets zwölf Bytes (für die zwölf untereinander liegenden Bildzeilen eines Zeichens), im vorliegenden Fall also die Adressen 8470... 847Bh. In jedem Byte bestimmt ein HIGH-Bit, ob der betreffende Bildpunkt in der jeweiligen Bildzeile aktiv ist (d.h. leuchtet) oder nicht.

Für das Zeichen 'G' ergibt sich im Standard-Zeichensatz folgender Inhalt des Video-Zeichengenerators (nach dem Laden ins RAM):

```
8470h:   - - - - - - - -    = 00h     Beginn des Video-Codes "G"
8471h:   - - * * * * - -    = 3Ch
8472h:   - * - - - - * -    = 42h
8473h:   - * - - - - - -    = 40h
8474h:   - * - - - - - -    = 40h
8475h:   - * - - * * * -    = 4Eh
8476h:   - * - - - - * -    = 42h
8477h:   - * - - - - * -    = 42h
8478h:   - * - - - - * -    = 42h
8479h:   - - * * * * * -    = 3Eh
847Ah:   - - - - - - - -    = 00h
847Bh:   - - - - - - - -    = 00h
847Ch:   . . . . . . . .    = xxh     wird nicht aktiviert
847Dh:   . . . . . . . .    = xxh           -  "  -
847Eh:   . . . . . . . .    = xxh           -  "  -
847Fh:   . . . . . . . .    = xxh           -  "  -
8480h:   - - - - - - - -    = 00h     Beginn des Video-Codes "H"
8481h:   - * - - - - * -    = 42h
...
```

7.7 System-Konstanten im EPROM

In einigen EPROM-Zellen sind Parameter programmiert, die das System-Verhalten bestimmen. Auch hier sind vor etwaigen Änderungen die Auswirkungen in allen Richtungen sorgfältig zu untersuchen.

FQUARZ (0006h): Quarzfrequenz in MHz
Standard-Einstellung: 06h (=6 MHz CPU-Takt)

TIMPRL (0007h): Tastatur-Prellzeit in ms
Standard-Einstellung: 0Ah (=10 ms für Entprellen; Änderung problemlos möglich)

V24BDR (000Eh): V.24-Baudrate (Identisch mit Drucker)

PRTBDR (000Eh): Drucker-Baudrate (Identisch mit V.24)
Standard-Einstellung: 4800h (=4800 Baud)

CASBDR (0016h): Cassetten-Baudrate
Standard-Einstellung: 1200h (=1200 Baud)

DFÜBDR (001Eh): DFÜ-/Modem-Baudrate
Standard-Einstellung: 4800h (=4800/16=300 Baud)

V24MOD (005Eh): V.24-Betriebsart (Identisch mit Drucker)

PRTMOD (005Eh): Drucker-Betriebsart (Identisch mit V.24)
Standard-Einstellung: 14h (=1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung)

CASMOD (005Fh): Cassetten-Betriebsart
Standard-Einstellung: 14h (=1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung)

DFÜMOD (0060h): DFÜ-/Modem-Betriebsart
Standard-Einstellung: 95h (=1 Startbit, 8 Datenbits, 1 Stopbit, keine Paritätsprüfung, Taktfrequenz geteilt durch 16, DTR bereit)

hms

7.8 Busbelegung ('/': Aktiv LOW)

c1 **Spannung +5 V**	b1 **Spannung +5 V**	a1 **Spannung +5 V**
c2 Datenbit DØ	b2 /Bell =BEL=	a2 Datenbit D5
c3 Datenbit D7	b3 Tx (V24)#	a3 Datenbit D6
c4 Datenbit D2	b4 TxRet (V24)#	a4 Datenbit D3
c5 AdreBbit AØ	b5 TxD (V24)	a5 Datenbit D4
c6 AdreBbit A3	b6 RxD (V24)	a6 AdreBbit A2
c7 AdreBbit A1	b7 RTS (V24)	a7 AdreBbit A4
c8 AdreBbit A8	b8 CTS (V24)	a8 AdreBbit A5
c9 AdreBbit A7	b9 DSR (V24)	a9 AdreBbit A6
c1Ø Control ALE	b1Ø GND (V24)	a1Ø Control READY
c11 TTL-Vid.VID (VID)	b11 DTR (V24)	a11 Control /HOLD
c12 Hor.Sync.HS (VID)	b12 Rx (V24)#	a12 AdreBbit A18*
c13 Ver.Sync.VS (VID)	b13 RxRet (V24)#	a13 **Spannung +12 V**
c14 Datenbit D1	b14 GND (PRT)	a14 Video-BAS (MON)
c15 **Spannung -12 V**	b15 /INPRM (CEN)	a15 **Spannung +8 V**
c16 Print.PRTOT (PRT)	b16 PE (CEN)	a16 Control INTA
c17 AdreBbit A11	b17 BUSY (CEN)	a17 AdreBbit A17*
c18 AdreBbit A1Ø	b18 /ACKNLG (CEN)	a18 AdreBbit A14
c19 AdreBbit A16*	b19 /INO $XC (KEY)	a19 Print.PRTIN (PRT)
c2Ø Control /TRAP	b2Ø /OUT $XØ (KEY)	a2Ø Control S1
c21 Control /INTR	b21 DATA 8 (CEN)	a21 Control /RST55
c22 Control /WR	b22 DATA 7 (CEN)	a22 Control /RST65
c23 Control SØ	b23 DATA 6 (CEN)	a23 Control /RST75
c24 Control /RD	b24 DATA 5 (CEN)	a24 Control RFSH
c25 Cas.Inp.CIN (CAS)	b25 DATA 4 (CEN)	a25 Decode /MEM3X
c26 Control /RESOUT	b26 DATA 3 (CEN)	a26 Decode /BANKX
c27 AdreBbit A12	b27 DATA 2 (CEN)	a27 Cas.Out.COT (CAS)
c28 AdreBbit A15	b28 DATA 1 (CEN)	a28 **Spannung +25 V**
c29 Control /CLK	b29 /DSTB (CEN)	a29 AdreBbit A13
c3Ø Control IO//M	b3Ø Relais Ø (CAS)	a3Ø AdreBbit A9
c31 Control /RESIN	b31 Relais 1 (CAS)	a31 Control /HLDA
c32 Bezugspegel GND	b32 Bezugspegel GND	a32 Bezugspegel GND

Axx*: Zusätzliches AdreBbit zur Bankumschaltung
=BEL=: Bus-Hardware
(xxx): Bus-Steckverbinder bzw. -Buchse
(V24)#: 2Ø-mA-Signal

hms

7.9 Belegung der Portadressen

x0h	-/O	SSG-Anzeigen/LED-Zeile		LED	Adressen/Daten/LEDs
x4h	-/O	Bankumschaltung	'175	CPU	Adre8bits A16*...A18*
xCh	I/-	Spaltenleitungen	'138	ASC	CTL/SFT/FCT/5 Daten
08h	I/O	Daten (8 Bits)		PRM	EPROM-Programmiermodul
09h	-/O	Adressen (lower)		PRM	- " -
0Ah	-/O	Adressen (upper)		PRM	- " -
0Bh	-/O	Control		PRM	- " -
18h	I/O	Daten (4 Bits)	5832	CPU	Echtzeit-Uhr
19h	-/O	Adressen/Steuerdaten	5832	CPU	- " -
28h	-/O	Register-Adressierung	6845	VID	Video-Steuerbaustein
29h	-/O	Daten (8 Bits)	6845	VID	- " -
2Ah	I/O	Control	'74/368	VID	Format/Blanking
38h	-/O	Daten (4 Bits)/Control		THO	Thermodrucker
39h	I/O	Daten (6 Bits)/Control		THO	- " -
40h	-/O	Adre8-Latch	'374	FDC	Hilfsbank 2000h
48h	I/O	Status(IN)/Contr.(OUT)	1770	FDC	FDC-Steuerbaustein
49h	I/O	Spur-Register	1770	FDC	- " -
4Ah	I/O	Sektor-Register	1770	FDC	- " -
4Bh	I/O	Datenregister (8 Bits)	1770	FDC	- " -
88h	I/O	Daten (8 Bits)		PAR	Parallel-Interface
98h	I/O	Daten (8 Bits)		PAR	- " -
A8h	I/O	Daten (8 Bits)		PAR	- " -
B8h	I/O	Daten (8 Bits)		PAR	- " -
C8h	I/O	Daten (8 Bits)		PAR	- " -
D8h	I/O	Daten (8 Bits)		PAR	- " -
E8h	I/O	Daten (8 Bits)		PAR	- " -
F8h	I/O	Daten (8 Bits)		PAR	- " -
89h	I/O	Daten (8 Bits)	8253	UIF	Zähler 0
99h	I/O	Daten (8 Bits)	8253	UIF	Zähler 1
A9h	-/O	Mode-Register	'374	UIF	Mux/Control
B9h	-/O	Schnarre ('Bell')		UIF	BEL (BUS)
C9h	I/O	Daten (8 Bits)	8253	UIF	Zähler 3
D9h	-/O	Control	8253	UIF	Zähler 0/1/2/3
E9h	I/O	Status(IN)/Contr.(OUT)	6850	UIF	V24/PRT/CAS (BUS)
F9h	I/O	Daten (8 Bits)	6850	UIF	- " -
AAh	I/O	Daten(IN)/Contr.(OUT)	0844	UIF	A/D-Umsetzer
BAh	-/O	Strobe-Impuls		UIF	CEN (BUS)
CAh	I/O	Control (IN)/Daten (OUT)		UIF	- " -
DAh	-/O	Daten (8 Bits)	0832	UIF	D/A-Umsetzer
BBh	I/O	Interrupt-Masken-Register		NSC	NSC800-INTR-Register

MOPPEL
System-Software

Assembler- und Disassembler-Handhabung

Modulares Prozessor-Programm zum Entwickeln und Lernen

® **Eingetragenes Warenzeichen**

hms

Inhaltsverzeichnis

Ø. Installation und Aufruf
Hardware-Voraussetzungen und Software-Aktivierung

1. Speicher-Organisation
Reservierte Speicherbereiche und zugehörige AdreBregister

2. Fehlermeldungen
Zusammenstellung aller Assembler-Fehlermeldungen

3. Aufbau des Quell-Listings
Zeilen-Organisation und Einteilung in Felder

4. Pseudo-Operatoren
Beeinflussung der Assembler-Aktivitäten durch Direkt-Kommandos

5. Symboltabelle anlegen bzw. erweitern: Pass #1 bzw. Pass #Ø
Zuordnung von symbolischen Namen und Absolutadressen

6. Objekt-Code erzeugen: Pass #2
Übersetzen des Quellprogramms in den Maschinen-Code

7. Dokumentation ausgeben: Pass #3
Listing von Quellprogramm, Maschinen-Code und Absolutadressen

8. Disassembler-Aufruf
Direkter Sprung zum Disassembler

9. Editor-Aufruf
Direkter Sprung zum Editor

1Ø. Floppy-Aufruf
Direkter Sprung zu den Floppy-Routinen

11. HEX-Mode einstellen
Zahlendarstellung in hexadezimaler Schreibweise veranlassen

12. In/Out on/off
Umschaltung auf externe Ein/Ausgabe-Kanäle

13. Monitor-Rücksprung
Direkter Rücksprung zum Monitor

14. OCT-Mode einstellen
Zahlendarstellung in oktaler Schreibweise veranlassen

15. Symboltabelle ausgeben
Symbolische Namen und zugehörige Absolutadressen listen

hms

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>o

A: Assembler
B: BASIC
D: Disassembler
E: Editor
F: Floppy
P: Prommer
X: Ext.2000h

Mo>a

MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 0/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>
```

Ø. Installation und Aufruf

Das ROM-residente Assembler-Programm ist Bestandteil des EPROMs 'blau'; es wird wahlweise im 4-K-EPROM (2732) oder 8-K-EPROM (2764) geliefert und auf Platz #6 der Speicherkarte eingesetzt (Brücke 'Type 64/32' entsprechend setzen).

Bei der Programmentwicklung übernimmt der Assembler die Aufgabe, ein im Assembler-Code (mnemotechnische Abkürzungen) vorliegendes Quellprogramm in den Maschinen-Code zu übersetzen, damit es vom Mikrocomputer ausgeführt werden kann. Der MOPPEL-8Ø85-Assembler ist ein sogenannter Drei-Pass-Assembler, d.h. der Vorgang der Umsetzung in den Maschinen-Code vollzieht sich normalerweise in drei Durchläufen (Passes; vgl. Fallbeispiel neben Abschnitt 3). Wesentliches Leistungsmerkmal jedes Assemblers ist die Verarbeitung von symbolischen Namen (Label), die für Operanden und Sprungziele verwendet werden können. Die Programme werden dadurch losgelöst von absoluten Zahlenwerten, wodurch sie sich jederzeit an jeden beliebigen Adreßbereich anpassen lassen.

Das Erstellen und Modifizieren des Quellprogramms geschieht mit Hilfe des Editors, der den Computer mit den dazu erforderlichen komfortablen Scheibmaschinen-Funktionen ausstattet (u.a. Einfügen und Löschen von Zeichen oder Zeilen). Sie sollten darüber hinaus auch die Unterstützung nutzen, die Ihnen das ROM-residente BASIC bietet, z.B. bei der Drucker-Ansteuerung im Zusammenhang mit der Programm-Dokumentation.

Vom Monitor-Grundmenü aus erfolgt der Assembler-Aufruf über das OUTWARD-Menü (Eingabe von 'o' ohne 'Return'), gefolgt von 'a' (wiederum ohne 'Return'). Der Assembler-Aufruf kann außerdem direkt aus dem Editor- oder Floppy-Menü heraus erfolgen. Sämtliche Assembler-Anweisungen können in Klein- oder Großschreibung erfolgen, Parameter-Eingaben sind nicht erforderlich.

Das Programm zeigt seine Eingabe-Bereitschaft durch das Blinken des Cursors an; bei ausgeschaltetem Cursor erfolgen interne Verarbeitungen, die keine Eingaben des Anwenders zulassen.

Für den praktischen Einsatz ist es sehr vorteilhaft, daß der Drucker jederzeit (auch bei laufenden Listings im Pass #3!) zu- oder abgeschaltet werden kann; zur Drucker-Ansteuerung ist das Universal-Interface erforderlich.

Auch der Assembler unterstützt zwei Bildformate, die abhängig sind von der Einstellung des Video-Interfaces: 24 Zeilen mit je 8Ø Zeichen oder 2Ø Zeilen mit je 4Ø Zeichen. Beim kleinen Bildformat steht am rechten Zeilenrand entsprechend weniger Platz für Kommentare zur Verfügung. Im übrigen sei auf die ausführlichen Beschreibungen des System-Handbuchs verwiesen.

hms

ØØ11 ORG EQU 2C4Øh (oder ORG 2C4Øh)
==========

Der Pseudo-Operator 'ORG' lädt den Operanden '2C4Øh' in den Pseudo-Programmzähler PSEUPC (Speicherzellen 2FØØ/Ø1h) und löscht gleichzeitig den Versatz in OFFSET (vgl. Abschnitt 4 'Pseudo-Operatoren').

ØØ12 OFS EQU 3C4Øh (oder OFS 3C4Øh)
==========
Der Pseudo-Operator 'OFS' lädt den Operanden '3C4Øh' in das Offset-Register OFFSET (Speicherzellen 2F6Ø/61h); dieser Versatz wird im Pass #2 zum Objekt-Pointer OBJPOI addiert (vgl. Abschnitt 4 'Pseudo-Operatoren').

Text-Pointer
==========
Der Text-Pointer TXTPOI ist ein 16-Bit-AdreBregister (Speicherzellen 2F62/63h), das im Quellprogramm ein Zeichen nach dem anderen adressiert.

Objekt-Pointer
==========
Der Objekt-Pointer OBJPOI ist ein 16-Bit-AdreBregister (Speicherzellen 2F5E/5Fh), das beim Ablegen des Maschinen-Codes im Pass #2 ein Byte nach dem anderen adressiert.
---

1. Speicher-Organisation

Die für den Assembler erforderlichen Randbedingungen werden beim Erstellen der Quellprogramme mit Hilfe des Editors bereits berücksichtigt. Einige grundlegende Organisationsformen müssen Sie allerdings im Auge behalten, damit Ihnen keine unliebsamen Überraschungen passieren (z.B. Überschreiben des Speicher-Inhalts):

8000...8FFFh: Symboltabelle
Fest reservierter Bereich, in dem beim Pass #1 alle symbolischen Namen mit den zugehörigen Absolutadressen abgelegt werden.

9000...EFFFh: Quellprogramm
Speicherbereich, in dem mit Hilfe des Editors das eigentliche Quellprogramm im Assembler-Code erstellt wird. Die erste Assembler-Anweisung muß stets das ORG-Statement sein (Origin; einmalige Angabe der ersten Speicher-Absolutadresse), und die letzte Anweisung muß immer 'END' sein. Beim Initialisieren des Speichers im Editor (Setup-Anweisung) werden diese Zeilen bereits automatisch eingefügt.

F000...FFFFh: Objekt-Code und temporärer Symbol-Buffer
Zielbereich für den bei der Assembler-Übersetzung (Pass #2) entstehenden Maschinen-Code; vor der Ausgabe der Symboltabelle (vgl. Abschnitt 15) werden in diesem Bereich alle symbolischen Namen alphabetisch sortiert (vor dem Zurückkopieren in den Bereich 8000...8FFFh).

Vier maßgebliche 16-Bit-Adreßregister (Pointer) dienen zur Verwaltung dieser Speicherbereiche; Sie sollten sich deren Funktion eingehend klar machen, um mit dem Assembler optimal umgehen zu können:

PSEUPC (Pseudo Program Counter): Pseudo-Programmzähler
Die mit der ORG-Anweisung bereitgestellte Adresse wird in den Pseudo-Programmzähler übertragen (Startadresse für das spätere Maschinenprogramm). Mit jedem übersetzten Assembler-Befehl wird dieses Adreßregister hochgezählt, und zwar um ein, zwei oder drei Adressen, je nach dem, ob ein Ein-, Zwei- oder Drei-Wort-Befehl vorliegt.

TXTPOI (Text Pointer): Adreßregister für das Quellprogramm
Wird bei Beginn jedes Durchlaufs auf 9000h gesetzt (Beginn des Quellprogramms) und adressiert die fortlaufend im Speicher stehenden ASCII-Zeichen.

OBJPOI (Object Pointer): Adreßregister für den Objekt-Code
Wird bei Beginn der Übersetzung (Pass #2) auf F000h gesetzt (Beginn des Objekt-Buffers) und ebenso wie der PSEUPC mit jedem übersetzten Assembler-Befehl um ein, zwei oder drei Adressen hochgezählt.

OFFSET (Offset): Versatz für den Maschinen-Code
Die mit der OFS-Anweisung bereitgestellte Adresse wird nach OFFSET übertragen; sie wird bei der Übersetzung im Pass #2 zum OBJPOI addiert und bewirkt dadurch eine Verschiebung des Maschinen-Codes außerhalb des Standard-Zielbereichs F000...FFFFh. Die OFS-Anweisung ist entbehrlich; wenn sie fehlt, wird OFFSET=0000h gesetzt (d.h. kein Versatz für den Maschinen-Code).

Wichtiger Hinweis: Bei jeder neuen ORG-Anweisung wird OFFSET wieder auf 0000h gesetzt, d.h. bei einem gewünschten Versatz muß nach jedem 'ORG' eine Neu-Definition des Offsets erfolgen.

unzulässige Eingabe:

```
A>y     .                       Error # 60
```

```
0017  PRINT::LXI  H,VIDBUF      *zweifacher Doppelpunkt
A>1                             Error # 61 in Line 0017
```

Symboltabelle reicht über 8FFFh hinaus:

```
A>1                             Error # 62 in Line xxxx
```

```
0017  PRINT: LXI  H,VIDBUF
0018  PRINT: MOV  C,M           *Label PRINT ist doppelt definiert
A>1                             Error # 63 in Line 0018
```

```
0021         CPI  0             *Syntaxfehler: hinter '0' fehlt 'h'
A>1                             Error # 64 in Line 0021
```

```
0017  PRINT: LXI  W,VIDBUF      *Mnemonic falsch
A>1                             Error # 65 in Line 0017
```

```
0018  PRTLOP:MOV  CM            *Feld-Separator fehlt
A>1                             Error # 66 in Line 0018
```

```
0017  $PRINT:LXI  H,VIDBUF      *Label-Fehler
A>1                             Error # 67 in Line 0017
```

2. Fehlermeldungen

In jedem der möglichen Durchläufe Pass #Ø...3 führt der Assembler eine Reihe von Fehler-Kontrollen durch; beim Erkennen eines Fehlers erfolgt unmittelbar die entsprechende Meldung (mit Angabe der Zeilennummer), und der betreffende Durchlauf wird abgebrochen.

Aufgrund der beschränkten Speicherkapazität des Assembler-EPROMs kann die Fehlerprüfung nur bis zu einem gewissen Grad duchgeführt werden, d.h. es werden bei weitem nicht alle denkbaren Fehlermöglichkeiten untersucht. Aufgetretene Fehler können durch unmittelbaren Aufruf des Editors behoben werden (direktes Hin- und Herspringen zwischen den beiden Menüs möglich).

Error # 6Ø: Unzulässige Eingabe (entstammt nicht dem Menü).

Error # 61: Label fehlt; der Label-Markierung ':' kann kein symbolischer Name zugeordnet werden.

Error # 62: Symboltabelle voll; Es sind so viele Label verwendet worden, daß der dafür reservierte Speicherplatz im Adreßbereich 8ØØØ...8FFFh nicht mehr ausreicht.

Error # 63: Mehrfach-Definition eines Labels; derselbe symbolische Name ist mehrfach als Label definiert worden.

Error # 64: Syntax-Fehler; bei einer Zahl fehlt das angehängte 'h' (für hexadezimale) oder 'o' (für oktale Schreibweise) bzw. vor dem Kommentar fehlt das Trennzeichen '*'.

Error # 65: Mnemonic falsch; fehlerhafte Schreibweise eines Assembler-Codes (auch bei Mehrfach-Definition von symbolischen Namen innerhalb einer Zeile).

Error # 66: Feld-Separator fehlt; Fehlende Register-Angabe oder fehlendes Komma zwischen zwei Register-Bezeichnungen.

Error # 67: Label-Fehler; ein symbolischer Name enthält falsche Elemente (beginnt z.B. nicht mit einem Buchstaben).

Fallbeispiel: Anhand eines einfachen Programmbeispiels soll die interaktive Arbeitsweise veranschaulicht werden, die sich bei der Software-Entwicklung zwischen Editor und Assembler abspielt. Sie finden hier (gleichlautend in den Abschnitten 3 der Assembler- und der Editor-Handhabung) die schematische Vorgehensweise zusammengestellt. Wenn Sie mit dem Einsatz von Editor und Assembler noch nicht vollkommen vertraut sind, sollten Sie sich die Mühe machen, die einzelnen Schritte durch entsprechendes Nachschlagen in der Software-Beschreibung nachzuvollziehen!

1. Aufgabenstellung:
Die fest im Monitor programmierte Funktionstaste FCT+V kopiert den Video-Speicher von 3000...377Fh um ans obere Ende des RAM-Bereichs (nach F800...FF7Fh) und schließt diese Daten mit der End-Kennzeichnung 00h ab. Es soll ein Programm PRINT geschrieben werden (ab 2C40h im Monitor-RAM), das den Speicher-Inhalt ab F800h wieder ausgibt (Darstellung der zuvor angefertigten Bildschirm-Momentaufnahme).

2. Programmentwurf (vgl. Editor, Abschnitt 9 'List'):
Nach dem Initialisieren des RAMs (Editor-Setup) werden ORG und OFS neu definiert; die verwendeten Label VIDBUF und CO müssen ebenfalls im Editor definiert werden, anschließend folgt die Programm-Eingabe.

3. Assembler-Testlauf (vgl. Assembler, Abschnitt 5 'Pass#1'):
Das Quellprogramm wird erstmals an den Assembler übergeben, um grobe Fehler aufzuspüren; der erste Durchlauf von Pass #1 liefert sofort die Fehlermeldung 'Error # 64 in Line 14', d.h. in Zeile 14 liegt ein Syntax-Fehler vor.

4. Editor-Korrekturlauf (vgl. Editor, Abschnitt 7 'Go to'):
Der Absolutadresse 0F800 fehlt das angehängte 'h'!

5. Assembler-Testlauf (vgl. Assembler, Abschnitt 5 'Pass#1'):
Das Quellprogramm wird erneut durchlaufen; wieder endeckt der Assembler im Pass #1 einen Fehler: Das Label PRTLOP ist nicht definiert worden (es wird in der Null-Liste am Ende von Pass #1 ausgeworfen).

6. Editor-Korrekturlauf (vgl. Editor, Abschnitt 7 'Go to'):
Sprungziel PRTLOP in Zeile 18 nachtragen (mit CTL+Q einleiten!).

7. Assembler-Durchläufe und Programmtest (vgl. Assembler, Abschn.5):
Pass #1 und Pass #2 liefern keine Fehlermeldung mehr; Programm im Monitor testen ('GO 2C40' im Monitor-START-Menü): Endabfrage fehlt!

8. Editor-Korrekturlauf (vgl. Editor, Abschnitt 8 'Insert'):
Endabfrage vor Zeile 20 einfügen (mit Aussprung aus PRINT -> 1003h).

9. Assembler-Durchläufe und Programmtest (vgl. Assembler, Abschn.5):
Pass #1 und Pass #2 liefern keine Fehlermeldung, und der Testlauf im Monitor funktioniert (Paralleldruck manuell einschalten).

10. Dokumentation (vgl. Editor, Abschn.5 und Assembler, Abschn.7):
Listing bei Bedarf äußerlich überar[redacted]n (z.B. überflüssige Zeilen löschen), Assembler-Durchlauf Pass #3 [redacted]rten und Symboltabelle ausgeben (alles mit Paralleldruck!); Programm auf Diskette sichern!

3. Aufbau des Quell-Listings

Der Assembler verarbeitet das Quellprogramm zeilenweise (intern abgeschlossen durch ØDh=Carriage Return). Jede Zeile besteht aus maximal vier Feldern unterschiedlicher Länge, die (im Editor) mit der Tabulator-Taste TAB spaltenrichtig angesprungen werden können (TABs bei Spalte 7, 12 und 22). Die Felder werden voneinander entweder durch einen Leerschritt ('Blank'), Doppelpunkt, Komma oder Sternchen getrennt (Feld-Separator). Sie haben folgende Bedeutung:

Label: Definition symbolischer Namen; den in diesem Feld definierten symbolischen Namen werden bei der Übersetzung im Pass #1 (bzw. Pass #Ø) Absolutadressen zugeordnet. Ein Label kann maximal sechs Stellen lang sein und muß mit einem Doppelpunkt abgeschlossen werden; es muß mit einem Buchstaben beginnen und kann ansonsten alle alphanumerischen Zeichen mit Ausnahme der Feld-Separatoren enthalten. Eine andere Möglichkeit der Label-Definition bietet der EQU-Operator (vgl. Abschnitt 4 'Pseudo-Operatoren').

Mnemonics: Feld für die mnemotechnischen Assembler-Abkürzungen, die entsprechend dem Intel-Copyright genormt sind (zwei bis vier Stellen lang, entsprechend den Assembler-Codes, z.B. 'EI' für 'Enable Interrupt' oder 'CALL' für einen Unterprogramm-Aufruf).

Operanden: Ergänzen bzw. spezifizieren die Mnemonics durch Angabe von Register-Bezeichnungen oder Nennung eines absoluten bzw. symbolischen Wertes.

Kommentar: Klartext-Erläuterungen (durch ein Sternchen abgetrennt), die bestimmte Vorgehensweisen transparent machen. Nutzen Sie die Möglichkeit für erklärende Kommentare, denn schon kurze Zeit nach Fertigstellung eines Programms sind selbst Ihnen als Programmierer manche Wege kaum oder gar nicht mehr nachvollziehbar!

Hinweis: Beim Anfertigen der Dokumentation im Pass #3 (vgl. Abschnitt 7) werden mit Ausnahme des Kommentar-Feldes alle Kleinbuchstaben automatisch in Großschreibung umgesetzt; Sie können also beim Erstellen des Quellprogramms im Editor alle Label und Mnemonics klein schreiben und brauchen nur bei den Kommentaren auf die gewünschte Klein/Großschreibung zu achten.

Beispiel: ORG 2800h

Beginn des späteren Programms bei 2800h; für aaaa kann jeder Wert aus dem gesamten Adreßbereich eingesetzt werden (0000...FFFFh).

Beispiel: OFS 3800h

Zielbereich für den Maschinencode, der im Pass #2 erzeugt wird, ist nicht F000h, sondern F000+3800=2800h; oooo ist so zu wählen, daß das Ergebnis aus F000+ooooh im RAM-Bereich liegt.

END (an jeder Stelle im Quellprogramm möglich)

Steht linksbündig in einer neuen Zeile und kann selbst nicht als Label definiert werden (wohl aber 'PRTEND').

Beispiel: TIMPRL EQU 20h oder TIMPRL 20h
Beispiel: LOOP: DCR A
JNZ LOOP

Label 'TIMPRL' steht für einen Operanden (Zahlenwert 20h=32d), während das Label 'LOOP' ein Sprungziel symbolisiert, dessen Absoluteadresse im Pass #1 errechnet wird.

Beispiel: DB 12h
Beispiel: DB "R"
Beispiel: DB "Test"

'DB' fügt Datenbytes in das aktuelle Maschinenprogramm ein; im ersten Fall wird das Datenbyte explizit angegeben: 12h; im zweiten Fall wird für "R" der entsprechende ASCII-Code 52h eingefügt; im dritten Fall setzt der Assembler eine ganze Code-Folge ein: 54h (für 'T'), 65h (für 'e'), 73h (für 's') und 74h (für 't').

Beispiel: DW 1234h

'DW' fügt ein Datenwort (=zwei aufeinanderfolgende Bytes, hier '34h' und '12h') in das aktuelle Maschinenprogramm ein; entsprechend der Intel-Notation kommt das niederwertige Byte vor dem höherwertigen.

Beispiel: DS 0F0h

'DS' reserviert im aktuellen Maschinenprogramm F0h=240d Datenbytes, um dort z.B. einen RAM-Buffer zu definieren.

Beispiel: 0h (Zahlenwert Null)
Beispiel: 3Fh (Zahlenwert 63d)
Beispiel: 0F3h (Zahlenwert 243d)
Beispiel: 0A3D1h (Zahlenwert 41937d)

Aus der Angabe einer Hexadezimal-(Oktal-)Zahl geht in keiner Weise hervor, ob sich dahinter eine Zahl (Operand) oder ein Sprungziel (Adresse) verbirgt.

4. Pseudo-Operatoren

Unter 'Pseudo-Operatoren' sind Direktiven an den Assembler zu verstehen, die selbst nicht zu den mnemotechnische Assembler-Codes gehören, aber ähnlich wie diese verarbeitet werden (vgl. auch Abschnitt 1 'Speicher-Organisation').

ORG aaaah: Laden des Pseudo-Programmzählers PSEUPC mit der hexadezimalen Adresse aaaa, bei der das zu übersetzende Programm beginnt; dies ist in der Regel die erste Assembler-Anweisung in einem Quellprogramm. Bei fehlender ORG-Angabe wird ORG=0000h gesetzt.

OFS ooooh: Laden des Offset-Registers OFFSET mit dem hexadezimalen Versatz oooo; Festlegung, wohin im Pass #2 der Objekt-Code geladen wird (Anfangsadresse für den Objekt-Code: F000h+ooooh). Bei fehlender OFS-Angabe und nach jeder neuen ORG-Anweisung wird OFS=0000h gesetzt.

END: Anweisung an den Assembler, den gerade durchgeführten Lauf zu beenden; bei fehlender END-Anweisung kann es zu undefinierten Reaktionen kommen, durch die u.U. auch das Quellprogramm zerstört wird!

LABEL EQU zzzzh: Zuweisung des hexadezimalen Zahlenwertes zzzz zu einem symbolischen Namen (Label) außerhalb eines Assembler-Befehls; ein Label kann auf diese Weise auch einem anderen gleichgesetzt werden. Die explizite Nennung der EQU-Anweisung ist entbehrlich, d.h für die Wertzuweisung genügt es, den symbolischen Namen und den betreffenden Wert (ohne 'EQU') nebeneinander zu schreiben.

DB bbh oder DB "X" oder DB "String": Define Byte; Einfügen des hexadezimalen Bytes 'bb' oder des ASCII-Codes für das Zeichen 'X' oder der in Anführungszeichen eingeschlossenen ASCII-Zeichenfolge 'String' in den aktuellen Objekt-Code.

DW wwwwh: Define Word; Einfügen des hexadezimalen Wortes 'wwww' in den laufenden Objekt-Code.

DS nnh: Define Storage; Anzahl von (hexadezimal) nn Bytes im Objekt-Code frei lassen, d.h. den Pseudo-Programmzähler PSEUPC um 'nn' erhöhen, ohne den Inhalt des Objekt-Buffers zu modifizieren.

Zahlenwerte: Bei der Angabe von absoluten Zahlenwerten, z.B. bei der Label-Definition oder Adreß-Zuweisung, ist der betreffenden Zahl ein 'h' nachzustellen, wenn sie hexadezimal zu verstehen ist, oder bei oktaler Schreibweise ist ein 'o' anzuhängen. Führende Nullen brauchen nicht angegeben zu werden mit Ausnahme von Hexadezimalzahlen, die mit A...F beginnen: Diesen Zahlen muß zur Unterscheidung von symbolischen Namen eine Null vorangestellt werden.

hms

Schritt 3: Assembler-Testlauf (vgl. Abschnitt 3):
==========

A>1 Error # 64 in Line ØØ14

Korrektur im Editor (vgl. dort, Abschnitt 7):
an die Absolutadresse ØF8ØØ ein 'h' anhängen.

Schritt 5: Assembler-Testlauf (vgl. Abschnitt 3):
==========

A>1
PRTLOP ØØØØh

Korrektur im Editor (vgl. dort, Abschnitt 7):
Sprungziel PRTLOP definieren (mit CTL+Q einleiten!).

Schritte 7&9: Assembler-Durchläufe (vgl. Abschnitt 3):
==========

A>1

A>2 o.k.; Code End: 2C5Øh

---

keine Fehlermeldungen.

5. Symboltabelle anlegen/erweitern: Pass #1/Ø

Format: A>1 oder A>Ø (ohne 'Return')

Funktion: Symboltabelle anlegen (1) oder erweitern (Ø)

Bevor im Pass #2 der eigentliche Übersetzungsvorgang beginnen kann, muß der Assembler alle verwendeten Label zusammenstellen und ihnen entsprechend ihrer Anordnung im Quellprogramm eine Absolutadresse zuordnen. Dies geschieht im Pass #1 (bzw. Pass #Ø) nach folgendem Schema:

Jeder symbolische Name wird, wenn er erstmals im Listing erscheint, in die sogenannte Symboltabelle eingetragen (reservierter Speicherbereich 8ØØØ...8FFFh). Erfolgt die Nennung des Namens zusammen mit einer Wertzuweisung (Label-Definition per Doppelpunkt oder EQU-Operator), wird der zugehörige Wert (16-Bit-Konstante) gleichzeitig mit dem Namen in die Symboltabelle eingetragen; andernfalls bleibt das zum Label gehörende Adreßfeld noch so lange leer (auf ØØØØh gesetzt), bis sich die Absolutadresse aus dem Zusammenhang ergibt. Dazu wird einfach der jeweilige Stand des Pseudo-Programmzählers PSEUPC ins Adreßfeld des Labels übertragen, wenn das Label (per Doppelpunkt) definiert wird.

Beim Durchlaufen von Pass #2 oder Pass #3 durchsucht der Assembler beim Auftauchen eines Labels die Symboltabelle und setzt an Stelle des symbolischen Namens die zuvor im Pass #1 (bzw. Pass #Ø) ermittelte Absolutadresse ein.

Der Unterschied zwischen Pass #Ø und Pass #1 ist folgender: Pass #1 legt die Symboltabelle neu an, eine etwa vorhandene wird also dabei überschrieben. Pass #Ø dagegen beläßt eine etwa vorhandene Symboltabelle und hängt alle neu hinzukommenden Label an deren Ende an; auf diese Weise kann ein neues Assembler-Programm auf die Symboltabelle eines anderen Programms zurückgreifen und die darin definierten Label mit benutzen.

Hinweis: Nach Anlegen der Symboltabelle listet der Assembler alle Label, deren Adreßfeld mit ØØØØh belegt ist (vgl. Beispiel links, Mitte). Auf diese Weise haben Sie eine Kontrollmöglichkeit darüber, ob ein Label versehentlich nicht definiert worden ist. Natürlich können Sie einem Label auch per Definition den Wert ØØØØh zuweisen; es taucht dann auch in der Null-Liste auf, ohne daß es sich dabei um einen Fehler handelt.

Die (alphabetisch sortierte) Symboltabelle können Sie sich mit der S-Anweisung ausgeben lassen (vgl. Abschnitt 15).

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>1

A>2                                    o.k.; Code End: 2C5Øh

A>
```

6. Objekt-Code erzeugen: Pass #2

Format: A>2 (ohne 'Return')

Funktion: Übersetzen des Quellprogramms in den Maschinen-Code (Objekt-Code) unter Zugrundelegung der im Pass #1 (bzw. Pass #Ø) angelegten Symboltabelle, d.h. vor Durchlaufen von Pass #2 muß Pass #1 (bzw. Pass #Ø) aufgrufen worden sein.

Der Zielbereich für den Objekt-Code beginnt bei Adresse FØØØh plus OFFSET (Versatz, der mit dem Pseudo-Operator OFS vorgegeben wird). Wenn ein Programm beispielsweise im Adreßbereich ab 2C4Øh laufen soll und der Maschinen-Code beim Assemblieren direkt dorthin geladen werden soll, müssen folgende Randbedingungen definiert werden:

```
ORG EQU 2C4Øh (oder ORG 2C4Øh)   *Programm-Startadresse markieren
                                 *d.h. PSEUPC mit 2C4Øh laden
OFS EQU 3C4Øh (oder OFS 3C4Øh)   *Standard-Objekt-Buffer=FØØØh plus
                                 *Offset=3C4Øh -> Zielbereich=2C4Øh
```

In einem Quellprogramm können ohne weiteres mehrere ORG-Anweisungen vorkommen, um definierte Adressen vorzugeben; dabei ist zu beachten, daß bei jeder ORG-Anweisung der Wert für OFFSET gelöscht (auf ØØØØh gesetzt) wird, d.h. normalerweise wird einer ORG-Neudefinition eine Neudefinition des Offsets folgen.

Nach Abschluß der Übersetzung im Pass #2 wird der aktuelle Stand des Pseudo-Programmzählers PSEUPC ausgegeben ('Code End'). Von Besonderheiten bei der Programmgestaltung abgesehen ist dies die letzte Adresse des Objekt-Codes, so daß Sie den Maschinen-Code des umgesetzten Quellprogramms in der Regel in folgendem Speicherbereich vorfinden:

Beginn bei FØØØh+OFFSET; Ende bei der angezeigten Adresse (im Beispiel links: 2C5Øh).

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>1

A>2                                        o.k.; Code End: 2C5Øh


A>3

ØØØ1 ØØØØ            ;*******************************************************
ØØØ2 ØØØØ            ;*                                                     *
ØØØ3 ØØØØ            ;*      PRINT  Video-Buffer ab F8ØØh ausdrucken        *
ØØØ4 ØØØØ            ;*                                                     *
ØØØ5 ØØØØ            ;*******************************************************
ØØØ6 ØØØØ            ;
ØØØ7 ØØØØ            ;   Stand:   Fr,Ø1.Ø5.87;12:28:28h
ØØØ8 ØØØØ            ;
ØØØ9 ØØØØ            ;   Druck:   Mo,Ø4.Ø5.87;14:51:23h
ØØ1Ø ØØØØ            ;
ØØ11 ØØØØ            ORG 2C4ØH   ;freier RAM-Bereich hinter FCT-Buffer
ØØ12 2C4Ø             OFS 3C4ØH   ;Objekt-Code nach FØØØ+3c4Ø=2c4Øh
ØØ13 2C4Ø             ;
ØØ14 2C4Ø             VIDBUF ØF8ØØH   ;Video-Buffer für FCT+V
ØØ15 2C4Ø             CO     49H      ;Monitor-Ausgabe-Routine
ØØ16 2C4Ø             ;
ØØ17 2C4Ø  21 ØØ F8   PRINT: LXI  H,VIDBUF  ;H&L: Text-Pointer
ØØ18 2C43  4E         PRTLOP:MOV  C,M       ;Text-Zeichen ins Reg C holen
ØØ19 2C44  CD 49 ØØ          CALL CO        ;und ausgeben
ØØ2Ø 2C47  79                MOV  A,C       ;Text-Zeichen ins Reg A
ØØ21 2C48  FE ØØ             CPI  ØH        ;Nullabfrage
ØØ22 2C4A  CA Ø3 1Ø           JZ   1ØØ3H     ;ja: Monitor-Warmstart
ØØ23 2C4D  23                INX  H         ;Text-Pointer erhöhen
ØØ24 2C4E  C3 43 2C           JMP  PRTLOP    ;Schleifendurchlauf
ØØ25 2C51             ;
ØØ26 2C51             END
```

7. Dokumentation ausgeben: Pass #3

Format: A>3<Ret>

Funktion: Komplette Ausgabe des Original-Quellprogramms mit zugehöriger Übersetzung des Maschinen-Codes einschließlich Speicheradressen und Zeilennummern.

Das Durchlaufen von Pass #3 setzt das Vorhandensein der entsprechenden Symboltabelle voraus, d.h. vor Pass #3 muß Pass #1 (bzw. Pass #Ø) aufgerufen worden sein.

Dieser letzte Assembler-Durchlauf dient zur Dokumentation des von Ihnen erstellten Quellprogramms und des daraus vom Assembler angefertigten Maschinen-Programms einschließlich der zugehörigen Speicher-Absolutadressen. Die vom Pass #3 ausgegebene Programmliste enthält also sämtliche relevanten Daten Ihres Programms, dargestellt in übersichtlicher, spaltengerechter Form. Alle Kleinbuchstaben (ausgenommen das Kommentarfeld) werden dabei in Großschreibung umgesetzt (vgl. Abschnitt 3 'Aufbau des Quell-Listings').

Das Listing im Pass #3 können Sie jederzeit stoppen oder fortsetzen, indem Sie die überbreite Leertaste ('Blank') betätigen; bei angehaltenem Listing läßt sich der Drucker zu- oder abschalten (durch CTL+7... CTL+4), um beispielsweise selektive Ausdrucke eines längeren Programms anzufertigen.

Für den Paralleldruck wird alle 64d Zeilen ein automatischer Seitenvorschub ausgegeben, gefolgt vom Ausdruck einer Kopfzeile mit Seitennumerierung. Diese Kopfzeile ist identisch mit der dritten Zeile des Quellprogramms, die zweckmäßigerweise die Bezeichnung des jeweiligen Programms sowie dessen Versions-Nummer enthält.

Ebenfalls zur Dokumentation dienen die beiden Zeilen 'Stand' und 'Druck'. Wenn Sie (im Editor) unter 'Stand' die Echtzeit-Uhr aufrufen, können Sie den zuletzt erreichten Entwicklungsstand mit Datum (und Uhrzeit) festhalten; unter 'Druck' erfolgt der automatische Aufruf der Echtzeit-Uhr, so daß Sie den Zeitpunkt des zuletzt angefertigten Ausdrucks angegeben bekommen.-

Speicheradressen und -inhalt werden wahlweise hexadezimal oder oktal ausgegeben, je nach dem, welche Einstellung Sie zuletzt vorgenommen haben (H- oder O-Anweisung; vgl. Abschnitte 11 bzw. 14 'HEX-' bzw. 'OCT-Mode einstellen'). Voreinstellung beim Kaltstart: HEX-Format (wie im Beispiel links).

Hinweis: Das Assembler-Listing beginnt den Ausdruck immer linksbündig, unabhängig von der im Editor vorgenommenen Consolen-Einstellung. Wenn Sie beim Pass #3 links einen Rand freilassen wollen, können Sie dies z.B. über die direkte Drucker-Ansteuerung in BASIC veranlassen.

hms

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>d

MOPPEL-Disassembler V 9.6
Copyright (C) hms'86

Start,End:

D>2c4Ø,2c5Ø

  2C4Ø  21 ØØ F8  ..... LXI  H,F8ØØh
  2C43  4E        ..... MOV  C,m
  2C44  CD 49 ØØ  ..... CALL ØØ49h
  2C47  79        ..... MOV  A,C
  2C48  FE ØØ     ..... CPI  ØØh
  2C4A  CA Ø3 1Ø  ..... JZ   1ØØ3h
  2C4D  23        ..... INX  H
  2C4E  C3 43 2C  ..... JMP  2C43h

D>
```

8. D: Disassembler-Aufruf

Format: A>D (ohne 'Return')

Funktion: Verlassen des Assembler-Menüs und direkter Sprung zum Disassembler, der wie folgt angesprochen wird:

Eingabe: D>aaaa,eeee<Ret>

Error # 02: Parameter-Fehler (kein HEX-Zeichen)
Error # 03: Parameter-Feld zu lang

Der Disassembler führt die Rückübersetzung von Maschinen-Programmen in den mnemotechnischen Assembler-Code durch. Er muß dazu Anfangs- (aaaa) und Endadresse (eeee) des betreffenden Speicherbereichs angegeben bekommen, wobei aaaa immer einen OpCode adressieren muß (das erste Byte in einem Mehrwort-Befehl). Der Disassembler-Durchlauf endet, sobald die Endadresse eeee erreicht oder überschritten ist, stets aber erst nach dem Disassemblieren eines kompletten Befehls (bei Mehrwort-Befehlen).

Die Leistungsfähigkeit des Disassemblers ist naturgemäß beschränkt: Er muß davon ausgehen, daß im angegebenen Speicherbereich lückenlos nur tatsächlich existierende Befehle aus dem 8085-Befehlssatz vorkommen; Tabellen oder andere Datenbytes führen zu Fehlinterpretationen. Unbekannte Befehls-Codes werden als ASCII-Code aufgefaßt und mit vorangestelltem Fragezeichen ausgegeben.

Der Disassembler kann natürlich auch keine symbolischen Namen vergeben; um Ihnen die diesbezügliche Dokumentation zu erleichtern, fügt er in jeder Zeile vor dem Assembler-Code sechs Punkte ein, an deren Stelle Sie etwaige Label eintragen können.

Im Beispiel links sehen Sie das rückübersetzte Beispielprogramm, das im Abschnitt 3 beschrieben wird. Der Sprungbefehl in der letzten Zeile (JMP 2C43h) weist die zweite Zeile (beginnend mit Adresse 2C43h) als Sprungziel aus, so daß Sie vor 'MOV C,m' ein Label eintragen könnten.

Die Ausgabe des Listings können Sie, wie gewohnt, durch die Betätigung der überbreiten Leertaste ('Blank') stoppen und wieder fortsetzen. Bei angehaltenem Listing läßt sich der Drucker zu- oder wieder abschalten (durch CTL+7...CTL+4). 'Escape' bzw. CTL+C bewirkt das Verlassen des Disassemblers mit einem Sprung ins MEMORY-Menü (!) des Monitors.

Ansonsten können Sie den Disassembler jederzeit durch Eingabe eines 'm' (ohne 'Return') verlassen und zum Monitor-Warmstart zurückkehren. Der Rücksprung zum Assembler ist nur auf dem Umweg über das Monitor-OUTWARD-Menü möglich.

hms

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>e

MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>
```

9. E: Editor-Aufruf

Format: A>E (ohne 'Return')

Funktion: Verlassen des Assembler-Menüs und direkter Sprung zum Editor (vgl. separate Beschreibung im Rahmen des vorliegenden Handbuchs).

Zwischen dem Editor, Assembler und den Floppy-Routinen können Sie ohne den Umweg über den Monitor direkt hin- und herspringen, um die rekursiven Aufrufe dieser Programmteile bei der Software-Entwicklung zu vereinfachen.

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>f

Drv.A:xx     Stp/Sid:xx       Dens:xD
Drv.B:xx     Tracks: xxd      Secs:xxd
Drv.C:xx     RAM-Beg:xxh      Byts:xxh
Drv.D:xx     RAM-End:xxh      Comd:xxh
--------------------------------------------------------------------------------


RAMb,RAMe,Tracks>9Ø,EF,4Ø

MOPPEL-FDC-Utilities V 1Ø.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor
```

10. F: Floppy-Aufruf

Format: A>F (ohne 'Return')

Funktion: Verlassen des Assembler-Menüs und direkter Sprung zu den Floppy-Disk-Utilities (vgl. separate Beschreibung im Rahmen eines eigenen Handbuchs).

Zwischen dem Editor, Assembler und den Floppy-Routinen können Sie ohne den Umweg über den Monitor direkt hin- und herspringen, um die rekursiven Aufrufe dieser Programmteile bei der Software-Entwicklung zu vereinfachen.

hms

```
A>h

A>1

A>2                                    o.k.; Code End: 2C50h


A>3

0001 0000          ********************************************************
0002 0000          *                                                      *
0003 0000          *      PRINT  Video-Buffer ab F800h ausdrucken         *
0004 0000          *                                                      *
0005 0000          ********************************************************
0006 0000          *
0007 0000          *   Stand:   Fr,01.05.87;12:28:28h
0008 0000          *
0009 0000          *   Druck:   Mo,04.05.87;15:01:33h
0010 0000          *
0011 0000          ORG 2C40H   *freier RAM-Bereich hinter FCT-Buffer
0012 2C40          OFS 3C40H   *Objekt-Code nach F000+3c40=2c40h
0013 2C40          *
0014 2C40          VIDBUF 0F800H   *Video-Buffer für FCT+V
0015 2C40          CO     49H      *Monitor-Ausgabe-Routine
0016 2C40          *
0017 2C40  21 00 F8 PRINT: LXI  H,VIDBUF  *H&L: Text-Pointer
0018 2C43  4E       PRTLOP:MOV  C,M       *Text-Zeichen ins Reg C holen
0019 2C44  CD 49 00        CALL CO        *und ausgeben
0020 2C47  79              MOV  A,C       *Text-Zeichen ins Reg A
0021 2C48  FE 00           CPI  0H        *Nullabfrage
0022 2C4A  CA 03 10        JZ   1003H     *ja: Monitor-Warmstart
0023 2C4D  23              INX  H         *Text-Pointer erhöhen
0024 2C4E  C3 43 2C        JMP  PRTLOP    *Schleifendurchlauf
0025 2C51          *
0026 2C51          END

A>
```

11. H: HEX-Mode einstellen

Format: A>H (ohne 'Return')

Funktion: Beim Durchlaufen von Pass #3 Speicheradressen und -inhalte in hexadezimaler Darstellungsweise ausgeben (Zahlenbasis 16d).

Diese Einstellung kann durch die O-Anweisung verändert werden (vgl. Abschnitt 14 'OCT-Mode einstellen'). Beim Aufruf des Assemblers (Kaltstart) ist die hexadezimale Darstellungsform voreingestellt.

```
1208  *
1209  RI2:    LDA   HQFLG      *Quell-Buffer einlesen
1210          ANI   80H
1211          JZ    RICAS      *RI > Externer Eingabe-Kanal
1212  RIMEM:  PUSH  H          *RI > Text-Buffer ab 9000h
1213          LHLD  TXTPOI
1214          MOV   A,M
1215          INX   H
1216          SHLD  TXTPOI
1217          POP   H
1218          ANI   7FH
1219          RET
1220  *
1221  PO2:    LDA   HQFLG      *Objekt-Code ausgeben
1222          ANI   80H
1223          JZ    POCAS      *PO > Externer Ausgabe-Kanal
1224  POMEM:  PUSH  H          *PO > Objekt-Buffer ab F000h
1225          PUSH  D
1226          LHLD  OBJPOI
1227          INX   H
1228          SHLD  OBJPOI
1229          DCX   H
1230          XCHG
1231          LHLD  OFFSET
1232          DAD   D          *Versatz addieren
1233          MOV   M,C        *Objekt-Code ablegen
1234          POP   D
1235          POP   H
1236          RET
1237  *
```

12. I: In/Out on/off

Format: A>I (ohne 'Return')

Funktion: Umschalten des Quell-Buffers vom RAM (ab Adresse 9000h) auf einen externen Eingabe-Kanal und Umschalten des Objekt-Buffers (ab Adresse F000h) auf einen externen Ausgabe-Kanal (Markierung im MSB des HQ-FLAGs).

Für Funktions-Erweiterungen des Assemblers ist im Menü diese Umschaltmöglichkeit vorgesehen. Die zusätzlich erforderlichen Treiber-Routinen sind nicht Bestandteil des Assemblers; sie müßten bei Bedarf vom Anwender ergänzt werden:

In **Adresse 6009h** Sprung zur neuen Eingabe-Routine RICAS, die das Quellprogramm zeichenweise vom externen Eingabe-Kanal einliest (MSB in jedem ASCII-Zeichen auf LOW). Der Text-Pointer TXTPOI in 2F62/63h muß nach Übergabe eines Zeichens um Eins erhöht werden, um das jeweils folgende Zeichen zu adressieren.

In **Adresse 600Fh** Sprung zur neuen Ausgabe-Routine POCAS, die den Objekt-Code zeichenweise an den externen Ausgabe-Kanal übergibt. Der Objekt-Pointer OBJPOI in 2F5E/5Fh muß nach Übergabe eines Bytes um Eins erhöht werden, um die nächste RAM-Zelle zu adressieren; beim Ablegen des Objekt-Codes ist der Inhalt des Offset-Registers OFFSET in 2F60/61h zum Wert des Objekt-Pointers zu addieren.

Links sind die beiden Treiber-Routinen aus dem Assembler gelistet, die ohne In/Out-Umschaltung (also nach jedem Kaltstart) aktiv sind.

hms

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 0/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>m

M>
```

13. M: Monitor-Rücksprung

Format: A>M (ohne 'Return')

Funktion: Verlassen des Assembler-Menüs und direkter Rücksprung zum Monitor (vgl. separate Beschreibung im Rahmen des System-Handbuchs).

Vom Monitor aus können Sie den Editor, den Assembler und die Floppy-Routinen über das OUTWARD-Menü aufrufen. Zwischen Editor, Assembler und Floppy-Utilities können Sie ohne den Umweg über den Monitor hin- und herspringen, um bei rekursiven Aufrufen dieser Programmteile die Software-Entwicklung zu vereinfachen.

hms

```
A>o

A>1

A>2                                        o.k.; Code End: 2C50h

A>3

0001 000000             ;*******************************************************
0002 000000             ;                                                      *
0003 000000             ;      PRINT  Video-Buffer ab F800h ausdrucken         *
0004 000000             ;                                                      *
0005 000000             ;*******************************************************
0006 000000             ;
0007 000000             ;   Stand:    Fr,01.05.87;12:28:28h
0008 000000             ;
0009 000000             ;   Druck:    Mo,04.05.87;15:15:05h
0010 000000             ;
0011 000000             ORG 2C40H   ;freier RAM-Bereich hinter FCT-Buffer
0012 054100             OFS 3C40H   ;Objekt-Code nach F000+3c40=2c40h
0013 054100             ;
0014 054100             VIDBUF 0F800H   ;Video-Buffer für FCT+V
0015 054100             CO     49H      ;Monitor-Ausgabe-Routine
0016 054100             ;
0017 054100  041 000 370 PRINT: LXI  H,VIDBUF  ;H&L: Text-Pointer
0018 054103  116         PRTLOP:MOV  C,M       ;Text-Zeichen ins Reg C holen
0019 054104  315 111 000        CALL CO        ;und ausgeben
0020 054107  171                MOV  A,C       ;Text-Zeichen ins Reg A
0021 054110  376 000            CPI  0H        ;Nullabfrage
0022 054112  312 003 020        JZ   1003H     ;ja: Monitor-Warmstart
0023 054115  043                INX  H         ;Text-Pointer erhöhen
0024 054116  303 103 054        JMP  PRTLOP    ;Schleifendurchlauf
0025 054121             ;
0026 054121             END

A>
```

14. O: OCT-Mode einstellen

Format: A>O (ohne 'Return')

Funktion: Beim Durchlaufen von Pass #3 Speicheradressen und -inhalte in oktaler Darstellungsweise ausgeben (Zahlenbasis 8d).

Diese Einstellung kann durch die H-Anweisung verändert werden (vgl. Abschnitt 11 'HEX-Mode einstellen'). Beim Aufruf des Assemblers (Kaltstart) ist die hexadezimale Darstellungsform voreingestellt.

Die Darstellung im oktalen Zahlensystem hat neben dem historischen vor allem den didaktischen Hintergrund der Umsetzung von Zahlen des einen Basis-Systems in ein anderes (probieren Sie es spaßeshalber ruhig einmal am Beispiel links!).

```
MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>s
CO     Ø049   PRINT  2C4Ø   PRTLOP 2C43   VIDBUF F8ØØ


Mm>l 8ØØØ,8Ø1e
8ØØØ   43 4F ØØ 49   ØØ 5Ø 52 49     4E 54 ØØ 4Ø   2C 5Ø 52 54
8Ø1Ø   4C 4F 5Ø ØØ   43 2C 56 49     44 42 55 46   ØØ ØØ F8

Mm>
```

15. S: Symboltabelle ausgeben

Format: A>S<Ret>

Funktion: Tabellarische Darstellung aller verwendeten Label und der im Pass #1 (bzw. Pass #Ø) ermittelten Absolutadressen.

Wichtiger Hinweis: Alle vorkommenden symbolischen Namen (Label) werden im Pass #1 (bzw. Pass #Ø) in der Reihenfolge ihrer Nennung in die Symboltabelle eingetragen (RAM-Bereich 8ØØØ...8FFFh. Erst beim Ausführen der S-Anweisung werden sie alphabetisch sortiert und anschließend ausgegeben; für diesen Sortier-Vorgang benutzt der Assembler den RAM-Bereich von FØØØ...FFFFh, der standardmäßig für den Objekt-Code vorgesehen ist (sofern kein Offset angegeben wurde). Nach dem Sortieren wird die geordnete Symboltabelle zurück in den dafür vorgesehenen RAM-Bereich 8ØØØ...8FFFh geschrieben. Der während dieses Vorgangs eventuell überschriebene Objekt-Code im Bereich ab FØØØh müßte nach der S-Anweisung erneut generiert werden (Pass #2 nochmals aufrufen).

Links sehen Sie die Symboltabelle für das im Abschnitt 3 beschriebene Programmbeispiel; sie enthält nur vier Label: CO, PRINT, PRTLOP und VIDBUF mit den zugehörigen Absolutadressen. Im RAM-Bereich ab 8ØØØh werden die Label als ASCII-Zeichenfolge abgelegt und mit ØØh von der danach eingetragenen Adresse getrennt (vgl. HEX-Listing links unten).

hms

MOPPEL
System-Software

Editor-
Handhabung

Modulares Prozessor-Programm zum Entwickeln und Lernen

® **Eingetragenes Warenzeichen**

hms

Inhaltsverzeichnis

Ø. Installation und Aufruf
Hardware-Voraussetzungen und Software-Aktivierung

1. Speicher-Organisation
Text-Buffer und Festadressen

2. Fehlermeldungen
Zusammenstellung aller Editor-Fehlermeldungen

3. Zeilen- und Zeichen-Eingabe
Unterschiede der beiden Eingabe-Modes

4. Assembler-Aufruf
Direkter Sprung zum Assembler

5. Clear
Zeilen löschen

6. Floppy-Aufruf
Direkter Sprung zu den Floppy-Routinen

7. Go to
Zeichen einfügen

8. Insert
Zeilen einfügen

9. List
Zeilen listen

1Ø. Monitor-Rücksprung
Direkter Rücksprung zum Monitor

11. Setup RAM
Text-Buffer initialisieren

12. Extend
Console einstellen

13. Hexadezimale Steuerzeichen
Einfügen von HEX-Codes in den Text

hms

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>o

A: Assembler
B: BASIC
D: Disassembler
E: Editor
F: Floppy
P: Prommer
X: Ext.2000h

Mo>e

MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>  '
```

Ø. Installation und Aufruf

Das ROM-residente Editor-Programm ist Bestandteil des EPROMs 'blau'; es wird wahlweise im 4-K-EPROM (2732) oder 8-K-EPROM (2764) geliefert und auf Platz #6 der 89er-Speicherkarte (Platz #7 der 87er-Speicherkarte) eingesetzt (Brücke 'Type 64/32' entsprechend setzen).

Erst der Editor versetzt Sie in die Lage, mit Ihrem Mikrocomputer auch Schreibmaschinen-Funktionen auszuführen, d.h. Klartexte erstellen und jederzeit beliebig verändern zu können. Diese Funktion geschieht mit drei unterschiedlichen Zielrichtungen:

1. Erstellen und Modifizieren von Assembler-Quellprogrammen.
Der Editor dient in diesem Fall dazu, als Handwerkszeug bei der Programmentwicklung die Texte (im Assembler-Code) zu erstellen, die anschließend vom Assembler-Programm in ein lauffähiges Mikrocomputer-Programm (im Maschinen-Code) umgesetzt werden. Dies ist die eigentliche Aufgabe des Editors, und dementsprechend sind seine Leistungsmerkmale organisiert (vgl. Fallbeispiel neben Abschnitt 3).

2. Erstellen und Modifizieren von BASIC-Programmen.
Sie können mit Hilfe des Editors komplette BASIC-Programme erstellen (einschließlich BASIC-Zeilennummern) und diese an den ROM-residenten BASIC-Interpreter zur Ausführung übergeben; auf diese Weise können Sie (im Editor) Ihr BASIC-Programm modifizieren, obwohl der BASIC-Interpreter selbst nicht über Editor-Funktionen verfügt (vgl. BASIC-Handhabung).

3. Erstellen von Texten für die Daten-Fernübertragung.
Sie können umfangreiche Texte vorbereiten, die Sie anschließend z.B. mit Hilfe der Monitor-Terminal-Funktion per Telefonleitung übertragen (vgl. System-Handbuch).

Vom Monitor-Grundmenü aus erfolgt der Editor-Aufruf über das OUTWARD-Menü (Eingabe von 'o' ohne 'Return'), gefolgt von 'e' (wiederum ohne 'Return'). Der Editor-Aufruf kann außerdem direkt aus dem Assembler- oder Floppy-Menü heraus erfolgen. Sämtliche Editor-Anweisungen können in Klein- oder Großschreibung erfolgen; die danach eingegebenen Parameter können, müssen aber nicht durch einen Leerschritt ('Blank') vom Kommando-Buchstaben getrennt sein. Mehrere Parameter sind wahlweise durch Komma oder Punkt voneinander zu trennen. Aus Gründen der Übersichtlichkeit ist das Kommando in den folgenden Beschreibungen stets ein Großbuchstabe, der von den Parametern durch einen Leerschritt abgesetzt ist.

Für den praktischen Einsatz ist es sehr vorteilhaft, daß der Drucker jederzeit (auch bei laufenden Listings!) zu- oder abgeschaltet werden kann; zur Drucker-Ansteuerung ist das Universal-Interface erforderlich.

hms

```
E>i1

      0001  test:  lxi  h,1234h   ;Beispiel-Eingabezeile.........

E>m

M>m

C: Copy
F: Fill
K: Kill
L: List
M: Memory
R: Revise
T: Text
X: Exchange

Mm>l 9000,902c
9000   74 65 73 74   3A 20 20 6C     78 69 20 20   68 2C 31 32
9010   33 34 68 20   20 20 2A 42     65 69 73 70   69 65 6C 2D
9020   45 69 6E 67   61 62 65 7A     65 69 6C 65   0D

Mm>t 9000,0d
test:  lxi  h,1234h   ;Beispiel-Eingabezeile

Mm>
```

1. Speicher-Organisation

Der Editor benutzt ausschließlich den RAM-Bereich von 9000...EFFFh als Text-Buffer, d.h. in diesem 24-K-Speicherraum legt er seine Texte ab. Der hierfür reservierte Platz ist ausreichend für über 24000 ASCII-Zeichen, was etwa einem Dutzend vollgeschriebener DIN-A4-Seiten entspricht oder rund 1200 Assembler-Zeilen mit normaler Kommentierung; dies ist der Umfang für ein ca.2 KByte langes Maschinenprogramm (entspricht etwa dem Umfang der Floppy- oder EPROM-Utilities).

Sämtliche Eingaben werden Zeilen-orientiert abgelegt und verarbeitet, d.h. jede Zeile wird bei der Eingabe durch einen Wagenrücklauf (Carriage Return) abgeschlossen und intern mit dem entsprechenden ASCII-Code 0Dh beendet (Beispiel links: Eingabe eine Zeile im Editor und anschließendes Listen des Speicherinhalts -hexadezimal und als ASCII-Zeichenfolge- im Monitor).

Zur Orientierung numeriert der Editor sämtliche Zeilen fortlaufend durch; die (dezimalen) Zeilennummern werden nicht mit ins Listing eingetragen, sondern bei jeder Ein- und Ausgabe neu errechnet (durch Zählen der Zeilen-End-Codes 0Dh). Die Ausgabe der Zeilennummern ist abschaltbar (vgl. Abschnitt 12 'Extend').

Zur Aufbereitung der Zeilennummern sind zwei Unterprogramme vorgesehen, die Sie bei Bedarf in eigene Programme einbauen können:

700Ah BINOUT: Den dezimalen Inhalt des Registerpaars H&L hexadezimal umsetzen und ausgeben. Beispiel: H&L mit 1000 laden und 700Ah aufrufen führt zur Ausgabe von 03E8 (=1000d)

700Dh LINOUT: Den hexadezimalen Inhalt des Registerpaars H&L dezimal umsetzen und ausgeben. Beispiel: H&L mit 0064 laden und 700Dh aufrufen führt zur Ausgabe von 0100 (=0064h)

Hinweis: Keins der Unterprogramme führt einen Test auf zulässige Zeichen durch.

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend                                  Error # 79

E>c                                         Error # 70

E>y                                         Error # 71

E>g 1e                                      Error # 72

E>l 8888

                                            Error # 75

E>l 7,6

                                            Error # 76

E>x 0,22,88                                 Error # 77

E>s 123,124                                 Error # 78

E>
```

2. Fehlermeldungen

Der Editor erkennt und meldet folgende Fehler:

Error # 70: Nulleingabe unzulässig; beim Löschen muß mindestens eine Zeilennummer angegeben werden.

Error # 71: Unzulässige Eingabe (entstammt nicht dem Menü).

Error # 72: Parameter-Fehler (keine Dezimalzahl; sämtliche Zeilennummern werden dezimal verabeitet).

Error # 75: Zeilennummer existiert nicht; die angegebene (oder beim Löschen bzw. Listen errechnete) Zeilennummer ist zu groß.

Error # 76: Parameter-Fehler (der zweite Parameter ist größer als der erste).

Error # 77: Vorgaben für die Console-Einstellung fehlerhaft.

Error # 78: Parameter-Fehler (beide Parameter sind bei der RAM-Initialisierung ungleich).

Error # 79: RAM ist nicht initialisiert (dazu wird nur der Inhalt der ersten RAM-Zelle des Text-Buffers ausgewertet; die Fehlermeldung erfolgt, wenn in 9000h nicht der ASCII-Code 2Ah für '*' steht).

Hinweis: Die Fehlermeldung 'Error # 79' kommt jedesmal vor der Editor-Bereitmeldung (Prompt 'E>'), wenn am Anfang des Text-Buffers kein '*' steht; um dies zu unterdrücken, brauchen Sie vor Zeile 1 nur eine Leerzeile einzufügen, die intern mit einem Sternchen aufgefüllt wird. Nach Fertigstellen des Textes löschen Sie dann die Zeile 1 einfach wieder.

Fallbeispiel: Anhand eines einfachen Programmbeispiels soll die interaktive Arbeitsweise veranschaulicht werden, die sich bei der Software-Entwicklung zwischen Editor und Assembler abspielt. Sie finden hier (gleichlautend in den Abschnitten 3 der Assembler- und der Editor-Handhabung) die schematische Vorgehensweise zusammengestellt. Wenn Sie mit dem Einsatz von Editor und Assembler noch nicht vollkommen vertraut sind, sollten Sie sich die Mühe machen, die einzelnen Schritte durch entsprechendes Nachschlagen in der Software-Beschreibung nachzuvollziehen!

1. Aufgabenstellung:
Die fest im Monitor programmierte Funktionstaste FCT+V kopiert den Video-Speicher von 3000...377Fh um ans obere Ende des RAM-Bereichs (nach F800...FF7Fh) und schließt diese Daten mit der End-Kennzeichnung 00h ab. Es soll ein Programm PRINT geschrieben werden (ab 2C40h im Monitor-RAM), das den Speicher-Inhalt ab F800h wieder ausgibt (Darstellung der zuvor angefertigten Bildschirm-Momentaufnahme).

2. Programmentwurf (vgl. Editor, Abschnitt 9 'List'):
Nach dem Initialisieren des RAMs (Editor-Setup) werden ORG und OFS neu definiert; die verwendeten Label VIDBUF und CO müssen ebenfalls im Editor definiert werden, anschließend folgt die Programm-Eingabe.

3. Assembler-Testlauf (vgl. Assembler, Abschnitt 5 'Pass#1'):
Das Quellprogramm wird erstmals an den Assembler übergeben, um grobe Fehler aufzuspüren; der erste Durchlauf von Pass #1 liefert sofort die Fehlermeldung 'Error # 64 in Line 14', d.h. in Zeile 14 liegt ein Syntax-Fehler vor.

4. Editor-Korrekturlauf (vgl. Editor, Abschnitt 7 'Go to'):
Der Absolutadresse 0F800 fehlt das angehängte 'h'!

5. Assembler-Testlauf (vgl. Assembler, Abschnitt 5 'Pass#1'):
Das Quellprogramm wird erneut durchlaufen; wieder endeckt der Assembler im Pass #1 einen Fehler: Das Label PRTLOP ist nicht definiert worden (es wird in der Null-Liste am Ende von Pass #1 ausgeworfen).

6. Editor-Korrekturlauf (vgl. Editor, Abschnitt 7 'Go to'):
Sprungziel PRTLOP in Zeile 18 nachtragen (mit CTL+Q einleiten!).

7. Assembler-Durchläufe und Programmtest (vgl. Assembler, Abschn.5):
Pass #1 und Pass #2 liefern keine Fehlermeldung mehr; Programm im Monitor testen ('GO 2C40' im Monitor-START-Menü): Endabfrage fehlt!

8. Editor-Korrekturlauf (vgl. Editor, Abschnitt 8 'Insert'):
Endabfrage vor Zeile 20 einfügen (mit Aussprung aus PRINT -> 1003h).

9. Assembler-Durchläufe und Programmtest (vgl. Assembler, Abschn.5):
Pass #1 und Pass #2 liefern keine Fehlermeldung, und der Testlauf im Monitor funktioniert (Paralleldruck manuell einschalten).

10. Dokumentation (vgl. Editor, Abschn.5 und Assembler, Abschn.7):
Listing bei Bedarf äußerlich überarbeiten (z.B. überflüssige Zeilen löschen), Assembler-Durchlauf Pass #3 starten und Symboltabelle ausgeben (alles mit Paralleldruck!); Programm auf Diskette sichern!

3. Zeilen- und Zeichen-Eingabe

Bei der Neueingabe von Texten unterscheidet der Editor zwei verschiedene Betriebsarten:

Zeilen-Eingabe (Insert-Mode, vgl. Abschnitt 8): Es werden in den Text so lange neue Zeilen eingefügt (mit Verschieben der vorhandenen nach hinten), bis der Insert-Mode wieder verlassen wird (durch 'Escape' bzw. CTL+C oder durch Umschalten auf Zeichen-Eingabe per CTL+F). Bereits eingegebene Zeichen werden (nach Zurückbewegen des Cursors) überschrieben, wenn Sie an derselben Stelle eine Neueingabe vornehmen. Der Insert-Mode ist durch den vollen Cursor erkennbar (volle Zeichenhöhe).

Zeichen-Eingabe (Go-to-Mode, vgl. Abschnitt 7): Es werden in eine bestehende Textzeile so lange neue Zeichen eingefügt (mit Verschieben der vorhandenen nach rechts), bis der Go-to-Mode wieder verlassen wird (durch 'Escape' bzw. CTL+C oder durch Umschalten auf Zeilen-Eingabe per CTL+A). Bereits eingegebene Zeichen werden bei Neueingaben nach rechts verschoben. Der Go-to-Mode ist durch den halben Cursor erkennbar (halbe Zeichenhöhe).

In beiden Betriebsarten zeigen Ihnen die bis zum Zeilenende ausgegebenen Punkte (Leerzeichen) den in der jeweiligen Zeile zur Verfügung stehenden Platz an; diese Punkte werden im Text-Buffer nicht mit abgespeichert. Alle Eingaben, die Sie in der Zeilen- oder Zeichen-Eingabe vornehmen (und die auf dem Bildschirm erscheinen), werden unmittelbar in den Text-Buffer übertragen, d.h. hierzu brauchen Sie nicht explizit die Return-Taste zu betätigen.

Mit der Delete-Taste DEL können Sie Falscheingaben korrigieren: Steht der Cursor auf einem Textzeichen, wird dies durch DEL gelöscht, und die rechts daneben stehenden Zeichen werden nach links nachgerückt. Steht der Cursor auf einem Leerzeichen (Punkt), wird er durch DEL nur nach links bewegt.

Drei weitere Lösch-Funktionen sind außerdem vorgesehen: SHIFT+DEL löscht die gesamte Zeile und füllt sie mit Leerzeichen auf; CTL+Q löscht die sieben links vom Cursor stehenden Zeichen und rückt den Zeilenrest nach links nach (Label löschen mit anschließender Neueingabe); CTL+W löscht ebenfalls die sieben links vom Cursor stehenden Zeichen, läßt den Zeilenrest aber am alten Platz (Label löschen ohne anschließende Neueingabe).

Das Verschieben des Cursors erfolgt außer mit den Pfeiltasten durch CTL+D (rechts) bzw. CTL+S (links); CTL+Y bringt den Cursor an den linken Zeilenrand, während 'Cursor abwärts' (=CTL+X) eine Zeilen-Fortschaltung bewirkt (genauso wie 'Return'). 'Cursor aufwärts' (=CTL+E) veranlaßt eine Zeilen-Rückschaltung mit Übergang in den Go-to-Mode.

Die Tabulator-Taste TAB rückt auf die Spalten 7, 12 oder 22 vor bzw. rechts von Spalte 22 bis zum ersten Leerzeichen (Punkt); übersprungene Leerzeichen werden durch Blanks ersetzt, Textzeichen bleiben unverändert erhalten.

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>a

MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 Ø/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>
```

4. A: Assembler-Aufruf

Format: E>A (ohne 'Return')

Funktion: Verlassen des Editor-Menüs und direkter Sprung zum Assembler (vgl. separate Beschreibung im Rahmen eines eigenen Handbuchs).

Zwischen dem Editor, Assembler und den Floppy-Routinen können Sie ohne den Umweg über den Monitor direkt hin- und herspringen, um die rekusiven Aufrufe dieser Programmteile bei der Software-Entwicklung zu vereinfachen.

hms

```
E>l 17

      0017  print: lxi   h,vidbuf  *H&L: Text-Pointer
      0018  prtlop:mov   c,m       *Text-Zeichen ins Reg C holen
      0019         call  co        *und ausgeben
      0020         mov   a,c       *Text-Zeichen ins Reg A
      0021         cpi   0h        *Nullabfrage
      0022         jz    1003h     *ja: Monitor-Warmstart
      0023         inx   h         *Text-Pointer erhöhen
      0024         jmp   prtlop    *Schleifendurchlauf
      0025  *
      0026  *
      0027  *
      0028  *
      0029  *
      0030  *
      0031  *
      0032  end

E>c 25,30

      0024         jmp   prtlop    *Schleifendurchlauf
      0025  *

E>
```

5. C: Clear (Zeilen löschen)

Format: E>C aaaa,eeee<Ret> oder E>C aaaa<Ret>

Funktion: Löschen der Zeile(n) aaaa...eeee (jeweils einschließlich).

Zur Kontrolle werden nach erfolgtem Löschen die Zeilen vor und hinter dem gelöschten Block gelistet (Zeilen-Nummern aaaa-1 und eeee+1). Beispiel links: Nach dem Löschen der Zeilen 25...30 wird die alte Zeile 31 mit 25 neu numeriert (durch Nachschieben) und zusammen mit der unveränderten Nummer 24 gelistet.

Hinweis: Beim Löschen wird der Text-Buffer ab Adresse EFFFh nach unten verschoben, und der Speicherplatz vom Textende bis zur Buffer-Obergrenze EFFFh wird mit ØØh vollgeschrieben.

Beim Löschen am Textende (bei der jeweils höchsten Zeilennummer) kann keine nachfolgende Zeile mehr gelistet werden; unabhängig vom korrekten Löschvorgang erfolgt in diesem Fall die Fehlermeldung 'Error # 75' (vgl. Abschnitt 2).

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>f

Drv.A:xx     Stp/Sid:xx        Dens:xD
Drv.B:xx     Tracks: xxd       Secs:xxd
Drv.C:xx     RAM-Beg:xxh       Byts:xxh
Drv.D:xx     RAM-End:xxh       Comd:xxh
-------------------------------------------------------------------------------


RAMb,RAMe,Tracks>90,EF,40

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor
```

6. F: Floppy-Aufruf

Format: E>F (ohne 'Return')

Funktion: Verlassen des Editor-Menüs und direkter Sprung zu den Floppy-Disk-Utilities (vgl. separate Beschreibung im Rahmen eines eigenen Handbuchs).

Zwischen dem Editor, Assembler und den Floppy-Routinen können Sie ohne den Umweg über den Monitor direkt hin- und herspringen, um die rekusiven Aufrufe dieser Programmteile bei der Software-Entwicklung zu vereinfachen.

```
E>g 14

      0011  org 2c40h    *freier RAM-Bereich hinter FCT-Buffer
      0012  ofs 3c40h    *Objekt-Code nach F000+3c40=2c40h
      0013  *
      0014  vidbuf 0f800h    *Video-Buffer für FCT+V.............

E>g 18

      0015  co      49h       *Monitor-Ausgabe-Routine
      0016  *
      0017  print: lxi   h,vidbuf   *H&L: Text-Pointer
      0018  mov   c,m        *Text-Zeichen ins Reg C holen.........

E>g 18

      0015  co      49h       *Monitor-Ausgabe-Routine
      0016  *
      0017  print: lxi   h,vidbuf   *H&L: Text-Pointer
      0018  prtlop:mov   c,m        *Text-Zeichen ins Reg C holen..
```

7. G: Go to (Zeichen einfügen)

Format: E>G aaaa<Ret>

Funktion: Sprung zur Zeilennummer aaaa und Einstellen der Betriebsart 'Zeichen einfügen'; die Kennzeichnung erfolgt durch einen halbhohen Cursor (halbe Zeichenhöhe).

Folgende Editier-Funktionen sind hierbei möglich (vorhandene Textzeichen werden bei jeder Neueingabe nach rechts verschoben; vgl. Abschnitt 3 'Zeilen- und Zeichen-Eingabe'):

CTL+S: Cursor links
CTL+D: Cursor rechts
CTL+Y: Cursor an linken Zeilenrand
TAB: Cursor auf Spalte 7, 12 oder 22 bzw. bis zum nächsten Leerzeichen (=Punkt) weiterbewegen

CTL+Q: Label löschen (sieben Zeichen links vom Cursor), Zeilenrest nach links nachrücken; Beispiel links: Mit CTL+Q wurde die gesamte Zeile um sieben Zeichen nach links verschoben (Mitte); nach Eingabe des neuen Labels 'PRTLOP' steht der Zeilenrest wieder am alten Platz (unten).
CTL+W: Label löschen (sieben Zeichen links vom Cursor), Zeilenrest am alten Platz belassen
DEL: Zeichen löschen, Zeilenrest nachrücken (bei Textzeichen) oder Cursor links (bei Leerzeichen = Punkt)
SHIFT+DEL: Zeile löschen und mit Leerzeichen (=Punkten) auffüllen

CTL+E: Zeile zurückschalten
CTL+X oder Return: Zeile weiterschalten

CTL+C oder Escape: Got-to-Mode verlassen und zum Editor-Grundmenü zurückkehren

Bei Zurückweisung einer Eingabe ertönt das akustische Signal ('Bell'), sofern Ihre Mikrocomputer-Hardware entsprechend ausgebaut ist; Zurückweisungen erfolgen z.B., wenn der Cursor bereits linksbündig steht und Sie die Taste 'Cursor links' betätigen.

Hinweis: Wenn Sie nach dem Löschen mit SHIFT+DEL ohne weitere Texteingabe weiterschalten zur nächsten Zeile, füllt der Editor die so entstandene Leerzeile mit einem Sternchen auf. Soll die Leerzeile als solche bestehen bleiben, müssen Sie nach dem Löschen mindestens ein Leerzeichen ('Blank') eingeben, ehe Sie zur folgenden Zeile weiterschalten.

hms

```
E>l 17,21

      0017  print: lxi   h,vidbuf  *H&L: Text-Pointer
      0018  prtlop:mov   c,m       *Text-Zeichen ins Reg C holen
      0019         call  co        *und ausgeben
      0020         inx   h         *Text-Pointer erhöhen
      0021         jmp   prtlop    *Schleifendurchlauf

E>i 20

      0017  print: lxi   h,vidbuf  *H&L: Text-Pointer
      0018  prtlop:mov   c,m       *Text-Zeichen ins Reg C holen
      0019         call  co        *und ausgeben
      0020         mov   a,c       *Text-Zeichen ins Reg A  <<<neu
      0021         cpi   0h        *Nullabfrage             <<<neu
      0022         jz    1003h     *ja: Monitor-Warmstart   <<<neu

E>
```

8. I: Insert (Zeilen einfügen)

Format: E>I aaaa<Ret>

Funktion: Vor der aktuellen Zeilennummer aaaa neue Zeilen einfügen (Einstellen der Betriebsart 'Zeilen einfügen'). Die Kennzeichnung erfolgt durch einen hohen Cursor (volle Zeichenhöhe).

Beispiel links: Durch 'I 20' wurde das Einfügen der drei mit 'neu' gekennzeichneten Zeilen eingeleitet; alle folgenden Zeilen werden automatisch neu numeriert.

Folgende Editier-Funktionen sind im Insert-Mode möglich (vorhandene Textzeichen werden ohne Verschieben überschrieben; vgl. Abschnitt 3 'Zeilen- und Zeichen-Eingabe'):

CTL+S: Cursor links
CTL+D: Cursor rechts
CTL+Y: Cursor an linken Zeilenrand
TAB: Cursor auf Spalte 7, 12 oder 22 bzw. bis zum nächsten Leerzeichen (=Punkt) weiterbewegen

CTL+Q: Label löschen (sieben Zeichen links vom Cursor), Zeilenrest nach links nachrücken
CTL+W: Label löschen (sieben Zeichen links vom Cursor), Zeilenrest am alten Platz belassen
DEL: Zeichen löschen, Zeilenrest nachrücken (bei Textzeichen) oder Cursor links (bei Leerzeichen = Punkt)
SHIFT+DEL: Zeile löschen und mit Leerzeichen (=Punkten) auffüllen

CTL+E: Zeile zurückschalten und Umschalten zur Betriebsart 'Zeichen einfügen' (vgl. Abschnitt 7)
CTL+X = Return: Zeile weiterschalten und Fortsetzen der Betriebsart 'Zeilen einfügen'

CTL+C = Escape: Insert-Mode verlassen und zum Editor-Grundmenü zurückkehren

Bei Zurückweisung einer Eingabe ertönt das akustische Signal ('Bell'), sofern Ihre Mikrocomputer-Hardware entsprechend ausgebaut ist; Zurückweisungen erfolgen z.B., wenn der Cursor bereits linksbündig steht und Sie die Taste 'Cursor links' betätigen.

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>l 17,19

      0017  print: lxi  h,vidbuf  *H&L: Text-Pointer
      0018         mov  c,m       *Text-Zeichen ins Reg C holen
      0019         call co        *und ausgeben


E>l:0029/A77F

      0001  ************************************************
      0002  *                                              *
      0003  *     PRINT  Video-Buffer ab F800h ausdrucken  *
      0004  *                                              *
      0005  ************************************************
      0006  *
      0007  *   Stand:   Fr,01.05.87;12:28:28h
      0008  *
      0009  *   Druck:   Mo,04.05.87;16:17:17h
      0010  *
      0011  org 2c40h   *freier RAM-Bereich hinter FCT-Buffer
      0012  ofs 3c40h   *Objekt-Code nach F000+3c40=2c40h
      0013  *
      0014  vidbuf 0f800    *Video-Buffer für FCT+V
      0015  co     49h       *Monitor-Ausgabe-Routine
      0016  *
      0017  print: lxi  h,vidbuf  *H&L: Text-Pointer
      0018         mov  c,m       *Text-Zeichen ins Reg C holen
      0019         call co        *und ausgeben
      0020         inx  h         *Text-Pointer erhöhen
      0021         jmp  prtlop    *Schleifendurchlauf
      0022  *
      0023  *
      0024  *
      0025  *
      0026  *
      0027  *
      0028  *
      0029  end


E>
```

9. L: List (Zeilen listen)

Format: E>L<Ret> oder E>L aaaa<Ret> oder E>L aaaa,eeee<Ret>

Funktion: 1. Gesamten Text-Buffer listen (mit Angabe des Füllgrades und der höchsten Zeilennummer (Beispiel links, unten: Die höchste vorkommende Zeilennummer ist 0029, und der Text-Buffer ist bis zur Adresse A77Fh gefüllt).

2. Text-Buffer ab Zeilennummer aaaa listen (bis zur höchsten Zeilennummer, wenn das Listen vorher nicht gestoppt und abgebrochen wird; s.u.).

3. Zeilennummern aaaa...eeee listen (Beispiel links, Mitte)

Das Listen kann jederzeit gestoppt und wieder fortgesetzt werden, indem die überbreite Leertaste ('Blank') betätigt wird. Bei angehaltenem Listing läßt sich der Drucker zu- oder wieder abschalten (durch CTL+7...CTL+4). Mit 'Escape' bzw. CTL+C wird das Listen abgebrochen, und der Editor meldet sich mit seinem Bereitzeichen (Prompt).

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>m

M>
```

10. M: Monitor-Rücksprung

Format: E>M (ohne 'Return')

Funktion: Verlassen des Editor-Menüs und direkter Rücksprung zum Monitor (vgl. separate Beschreibung im Rahmen des System-Handbuchs).

Vom Monitor aus können Sie den Editor, den Assembler und die Floppy-Routinen über das OUTWARD-Menü aufrufen. Zwischen Editor, Assembler und Floppy-Utilities können Sie ohne den Umweg über den Monitor hin- und herspringen, um bei rekursiven Aufrufen dieser Programmteile die Software-Entwicklung zu vereinfachen.

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend                                Error # 79

E>s 456,456


E>s:0015/914C

      0001  ************************************************
      0002  *                                              *
      0003  *                                              *
      0004  *                                              *
      0005  ************************************************
      0006  *
      0007  *   Stand:
      0008  *
      0009  *   Druck:   Fr,01.05.87;11:45:38h
      0010  *
      0011  org 0f000h
      0012  ofs 0h
      0013  *
      0014  *
      0015  end


E>
```

11. S: Setup RAM (Text-Buffer initialisieren)

Format: E>S aaaa,aaaa<Ret>

Funktion: Löschen des gesamten Text-Buffers im Bereich 9000... EFFFh und Einfügen der Assembler-Kopfzeilen; am Ende dieses Rumpf-Textes wird ein Byte 03h (=ETX) eingefügt, das bei der Verarbeitung von Klartexten das Textende markiert.

Die in Zeile 3 vorgesehene Programm-Kennzeichnung (Name und Versionsnummer) wird bei der Dokumentation von Assembler-Programmen (im Pass # 3) auf jeder neuen Druckseite mit ausgedruckt.

Wichtiger Hinweis: Vor Eingabe eines neuen Textes, gleichgültig, ob es sich um einen Klartext, ein BASIC-Listing oder ein Assembler-Quellprogramm handelt, soll die Setup-Funktion aufgerufen werden; nur so ist sichergestellt, daß alle folgenden Editor-Anweisungen auch ordnungsgemäß ausgeführt werden.

Wenn Sie die Assembler-Kopfzeilen nicht benötigen, löschen Sie sie einfach wieder (vgl. Abschnitt 5 'Clear'). Ansonsten beginnen Sie die Eingabe von Assembler-Quellprogrammen mit 'Insert 14', nachdem Sie ORG und OFS definiert und hinter 'Stand' das Datum eingetragen haben (durch FCT+U). An der Stelle 'Druck' wird bei jedem Listen automatisch die Echtzeit-Uhr ausgelesen und angezeigt; auf diese Weise erhalten Sie eine stets aktuelle Dokumentation des letzten Entwicklungsstandes und Zeitpunkt des Ausdrucks.

Zur Sicherung gegen unbeabsichtigtes Löschen müssen zusammen mit dem Kommando-Buchstaben die beiden (durch Komma oder Punkt getrennten) Parameter aaaa eingegeben werden; aaaa kann eine ein- bis vierstellige Dezimalzahl sein, wichtig ist nur, daß sie zweimal gleichlautend eingegeben wird.

hms

```
E>gi0

      0007  *
      0008  *
      0009  *
      0010  Standard-Consolen-Einstellung:.....
      0011  6 Leerstellen am linken Rand,.....
      0012  Zeilennummer ausgeben,.....
      0013  Consolenbreite 59d Stellen, d.h. 59-6 Anschläge.....
      0014  *.....

E>x 0,0,79

E>i 16

Consolenbreite 59d Stellen, d.h. 59-6 Anschläge
*
*
Modifizierte Einstellung:.....
keine Leerstellen am linken Rand (erste Null),.....
Zeilennummer abschalten (zweite Null),.....
Consolenbreite 79d, d.h. 79-0=79 Anschläge.....

E>
```

12. X: Extend (Console einstellen)

Format: E>X ll,n,cc<Ret>

Funktion: Linken Rand einstellen (ll: Leerzeichen links),
Zeilennummer unterdrücken (n=Ø) oder darstellen (n<>Ø) und
Consolen-(Zeilen-)Breite auf cc Zeichen einstellen.

Diese Funktion ist dazu vorgesehen, die vom Editor gebotenen Textverarbeitungs-Möglichkeiten zu nutzen; Sie können damit die Zeilenbreite an ein bestimmtes Formular anpassen und die Zeilennummern (z.B. für Text-Ausdrucke) unterdrücken.

Die maximale Consolenbreite darf 79d Zeichen nicht überschreiten (einschließlich der führenden Leerzeichen am linken Rand). Die beim Kaltstart vorgenommene Standard-Einstellung ist an das MOPPEL-Assembler-Format angepaßt.

hms

```
MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend


E>g5

0002  *
0003  *
0004  *
0005  hØc ...<<< h: HEX-Zeichen; ØC: Form Feed; Blank: Ende HEX-Zeichen.....


E>
```

13. Hexadezimale Steuerzeichen

Format: CTL+B (einleiten) und Blank (beenden)

Funktion: Einfügen von hexadezimalen Steuerzeichen in den Text und Übergabe des entsprechenden HEX-Codes an die Ausgabe-Routine CO bei allen späteren Listings.

Mit dieser Funktion haben Sie die Möglichkeit, auch solche Codes in einen Text einzufügen, die vom Editor normalerweise zurückgewiesen werden. Auf diese Weise können Sie z.B. einen expliziten Seitenvorschub einfügen (=ØCh) oder auch Escape-Sequenzen zum Hervorheben bestimmter Textpassagen (Inversdarstellung ein- und ausschalten; vgl. Abschnitt 6.5 des System-Handbuchs).

Mit CTL+B leiten Sie eine Folge von hexadezimalen Steuerzeichen ein; sie wird optisch duch das halbfette 'h' kenntlich gemacht. Jeder HEX-Code muß danach zweistellig angegeben werden (das gilt auch für eine führende Null). Sie können beliebig viele Code-Folgen nacheinander eingeben und verlassen den HEX-Mode einfach dadurch wieder, daß Sie ein Leerzeichen ('Blank') eingeben.

hms

MOPPEL
System-Software

Floppy-Handhabung

Modulares Prozessor-Programm zum Entwickeln und Lernen

® **Eingetragenes Warenzeichen**

hms

Inhaltsverzeichnis

Ø. Installation und Aufruf
Hardware-Voraussetzungen und Software-Aktivierung

1. Floppy-Menü
Bedeutung der Parameter in den Kopfzeilen

2. Fehlermeldungen
Zusammenstellung aller Floppy-Fehlermeldungen

3. Parameter-Eingabe
3.1 Eingabe der RAM-Grenzen und Spur-Anzahl
3.2 Eingabe des Blockanfangs auf der Diskette

4. A: Assembler
Direkter Sprung zum Assembler

5. B: Batch out
Vorspann für Selbst-Start ausgeben

6. C: Copy
Spuren kopieren

7. D: Disk in
Batch-Vorspann einlesen und Programm starten

8. E: Editor
Direkter Sprung zum Editor

9. F: Format
Diskette formatieren

1Ø. M: Monitor
Monitor-Rücksprung

11. R: Read
Diskette lesen

12. W: Write
Diskette beschreiben

13. System-Parameter
13.1 Anschlußbelegung der Verbindungskabel
13.2 Fest im EPROM eingestellte Parameter
13.3 Batch-Parameter
13.4 Kommandos des FDC-Steuerbausteins WD177Ø

14. CP/M-Betriebssystem
14.1 Laufwerk-Anpassung für CP/M
14.2 Voreingestellte Laufwerk-Typen

hms

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>o

A: Assembler
B: BASIC
D: Disassembler
E: Editor
F: Floppy
P: Prommer
X: Ext.2000h

Mo>f
```

Ø. Installation und Aufruf

Die Europa-Karte ist (bei ausgeschalteter Versorgungsspannung) in einen freien Bus-Platz einzustecken; es empfiehlt sich hierfür der Steckplatz ganz außen rechts, weil dann das Flachbandkabel zum Laufwerk-Anschluß problemlos zwischen der Bus-Platine und der Seitenwand des 19-Zoll-Einschubs hindurchgeführt werden kann (hierzu gegebenenfalls die rechte Seitenwand lösen). Die Floppy-Software ist zusammen mit den EPROM-Utilities in einem EPROM enthalten, das auf dem Floppy-Disk-Controller eingesetzt wird.

Über das Flachbandkabel werden die Laufwerke elektrisch angeschlossen (Polung anhand der Pin-Numerierung beachten!) Es können wahlweise 3-Zoll-, 3.5-Zoll- oder 5.25-Zoll-Laufwerke mit 34poliger Standard-Schnittstelle verwendet werden (vgl. Abschnitt 13.1 'Anschlußbelegung der Verbindungskabel'), und zwar mit einem oder zwei Schreib/Lese-Köpfen, auch im gemischten Betrieb (maximal vier Laufwerke).

Bei Anschluß mehrerer Laufwerke sollen die Abschlußwiderstände auf der Laufwerk-Elektronik nur in demjenigen Gerät vorhanden sein, das am Ende des Flachband-Kabels liegt; in den dazwischenliegenden Geräten sind diese Widerstände (in der Regel Widerstands-Netzwerke) zu entfernen. Beachten Sie hierzu die Angaben des Laufwerk-Herstellers.

Bei mehreren Laufwerken muß außerdem jedes einzelne seine eigene Kennung bekommen, um es gegenüber den anderen zu unterscheiden; dies geschieht in der Regel durch kleine Steckbrücken auf der Laufwerk-Elektronik, die wie folgt zu verdrahten sind:

DSØ (Drive Select Ø) für Drive A (Laufwerk A)
DS1 (Drive Select 1) für Drive B (Laufwerk B)
DS2 (Drive Select 2) für Drive C (Laufwerk C)
DS3 (Drive Select 3) für Drive D (Laufwerk D)

Die Stromversorgung von maximal zwei Mikro-Laufwerken (3 Zoll oder 3.5 Zoll) kann normalerweise noch vom Schaltnetzteil übernommen werden (d.h. mittlere Stromaufnahme pro Laufwerk ca. 35Ø mA aus +5 V bzw. ca. 55Ø mA aus +12 V). Bei Ansprache eines Laufwerks werden die Motoren aller übrigen angeschlossenen Laufwerke mit in Gang gesetzt (nicht aber deren Elektronik).

Vom Monitor-Grundmenü aus erfolgt der Floppy-Aufruf über das OUTWARD-Menü (Eingabe von 'o' ohne 'Return'), gefolgt von 'f' (wiederum ohne 'Return'). Die Eingabe der Kommando-Buchstaben kann wahlweise in Klein- oder Großschreibung erfolgen; die danach eingegebenen Parameter können, müssen aber nicht durch einen Leerschritt ('Blank') vom Kommando-Buchstaben getrennt sein. Mehrere Parameter sind wahlweise durch Komma oder Punkt voneinander zu trennen, führende Nullen brauchen nicht eingegeben zu werden. Aus Gründen der Übersichtlichkeit ist das Kommando in den folgenden Beschreibungen stets ein Großbuchstabe, der von den Parametern durch einen Leerschritt abgesetzt ist (vgl. auch Abschnitt 3 'Parameter-Eingabe').

Auch die Floppy-Software unterstützt zwei Bildformate, die abhängig sind von der Einstellung des Video-Interfaces: 24 Zeilen mit je 8Ø Zeichen oder 2Ø Zeilen mit je 4Ø Zeichen. Im übrigen sei auf die ausführlichen Beschreibungen des System-Handbuchs verwiesen.

hms

```
Drv.A:xx     Stp/Sid:xx       Dens:xD
Drv.B:xx     Tracks: xxd      Secs:xxd
Drv.C:xx     RAM-Beg:xxh      Byts:xxh
Drv.D:xx     RAM-End:xxh      Comd:xxh
-------------------------------------------------------------------------------

RAMb,RAMe,Tracks>90,EF,40

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor


 F>
```

1. Floppy-Menü

Nach jedem Kalt- oder Warmstart (über 'Escape' bzw. CTL+C) wird das Floppy-Menü mit den neun möglichen Kommandos 'A', 'B', 'C', 'D', 'E', 'F', 'M', 'R' und 'W' auf dem Bildschirm dargestellt.

Eingaben des Anwenders werden von unten nach oben protokolliert und fortgeschrieben. Ausgenommen hiervon sind die Parameter in den Kopfzeilen; sie werden -soweit erforderlich- für die einzelnen Operationen unverändert übernommen, müssen also nicht jedesmal neu eingegeben werden. 'System-Parameter' sind fest im EPROM gespeicherte Werte, die während des normalen Betriebes nicht verändert werden (vgl. Abschnitt 13.2 'Fest im EPROM eingestellte Parameter').

Die Felder in den Kopfzeilen haben folgende Bedeutung (ein angehängtes 'h' kennzeichnet hexadezimale, ein 'd' dezimale Schreibweise):

Drv.A: Drive A (während des Betriebes aktualisiert)
Kopfposition von Laufwerk A ('xx': undefiniert).
Drv.B: Drive B (während des Betriebes aktualisiert)
Kopfposition von Laufwerk B ('xx': undefiniert).
Drv.C: Drive C (während des Betriebes aktualisiert)
Kopfposition von Laufwerk C ('xx': undefiniert).
Drv.D: Drive D (während des Betriebes aktualisiert)
Kopfposition von Laufwerk D ('xx': undefiniert).

Stp/Sid: Step/Side (System-Parameter/Eingabe-abhängig)
Zusammengefaßte Anzeige von Step-Rate und gewählter Disketten-Seite ('Side' nur für Laufwerke mit zwei Schreib/Lese-Köpfen).
Tracks: Spur-Anzahl (während des Betriebes aktualisiert)
Anzahl der Spuren beim Kopieren oder Formatieren.
RAM-Beg: RAM-Beginn
RAM-Anfangsadresse beim Datentransfer (obere Adreßhälfte; die untere Hälfte ist immer ØØh).
RAM-End: RAM-Ende
RAM-Endadresse beim Datentransfer (obere Adreßhälfte; die untere Hälfte ist immer FFh).

Dens: Density (System-Parameter)
Einfache/doppelte Schreibdichte.
Secs: Sectors per Track (System-Parameter)
Anzahl der Sekoren pro Spur.
Byts: Bytes per Sector (System-Parameter)
Anzahl der Bytes pro Sektor.
Comd: Command (während des Betriebes aktualisiert)
Vom FDC-Steuerbaustein WD177Ø augenblicklich bzw. zuletzt ausgeführtes Kommando (vgl. Abschnitt 13.4 'Kommandos des FDC-Steuerbausteins WD177Ø').

hms

```
Drv.A:xx     Stp/Sid:30       Dens:DD
Drv.B:xx     Tracks: 40d      Secs:16d
Drv.C:xx     RAM-Beg:90h      Byts:01h
Drv.D:xx     RAM-End:EFh      Comd:D0h
```

```
 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor
```

```
F>                                          Error # 90

F>w
Dst-Drv,Trk,Sec> A, X, y                    Error # 91

F>
```

2. Fehlermeldungen

Bei der Parameter-Eingabe werden folgende Fehler sofort erkannt und gemeldet:

Error # 90: Fehlende oder unzulässige Eingabe (entstammt nicht dem Menü).

Error # 91: Parameter-Fehler (zu lang oder kein HEX-Digit).

Die übrigen Fehlermeldungen beziehen sich auf das Lesen bzw. Beschreiben der Diskette (auch beim Kopieren oder Formatieren); sie haben folgende Bedeutung:

Error # 92: Datenverlust; ein Datenbit wurde nicht rechtzeitig abgerufen (beim Lesen) bzw. nicht rechtzeitig bereitgestellt (beim Schreiben), jeweils bezogen auf die Interrupt-Anforderung des FDC-Steuerbausteins und die zulässige Zeit für ein Datenbyte (TRAP Respond Time).

Error # 93: CRC-Fehler; bei der dynamischen Prüfsummen-Bildung (Cyclic Redundancy Check) wurde im ID- oder Datenfeld ein Fehler erkannt (beim Vergleich mit der Aufzeichnung).

Error # 94: Angegebene(r) Sektor/Spur/Seite wurde nicht gefunden

Error # 95: Daten-Adreß-Marke gelöscht (das Byte vor dem eigentlichen Datenblock eines Sektors).

Error # 96: Diskette ist schreibgeschützt (nur bei Schreib-Operationen, auch beim Formatieren).

Error # 97: RAM-Grenzen fehlerhaft (RAMe < RAMb).

Error # 99: Laufwerk nicht betriebsbereit; keine Fertig-Meldung innerhalb der zulässigen Zeit (Time Out).

Hinweis: Jede Fehlermeldung löst beim FDC-Steuerbaustein einen sofortigen Interrupt aus, durch den die laufende Operation unmittelbar abgebrochen wird. Das dadurch aktivierte Laufwerk wird nur dann automatisch wieder abgeschaltet, wenn sich darin eine Diskette befindet; andernfalls bleibt das betreffende Laufwerk so lange aktiviert (mit laufendem Motor), bis die nächste Operation fehlerfrei beendet wurde.

hms

```
Drv.A:xx    Stp/Sid:xx       Dens:xD
Drv.B:xx    Tracks: xxd      Secs:xxd
Drv.C:xx    RAM-Beg:xxh      Byts:xxh
Drv.D:xx    RAM-End:xxh      Comd:xxh
-------------------------------------------------------------------------------

RAMb,RAMe,Tracks>90,EF,40    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor
```

3. Parameter-Eingabe

Die für den Disketten-Zugriff notwendigen Parameter werden in zwei Gruppen eingegeben:

* **RAM-Beginn und -Ende sowie Spur-Anzahl;**
  (Übernahme dieser Parameter in die Kopfzeilen).

* **Quell- bzw. Ziel-Laufwerk, Seite, Spur- und Sektor-Nummer;**
  (Neueingabe dieser Parameter bei jeder Operation).

Durch diese Organisationsform erleichtert sich die Bedienung für den Anwender ganz wesentlich, insbesondere bei häufigen Disketten-Zugriffen, wie z.B. bei der Programm-Entwicklung mit Editor und Assembler. Die Grenzen des angesprochenen RAM-Bereichs (RAM-Beginn und -Ende) brauchen nur einmal eingegeben zu werden; sie erscheinen anschließend (ebenso wie die Spur-Anzahl) in den Kopfzeilen und werden vom Floppy-Menü so lange übernommen, bis der Anwender neue Grenzwerte eingibt.

Die für jede Operation notwendigen Eingaben beschränken sich auf das gewünschte Laufwerk, die Disketten-Seite und die betreffende Spur- sowie Sektor-Nummer.

Die vom Anwender vorzunehmenden Eingaben vereinfachen sich weiterhin dadurch, daß für jeden Parameter Voreinstellungen erscheinen, die, soweit sie zutreffen, nur quittiert zu werden brauchen (durch 'Return' bzw. Weiterbewegen des Cursors). Die voreingestellten Werte gehören zu den System-Parametern, die fest im EPROM gespeichert sind, sich dort bei Bedarf aber ohne weiteres ändern lassen (vgl. Abschnitt 13.2 'Fest im EPROM eingestellte Parameter').

Diese über lange Zeit erprobte Eingabe-Form hat sich in der Praxis bestens bewährt; da sie vom stereotypen Standard abweicht, sollten Sie sich als Anwender eingehend damit vertraut machen, um die gebotene Leistungsfähigkeit auch wirklich auszunutzen!

```
Drv.A:05    Stp/Sid:30       Dens:DD
Drv.B:12    Tracks: 40d      Secs:16d
Drv.C:xx    RAM-Beg:90h      Byts:01h
Drv.D:xx    RAM-End:EFh      Comd:80h
-------------------------------------------------------------------------------

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor




F>r
Src-Drv,Trk,Sec>0b,07,01                o.k.

F>r
Src-Drv,Trk,Sec>b,7,1                   <<<<<<<<<<<<<<<<<<<<<<<<<<
```

3.1 Eingabe der RAM-Grenzen und Spur-Anzahl

Aufruf: Mit der Escape-Taste oder durch Betätigen von CTL+C wird die betreffende Eingabezeile unterhalb der fünf festen Kopfzeilen eingeblendet. In dieser Zeile erscheinen die fest eingestellten System-Parameter, während die entsprechenden Felder in den Kopfzeilen gelöscht werden (mit 'xx' überschrieben).

Eingabe: Geben Sie die gewünschten Parameter ein, soweit sie von den angegebenen Werten abweichen; bei Übereinstimmung können Sie den Cursor einfach zum nächsten Parameter-Feld weiterbewegen (mit der Taste 'Cursor rechts' oder mit CTL+D).

Bedeutung: RAM-Beginn und -Ende legen die Speichergrenzen für den Datentransfer fest; es wird jeweils nur die obere Adreßhälfte angegeben, die beim RAM-Beginn auf '00h' und beim RAM-Ende auf 'FFh' ergänzt wird. Die (dezimale) Angabe der Spur-Anzahl bezieht sich auf das Formatieren und Kopieren.

Ändern: Bei Änderungen innerhalb der Eingabezeile gehen Sie einfach mit dem Cursor zurück (mit der Taste 'Cursor links' oder mit CTL+S) und geben an der betreffenden Stelle die richtigen Werte ein.

Übernahme: Durch Betätigen der Return-Taste werden die drei Parameter in die entsprechenden Felder der Kopfzeilen übertragen ('Tracks', 'RAM-Beg' und 'RAM-End'). Das Floppy-Menü meldet sich daraufhin wieder mit seinem Bereit-Zeichen 'F>' (Prompt).

Korrektur: Bei Änderungen nach erfolgter Übernahme verfahren Sie wie bei einer Neueingabe: Sie betätigen die Escape-Taste (oder CTL+C) und gehen weiter so vor, wie unter 'Aufruf' beschrieben (s.o.).

Hinweis: Die drei Parameter für die RAM-Grenzen und die Spur-Anzahl werden in einem internen RAM-Buffer zwischengespeichert; sie können von dort jederzeit unverändert übernommen werden, wenn zur Quittierung der Eingabe-Zeile nicht 'Return' sondern 'Escape' oder CTL+C betätigt wird. Diese Übernahme-Möglichkeit der zuletzt eingestellten Parameter gilt auch dann, wenn der Floppy-Aufruf im Kaltstart erfolgt, also aus dem Monitor, dem Assembler oder Editor heraus! Sie können also das Floppy-Menü verlassen, anschließend im Assembler oder Editor arbeiten und nach dem erneuten Floppy-Aufruf mit den zuletzt eingestellten Parametern 'RAMb', 'RAMe' sowie 'Tracks' weiterarbeiten. Nutzen Sie bei Bedarf diese weitere Möglichkeit der optimierten Parameter-Eingabe.

3.2 Eingabe des Block-Anfangs auf der Diskette

Aufruf: Erfolgt automatisch vor jedem Disketten-Zugriff. Es erscheinen jeweils die fest eingestellten System-Parameter, während die in den Kopfzeilen angezeigten Kopf-Positionen unverändert erhalten bleiben.

Eingabe: Geben Sie die gewünschten Parameter ein, soweit sie von den angegebenen Werten abweichen; bei Übereinstimmung können Sie den Cursor einfach zum nächsten Parameter-Feld weiterbewegen (mit der Taste 'Cursor rechts' oder mit CTL+D).

Bedeutung: 'Src-Drv' (Source-Drive) gibt das Quell-Laufwerk A...D an (beim Lesen), und 'Dst-Drv' (Destination Drive) bezeichnet das Ziel-Laufwerk A...D (beim Schreiben). Die Ziffer 'Ø' oder '1' vor der Laufwerk-Bezeichnung kennzeichnet die Disketten-Seite (nur für Laufwerke mit zwei Schreib/Lese-Köpfen von Bedeutung; vgl. untenstehenden Hinweis).

Die Angaben für 'Trk' und 'Sec' spezifizieren die Spur- und Sektor-Nummer, bei denen der Datentransfer beginnt.

Ändern: Bei Änderungen innerhalb der Eingabezeile gehen Sie einfach mit dem Cursor zurück (mit der Taste 'Cursor links' oder mit CTL+S) und geben an der betreffenden Stelle die richtigen Werte ein.

Starten: Durch Betätigen der Return-Taste wird das entsprechende Kommando mit den gewählten Parametern gestartet. Von diesem Zeitpunkt an werden die Kopfpositionen der angesprochenen Laufwerke dynamisch aufdatiert, und der vom FDC-Steuerbaustein jeweils ausgeführte Befehl erscheint im Feld 'Comd' (Command, vgl. Abschnitt 13.4 'Kommandos des FDC-Steuerbausteins WD177Ø). Nach Ausführung des Kommandos (oder nach Abbruch wegen eines Fehlers) meldet sich das Floppy-Menü mit seinem Bereit-Zeichen 'F>' (Prompt).

Hinweis: In diesem Zusammenhang wirkt es sich besonders vorteilhaft aus, führende Nullen nicht mit eingeben zu müssen: Die vor der Laufwerk-Bezeichnung einzugebende Ziffer Ø/1 für die Disketten-Seite kann immer dann entfallen, wenn Seite Ø angesprochen wird (Standard-Laufwerk). Beispiel (vgl.linkes Blatt): Seite Ø/Laufwerk B, Spur 7, Sektor 1 wird eingegeben als 'B,7,1', gefolgt von drei Leerschritten ('Blanks'), um den Rest der vorgeschlagenen Parameter zu löschen.

```
M>
Drv.A:xx    Stp/Sid:xx      Dens:xD
Drv.B:xx    Tracks: xxd     Secs:xxd
Drv.C:xx    RAM-Beg:xxh     Byts:xxh
Drv.D:xx    RAM-End:xxh     Comd:xxh
--------------------------------------------------------------------------------


RAMb,RAMe,Tracks>90,EF,40

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor

 F>a


MOPPEL-Assembler V 8.6
Copyright (C) hms'86

 0/1/2/3: Pass #
 D: Disassembler
 E: Editor
 F: Floppy
 H: HEX-Mode
 I: In/Out on/off
 M: Monitor
 O: OCT-Mode
 S: Symbol Table

A>
```

4. A: Assembler-Aufruf

Format: F>A (ohne 'Return')

Funktion: Verlassen des Floppy-Menüs und direkter Sprung zum Assembler.

Beim Rücksprung ins Floppy-Menü, egal, ob er aus dem Assembler, Editor oder Monitor erfolgt, können die zuletzt gewählten Werte für die Parameter 'RAMb', 'RAMe' und 'Tracks' unverändert übernommen werden, wenn die entsprechende Abfrage-Zeile nicht mit 'Return' abgeschlossen, sondern mit 'Escape' oder CTL+C übergangen wird (vgl. Abschnitt 3 'Parameter-Eingabe').

hms

```
Drv.A:00     Stp/Sid:30        Dens:DD
Drv.B:xx     Tracks: 40d       Secs:16d
Drv.C:xx     RAM-Beg:90h       Byts:01h
Drv.D:xx     RAM-End:AFh       Comd:A2h
-------------------------------------------------------------------------------

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor




F>b
Num,Start>00,70
Src-Drv,Trk,Sec>0A,01,01                o.k.

F>
```

5. B: Batch Out (Vorspann für Selbst-Start ausgeben)

Format: F>B (ohne 'Return')

Num,Start>nn,aa<Ret>

Eingabe der übrigen Parameter wie in Abschnitt 3 beschrieben.

Funktion: Aufzeichnung eines Batch-Vorspanns mit der Nummer 'nn' auf Seite Ø, Spur ØØ, Sektor Ø1 der Diskette im Laufwerk A. Die Adresse aa gibt die obere Hälfte Startadresse an (untere Hälfte: ØØh), zu der nach dem Wieder-Einlesen des Vorspanns verzweigt werden soll (vgl. Abschnitt 7 'Disk in').

In dem betreffenden Batch-Vorspann werden außerdem die eingestellten Parameter 'RAMb', 'RAMe' sowie 'Src-Drv,Trk,Sec' mit aufgezeichnet; sie legen die Randbedingungen für den Datentransfer fest, der mit dem Disk-in-Aufruf gestartet wird (vgl.Abschnitt 13.3 'Batch-Parameter').

Zulässig: Für die Batch-Nummer 'nn' können die Werte ØØ...31 eingetragen werden (32 verschiedene Batch-Vorspanne pro Diskette möglich). Für die Startadresse aaaa kann jeder beliebige Wert aus dem AdreßBbereich eingesetzt werden; dies kann, muß aber keinesfalls die Startadresse desjenigen Datenblocks sein, der über den betreffenden Batch-Vorspann eingelesen wird.

Für den im Batch-Vorspann spezifizierten Datenblock gibt es keine prinzipiellen Einschränkungen, es muß für das Einlesen nur ausreichende RAM-Kapazität zur Verfügung stehen.

Einschränkung: Das automatische Einlesen mit anschließendem Programmstart ist auf das Basis-Laufwerk A, Seite Ø beschränkt ('Src-Drv' immer 'ØA'). Der Batch-Vorspann belegt dort den ersten Sektor von Spur ØØ, so daß dieser beim Batch-Betrieb nicht für die eigentliche Datenaufzeichnung genutzt werden kann.

Beim Generieren des Batch-Vorspanns wird der RAM-Bereich 2DØØ...2DFFh benutzt, d.h. die darin befindlichen Daten werden beim Batch-Betrieb überschrieben.

Quittung: o.k.

Error # 9x: Vgl. Abschnitt 2; beim Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

hms

```
Drv.A:25    Stp/Sid:30       Dens:DD
Drv.B:16    Tracks: 17d      Secs:16d
Drv.C:xx    RAM-Beg:90h      Byts:01h
Drv.D:xx    RAM-End:EFh      Comd:A2h
-------------------------------------------------------------------------------

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor




F>c
Src-Drv,Trk,Sec> b,00,01
Dst-Drv,Trk,Sec>0A, 9,01                    o.k.

F>
```

6. C: Copy (Spuren kopieren)

Format: F>C (ohne 'Return')

Eingabe der übrigen Parameter wie in Abschnitt 3 beschrieben.

Die Anzahl der zu kopierenden Spuren steht im Feld 'Tracks' der Kopfzeilen und muß, wenn sie vom angezeigten Wert abweicht, vor dem Starten des Kopiervorgangs entsprechend abgeändert werden.

Funktion: Sequentiell eine Spur nach der anderen vom Quell-Laufwerk (Src-Drv) auf das Ziel-Laufwerk (Dst-Drv) umkopieren.

Zulässig: Für die Parameter des Quell- und Ziel-Laufwerks kann jedes der vier möglichen Laufwerke A...D, jede Seite und jede Spur angegeben werden.

Einschränkung: Es können nur ganze Spuren kopiert werden, d.h. in beiden Parameter-Blöcken muß die Sektor-Angabe unverändert mit 'Øi' übernommen werden.

Beim Kopieren wird jeweils eine komplette Spur ins RAM geschrieben, beginnend bei der Adresse 'RAMb', die in den Kopfzeilen erscheint. Die in diesem RAM-Bereich befindlichen Daten werden beim Kopieren also überschrieben.

Quittung: o.k.

Error # 9x: Vgl. Abschnitt 2; beim Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

hms

```
Mm>
Drv.A:ØØ    Stp/Sid:3Ø       Dens:DD
Drv.B:xx    Tracks: 4Ød       Secs:16d
Drv.C:xx    RAM-Beg:2Dh       Byts:Ø1h
Drv.D:xx    RAM-End:2Dh       Comd:8Øh
```

```
MOPPEL-FDC-Utilities V 1Ø.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor

F>d
Num>ØØ                          o.k.
```

Ablauf beim Aufruf von Disk in Ø (Batch Ø) zum CP/M-Start:

Tr.ØØ,Sec.Ø1 -> 2DØØ,2DFFh Batch-Vorspann einlesen (Reaktion auf F>dØØ)

Tr.ØØ,Sec.Ø2 -> 28ØØ,29FFh Boot-Loader einlesen (vom Batch-Vorspann veranlaßt)

Tr.ØØ,Sec.Ø4 -> 2AØØ,2AFFh System-Parameter einlesen (gehört zum Boot-Loader)

Tr.ØØ,Sec.Ø5 -> 2BØØ,2BFFh Function-Codes einlesen (weil Byte 2AEØh nicht ØØ)

Tr.ØØ,Sec.Ø9 -> 38ØØ,3FFFh Video-Codes einlesen (weil Byte 2AE1h nicht ØØ)

Tr.Ø1,Sec.Ø1 -> D4ØØ,DBFFh CCP einlesen (vom Boot-Loader veranlaßt)

Tr.Ø1,Sec.Ø9 -> DCØØ,E9FFh BDOS einlesen (vom Boot-Loader veranlaßt)

Tr.Ø2,Sec.Ø7 -> EAØØ,F3FFh BIOS einlesen (vom Boot-Loader veranlaßt)

Sprung zur Adresse EAØØh (BIOS-Anfang; letzte Aktivität des Boot-Loaders)

Die auf dem Bildschirm ablaufende Reaktion sehen Sie links neben Blatt 14.

7. D: Disk in (Batch-Vorspann einlesen und Programm starten)

Format: F>D (ohne 'Return')

Num>nn<Ret>

Funktion: Einlesen des Batch-Vorspanns mit der Nummer 'nn', danach Einlesen des darin spezifizierten Datenblocks und anschließender Sprung zur ebenfalls angegebenen Startadresse aaØØh (vgl. Abschnitt 5 'Batch out'), d.h. automatisches Einlesen und Starten eines auf Diskette vorliegenden Programms.

Zulässig: Für nn könnnen die Nummern ØØ...31 eingegeben werden. Die Startadresse aaØØh kann, muß aber nicht die Anfangsadresse des Datenblocks sein, der mit dem Batch-Vorspann eingelesen wird.

Quittung: o.k. (vor dem Programmstart).

Error # 9x: Vgl. Abschnitt 2; bei Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

Hinweis: Der Batch-Vorspann (von Seite Ø, Spur ØØ, Sektor Ø1 der Diskette im Basis-Laufwerk A) wird in den RAM-Bereich 2DØØ...2DFFh eingelesen, d.h. die dort befindlichen Daten werden beim Disk-in-Betrieb überschrieben.

Beim Sprung zur Floppy-Festadresse 2ØØCh, der aus jedem anderen Programmteil heraus erfolgen kann, wird die automatische Disk-in-Funktion mit Batch-Vorspann Nr.ØØ ausgelöst (vgl. Abschnitt 13.2 'Fest im EPROM eingestellte Parameter'). Wenn eine Funktions-Taste also mit der Code-Folge C3h, ØCh, 2Øh (=JMP 2ØØCh) belegt ist, genügt allein die Betätigung dieser Funktions-Taste, um den vollautomatischen Disk-in-Betrieb zu starten (vgl. Abschnitt 6.1 des System-Handbuchs 'Monitor-Eingabe-Routinen CI und CSTS'). Sie ersparen sich dadurch eine ganze Reihe von Eingaben, die ansonsten für den Disk-in-Aufruf erforderlich wären (Sprung ins Floppy-Menü, RAM-Grenzen eingeben/quittieren, Disk-in-Kommando aufrufen und starten).

hms

```
Drv.A:xx     Stp/Sid:xx       Dens:xD
Drv.B:xx     Tracks: xxd      Secs:xxd
Drv.C:xx     RAM-Beg:xxh      Byts:xxh
Drv.D:xx     RAM-End:xxh      Comd:xxh
--------------------------------------------------------------------------------


RAMb,RAMe,Tracks>90,EF,40

MOPPEL-FDC-Utilities V 10.6
Copyright (C) hms'86

 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor

 F>e


MOPPEL-Editor V 6.6
Copyright (C) hms'86

 A: Assembler
 C: Clear
 F: Floppy
 G: Go to
 I: Insert
 L: List
 M: Monitor
 S: Setup RAM
 X: Extend

E>
```

8. E: Editor-Aufruf

Format: F>E (ohne 'Return')

Funktion: Verlassen des Floppy-Menüs und direkter Sprung zum Editor.

Beim Rücksprung ins Floppy-Menü, egal, ob er aus dem Assembler, Editor oder Monitor erfolgt, können die zuletzt gewählten Werte für die Parameter 'RAMb', 'RAMe' und 'Tracks' unverändert übernommen werden, wenn die entsprechende Abfrage-Zeile nicht mit 'Return' abgeschlossen, sondern mit 'Escape' oder CTL+C übergangen wird (vgl. Abschnitt 3 'Parameter-Eingabe').

hms

```
Drv.A:xx     Stp/Sid:30       Dens:DD
Drv.B:ØØ     Tracks: 8Ød      Secs:16d
Drv.C:xx     RAM-Beg:9Øh      Byts:Ø1h
Drv.D:xx     RAM-End:EFh      Comd:Ø3h
```

```
 A/E: Assembler/Editor
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor



F>f
Drv>b                                  o.k.

F>
```

9. F: Format (Diskette formatieren)

Format: F>F nn<Ret>

Drv>sL<Ret>

Funktion: Die im Feld 'Tracks' der Kopfzeilen angezeigte Anzahl von Spuren einschließlich Spur- und Sektor-Kennung auf Seite s der Diskette im Laufwerk L schreiben; im Datenbereich jeder neu formatierten Spur stehen nur Bytes mit Inhalt 'E5h, die bei CP/M-Programmen als Leer-Information interpretiert werden.

Unabhängig von der Anzahl der neu zu formatierenden Spuren beginnt das Formatieren immer bei Spur ØØ, die ganz außen auf jeder Diskette liegt.

Quittung: o.k.

Error # 9x: Vgl. Absschnitt 2; bei Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

Hinweis: Jede fabrikneue Diskette muß auf diese Weise zunächst formatiert werden, bevor man sie mit Daten beschreiben kann. Wenn eine bereits beschriebene Diskette ganz oder teilweise neu formatiert wird, gehen die darauf enthaltenen Informationen verloren (sie werden mit 'E5h' überschrieben).

hms

11. R: Read (Diskette lesen)

Format: F>R (ohne 'Return')

Eingabe der übrigen Parameter wie in Abschnitt 3 beschrieben.

Funktion: Daten sektorweise sequentiell von der Diskette ins RAM einlesen, beginnend bei den eingegebenen Parametern. Der Umfang des Datenblocks ergibt sich aus der Differenz zwischen RAM-End- und -Anfangsadresse.

Zulässig: Die kleinste Blockgröße, auf die auf der Diskette zugegriffen werden kann, ist ein Sektor; bei der fest eingestellen Sektorlänge von 256 Bytes (vgl. Abschnit 13.2 'Fest im EPROM abgespeicherte Parameter') müßte bei Angabe der RAM-Grenzen derselbe Wert für 'RAMb' und 'RAMe' eingegeben werden, wenn nur ein einziger Sektor gelesen werden soll.

Quittung: o.k.

Error # 9x: Vgl. Abschnitt 2; bei Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

hms

```
Drv.A:ØØ     Stp/Sid:3Ø       Dens:DD
Drv.B:xx     Tracks: 4Ød      Secs:16d
Drv.C:xx     RAM-Beg:2Ah      Byts:Ø1h
Drv.D:xx     RAM-End:2Ah      Comd:A2h
```

```
 B/D: Batch out/Disk in
 C/F: Copy/Format
 R/W: Read/Write
  M:  Monitor
```

```
F>w
Dst-Drv,Trk,Sec>ØA,ØØ,Ø4                    o.k.

F>w
Dst-Drv,Trk,Sec>a,Ø,4                       o.k.

F>
```

12. W: Write (Diskette beschreiben)

Format: F>W (ohne 'Return')

Eingabe der übrigen Parameter wie in Abschnitt 3 beschrieben.

Funktion: Daten sektorweise sequentiell aus dem Speicher auf die Diskette überschreiben, beginnend bei den eingegebenen Parametern. Der Umfang des Datenblocks ergibt sich aus der Differenz zwischen RAM-End- und -Anfangsadresse.

Zulässig: Die kleinste Blockgröße, auf die auf der Diskette zugegriffen werden kann, ist ein Sektor; bei der fest eingestellen Sektorlänge von 256 Bytes (vgl. Abschnit 13.2 'Fest im EPROM gespeicherte Parameter') müßte bei Angabe der RAM-Grenzen derselbe Wert für 'RAMb' und 'RAMe' eingegeben werden, wenn nur ein einziger Sektor gelesen werden soll.

Quittung: o.k.

Error # 9x: Vgl. Abschnitt 2; bei Erkennen eines Fehlers wird das laufende Kommando unmittelbar abgebrochen.

hms

13.1 Anschlußbelegung der Verbindungskabel

Die Verbindung der Laufwerk-Elektronik mit dem Floppy-Disk-Controller erfolgt über ein 34poliges Flachbandkabel mit Direktsteckern. Die Belegung dieser Schnittstelle ist standardisiert und unter der Bezeichnung '34polige Shugart-Schnittstelle' ein marktüblicher Begriff.

Bezogen auf die Stiftleiste des Floppy-Disk-Controllers sind alle Ausgangs-Signale ('O') Open-Collector-Ausgänge (aktiv LOW), die auf der Laufwerk-Elektronik mit jeweils 470 Ohm abgeschlossen sein sollen; alle Eingangs-Signale ('I', ebenfalls aktiv LOW) werden auf dem Floppy-Disk-Controller mit 470-Ohm-Pull-Up-Widerständen abgeschlossen.

Die Anschlüsse sind wie folgt belegt ('n.c.'=no connection, d.h. Anschluß nicht belegt; in Klammern Angabe der Bus-Leitung a/b/c32):

```
                    ========
GND  (x32)   ! 1     2 !     n.c. (schwarze Kenn-Ader)
GND  (x32)   ! 3     4 !     n.c.
GND  (x32)   ! 5     6 !  O: Drive 3 Select
GND  (x32)   ! 7     8 !  I: Index Pulse
GND  (x32)   ! 9    10 !  O: Drive 0 Select
GND  (x32)   ! 11   12 !  O: D
GND  (x32)   ! 13   14 !  O: Drive 2 Select
GND  (x32)   ! 15   16 !  O: Motor On
GND  (x32)   ! 17   18 !  O: Direction Select
GND  (x32)   ! 19   20 !  O: Step
GND  (x32)   ! 21   22 !  O: Write Data
GND  (x32)   ! 23   24 !  O: Write Gate
GND  (x32)   ! 25   26 !  I: Track 00
GND  (x32)   ! 27   28 !  I: Write Protect
GND  (x32)   ! 29   30 !  I: Read Data
GND  (x32)   ! 31   32 !  O: Side One Select
GND  (x32)   ! 33   34 !     n.c.
                --------
```

Die jeweils vierpoligen Stromversorgungs-Anschlüsse werden direkt mit dem Netzteil verlötet. Die acht dafür auf der Bus-Platine vorgesehenen Lötaugen sind entsprechend gekennzeichnet (GND=Masse; +5 V; +12 V; in Klammern die Pins des Netzteil-Steckplatzes am Bus):

```
                  ----
+12 V (c16)   ! 1 !    Motor-Stromversorgung
 GND  (x32)   ! 2 !    Masse für +12 V
 GND  (x32)   ! 3 !    Masse für +5 V
 +5 V (x24)   ! 4 !    Elektronik-Stromversorgung
                  ----
```

Wichtiger Hinweis: Für den einwandfreien Betrieb der Laufwerke (Datenrate 250000 Bits pro Sekunde!) ist eine saubere und stabile Stromversorgung die Grundvoraussetzung! Bei Speisung über fremde Netzteile muß eine Masse-Verbindung zwischen externem und Mikrocomputer-Netzteil hergestellt werden! Solange die Stromversorgung nicht in Ordnung ist, werden ständige Störungen auf Dauer keinen ordnungsgemäßen Betrieb zulassen.

13.2 Fest im EPROM eingestellte Parameter

Beim externen Einsprung ins Floppy-Menü (aus dem Monitor, Assembler oder Editor) werden folgende Kaltstart-Parameter aus dem EPROM in einen dafür vorgesehenen RAM-Buffer überschrieben. Die enstsprechenden RAM-Zellen können auch manuell mit anderen Werten geladen werden (mit Hilfe des Monitors); in diesem Fall muß der Einsprung ins Floppy-Menü über die Warmstart-Adresse 2003h erfolgen:

	Bedeutung	EPROM		RAM	Grenzwerte
Stp	(Stepping Rate)	2029h	(=03h) ->	2E9Ah;	0=6; 1=12 ms 2=20;3=30 ms
Tracks	(Tracks per Side)	202Ch	(=28h) ->	2E9Bh;	1...4Fh
RAM-Beg	(RAM-Begin)	202Ah	(=90h) ->	2E9Ch;	90=9000h
RAM-End	(RAM-End)	202Bh	(=EFh) ->	2E9Dh;	EF=EFFFh
Dens	(Density)	2026h	(=00h) ->	2E90h;	0=DD; 1=SD
Secs	(Sectors per Track)	2027h	(=10h) ->	2E91h;	1...10h
Byts	(Bytes per Sector)	2028h	(=01h) ->	2E92h;	0=128;1=256 2=512;3=1024
Sid	(Side No.)	Bildschirm	->	2E99h;	0=lo; 1=hi
Drive	(Drive No.)	Bildschirm	->	2EA7h;	0...3=A...D
Track	(Track No.)	Bildschirm	->	2EA8h;	0...4Fh
Sector	(Sector No.)	Bildschirm	->	2EA9h;	1...10h
Records	(Number of Records)	vgl.Hinweis	->	2EA0h;	1...RAMmax

Für externe Einsprünge sind folgende Festadressen reserviert:

2000h FKALT	Floppy-Kaltstart	Aufruf vom Monitor, Assembler, Editor
2003h WARMEX	Floppy-Warmstart	Externer Einsprung (Warmstart) nach manueller Parameter-Vorgabe (s.o.)
2006h WRBLOK	Block schreiben	Parameter-Buffer 2Exx laden (s.o.)
2009h RDBLOK	Block lesen	Parameter-Buffer 2Exx laden (s.o.)
200Ch BATCH0	Batch-0-Aufruf	Vgl. Abschnitt 7 'Disk in'

Hinweis: Die Anzahl der Sektoren (Records) wird automatisch aus den RAM-Grenzen errechnet: Records=RAMe-RAMb+1

hms

13.3 Batch-Parameter

Die Aufzeichnung und das spätere Einlesen eines Batch-Vorspanns (vgl. Abschnitt 5 'Batch out' bzw. Abschnitt 7 'Disk in') dienen dazu, Programme oder andere Datenblöcke automatisch von Diskette einzulesen und anschließend (ebenfalls automatisch) ein Programm zu starten.

Zur Abwicklung dieser Aufgaben ist auf der Diskette der erste Sektor reserviert (Seite Ø, Laufwerk A, Spur ØØ, Sektor Ø1); die Aufbereitung der entsprechenden Parameter erfolgt im RAM-Bereich 2DØØ... 2DFFh, von wo aus sie auf die Diskette geschrieben (bei 'Batch out') bzw. wohin sie von Diskette geladen werden (bei 'Disk in').

Pro Batch-Vorspann werden acht Bytes erzeugt; bei 32 möglichen Batches (Nr.Ø...31d) wird also zur Aufbereitung bzw. Aufzeichnung (beides erfolgt automatisch) ein Speicherplatz von 256 Bytes (=1 Sektor) benötigt, der folgendermaßen organisiert ist:

```
2DØØ...2DØ7h: Parameter für Batch-Vorspann Nr.Ø:
  ØØ          Startadresse nach Einlesen (lower)
  Ø1          Startadresse nach Einlesen (upper)
  Ø2          Anzahl der einzulesenden Sektoren (Records)
  Ø3          >Byte wird nicht ausgewertet<
  Ø4          Sektor-Nr. (Beginn des Programms/Datenblocks)
  Ø5          Spur-Nr. (Beginn des Programms/Datenblocks)
  Ø6          RAM-Adresse (lower; Beginn d. Programm/Daten-Speichers)
  Ø7          RAM-Adresse (upper; Beginn d. Programm/Daten-Speichers)
2DØ8...2DØFh: Parameter für Batch-Vorspann Nr.1
2D1Ø...2D17h: Parameter für Batch-Vorspann Nr.2
2D18...2D1Fh: Parameter für Batch-Vorspann Nr.3
2D2Ø...2D27h: Parameter für Batch-Vorspann Nr.4
2D28...2D2Fh: Parameter für Batch-Vorspann Nr.5
2D3Ø...2D37h: Parameter für Batch-Vorspann Nr.6
2D38...2D3Fh: Parameter für Batch-Vorspann Nr.7
2D4Ø...2D47h: Parameter für Batch-Vorspann Nr.8
2D48...2D4Fh: Parameter für Batch-Vorspann Nr.9
2D5Ø...2D57h: Parameter für Batch-Vorspann Nr.1Ø
2D58...2D5Fh: Parameter für Batch-Vorspann Nr.11
2D6Ø...2D67h: Parameter für Batch-Vorspann Nr.12
2D68...2D6Fh: Parameter für Batch-Vorspann Nr.13
2D7Ø...2D77h: Parameter für Batch-Vorspann Nr.14
2D78...2D7Fh: Parameter für Batch-Vorspann Nr.15
2D8Ø...2D87h: Parameter für Batch-Vorspann Nr.16
2D88...2D8Fh: Parameter für Batch-Vorspann Nr.17
2D9Ø...2D97h: Parameter für Batch-Vorspann Nr.18
2D98...2D9Fh: Parameter für Batch-Vorspann Nr.19
2DAØ...2DA7h: Parameter für Batch-Vorspann Nr.2Ø
2DA8...2DAFh: Parameter für Batch-Vorspann Nr.21
2DBØ...2DB7h: Parameter für Batch-Vorspann Nr.22
2DB8...2DBFh: Parameter für Batch-Vorspann Nr.23
2DCØ...2DC7h: Parameter für Batch-Vorspann Nr.24
2DC8...2DCFh: Parameter für Batch-Vorspann Nr.25
2DDØ...2DD7h: Parameter für Batch-Vorspann Nr.26
2DD8...2DDFh: Parameter für Batch-Vorspann Nr.27
2DEØ...2DE7h: Parameter für Batch-Vorspann Nr.28
2DE8...2DEFh: Parameter für Batch-Vorspann Nr.29
2DFØ...2DF7h: Parameter für Batch-Vorspann Nr.3Ø
2DF8...2DFFh: Parameter für Batch-Vorspann Nr.31
```

13.4 Kommandos des FDC-Steuerbausteins WD1770

Im Feld 'Comd' der Kopfzeilen erscheint ständig das augenblicklich (bzw. zuletzt) ausgeführte Kommando des FDC-Steuerbausteins WD1770. Die Arbeitsweise des Disketten-Betriebes wird damit transparenter, und Sie können jede einzelne Aktion bis ins Detail verfolgen.

Ein vermeintliches Flimmern im Comd-Feld rührt nicht etwa von einer Fehlfunktion her, sondern die vom Steuerbaustein ausgeführten Kommandos wechseln dann nur in so schneller Folge, daß sie mit bloßem Auge nicht mehr ohne weiteres auseinanderzuhalten sind.

Im einzelnen haben die Anzeigen folgende Bedeutung:

```
00h +s: Restore; Kopf auf Spur 00 zurückfahren (ganz außen).
10h +s: Seek; Kopf positionieren; die Nummer der gewünschten
              Spur erscheint in der betreffenden Kopfzeile 'Drv.x'
50h +s: Step in; Kopf um eine Spur nach innen verschieben.
80h:    Read Sector; Sektor lesen.
A0h +p: Write Sector; Sektor schreiben.
D0h:    Force Interrupt; Kommando abbrechen (bei Fehler).
F0h +p: Write Track; Komplette Spur schreiben (formatieren).
-----^ ------------------------------------------------------------
     p: Precompensation off (=2) bis Spur 42d; danach on (p=0)
     s: Stepping Rate 0...3 (im Feld 'Stp/Sid' angezeigt)
```

```
Drv.A:02    Stp/Sid:30       Dens:DD
Drv.B:xx    Tracks: 40d      Secs:16d
Drv.C:xx    RAM-Beg:90h      Byts:01h
Drv.D:xx    RAM-End:EFh      Comd:80h
-------------------------------------------------------------------------------

MOPPEL-CP/M-Boot-Loader V 11.6   <<<< vom Batch-0-Vorspann veranlaßt
Copyright (C) hms'86

Initializing Printer             <<<< Drucker-Initialisierung standardmäßig

Loading Function Codes           <<<< weil das Byte 2AE0h ungleich 00 ist

Copying Video Codes              <<<< weil das Byte 2AE1h ungleich 00 ist

Switching to Bank 1              <<<< automatische Bankumschaltung


This is 60 K CP/M Rel. 2.2       <<<< CP/M-Bereitmeldung

A>                               <<<< Prompt
```

14. CP/M-Betriebssystem

Das von der Firma Digital Research Incorporated entwickelte und in Lizenz vertriebene Betriebssystem 'CP/M' (Control Programming for Microprocessors) ist auch für den MOPPEL angepaßt worden.

Es handelt sich hierbei um ein Disketten-orientiertes Betriebssystem (Version 2.2), das einen RAM-Arbeitsspeicher ab Adresse ØØØØh voraussetzt und das genau definierte Schnittstellen zur System-Hardware verwendet. Diese Randbedingungen erfüllt die MOPPEL-Hard- und Software, u.a. durch die automatische Bank-Umschaltung auf der Speicher-Karte.

Ziel dieses Software-Pakets ist es, durch die Normung der Hard- und Software-Schnittstellen einheitliche Betriebsbedingungen für Computer unterschiedlicher Fabrikate zu schaffen, so daß auf verschiedenartigen Geräten dieselben Programme laufen können (Standardisierung).

CP/M selbst (CCP und BDOS) sowie die Anpassungs-Routinen (Boot-Loader und BIOS) sind uneingeschränkt auf dem 8Ø85-Mikroprozessor lauffähig; ein Großteil der unter CP/M laufenden Programme, die auf dem Markt angeboten werden (z.B. 'WordStar', MBASIC oder PASCAL), setzt allerdings Z-8Ø-Kompatibilität voraus, d.h. in diesen Fällen muß auf der CPU das Z-8Ø-Modul nachgerüstet werden; andere Anpassungen sind nicht erforderlich, die Treiber-Routinen sind sämtlich Z-8Ø-kompatibel geschrieben.

Das vom Floppy-Disk-Controller und dem BIOS unterstützte Disketten-Format entspricht der ECMA-7Ø-Norm: 16 Sektoren pro Spur, Sektorlänge 256 Bytes und doppelte Schreibdichte (MFM), CP/M-Version 2.2 (von Bedeutung für die Format-Angabe bei fremder Software).

Der Aufruf von CP/M erfolgt über die Disk-in-Funktion Nr.Ø (Batch Ø); Sie können dies von jedem anderen Programmteil aus veranlassen, wenn Sie eine Funktionstaste mit der Code-Folge C3h, ØCh, 2Øh belegen (=JMP 2ØØCh; vgl. Abschnitt 7 'Disk in').

Beachten Sie ferner die Möglichkeit, jeder Taste der ASCII-Tastatur (und der ASCII-Erweiterung) eine Folge von vier frei definierbaren Bytes zuzuordnen; viele CP/M-Programme (insbesondere solche zur Textverarbeitung!) verlangen für bestimmte Funktionen vorgeschriebene Tasten-Sequenzen, die sich auf diese Weise extrem einfach simulieren lassen: Die gleichzeitige Betätigung der Funktionstaste mit nur einer weiteren Taste liefert bis zu vier aufeinanderfolgende Codes (vgl. Abschnitt 6.1 des System-Handbuchs 'Monitor-Eingabe-Routinen CI und CSTS').

Natürlich sind auch unter CP/M die fest programmierten Funktionstasten FCT+U (=Uhrzeit-Aufruf) und FCT+V (=Bildschirm-Kopie ins RAM) aktiv, von denen Sie jederzeit Gebrauch machen können.

14.1 Laufwerk-Anpassung für CP/M

Auf jeder für den MOPPEL gelieferten CP/M-System-Diskette ab Version V.6 sind die Systemspuren ab Tr.ØØ wie folgt belegt:

```
Tr.ØØ,Sec.Ø1 -> 2DØØ...2DFFh: Batch-Parameter (vgl. Abschnitt 13.3)
      Sec.Ø2 -> 28ØØ...29FFh: CP/M-Boot-Loader  V 11.6; Start:  F>DØØ
      Sec.Ø4 -> 2AØØ...2AFFh: System-Parameter (s.u.)
      Sec.Ø5 -> 2BØØ...2CFFh: Function-Codes >(vgl. Abschnitt 7.5 des
      Sec.Ø9 -> 38ØØ...3FFFh: Video-Codes    > System-Handbuchs)

Tr.Ø1,Sec.Ø1 -> D4ØØ...DBFFh: Console Command Processor (6Ø-K-CCP)
      Sec.Ø9 -> DCØØ...E9FFh: Basic Disk Operating System (6Ø-K-BDOS)
Tr.Ø2,Sec.Ø7 -> EAØØ...F3FFh: Basic Input/Output System (6Ø-K-BIOS)

Tr.Ø2,Sec.16 -> 9ØØØ...9ØFFh: (C)opyright-Vermerk DIR           F>DØ1

Tr.Ø3,Sec.Ø1 -> 9ØØØ...9FFFh: Text 'Laufwerk-Anpassung für CP/M'F>DØ2
```

System-Parameter Tr.ØØ,Sec.Ø4/RAM 2AØØ...2AFFh:

```
2AØØ...2A7Fh: Translate Tables XLTØ...XLT3 (je 2Øh Bytes)
2A8Ø...2ABFh: Angepaßte DPBs Ø...3 (vgl. Abschnitt 14.2)
2ACØh:        CP/M-Kaltstart-I/O-Byte (Voreinstellung: 8Øh)
2AC1h:        CP/M-Kaltstart-Driver/User (Voreinstellung: ØØh)
2ADØ...2AD3h: Drive-A-Parameter (4 Bytes):
  DØ          Typ    (ØØ=4ØTr/1K;Ø1=8ØTr/2K;Ø2=2x8ØTr/2K;Ø3=2x8ØTr/4K)
  D1          Step   (ØØ= 6 ms;  Ø8=12 ms;  1Ø=2Ø ms;  18=3Ø ms)
  D2          Sides (ØØ=ein, Ø1=zwei Schreib/Lese-Köpfe)
  D3          Tracks (Spuren pro Seite; maximal 4Fh=79d)
2AD4...2AD7h: Drive-B-Parameter (4 Bytes, vgl. Drive-A-Parameter)
2AD8...2ADBh: Drive-C-Parameter (4 Bytes, vgl. Drive-A-Parameter)
2ADC...2ADFh: Drive-D-Parameter (4 Bytes, vgl. Drive-A-Parameter)
2AEØh:        Function Codes nach 2BØØ...2CFFh laden (wenn nicht ØØ)
2AE1h:        Video-Codes nach 38ØØ...3FFFh laden (wenn nicht ØØ)
```

Um die System-Parameter zu modifizieren, laden Sie von einer CP/M-System-Diskette Spur ØØ, Sektor Ø4 nach 2AØØ...2AFFh ins RAM:

```
  RAM-Beg:2A; RAM-End:2Ah; Read Src-Drv,Trk,Sec>ØA,ØØ,Ø4<Ret>
```

Sie können jetzt jedes der verwendeten Laufwerke anpassen, indem Sie im Bereich 2ADØ...2ADFh die Daten für Laufwerk-Typ, Stepping-Rate, Sides (Kopf-Anzahl) und Tracks eingeben und dann den RAM-Bereich 2AØØ...2AFFh zurück auf die Diskette schreiben:

```
  RAM-Beg:2A; RAM-End:2Ah; Write Dst-Drv,Trk,Sec>ØA,ØØ,Ø4<Ret>
```

Ähnlich gehen Sie vor, um die CP/M-System-Parameter 'I/O-Byte' und 'Driver/User' zu ändern oder um Function- und/oder Video-Codes beim Kaltstart neu zu laden (korrespondierende Bytes ungleich ØØ setzen).

Die zu jedem Laufwerk gehörenden Disk-Parameter-Blocks werden vom Boot-Loader automatisch errechnet, in den RAM-Bereich 2A8Ø...2ABFh kopiert und beim Kaltstart an das BIOS übergeben; Sie brauchen sich also darum nicht mehr zu kümmern, nachdem Sie die Laufwerke typmäßig spezifiziert haben (vgl. nächsten Abschnitt 14.2 'Voreingestellte Laufwerk-Typen').

hms

14.2 Voreingestellte Laufwerk-Typen

Im Bereich 2AD0...2ADFh der System-Parameter (auf Spur 00, Sektor 04 jeder CP/M-System-Diskette) sind die vier möglichen Laufwerke A...D typmäßig spezifiziert (vgl. vorigen Abschnitt 14.1 'Laufwerk-Anpassung für CP/M'). Diese Spezifikationen können Sie, wie im vorigen Abschnitt beschrieben, jederzeit ändern.

Folgende Voreinstellungen sind auf der Diskette vorgenommen:

```
2A00h = 00h: Laufwerk A: Typ 0 = 1x40 Spuren (1-K-Blockgröße)
2A01h = 08h:             Stepping Rate = 6 ms
2A02h = 00h:             einseitig (d.h. ein Schreib/Lese-Kopf)
2A03h = 28h:             Spur-Anzahl pro Seite = 40d
2A04h = 01h: Laufwerk B: Typ 1 =2x40 bzw.1x80 Spuren (2-K-Blockgröße)
2A05h = 08h:             Stepping Rate = 6 ms
2A06h = 00h:             einseitig (d.h. ein Schreib/Lese-Kopf)
2A07h = 50h:             Spur-Anzahl pro Seite = 80d
2A08h = 02h: Laufwerk C: Typ 2 = 2x80 Spuren (2-K-Blockgröße)
2A09h = 18h:             Stepping Rate = 30 ms
2A0Ah = 01h:             zweiseitig (d.h. zwei Schreib/Lese-Köpfe)
2A0Bh = 50h:             Spur-Anzahl pro Seite = 80d
2A0Ch = 03h: Laufwerk D: Typ 3 = 2x80 Spuren (4-K-Blockgröße)
2A0Dh = 18h:             Stepping Rate = 30 ms
2A0Eh = 01h:             zweiseitig (d.h. zwei Schreib/Lese-Köpfe)
2A0Fh = 50h:             Spur-Anzahl pro Seite = 80d
```

Die Daten für die zugehörigen Disk-Parameter-Blocks (DPBs) stehen am Ende des Boot-Loaders (im RAM-Bereich 29C0...29FFh nach dem Einlesen von Spur 00, Sektoren 2&3 ins RAM 2800...29FFh). Pro Laufwerk-Typ sind hier 16d Bytes pro DPB reserviert, die wie folgt organisiert sind:

```
29C0...29CFh: DPB für Laufwerk-Typ 0 (1x40 Tr./1-K-Blocks)
  C0          SPT (Sectors per Track); 2 Bytes
  C2          BSH (Block Shift); 1 Byte
  C3          BLM (Block Mask); 1 Byte
  C4          EXM (Extension Mask); 1 Byte
  C5          DSM (Disk Size Memory); 2 Bytes
  C7          DRM (Directory Memory); 2 Bytes
  C9          AL0 (Allocation Vector 0); 1 Byte
  CA          AL1 (Allocation Vector 1); 1 Byte
  CB          CKS (Check Size Vector); 2 Bytes
  CD          OFF (Offset); 2 Bytes
  CF          --- redundant; 1 Byte
29D0...29DFh: DPB für Laufwerk-Typ 1 (2x40 oder 1x80 Tr./2-K-Blocks)
29E0...29EFh: DPB für Laufwerk-Typ 2 (2x80 Tr./2-K-Blocks)
29F0...29FFh: DPB für Laufwerk-Typ 3 (2x80 Tr./4-K-Blocks)
```

Die Parameter-Blocks können bei Bedarf auch an andere Laufwerke angepaßt werden (einfache Orientierungsmöglichkeit durch die 'gerade' Adreßzuordnung). Bei Änderungen ist zu beachten, daß die im Bereich 29C0... 29FFh liegenden Parameter-Blocks Laufwerk-Typen kennzeichnen und nicht direkt mit den Laufwerken A...D korrespondieren! Die Zuordnung von Laufwerk A...D und Typ 0...3 erfolgt über die Drive-Parameter A...D im RAM-Bereich 2AD0...2ADFh (vgl. 'System-Parameter' im Abschnitt 14.1 'Laufwerk-Anpassung für CP/M').

hms

MOPPEL
System-Software

BASIC-
Handhabung

Modulares Prozessor-Programm zum Entwickeln und Lernen

® **Eingetragenes Warenzeichen**

hms

Inhaltsverzeichnis

Ø. Installation und Aufruf
Hardware-Voraussetzungen und Software-Aktivierung

1. BASIC-Grundmenü
1.1 Convert (alte BASIC-Programme umsetzen)
1.2 Kalt (Interpreter-Kaltstart)
1.3 Monitor (Direkter Monitor-Rücksprung: 'm'./.BYE)
1.4 Warm (Interpreter-Warmstart)
1.5 Extern (Externen Eingabe-Kanal aktivieren)

2. Fehlermeldungen
Zusammenstellung aller BASIC-Fehlermeldungen

3. BASIC-Befehlssatz
Allgemeine Bedienungshinweise

4. Bildschirm- und Drucker-Ansteuerung, Zufallszahl
Befehle: CLRSCN, CLRLIN, CRSPOS; NULL, SPC, POS, TAB; RND;
PRINT, <,>, <;>, LPRINT, PPRINT

5. Operatoren, Rechen- und Anwender-Funktionen
Operatoren: +, -, *, /, ( ); <, =, >; ^, E;
Befehle: SQR, EXP, LOG; LET, DEF FN

6. Ein/Ausgaben (Console, RAM und Ports), logische Operationen
Befehle: PEEK, POKE, INP, OUT, WAIT;
AND, OR, NOT

7. Stringverarbeitung, Typ-Umwandlung
Befehle: TI$, LEFT$, MID$, RIGHT$;
VAL, STR$, LEN; CHR$, ASC, HEX

8. Daten-Bereitstellung und -Abruf
Befehle: DATA, READ, RESTORE; DIM, CLEAR;
INT, SGN, ABS

9. Trigonometrische Funktionen
Befehle: SIN, COS, TAN, ATN

1Ø. Programmausführung
Befehle: RUN, STOP, CONT, END; GOTO, USR;
ON...GOTO, IF...GOTO;
FOR...NEXT; GOSUB, RETURN, CALL

11. Programm-Verwaltung und -Speicherung
Befehle: NEW, FRE, REM; LIST, LLIST; CSAVE, CLOAD

hms

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>o

A: Assembler
B: BASIC
D: Disassembler
E: Editor
F: Floppy
P: Prommer
X: Ext.2000h

Mo>b
```

Ø. Installation und Aufruf

Der ROM-residente BASIC-Interpreter ist im EPROM 'grün' enthalten; er wird auf Platz #4 eingesetzt (8-K-EPROM 2764) oder auf den Plätzen #4&5 (bei zwei 4-K-EPROMs 2732).

Die Zusammensetzung des Kunstwortes 'BASIC' macht bereits die hauptsächliche Zielsetzung dieser Programmiersprache deutlich (Beginner's All Purpose Symbolic Instruction Code): Eine für den Anfänger (und für einfache Aufgaben) konzipierte symbolische Programmiersprache.

Das MOPPEL-BASIC enthält den Grundbefehlssatz des ursprünglich von der Firma Microsoft erstellten 8-K-BASIC-Interpreters, der im Hinblick auf maschinennahe Einsatzmöglichkeiten erweitert worden ist. Und genau auf diesem Aspekt der engen Verknüpfung von Hard- und Software liegt das Schwergewicht des ROM-residenten MOPPEL-BASICs, so z.B. durch Realisierung von Befehlen zur direkten Bildschirm- oder Drucker-Steuerung.

Es ist nicht Ziel dieses auf den engen Speicherplatz von 8 K beschränkten Interpreters, eine hohe Rechengenauigkeit zu bieten, wie sie z.B. für Hypotheken- oder Zinsberechnungen benötigt wird; dafür stehen Disketten-orientierte BASIC-Versionen zur Verfügung (z.B. solche, die unter CP/M laufen). Dagegen unterstützt die ASCII-Tastatur-Erweiterung den Aufruf der Grundrechenarten, damit Sie Ihren Computer im Labor auch als Rechner einsetzen können (fest verdrahtete Operatoren +, -, *, / sowie '?' für 'PRINT').

Vom Monitor-Grundmenü aus erfolgt der BASIC-Aufruf über das OUTWARD-Menü: Eingabe von 'o' (ohne 'Return'), gefolgt von 'b' (wiederum ohne 'Return'). Im Unterschied zu allen anderen ROM-residenten Programmpaketen müssen Sie anschließend den Einsprung in den Interpreter noch näher spezifizieren (vgl. Abschnitt 1 'BASIC-Grundmenü').

Durch die Standardisierung von BASIC weichen einige Bedienfunktionen vom gewohnten Standard ab (Ersatz der Escape-Taste durch CTL+C; vgl. auch Abschnitte 1Ø und 11). An der Drucker-Parallelschaltung ändert sich gegenüber der üblichen Vorgehensweise nichts, d.h. Sie können (sofern das Universal-Interface vorhanden ist), einen angeschlossenen Drucker jederzeit zu- oder wieder abschalten (über CTL+7...CTL+4).

Bei Verwendung der von BASIC gebotenen Drucker-Befehle wid das I/O-Byte aus Speicherzelle 4ØØ6h geladen (Standardwert: 8Øh, d.h. die Drucker-Ansteuerung erfolgt in diesen Fällen über die serielle Schnittstelle).

Auch der BASIC-Interpreter unterstützt zwei Bildformate, die abhängig sind von der Einstellung des Video-Interfaces: 24 Zeilen mit je 8Ø Zeichen oder 2Ø Zeilen mit je 4Ø Zeichen. Im übrigen sei auf die ausführlichen Beschreibungen des System-Handbuchs verwiesen.

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->
```

1. BASIC-Grundmenü

Nach dem Aufruf des BASIC-Interpreters vom Monitor-OUTWARD-Menü aus müssen Sie spezifizieren, auf welchem von vier möglichen Wegen Sie fortfahren wollen, oder ob Sie vom BASIC-Grundmenü aus direkt wieder zurück in den Monitor springen wollen; die Kennbuchstaben des Grundmenüs können als Kleinbuchstaben eingegeben werden; zur Hervorhebung sind sie in den folgenden Abschnitten in Großschreibung dagestellt.

C (Convert): Dient zur Umsetzung alter BASIC-Programme (vgl. Abschnitt 1.1); vor dem Aufruf der C-Anweisung muß der BASIC-Interpreter mindestens einmal kaltgestartet worden sein (Initialisierung aller internen Parameter).

K (Kaltstart): Der normale Einstieg in den Interpreter, bei dem alle internen Parameter initialisiert und etwa vorhandene Programme gelöscht werden (vgl. Abschnitt 1.2).

M (Monitor-Rücksprung): Zusätzlich zum BASIC-Befehl 'BYE' besteht im Grundmenü die Möglichkeit, durch Eingabe eines 'm' (ohne 'Return') direkt zum Monitor-Kaltstart zurückzuspringen (vgl. Abschnitt 1.3).

W (Warmstart): Einstieg in den Interpreter mit Übernahme aller vorhandenen Parameter und Programme; der Warmstart setzt voraus, daß zuvor mindestens einmal kaltgestartet wurde (vgl. Abschnitt 1.4).

X (Extern): Aktivieren des externen Eingabe-Kanals (über das RAM), über den (anstelle der Tastatur) alle folgenden Anweisungen an den Interpreter ablaufen (einschließlich der automatischen Start-Möglichkeit für ein auf diesem Weg eingelesenes Programm; (vgl. Abschnitte 1.5 und 11).

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->w

o.k.

B>list

10 DIM D(12)
30 FOR N^1 CLRLIN 12
40 T^TTI$D(M)
45 Y^JTHEN1601

o.k.

BASIC->c

o.k.

B>list

10 DIM D(12)
30 FOR N=1 TO 12
40 T=T+D(M)
45 Y=J-1601

o.k.

B>
```

1.1 C: Convert (alte BASIC-Programme umsetzen)

Format: B>C (ohne 'Return')

Funktion: Umsetzen von BASIC-Programmen, die mit der Version 3.3 des MOPPEL-BASIC-Interpreters erstellt worden sind, so daß sie auf der aktuellen Interpreter-Version 3.6 lauffähig sind. Das Programm steht anschließend in aktualisierter Form im RAM und kann bei Bedarf wieder abgespeichert werden (vgl. Abschnitt 11).

Quittung: o.k.

Voraussetzung: Der BASIC-Interpreter muß (nach dem Einschalten der Versorgungsspannung) mindestens einmal kaltgestartet worden sein, und das ursprüngliche, umzusetzende Programm muß in demselben Speicherbereich ab Adresse 8000h stehen, von dem aus es abgespeichert wurde.

Hinweis: Programme, die mit der Version 3.4 oder 3.5 des BASIC-Interpreters erstellt worden sind, können ohne Änderungen auf die Version 3.6 übertragen werden. Sie müssen nicht (und dürfen nicht!) mit der C-Anweisung umgesetzt werden, weil dies zu Fehl-Interpretationen führen würde.

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->k              <<<  Kaltstart (alles löschen)

o.k.

B>list                <<<  Kontrolle: 'LIST' liefert keine Programmzeile


o.k.

B>? fre(y)
 32500                <<<  freier RAM-Bereich für den Anwender

o.k.

B>
```

1.2 K: Kalt (Interpreter-Kaltstart)

Format: B>K (ohne 'Return')

Funktion: Initialisierung des BASIC-Interpreters durch Rücksetzen aller Parameter in den Grundzustand (u.a. Rücksetzen des DATA-Pointers sowie aller Feld- und Bereichs-Dimensionierungen, Setzen des dynamisch verschiebbaren Stack-Pointers ans obere Ende des hierbei ermittelten Speicherbereichs).

Quittung: o.k.

Hinweis: Obwohl der Inhalt des gesamten (bei Adresse 80C4h beginnenden) Programmspeichers hierdurch nicht verändert wird, ist ein eventuell vorhandenes Programm nach einem Kaltstart verloren, weil die zu seiner Verwaltung notwendigen Parameter gelöscht sind. Achten Sie darum sorgfältig darauf, daß Sie anstelle eines beabsichtigten Warmstarts nicht versehentlich einen Kaltstart auslösen!

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->m


MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>
```

1.3 M: Monitor (Direkter Monitor-Rücksprung)

Format: B>M (ohne 'Return')

Funktion: Direkter Rücksprung aus dem BASIC-Interpreter zum Monitor-Kaltstart. Hierbei wird am Inhalt des gesamten Speichers nichts verändert, d.h. ein etwa vorhandenes Programm kann nach einem späteren Warmstart des Interpreters im alten Zustand übernommen und weiterbearbeitet werden.

Hinweis: Außer vom BASIC-Grundmenü aus ist der Monitor-Rücksprung auch über das BASIC-Direkt-Kommando 'BYE' möglich.

Eventuell vorgenommene Drucker-Einstellungen (vgl. Abschnitt 4: PPRINT) werden im Monitor unverändert übernommen.

hms

```
MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->w                    <<< Warmstart (vorhandene Programme übernehmen)

o.k.

B>list                      <<< Kontrolle: Altes Programm listen

10 REM  Test-Beispiel
20 PRINT TI$
30 REM  das war's

o.k.

B>
```

1.4 W: Warm (Interpreter-Warmstart)

Format: B>W (ohne 'Return')

Funktion: Einsprung in den BASIC-Interpreter ohne Modifikation des Speicherinhalts, d.h. etwa vorhandene Programme werden (mit allen für ihre Verwaltung erforderlichen Parametern) unverändert übernommen.

Voraussetzung: Ein Aufruf des W-Kommandos setzt voraus, daß (nach dem Einschalten der Versorgungsspannung) mindestens einmal kaltgestartet worden ist, um sämtliche Parameter in den Ausgangszustand zu bringen.

Quittung: o.k.

Hinweis: Der direkte Start eines im Speicher befindlichen BASIC-Programms ist aus jedem anderen Programm heraus möglich, indem zur Adresse 4003h gesprungen wird; dies wirkt wie ein automatischer Warmstart mit anschließender, automatischer Eingabe von 'RUN' (vgl. auch Abschnitt 11).

hms

```
E>1:0013/92A1

0001  10 rem  Dieses BASIC-Programm wurde (einschließlich der BASIC-Zeilen-
0002  11 rem  nummern) im Editor erstellt und per X-Anweisung direkt aus dem
0003  12 rem  RAM an den Interpreter übergeben; da als letzte Anweisung (in
0004  13 rem  der Editor-Zeile Nr.12) 'RUN' steht, wird das Programm nach dem
0005  14 rem  Einlesen automatisch gestartet
0006  13 rem
0007  100 x=1                :rem  Programmbeispiel 'Zufallszahlen erzeugen'
0008  110 for i=1 to 5        :rem  fünf Zahlen pro Zeile ausgeben
0009  120 z=rnd(x):print z,   :rem  Komma: Tabulator-Position anspringen
0010  130 next                :rem  Ende der mit 'FOR' eingeleiteten Schleife
0011  140 ? " "               :rem  Blank ausgeben (incl. Zeilenvorschub)
0012  run
0013  end

                                        Error # 79

MOPPEL-BASIC-Interpreter V 3.6
Copyright (C) hms'86

 C: Convert
 K: Kalt
 M: Monitor
 W: Warm
 X: Extern

BASIC->x

o.k.

B>10 rem  Dieses BASIC-Programm wurde (einschließlich der BASIC-Zeilen-
B>11 rem  nummern) im Editor erstellt und per X-Anweisung direkt aus dem
B>12 rem  RAM an den Interpreter übergeben; da als letzte Anweisung (in
B>13 rem  der Editor-Zeile Nr.12) 'RUN' steht, wird das Programm nach dem
B>14 rem  Einlesen automatisch gestartet
B>13 rem
B>100 x=1                :rem  Programmbeispiel 'Zufallszahlen erzeugen'
B>110 for i=1 to 5       :rem  fünf Zahlen pro Zeile ausgeben
B>120 z=rnd(x):print z,  :rem  Komma: Tabulator-Position anspringen
B>130 next               :rem  Ende der mit 'FOR' eingeleiteten Schleife
B>140 ? " "              :rem  Blank ausgeben (incl. Zeilenvorschub)
B>run
 .42502         .90657         1.2991E-03     .59043         .22676


o.k.

B>
```

1.5 X: Extern (Externen Eingabe-Kanal aktivieren)

Format: B>X (ohne 'Return')

Funktion: Einlesen der im RAM ab 9000h stehenden Klartext-Daten und Übergabe an den Interpreter, als ob sie in kontinuierlicher Folge von der Tastatur stammen würden.

Quittung: o.k.

Im RAM-Bereich ab Adresse 9000h können Sie mit Hilfe des Editors komplette BASIC-Programme erstellen (einschließlich der BASIC-Zeilennummern) und diese mit dem X-Kommando direkt an den BASIC-Interpreter übergeben; wenn als letzte Anweisung 'RUN' folgt, startet das auf diese Weise eingelesene BASIC-Programm sogar vollautomatisch (vgl. auch Abschnitt 11).

Die Umschaltung zurück auf die Tastatur als Eingabe-Kanal erfolgt, sobald das Byte 03h (=ETX) im Text auftaucht. Beim Initialisieren des Editors mit der Setup-Anweisung wird das ETX-Byte automatisch ans Ende der Kopfzeilen angehängt.

Hierüber besteht durchaus die Möglichkeit der interaktiven Programmentwicklung zwischen Editor und BASIC: Sie können zwischen beiden Menüs hin- und herspringen (auf dem Umweg über den Monitor bzw. BASIC-Warmstart) und die mit dem Editor vorgenommenen Modifikationen unmittelbar im BASIC testen. Den Umweg können Sie sogar umgehen, indem Sie zwei Funktionstasten mit den betreffenden Warmstart-Adressen belegen (4003h für BASIC, vgl. Abschnitt 1.4 und 7003h für den Editor)!

Beim Einsprung nach 4007h können Sie im Registerpaar H&L eine andere als die Standard-RAM-Adresse 9000h übergeben.

hms

```
B>pint x                        <<< Schreibfehler 'PRINT'

SN Fehler


B>?sqr(-2)                      <<< Wurzel aus '-2'

FC Fehler


B>? 10^99                       <<< max. Zahlenwert ist 10^37

OV Fehler


B>10 goto 166                   <<< Zeilennummer '166' gibt es nicht
B>run

US Fehler in Zeile 10


B>? 7/0                         <<< Division durch Null

/0 Fehler


B>poke 3,a$                     <<< 'A$' ist String-Variable!

TM Fehler


B>cload                         <<< kein Bandgerät angeschlossen

CC Fehler
B>
```

2. Fehlermeldungen

Der BASIC-Interpreter erkennt und meldet folgende Fehler (im Programmlauf mit zusätzlicher Angabe der Zeilennummer; vgl. Beispiele links):

NF (NEXT without FOR): Bei der Programmierung einer Schleife fehlt das einleitende Element 'FOR' (vgl. Abscchnitt 1Ø).

SN (Syntax): Die Schreibweise eines BASIC-Befehls entspricht nicht der Norm (vgl. Abschnitt 3).

RG (RETURN without GOSUB): Im Programm taucht ein 'RETURN' ohne den vorherigen Unterprogramm-Aufruf 'GOSUB' auf (vgl. Abschnitt 1Ø).

OD (Out of Data): Es wurden mehr READ-Befehle ausgeführt, als an Daten bereitgestellt worden sind (vgl. Abschnitt 8).

FC (Function Call): Bei Ausführen einer Funktion verletzt ein Operand die zulässigen Bedingungen (z.B. Wurzel aus negativer Zahl).

OV (Overflow): Das Ergebnis einer Operation überschreitet den maximal möglichen Rechenbereich (vgl. Abschnitt 3).

OM (Out of Memory): Überlauf eines Speicherbereichs (z.B. Variablen- oder Programmspeicher, Stack).

US (Undefined Statement): Eine Anweisung entstammt nicht dem BASIC-Befehlssatz oder ein Sprungziel (Zeilennummer) existiert nicht.

BS (Bad Subscript): Der Index eines Matrix-Elements liegt außerhalb des durch 'DIM' definierten Bereichs.

DD (Double Dimension): Dasselbe Matrix-Element ist mehrfach definiert worden (Redimensionierungsfehler; (vgl. Abschnitt 8).

/Ø (Division by Zero): Bei einer Division ist der Divisor verbotenerweise Null.

ID (Illegal Direct): Der eingegebene BASIC-Befehl ist im Direkt-Mode nicht zulässig.

TM (Type Mismatch): Bei der Befehlsausführung taucht ein falscher Variablentyp auf (Zahl anstelle eines Strings oder umgekehrt).

OS (Out of String Space): Der für einen String reservierte Speicherplatz ist übergelaufen (vorher mit 'CLEAR' erweitern!).

ST (String too Complex): Ein String überschreitet die maximal zulässige Länge.

CN (Can't Continue): Nach einer Programmunterbrechung ist die Fortsetzung über 'CONT' nicht möglich (vgl. Abschnitt 1Ø).

UF (Undefined Function): Eine im Programm aufgerufene Funktion ist zuvor nicht per 'DEF FN' definiert worden (vgl. Abschnitt 5).

CC (Compact Cassette): Beim Einlesen über das Magnetband-Interface ist ein Fehler aufgetreten (vgl. Abschnitt 11).

hms

```
END   FOR   NEXT   DATA   INPUT   DIM   READ   LET
GOTO   RUN   IF   RESTORE   GOSUB   RETURN   REM   STOP
OUT   ON   NULL   WAIT   POKE   PRINT   DEF   CONT
LIST   CLEAR   CLOAD   CSAVE   NEW   BYE   LLIST   LPRINT
CALL   CLRSCN   CLRLIN   CRSPOS   PPRINT   TAB(   TO   SPC(
FN   TI$   THEN   NOT   STEP   +   -   *
/   ^   AND   OR   >   =   <   SGN
INT   ABS   USR   FRE   INP   POS   SQR   RND
LOG   EXP   COS   SIN   TAN   ATN   PEEK   LEN
STR$   VAL   ASC   CHR$   HEX   LEFT$   RIGHT$   MID$
```

3. BASIC-Befehlssatz

Die folgende Vorstellung der im MOPPEL-BASIC enthaltenen Befehle erfolgt unter dem Aspekt der jeweiligen Besonderheiten; die vorliegende Beschreibung ist also keinesfalls mit einem BASIC-Lehrbuch zu verwechseln!- Entsprechend der üblichen Nomenklatur sind Zahlen zur eindeutigen Kennzeichnung spezifiziert: Ein angehängtes 'h' steht für 'hexadezimal', ein 'd' für 'dezimal' und ein 'b' für 'binär'.

Die Befehlszusammenstellung auf der linken Seite ist nach aufsteigender Reihenfolge der Tokens erfolgt, beginnend bei 80h für 'END' bis zu CFh für 'MID$'; bei einer Inspektion des BASIC-Programmspeichers können Sie mit Kenntnis dieses Zusammmenhangs den Speicherinhalt analysieren.

Sämtliche Befehle (auch die Buchstaben in HEX-Zahlen) können in Kleinschreibung eingegeben werden; bei der Vorstellung der Befehle sind sie zur Hervorhebung versal geschrieben, bei den Beispielen (auf den linken Seiten) durchweg klein; anstelle von 'PRINT' ist die abkürzende Eingabe von '?' zulässig.

Korrekturen sind nur in der jeweiligen Zeile möglich, nachdem der Cursor per Delete-Taste zurückbewegt worden ist; die Cursor-Steuertasten würden zu undefinierten Reaktionen führen! Spätere Korrekturen machen die Neueingabe der betreffenden Zeile erforderlich. Beachten Sie in diesem Zusammenhang die Möglichkeit, BASIC-Programme mit dem MOPPEL-Editor zu bearbeiten (einschließlich sämtlicher gebotenen Korrektur-Funktionen; vgl. Abschnitt 1.5). Auf der Tastatur-Erweiterung vereinfachen die Rechentasten '+', '-', '*' und '/' sowie die PRINT-Taste '?' die Bedienung im Direkt-Modus.

Der Rechenbereich umfaßt +/-10^37, betragsmäßig größere Zahlen führen zur Überlauf-Fehlermeldung 'OV' (vgl. Abschnitt 2). Bei Byte-Operanden (z.B. bei Port-Ein/Ausgaben oder Speicher-Direktzugriffen) ist deren Maximalwert auf 255d begrenzt; größere (oder negative) Werte bewirken die Fehlermeldung 'FC' (Bereichsüberschreitung).

Variablen können maximal zweistellig definiert werden, jedoch lassen sich ohne weiteres die Variablen 'VA', 'VA(i)' und 'VA$' gleichzeitig verwenden.- In der vorliegenden Beschreibung sind zur Deutlichmachung folgende Symbole eingeführt worden:

a$:	String-Variable
i:	Laufvariable (z.B. bei 'FOR/NEXT')
m,n:	Zeilennummer oder Adresse
x,z:	Numerische Variablen
y:	Dummy-Variable (beliebiger Wert, nur von der Syntax her erforderlich)

hms

```
B>null(10)                              <<< linken Rand verschieben

o.k.

        B>? spc(21) : x=pos(y) : ? x    <<< 21 Blanks ausgeben
                              21        <<< anschließend Cursor-Position

o.k.

        B>10 input a
        B>20 ? a; tab(28); "!"          <<< nach Ausgabe von 'a' auf Spalte 28
        B>30 goto 10                        tabulieren (d.h. das Ausrufungs-
        B>run                               zeichen steht immer in Spalte 29,
        ? 123                               unabhängig von der Länge von 'a')
         123
        ? 123456789
         1.2345E+08
        ? .1
         .1


B>list

10 X=1                      <<< 'x' einmalig definieren (wahlfrei, aber nicht Null)
20 FOR I=1 TO 5
30 Z=RND(X) : PRINT Z,      <<< Zufallszahl 'z' erzeugen und fünf pro Zeile ausgeb.
40 NEXT
50 PRINT " " : GOTO 20

o.k.

B>run
 .83366         .84028         .11061         .29844         .40389
 .62664         .64523         .96843         .76384         .71663
 .82772         .30093         .9578          .19382         .43567
 5.0569E-02     .33853         .71273         .85621         .28204


B>pprint hex("1b"),hec("6c")    <<< Escape-Sequenz beim EPSON-Drucker:
B>pprint 6,255                  <<< linken Rand um nn Stellen nach rechts
                                    (1Bh, 6Ch, nn)
```

4. Bildschirm- und Drucker-Ansteuerung, Zufallszahl

CLRSCN: Bildschirm löschen
Vor Bildschirm-Operationen sollte der Schirm jedesmal gelöscht werden (vgl. Abschnitte 5 und 9). Bei parallelgeschaltetem Drucker erhält dieser ein 'Formfeed' (ØCh).

CLRLIN: Zeile ab Cursor löschen
Wenn Operanden unterschiedlicher Länge an derselben Cursor-Position ausgegeben werden, sollte die alte Ausgabe vorher gelöscht werden (vgl. Abschnitt 6).

CRSPOS ss,zz: Cursor positionieren (ss: Spalte; zz: Zeile)
Der Cursor kann von Spalte ØØ (linksbündig) bis 79 und von Zeile 1 (oben) bis 24 positioniert werden; alle folgenden Ausgaben werden an der so definierten Position fortgesetzt (vgl. Abschnitte 5 und 9).

NULL(x): Linken Rand um x Stellen nach rechts verschieben (vgl. links, oben).

SPC(x): x Leerzeichen (Spaces) ausgeben (vgl.links, oben).

POS(y): Cursor-Position ausgeben
Liefert die aktuelle Cursor-Position in der Zeile (vgl.links, oben).

TAB(x): Cursor auf Spalte x tabulieren
Bewegt den Cursor unabhängig von der augenblicklichen Position auf Spalte x vor, sofern er noch links davon steht (vgl.links, Mitte).

RND(x): Zufallszahl erzeugen
Bei x=1 werden Zufallszahlen im Bereich Ø...1 erzeugt; bei x=Ø (oder fehlender x-Definition) wird fortlaufend die zuletzt erzeugte Zufallszahl ausgegeben (vgl.links, unten).

PRINT x abgekürzt ? x: Bildschirm-Ausgabe von 'x'
';': Unterdrückt in einer PRINT-Zeile den Zeilenvorschub.
(vgl.links, oben, Zeile 2Ø)
',': Rückt den Cursor in einer PRINT-Zeile auf die nächste Tabulator-Position vor (TABs sind alle 14d Spalten; vgl.links, Mitte, Zeile 3Ø).

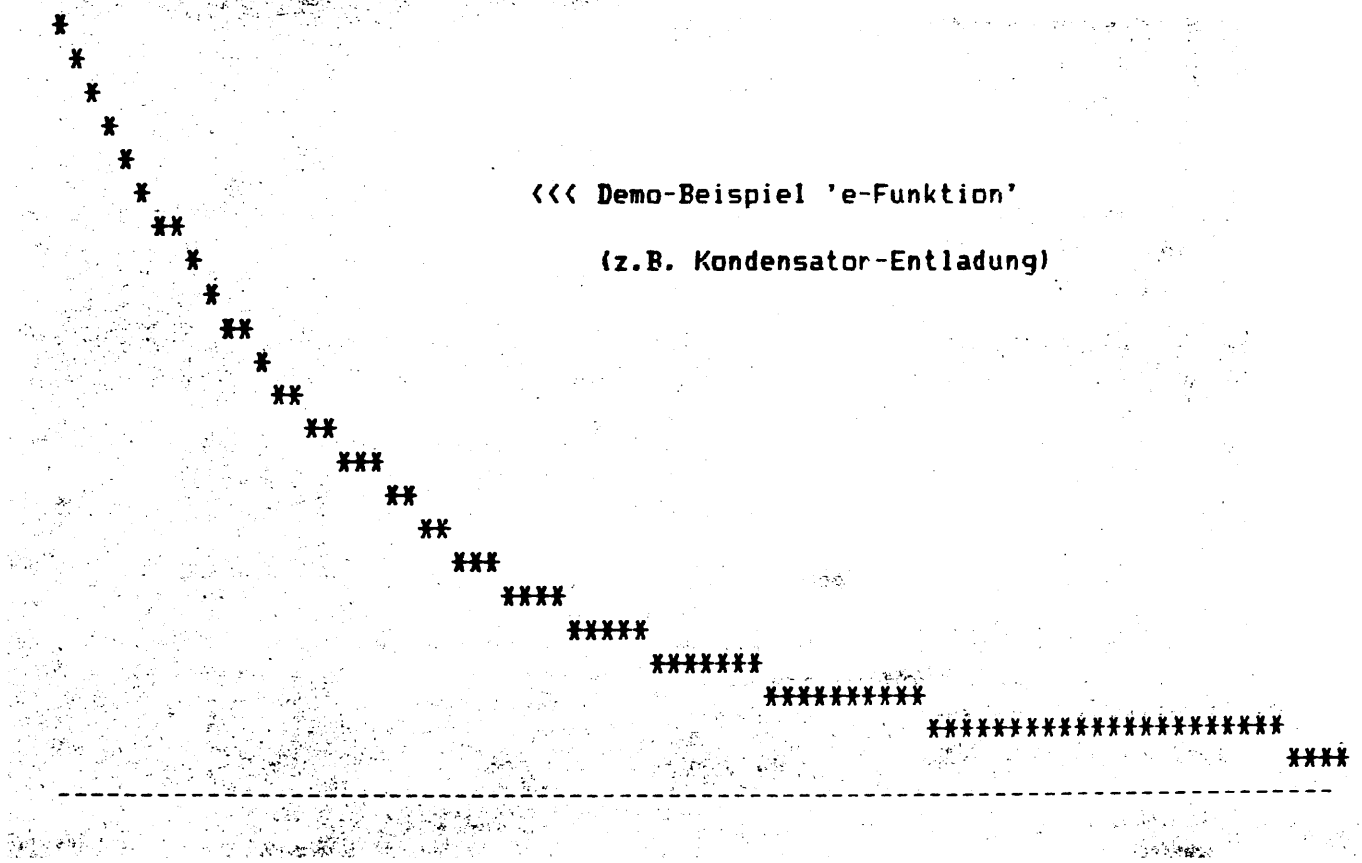
LPRINT x: Wie 'PRINT x', aber mit parallelgeschaltetem Drucker.

PPRINT x,z: Exklusive Drucker-Ansteuerung
Die Bytes 'x' und 'z' gelangen direkt und ohne Bildschirm-Ausgabe an den Drucker; bei z=255d wird nur x ausgegeben. Mit dieser Anweisung läßt sich beispielsweise die Schriftart des angeschlossenen Druckers umschalten, ohne andere Ausgabe-Funktionen zu beeinflussen.

Im Beispiel links unten werden im Direkt-Modus nacheinander die drei Codes 1Bh, 6Ch und 6 an den Drucker übergeben (das letzte Zeichen '255' wird unterdrückt). Die damit bewirkte Randverschiebung bleibt nach einem Rücksprung in den Monitor (oder Assembler) erhalten.

hms

```
B>10 input x
B>20 def fn dl(y)=log(x)/log(10)
B>30 z=fn dl(y)
B>40 ? z
B>run
? 2
 .301Ø3


o.k.

B>new


o.k.

B>list

5 CLRSCN
10 CRSPOS Ø,24
20 FOR I=1 TO 79
25 PRINT "-"; : NEXT
30 CRSPOS Ø,1
40 FOR T=Ø TO 4-1/2Ø STEP 1/2Ø
50 S=2Ø*T : Y=1-EXP(-T) : Z=22*Y+1.5
60 CRSPOS S,Z : PRINT"*" : NEXT
70 GOTO 70

o.k.

B>run
```



5. Operatoren, Rechen- und Anwender-Funktionen

+ - * / ( ): Addition, Subtraktion, Multiplikation, Division; die Klammern fassen Operatoren zusammen, die per Definition nicht zusammen verarbeitet werden (z.B. Zusammenfassung von Summanden vor einer Division).

< = >: Operatoren für 'kleiner als', 'gleich' und 'größer als' (vgl. Abschnitt 8).

^: Präfix für Exponenten ('E': Exponent zur Basis 10d).

SQR(x): Quadratwurzel aus x

EXP(x): e^x (e hoch x)
Benannt nach Leonhard Euler (1707-1783); e=2,7182818... ist die Basis der natürlichen Logarithmen. Beispiel für eine e-Funktion: Entladekurve eines Kondensators (RC-Glied; vgl.links, unten).

LOG(x): ln x (natürlicher Logarithmus von x)
Zwischen dem natürlichen und dekadischen Logarithmus DL läßt sich ohne weiteres umrechnen: DL(x)=ln x/ln 10 (vgl.links, oben).

LET x=z: Wertzuweisung ('x' sei gleich 'z')
Der Operator 'LET' ist entbehrlich, d.h. für die Wertzuweisung genügt die Schreibweise 'x=z'; ebenfalls zulässig ist die Formulierung 'x=x+1' für 'LET x=x+1', d.h. 'x' wird ersetzt durch 'x+1' (also um Eins hochgezählt).

DEF FN F(y): Funktionsnamen 'f(y)' definieren
Hiermit läßt sich eine Funktion definieren, die z.B. im BASIC-Befehlssatz nicht enthalten ist, und die anschließend im Programm immer wieder aufgerufen werden kann. Beispiel links, oben: Unter 'DL(y)' wird der dekadische Logarithmus der Zahl x definiert.

Hinweis: Der Ausdruck auf der linken Seite ist auf dem Umweg über die festprogrammierte Funktionstaste FCT+V entstanden (Retten des Bildspeichers ins RAM ab F800h und anschließendes Ausdrucken des RAM-Inhalts). Eine Drucker-Parallelschaltung ist beim vorliegenden Programmbeispiel nicht möglich, weil die Cursor-Positionierung nicht den Druckkopf beeinflußt.

hms

```
B>list                                  <<< BEISPIEL 1:

10 VI=HEX("30a0")
20 X=PEEK(VI) : PRINT X

o.k.

B>run
 49                                     <<< direktes Einlesen vom Video-RAM
                                            (49d=31h: ASCII-Code für '1' von
o.k.                                         der Zeilen-Nummer '10')


B>list                                  <<< BEISPIEL 2:

50 INPUT X
60 POKE HEX("3250"),X
                                    A   <<< direkte Video-Ausgabe nach 'run'
o.k.

B>run
? 65                                    <<< 65d=41h (ASCII-Code für 'A')


B>list                                  <<< BEISPIEL 3:

70 PO=HEX("c8") : SD=HEX("10")          <<< Inp.Port'C8h'; SSG-Anzeige'10h'
80 X=INP(PO) : PRINT X
90 OUT SD,X

o.k.

B>run
 124                                    <<< 124d=7Ch=0111.1100b

o.k.


B>list                                  <<< BEISPIEL 4:

100 INPUT X,Z                           <<< Variablen 'x' und 'z' einlesen
110 K1=X AND Z
120 K2=X OR Z
130 K3=NOT K2
140 PRINT "K1:";K1, "K2:";K2, "K3:";K3

o.k.

B>run
? 12,6                                  <<< 12d=0Ch=0000.1100b; 6d=6h=0000.0110b
K1: 4         K2: 14        K3:-15


o.k.

B>
```

6. Ein/Ausgaben (Console, RAM und Ports), logische Operationen

---

INPUT x: Variable 'x' von der Console einlesen
Tastatur-Eingabe-Anweisung, die mit einer Text-Ausgabe gekoppelt werden kann (vgl. Abschnitt 8). Als Aufforderung zur Eingabe erscheint ein '?' (vgl.links, Mitte und unten). Consolen-Ausgabe 'PRINT': vgl. Abschnitt 4.

PEEK(n): Daten aus der Speicheradresse n auslesen
Direkter Zugriff auf den Mikrocomputer-Speicher (RAM und ROM; vgl. links, oben).

POKE n,x: Variable 'x' in die Speicheradresse n einschreiben.
Direkter Zugriff auf den Mikrocomputer-Speicher (nur RAM; vgl.links, Mitte).

INP(n): Daten vom Port n einlesen
Die Port-Eingabe-Funktion 'INP' kann nicht, wie sonst üblich, mit CTL+C unterbrochen werden (vgl.links, Mitte; Port 'PO'=C8h ist ein 8-bit-Parallel-Interface mit Kippschalter-Eingabe).

OUT n,x: Variable 'x' über den Port n ausgeben
Port 'SD'=1Øh ist eine Stelle des Siebensegment-Displays auf der LED-Einheit, in der bei Ansteuerung mit 124d=7Ch ein stilisiertes 'b' erscheint (Aktivierung der Segmente c, d, e, f und g; vgl.links, Mitte).

WAIT n,x,z: Daten vom Port n einlesen, XOR z, UND x
Wie 'INP(n)', aber die Daten werden zusätzlich mit dem Bitmuster 'z' Exklusiv-ODER-verknüpft (d.h. selektiv invertiert) und anschließend mit dem Bitmuster 'x' UND-verknüpft; erst wenn das so entstandene Ergebnis ungleich Null ist, wird die Programmausführung fortgesetzt. Der WAIT-Befehl kann nicht, wie sonst üblich, mit CTL+C unterbrochen werden.

x AND z: Konjunktion
Die Operanden 'x' und 'z' werden logisch UND-verknüpft, d.h. im binären Format werden jeweils zwei korrespondierende Bits UND-verknüpft (vgl.links, unten).

x OR z: Disjunktion
Die Operanden 'x' und 'z' werden logisch ODER-verknüpft, d.h. im binären Format werden jeweils zwei korrespondierende Bits ODER-verknüpft (vgl.links, unten).

NOT x: Komplement
Es wird das Zweier-Komplement des im binären Format vorliegenden Operanden 'x' gebildet (vgl.links, unten).

Hinweis: In der Programmzeile 8Ø (vgl.links, Mitte) werden Daten vom Port 'PO' eingelesen und ausgegeben ('PRINT x'); wenn die Ausgabe immer an derselben Bildposition erfolgen soll, ist der Cursor vorher dorthin zu positionieren, gefolgt vom Löschen des Zeilenrests (vgl. Abschnitt 4).

hms

```
B>A$=TI$                          <<< Uhrzeit-String unter 'A$' "einfrieren"

o.k.

B>PRINT A$
Mo,11.05.87;14:36:24h

B>WO$=LEFT$(A$,2) : PRINT WO$
Mo                                <<< Wochentag abspalten (String!)

B>MO$=MID$(A$,7,2) : PRINT MO$
05                                <<< Monat herauslösen (String!!)

B>UR$=RIGHT$(A$,9) : PRINT UR$
14:36:24h                         <<< Uhrzeit abspalten (String!!)

o.k.



B>MO=VAL(MO$) : PRINT MO
 5                                <<< String '05' wird Zahl (+)5

B>ZA=123456
B>ZA$=STR$(ZA)
B>LI=LEFT$(ZA$,3) : PRINT LI
 12

B>PRINT LEN(ZA$)
 7



B>PRINT CHR$(69)
E


B>PRINT ASC("E")
69


B>PRINT HEX("45")
69
```

7. Stringverarbeitung, Typ-Umwandlung

TI$: Echtzeit-Uhr auslesen
Die interne Echtzeit-Uhr wird ausgelesen und als String-Variable abgelegt (vgl.links, oben).

LEFT$(a$,n): Teil-String von a$ abtrennen
Vom String 'a$' werden linksbündig n Zeichen abgetrennt (vgl.links, oben).

MID$(a$,m,n): Teil-String aus a$ heraustrennen
Vom String 'a$' werden n Zeichen herausgetrennt, beginnend beim m-ten Zeichen von links (vgl.links, oben; die Ziffernfolge des Monats ist ein String, weil auch die ursprüngliche Variable 'a$' ein String war!).

RIGHT$(a$,n): Teil-String von a$ abtrennen
Vom String 'a$' werden rechtsbündig n Zeichen abgetrennt (vgl.links, oben; die Ziffernfolge der Uhrzeit ist ein String, weil die ursprüngliche Variable 'a$' auch ein String war!).

VAL(a$): Variablen-Typ umwandeln
Aus der String-Variablen 'a$' wird eine numerische Variable gebildet (vgl. links, Mitte; das positive Vorzeichen wird, wie üblich, nicht mit ausgegeben).

STR$(x): Variablen-Typ umwandeln
Aus der numerischen Variablen 'x' wird eine String-Variable gebildet (vgl. links, Mitte; bei der Zahl 'ZA' zählt das positive Vorzeichen bei der Anzahl der Stellen mit!).

LEN(a$): Länge eines Strings ermitteln
Liefert die Stellenzahl (numerische Variable) eines Strings (vgl. links, unten).

CHR$(x): Code umsetzen
Gibt das zum Code 'x' korrespondierende ASCII-Zeichen aus (String-Variable; vgl.links, unten).

ASC("x"): Zeichen umsetzen
Gibt den zum Zeichen 'x' korrespondierenden ASCII-Code aus (numerische Variable; vgl.links, unten).

HEX("x"): Code umsetzen
Setzt den Hexadezimal-Code 'x' um ins dezimale Äquivalent (numerische Variable; vgl.links, unten). Die Alpha-Zeichen a...f in einer HEX-Zahl können in Kleinschreibung eingegeben werden (vgl. Abschnitt 3).

hms

```
B>list                                  <<< 'Ewiger Kalender'

10 DIM D(12)                            <<< für D(i) 12 Indizes zulassen
20 INPUT "Datum (tt,mm,jjjj)";T,M,J
30 FOR I=1 TO 12
35 READ D(I) : NEXT                     <<< Daten aus Zeile 190/191 auslesen
40 T=T+D(M)                             <<< Tag-Eingabe um D(m) erhöhen
45 Y=J-1601                             <<< Jahr-Eingabe reduzieren
50 S1=INT(Y/4)                          <<< durch 4 teilen (wg.Schaltjahr)
55 S2=INT(Y/100)
60 S3=INT(Y/400)
70 T=T+Y+S1-S2+S3+6
80 IF M<=2 GOTO 120                     <<< d.h. Monat Januar oder Februar
85 IF INT(J/4)<>J/4 GOTO 120            <<< nicht durch 4 teilbar: weiter
90 T=T+1                                <<< also Schaltjahr
95 IF INT(J/100)<>J/100 GOTO 120        <<< nicht durch 100 teilbar: weiter
100 T=T-1                               <<< doch kein Schaltjahr
105 IF INT(J/400)<>J/400 GOTO 120       <<< nicht durch 400 teilbar: weiter
110 T=T+1                               <<< endgültig: Schaltjahr (gilt für 2000)
120 K=(T/7)-INT(T/7)+.01                <<< '0,01' gleicht Rundungsfehler aus
125 K=INT(7*K)                          <<< lfd.Nummer für den Wochentag
130 FOR I=1 TO 7
135 READ W$(I) : NEXT                   <<< Daten aus Zeilen 195/196/197 auslesen
140 W$=W$(K+1)                          <<< Text zur lfd.Nummer zuordnen
150 PRINT "Wochentag: ";W$
160 RESTORE : PRINT : PRINT : GOTO 20 <<< DATA-Pointer rücksetzen, 2 Leerzeilen
190 DATA 0,31,59,90,120,151,181
191 DATA 212,243,273,304,334
195 DATA Montag,Dienstag,Mittwoch
196 DATA Donnerstag,Freitag
197 DATA Sonnabend,Sonntag

o.k.

B>run
Datum (tt,mm,jjjj)? 11,05,1987
Wochentag: Montag                       <<< Ergebnis des Programmlaufs


B>? ti$
Mo,11.05.87;15:27:06h                   <<< Auslesen der Echtzeit-Uhr

o.k.


B>
```

8. Daten-Bereitstellung und -Abruf

DATA x: Datenzeile definieren
Bereitstellung einer Folge von Daten, die per READ-Anweisung abgerufen und dann weiterverarbeitet werden (vgl.links)

READ x: Daten abrufen
Die mit 'DATA' bereitgestellten Daten sequentiell abrufen und für die Weiterverarbeitung einer Variablen zuweisen; es dürfen nicht mehr READ-Anweisungen aufgerufen werden als an Daten bereitgestellt worden sind (vgl.links).

RESTORE: Daten-Pointer rücksetzen
Das zur sequentiellen Adressierung der Daten dienende Adreßregister an den Anfang der DATA-Zeile(n) zurücksetzen (vgl.links).

DIM x(z): Feld-Dimensionierung
Für die indizierte Variable 'x' werden 'z+1' Feldelemente reserviert (beginnend bei z=Ø; vgl.links). Bis z=1Ød braucht die DIM-Anweisung nicht aufgerufen zu werden (automatische Dimensionierung beim Kaltstart).

CLEAR x: String-Speicherplatz reservieren
Alle Variablen werden auf Null gesetzt und x Bytes für die String-Verarbeitung reserviert. Bis x=5Ød braucht die CLEAR-Anweisung nicht aufgerufen zu werden (automatisches Reservieren beim Kaltstart).

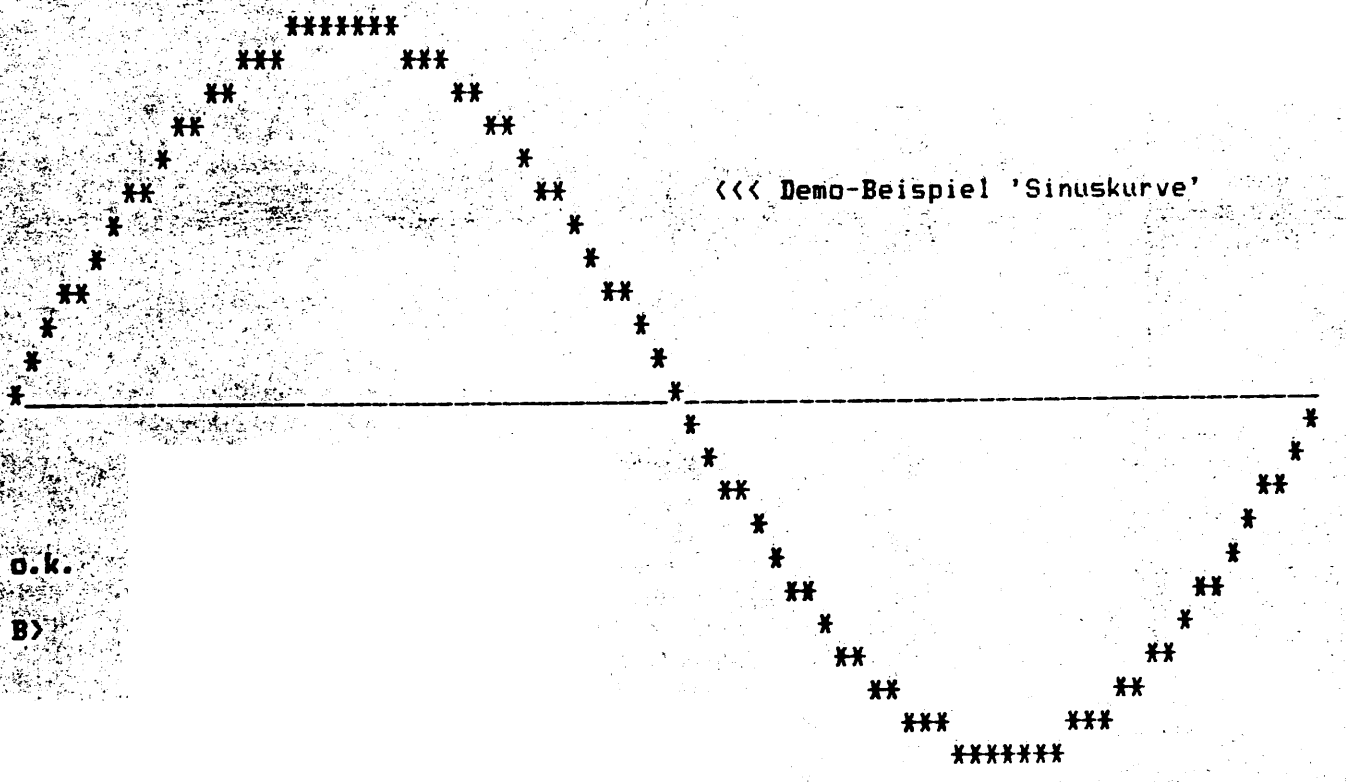
INT(x): Integer-Funktion
Von der Variablen 'x' wird der Nachkomma-Anteil unterdrückt (Ganzzahl-Bildung; vgl.links).

SGN(x): Signum-Funktion
Liefert das Vorzeichen der Variablen 'x': (+)1 bei positivem 'x', -1 bei negativem 'x' und Ø bei x=Ø.

ABS(x): Absolutwert-Funktion
Bildet den Absolutwert der Variablen 'x', d.h. ein negatives Vorzeichen von 'x' wird invertiert.

Das Programmbeispiel 'Ewiger Kalender' (Ermittlung des Wochentages) auf der linken Seite zeigt neben einer Reihe mathematischer Funktionen, daß es mit geschicktem Ansatz trotz des eingeschränkten Rechenbereichs möglich ist, große Zahlenwerte zu verarbeiten: Im Prinzip errechnet das Programm nur die Anzahl der seit einem Stichtag verstrichenen Tage (unter Berücksichtigung der aufgetretenen Schaltjahre) und ordnet zyklisch die Wochentage zu. Dabei wird aber nicht die Anzahl von Jahren mit 365 multipliziert (das würde den Rechenbereich überschreiten), sondern pro Gemeinjahr wird nur <u>ein</u> Wochentag addiert (jährliche Verschiebung eines Datums um einen Wochentag).

```
BASIC->w

o.k.

B>list

5 CLRSCN                Schirm löschen
10 CRSPOS 0,12          Cursor auf Zeile 12, linksbündig
15 FOR I=1 TO 80       >Beginn erste Schleife
20 PRINT "_";           Querstrich in der Bildmitte
25 CALL HEX("13")       Monitor-Unterprogramm 'DLY100' (100 ms Zeitverzögerung)
30 NEXT                >Ende erste Schleife
50 FOR S=0 TO 79       <Beginn zweite Schleife
55 PI=4*ATN(1)          Hilfs-Definition von Pi
60 X=S*2*PI/80          S(=Schrittweite): 180 Grad (=2 Pi) auf 80 Spalten
70 Y=11*SIN(X)          Sinus (-1,0...+1,0) auf -11...+11 dehnen
80 Z=12.5-Y             auf Bildmitte zentrieren
85 CRSPOS S,Z           S: Horizontale; Z: Vertikale
90 PRINT "*"            Markierungs"punkt"
95 NEXT                <Ende zweite Schleife

o.k.

B>run
```




```
o.k.

B>? PI                  <<< ist oben definiert worden
 3.1415                 <<< Rundungsfehler: Pi=3,1415926535... (also: 3.1416!)
```

9. Trigonometrische Funktionen

SIN(x): Sinus-Funktion
Liefert das zum Winkel 'x' (im Bogenmaß) gehörende Verhältnis von Gegenkathete zu Hypothenuse (vgl.links).

COS(x): Kosinus-Funktion
Liefert das zum Winkel 'x' (im Bogenmaß) gehörende Verhältnis von Ankathete zu Hypothenuse.

TAN(x): Tangens-Funktion
Liefert das zum Winkel 'x' (im Bogenmaß) gehörende Verhältnis von Gegenkathete zu Ankathete.

ATN(x): Arkustangens-Funktion
Liefert den zum Verhältnis 'x' (von Gegenkathete zu Ankathete) gehörenden Winkel (im Bogenmaß).

Hinweis: Die Kreiszahl 'Pi' ist nicht als Konstante abrufbar; sie muß explizit definiert werden (z.B. über eine Winkelfunktion wie links in Zeile 55). Ähnliches gilt für die Kotangens-Funktion, die aus dem Verhältnis COS(x)/SIN(x) abgeleitet werden kann.- Die Angaben im Bogenmaß bedeuten, daß 180 (Alt-)Grad gleich Pi entsprechen.

Der Ausdruck auf der linken Seite ist auf dem Umweg über die festprogrammierte Funktionstaste FCT+V entstanden (Retten des Bildspeichers ins RAM ab F800h und anschließendes Ausdrucken des RAM-Inhalts). Eine Drucker-Parallelschaltung ist beim vorliegenden Programmbeispiel nicht möglich, weil die Cursor-Positionierung nicht den Druckkopf beeinflußt.

10. Programmausführung

RUN: Programm starten
Wenn zusammen mit 'RUN' keine Zeilennummer angegeben wird, beginnt die Programmausführung bei der niedrigsten Zeilennummer.

CTL+S: Programaausführung stoppen
Anwender-Eingabe über die Tastatur, die die Programmausführung an jeder beliebigen Stelle unterbricht (Ausnahme: vgl. Abschnitt 6).

CTL+Q: Programmausführung fortsetzen
Anwender-Eingabe über die Tastatur, die die zuvor per 'CTL+S' gestoppte Programmausführung fortsetzt.

CTL+C: Programmausführung abbrechen
Anwender-Eingabe über die Tastatur, die einen Abbruch des laufenden Programms zur Folge hat.

STOP: Programmausführung unterbrechen
BASIC-Befehl, der ein laufendes Programm an einer definierten Stelle unterbricht.

CONT: Programmausführung fortsetzen
Direkt-Befehl, der ein per 'STOP' unterbrochenes Programm fortsetzt. Während der Unterbrechnung dürfen keine Modifikationen am Programm vorgenommen worden sein, andernfalls ergeht die Fehlermeldung 'CN' (Can't Continue; vgl. Abschnitt 2).

END: Programmende markieren
Beendet die Programmausführung und ist als Befehl nur dann erforderlich, wenn andere Programme (oder Unterprogramme) von einem davor liegenden Programm getrennt werden müssen.

GOTO n: Unbedingter Sprung zur Zeilennummer n

USR(n): Unbedingter Sprung zur Speicheradresse n

ON x=... GOTO n1,n2,...: Verzweigung in Abhängigkeit von x
Bei x=1 erfolgt ein Sprung zur Zeilennummer n1, bei x=2 zu n2 usf.

IF x=... GOTO n: Bedingter Sprung zur Zeilennummer n (vgl. Abschnitt 8).

FOR i=x TO z / NEXT: Programmschleife definieren
Die von 'FOR' und 'NEXT' eingeschlossenen Befehle so oft wiederholen, bis die Laufvariable 'i' vom Anfangswert 'x' ausgehend den Endwert 'z' erreicht hat (normale Schrittweite pro Durchlauf: +1). Es können beliebige andere Schrittweiten angegeben werden (vgl. Abschnitt 5).

GOSUB n: Unterprogramm bei Zeilennummer n aufrufen
BASIC-Unterprogramm aufrufen, das mit 'RETURN' abgeschlossen ist.

CALL n: Unterprogramm bei Speicheradresse n aufrufen
Maschinen-Unterprogramm aufrufen, das mit C9h (=RET) abgeschlossen ist (vgl. Abschnitt 9).

hms

11. Programm-Verwaltung und -Speicherung

NEW: Programmspeicher initialisieren
Löschen aller vorhandenen Programme und Rücksetzen aller Variablen.

FRE(y): Freien Speicherplatz ermitteln
Ermittelt den für den Anwender verfügbaren freien Speicherplatz (in Bytes; vgl. Abschnitt 1.2). **FRE(y$)** liefert den freien Platz im String-Speicher.

REM: Kommentarzeile definieren
Die mit 'REM' eingeleitete Zeile bei der Programmausführung übergehen (Kommentarzeile; vgl. Abschnitt 1.5).

LIST: Programmzeilen listen
Wenn zusammen mit 'LIST' keine Zeilennummer angegeben wird, beginnt das Listen mit der niedrigsten Zeilennummer. Es kann jederzeit mit CTL+S gestoppt und mit CTL+Q wieder fortgesetzt werden; CTL+C bricht den List-Vorgang ab.

LLIST: Wie 'LIST', aber mit parallelgeschaltetem Drucker.

CSAVE: Programm auf Magnetband ausgeben
Gibt das Programm bis zum Ende des Variablen-Bereichs (Endadresse in 8083/84h) über das Magnetband-Interface aus. Die Bedienung entspricht der Monitor-Funktion 'Cassetten-Ausgabe' (vgl. Abschnitt 3.2 des System-Handbuchs).

CLOAD: Programm vom Magnetband einlesen
Liest ein mit der CSAVE-Anweisung ausgegebenes Programm über das Magnetband-Interface ein. Die Bedienung entspricht der Monitor-Funktion 'Cassetten-Eingabe' (vgl. Abschnitt 3.1 des System-Handbuchs).

Auto-Start-Funktion: Um ein extern (z.B. auf Diskette) gespeichertes Programm nach dem Einlesen selbsttätig zu starten, ist wie folgt zu verfahren:

Programmspeicher ab 8000h auf Diskette überspielen (die Endadresse des Programms steht in den RAM-Zellen 8087/88h, und das Ende des Variablen-Bereichs steht in 8083/84h); Sektor 01 auf Spur 00 für den Batch-Vorspann frei lassen!

Batch-Vorspann generieren (Nr.00, Start:40, RAM-Beg: 80h!) und auf Diskette schreiben (vgl. Abschnitt 5 der Floppy-Handhabung).

Programm-Startadresse von 4000h auf 4003h ändern (im Monitor; dazu genügt die Modifikation des RAM-Inhalts 2D00h von 00h auf 03h). Anschließend den RAM-Bereich 2D00...2DFFh auf Sektor 01, Spur 00 schreiben (modifizierter Batch-Vorspann).

Mit der Disk-In-Funktion Nr.0 kann nun das BASIC-Programm automatisch von Diskette eingelesen und selbsttätig gestartet werden.

MOPPEL
System-Software

Prommer-Handhabung

Modulares Prozessor-Programm zum Entwickeln und Lernen

® **Eingetragenes Warenzeichen**

hms

Inhaltsverzeichnis

Ø. Installation und Aufruf
Hardware-Voraussetzungen und Software-Aktivierung

1. Prommer-Menü
Bedeutung der Parameter in den Kopfzeilen

2. Fehlermeldungen
Zusammenstellung aller Prommer-Fehlermeldungen

3. A: Adjust
Spannungen abgleichen

4. B: Blank Check
Leertest

5. C: Compare
EPROM- mit Speicher-Inhalt vergleichen

6. M: Monitor
Monitor-Rücksprung

7. P: Program
EPROM programmieren

8. R: Read
EPROM auslesen

9. T: Typ
EPROM-Typ einstellen

1Ø. V: Vpp
Programmierspannung einstellen

hms

```
MOPPEL-Video-Monitor V 7.6
Copyright (C) hms'86

M: Memory
O: Outward
P: Peripheral
S: Start

M>o

A: Assembler
B: BASIC
D: Disassembler
E: Editor
F: Floppy
P: Prommer
X: Ext.2000h

Mo>p
```

Ø. Installation und Aufruf

Folgende Hardware-Voraussetzungen sind für den Betrieb des EPROM-Programmiermoduls erforderlich:

Das Modul ist (bei ausgeschalteter Versorgungsspannung) frontseitig auf den Floppy-Disk-Controller aufzustecken; zur mechanischen Abstützung empfiehlt es sich, links neben dem Floppy-Disk-Controller eine weitere Europa-Karte einzuschieben (z.B. die Speicher-Karte). Die Prommer-Software ist zusammen mit den Floppy-Disk-Utilities in einem EPROM enthalten, das auf dem Floppy-Disk-Controller eingesetzt ist.

Außer der Standard-5-V-Versorgung (Bus-Leitungen a,b,c1 und a,b,c32) werden für den Betrieb +8 V (Bus-Leitung a15) sowie +25 V (Bus-Leitung a28) benötigt; diese zum Programmieren erforderlichen Hilfsspannungen werden vom Schaltnetzteil bereitgestellt.

Alle EPROMs sind mit der Markierungskerbe nach oben in die Schnellspannfassung einzusetzen. Typen im 24poligen Gehäuse (2716 und 2732) sind nach unten hin bündig zu bestücken, d.h. die vier oberen Pole der Fassung bleiben in diesen Fällen frei.

Vom Monitor-Grundmenü aus erfolgt der Prommer-Aufruf über das OUTWARD-Menü (Eingabe von 'o' ohne 'Return'), gefolgt von 'p' (wiederum ohne 'Return'). Sämtliche Einstellungen (Typen-Anpassung, Programmierspannung) gehen kontaktlos ohne mechanische Schalter vor sich. Zum Programmieren der Typen 2764, 27128 und 27256 dient der sogenannte 'Intelligente Algorithmus', der dynamisch die benötigte Pulslänge ermittelt und auf diese Weise eine drastische Reduzierung der Programmierzeit bewirkt.

Die Eingabe der Kommando-Buchstaben kann wahlweise in Klein- oder Großschreibung erfolgen; die danach eingegebenen Parameter können, müssen aber nicht durch einen Leerschritt ('Blank') vom Kommando-Buchstaben getrennt sein. Mehrere Parameter sind wahlweise durch Komma oder Punkt voneinander zu trennen, führende Nullen brauchen nicht eingegeben zu werden. Aus Gründen der Übersichtlichkeit ist das Kommando in den folgenden Beschreibungen stets ein Großbuchstabe, der von den Parametern durch einen Leerschritt abgesetzt ist.

Auch die Prommer-Software unterstützt zwei Bildformate, die abhängig sind von der Einstellung des Video-Interfaces: 24 Zeilen mit je 8Ø Zeichen oder 2Ø Zeilen mit je 4Ø Zeichen. Im übrigen sei auf die ausführlichen Beschreibungen des System-Handbuchs verwiesen.

```
Typ: 2764          SB:----                                  Mi,08.04.87;09:08:31h
Vpp: 21 V          SE:----
Tim: 0'00          DB:----
--------------------------------------------------------------------------------


MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>
```

1. Prommer-Menü

Nach jedem Kalt- oder Warmstart (über 'Escape' bzw. CTL+C) wird das Prommer-Menü mit den acht möglichen Kommandos 'A', 'B', 'C', 'M', 'P', 'R', 'T' und 'V' auf dem Bildschirm dargestellt.

Eingaben des Anwenders werden von unten nach oben protokolliert und fortgeschrieben. Ausgenommen hiervon sind die Parameter in den Kopfzeilen; sie werden -soweit erforderlich- für die einzelnen Operationen unverändert übernommen, müssen also nicht jedesmal neu eingegeben werden.

Dabei haben die Felder in den Kopfzeilen folgende Bedeutung:

Typ: EPROM-Typenangabe: 2716, 2732, 2764, 27128 oder 27256
Da das Lesen und Programmieren der möglichen EPROM-Typen nach unterschiedlichen Vorgehensweisen erfolgt,. muß die Prommer-Software entsprechend eingestellt werden, sofern der aktuelle vom voreingestellten Typ 2764 abweicht (vgl. Abschnitt 9 'EPROM-Typ einstellen').

Vpp: Programmierspannung: 12,5 V; 21,Ø V oder 25,Ø V
Entsprechend den Hersteller-Angaben ist die zum Programmieren benötigte Spannung vorzugeben, sofern sie vom voreingestellten Wert 21,Ø V abweicht (vgl. Abschnitt 1Ø 'Programmierspannung einstellen').

Tim: Zeitzähler beim Programmieren
Um eine Kontrolle des Programmier-Fortschritts zu haben, wird dieser Zeitzähler während des Programmierens im Sekundentakt weitergezählt (vgl. Abschnitt 7 'EPROM programmieren').

SB: Source Begin
Anfangsadresse des Ursprungs-Datenblocks (beim Programmieren Speicher-Adresse, bei den übrigen Kommandos EPROM-Adresse).

SE: Source End
Endadresse des Ursprungs-Datenblocks (beim Programmieren Speicher-Adresse, bei den übrigen Kommandos EPROM-Adresse).

DB: Destination Begin
Anfangsadresse des Ziel-Datenblocks (beim Programmieren EPROM-adresse, bei den übrigen Kommandos Speicher-Adresse).

EZU: Echtzeit-Uhr
Als Basis für den Zeitzähler 'Tim' erscheint im Prommer-Menü die Uhrzeit- und Datums-Information der internen Echtzeit-Uhr.

hms

```
Typ: 2764       SB:----                                  Mi,08.04.87;09:44:09h
Vpp: 21 V       SE:----
Tim: 0'00       DB:----
--------------------------------------------------------------------------------
MOPFEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>                                        Error # 80

P>r yyyy                                  Error # 82

P>r 12345,6789a,bcdef                     Error # 83

P>
```

2. Fehlermeldungen

Bei der Parameter-Eingabe werden folgende Fehler sofort erkannt und gemeldet:

Error # 80: Eingabe fehlt.

Error # 82: Parameter-Fehler; nur hexadezimale Eingaben sind zulässig.

Error # 83: Eingabe unzulässig; sie entstammt entweder nicht dem Menü oder ist zu lang.

Die übrigen Fehlermeldungen haben folgende Bedeutung:

Error # 84: Typangabe unzulässig (vgl. Abschnitt 9 'EPROM-Typ einstellen').

Error # 85: Spannungsangabe unzulässig (vgl. Abschnitt 10 'Programmierspannung einstellen').

Error # 86: EPROM-Inhalt nicht gelöscht (vgl. Abschnitt 4 'Leertest').

Error # 87: EPROM- und Speicher-Inhalt sind nicht gleich (vgl. Abschnitt 5 'EPROM- mit Speicher-Inhalt vergleichen').

Error # 88: Programmieren fehlerhaft (vgl. Abschnitt 7 'EPROM programmieren').

Error # 89: Einschreiben ins RAM fehlerhaft (vgl. Abschnitt 8 'EPROM auslesen').

```
Typ: 2764          SB:----                                   Mi,08.04.87;09:09:06h
Vpp: 21 V          SE:----
Tim: 0'00          DB:----
--------------------------------------------------------------------------------
MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>a

P1: 6,0 V an Pin 28

P2:12,5 V an Pin 1

P3:21,0 V an Pin 1
```

3. A: Adjust (Spannungen abgleichen)

Format: P>A<Ret>

Funktion: Einmaliges Justieren der zum Programmieren benötigten Spannungen von 12,5 V, 21,Ø V (Programmierspannung) sowie 6 V (erhöhte Versorgungsspannung). Dieser Abgleich ist bei der fertig gelieferten Baugruppe bereits herstellerseitig durchgeführt worden.

Für den ersten Abgleich ist ein Widerstand von ca. 4,7 kOhm von Pin 28 (oben rechts) zum Pin 14 (unten links) anzuklemmen. An diesen Widerstand ist ein Voltmeter anzuschließen (Pluspol an Pin 28) und das Potentiometer P1 so zu verdrehen, daß sich eine Spannung von 6,Ø V einstellt (erhöhte Versorgungsspannung beim Intelligenten Programmier-Algorithmus der Typen 2764, 27128 und 27256).

Für den zweiten Abgleich ist zunächst die Return-Taste zu betätigen und ein Widerstand von ca. 4,7 kOhm von Pin 1 (oben links) zum Pin 14 (unten links) anzuklemmen. Das Voltmeter wird an diesen Widerstand angeschlossen (Pluspol an Pin 1), und das Potentiometer P2 wird so verdreht, daß sich eine Spannung von 12,5 V einstellt (Programmierspannung für die Typen 27256).

Für den dritten Abgleich ist zunächst wiederum die Return-Taste zu betätigen; der im zweiten Schritt angeklemmte Widerstand und das Voltmeter bleiben mit den Pins 1 und 14 verbunden. Das Potentiometer P3 wird nun so verdreht, daß sich eine Spannung von 21,Ø V einstellt (Programmierspannung für die Typen 2732A, 2764 und 27128).

Die Programmierspannung von 25,Ø V (für die Typen 2716 und 2732) wird vom Schaltnetzteil direkt geliefert; ein Abgleich ist nicht erforderlich.

Weitere Betätigungen der Return-Taste schalten zyklisch weiter zu den Schritten 1, 2, 3 und so fort, um beispielsweise die vorgenommenen Einstellungen zu kontrollieren. Der Rücksprung ins Prommer-Menü erfolgt durch die Betätigung der Escape-Taste bzw. CTL+C.

hms

```
Typ: 2764       SB:0400                                   Mi,08.04.87;09:09:56h
Vpp: 21 V       SE:047F
Tim: 0'00       DB:----
--------------------------------------------------------------------------------

MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>b                                        o.k.

P>b 400,47f                                o.k.

P>wB                                       Error # 86 at Adr. 0400:27=00100111
```

4. B: Blank Check (Leertest)

Format: P>B<Ret> oder P>B aaaa,eeee<Ret>

Funktion: EPROM ganz oder teilweise auf gelöschten Zustand (=Inhalt FFh) testen.

Parameter-Eintrag: SB:aaaa (Source Begin; EPROM-Anfangsadresse)
SE:eeee (Source End; EPROM-Endadresse)
DB:----

Zusatzfunktion: Bei der Kurzform-Eingabe (keine Angaben für die Adressen aaaa und eeee) werden folgende Parameter eingetragen:

SB:0000 (niedrigste EPROM-Adresse)
SE:07FF (2716-Endadresse)
0FFF (2732-Endadresse)
1FFF (2764-Endadresse)
3FFF (27128-Endadresse)
7FFF (27256-Endadresse)

Zulässig: Wiederholen des Kommandos mit denselben Parametern durch Eingabe von 'w' (ohne 'Return').

Quittung: o.k.

Error # 86: EPROM-Inhalt ist nicht gelöscht (mit Abbruch der Operation und Fehler-Listing). Mit 'Return' wird der Leertest bei der folgenden EPROM-Adresse fortgesetzt, und mit 'Escape' bzw. CTL+C erfolgt der Rücksprung ins Prommer-Menü.

```
Typ: 2764       SB:0D00                                  Mi,08.04.87;09:13:39h
Vpp: 21 V       SE:0FFF
Tim: 0'00       DB:8D00
--------------------------------------------------------------------------------


MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>c 8                                    o.k.

P>c d00,fff,8d00                         o.k.

P>wC                                     Error # 87 at Adr. 0D00:FF=11111111
```

5. C: Compare (EPROM- mit Speicher-Inhalt vergleichen)

Format: P>C k<Ret> oder P>C aaaa,eeee,zzzz<Ret>

Funktion: EPROM ganz oder teilweise mit Speicher-Inhalt vergleichen.

Parameter-Eintrag: SB:aaaa (Source Begin; EPROM-Anfangsadresse)
SE:eeee (Source End; EPROM-Endadresse)
DB:zzzz (Destination Begin; Speicher-Anfangsadr.)

Zusatzfunktion: Bei der Kurzform-Eingabe 'k' (einstellige Parameter-Eingabe) werden folgende Parameter eingetragen:

SB:0000 (niedrigste EPROM-Adresse)
SE:07FF (2716-Endadresse)
0FFF (2732-Endadresse)
1FFF (2764-Endadresse)
3FFF (27128-Endadresse)
7FFF (27256-Endadresse)
DB:k000 ('k' mit drei Nullen aufgefüllt)

Zulässig: Wiederholen des Kommandos mit denselben Parametern durch Eingabe von 'w' (ohne 'Return').

Quittung: o.k.

Error # 87: EPROM-Inhalt weicht vom Inhalt der korrespondierenden Speicherstelle ab (mit Abbruch der Operation und Fehler-Listing). Mit 'Return' wird der Vergleich bei der folgenden EPROM-Adresse fortgesetzt, und mit 'Escape' bzw. CTL+C erfolgt der Rücksprung ins Prommer-Menü.

hms

```
Vpp: 21 V        SE:----                              Mi,08.04.87;10:27:34h
Tim: Ø'00        DB:----
-------------------------------------------------------------------------------


MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>


M>
```

6. M: Monitor (Monitor-Rücksprung)

Format: P>M (ohne 'Return')

Funktion: Verlassen des Prommer-Menüs und Rücksprung zum Monitor-Warmstart.

hms

```
Typ: 2764          SB:7A00                                  Mi,08.04.87;10:36:34h
Vpp: 21 V          SE:7B00
Tim: 0'01          DB:0A00
-------------------------------------------------------------------------------
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>p 7
Vpp: 21 V?                                  o.k.

Stop at Adr. 096E:FF=11111111               o.k.

P>p 7a00,7b00,a00                           o.k.

P>wP                                        Error # 88 at Adr. 0AF0:FF=11111111
```

7. P: Program (EPROM programmieren)

Format: P>P k<Ret> oder P>P aaaa,eeee,zzzz<Ret>

Vor dem erstmaligen Programmieren ist die angezeigte Programmierspannung durch 'Return' zu bestätigen, um eine Beschädigung durch versehentliches Fehlbedienen zu vermeiden.

Funktion: EPROM ganz oder teilweise programmieren.

Parameter-Eintrag: SB:aaaa (Source Begin; Speicher-Anfangsadresse)
SE:eeee (Source End; Speicher-Endadresse)
DB:zzzz (EPROM-Anfangsadresse)

Zusatzfunktion: Bei der Kurzform-Eingabe 'k' (einstellige Parameter-Eingabe) werden folgende Parameter eingetragen:

SB:k000 (Speicher-Anfangsadresse)
SE:k+07FF (2716-Endadresse)
k+0FFF (2732-Endadresse)
k+1FFF (2764-Endadresse)
k+3FFF (27128-Endadresse)
k+7FFF (27256-Endadresse)
DB:0000 (niedrigste EPROM-Adresse)

Zulässig: Wiederholen des Kommandos mit denselben Parametern durch Eingabe von 'w' (ohne 'Return').

Quittung: o.k.

Error # 88: Programmieren fehlerhaft (mit Abbruch der Operation und Fehler-Listing). Mit 'Return' wird das Programmieren bei der folgenden EPROM-Adresse fortgesetzt, und mit 'Escape' bzw. CTL+C erfolgt der Rücksprung ins Prommer-Menü.

Hinweis: Während des Programmierens wird der Zeitzähler 'Tim' in den Kopfzeilen hochgezählt, um den Programmier-Verlauf kontrollieren zu können. Durch Betätigen der Leertaste ('Blank') kann das Programmieren jederzeit gestoppt werden (Anhalten des Zeitzählers und Angabe der erreichten Adresse); über 'Escape' bzw. CTL+C ist der Rücksprung ins Prommer-Menü möglich, während mit 'Return' das Programmieren folgerichtig fortgesetzt wird.

Die Typen 2716 und 2732 werden pro Byte mit einem Puls fester Länge (ca.50 ms) beaufschlagt; daraus resultiert eine feste Programmierzeit von Tim: 1'38 (1 Minute und 38 Sekunden) beim 2716 bzw. Tim: 3'14 beim 2732. Die Typen 2764, 27128 und 27256 werden im sogenannten 'Intelligenten Mode' mit variabler Pulslänge programmiert, was bedeutend schneller vor sich geht (Anhaltswerte: Tim: 0'40 beim 2764, Tim: 1'20 beim 27128 und Tim: 2'40 beim 27256).

```
Typ: 2764          SB:1000                          Mi,08.04.87;09:30:57h
Vpp: 21 V          SE:16F9
Tim: 0'00          DB:4000
--------------------------------------------------------------------------------


MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>r 9                                   o.k.

P>r 1000,16f9,a600                      o.k.

P>r 1000,16f9,4000                      Error # 89 at Adr. 1000:FF=11111111
```

8. R: Read (EPROM auslesen)

Format: P>R k<Ret> oder P>R aaaa,eeee,zzzz<Ret>

Funktion: EPROM-Inhalt ganz oder teilweise ins RAM übertragen.

Parameter-Eintrag: SB:aaaa (Source Begin; EPROM-Anfangsadresse)
SE:eeee (Source End; EPROM-Endadresse)
DB:zzzz (Destination Begin; Speicher-Anfangsadr.)

Zusatzfunktion: Bei der Kurzform-Eingabe 'k' (einstellige Parameter-Eingabe) werden folgende Parameter eingetragen:

SB:ØØØØ (niedrigste EPROM-Adresse)
SE:Ø7FF (2716-Endadresse)
ØFFF (2732-Endadresse)
1FFF (2764-Endadresse)
3FFF (27128-Endadresse)
7FFF (27256-Endadresse)
DB:kØØØ ('k' mit drei Nullen aufgefüllt)

Zulässig: Wiederholen des Kommandos mit denselben Parametern durch Eingabe von 'w' (ohne 'Return').

Quittung: o.k.

Error # 89: Einschreiben ins RAM fehlerhaft (mit Abbruch der Operation und Fehler-Listing). Mit 'Return' wird das Auslesen bei der folgenden EPROM-Adresse fortgesetzt, und mit 'Escape' bzw. CTL+C erfolgt der Rücksprung ins Prommer-Menü.

```
Typ:27128        SB:----                                   Mi,08.04.87;09:34:46h
Vpp: 21 V        SE:----
Tim: 0'00        DB:----
--------------------------------------------------------------------------------
MOPPEL-EPROM-Utilities V 16.6
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P>t 28                                   o.k.

P>t 128                                  o.k.

P>t 18                                   Error # 84

P>
```

9. T: Typ (EPROM-Typ einstellen)

Format: P>T tt<Ret>

Funktion: EPROM-Typenangabe einstellen bzw. ändern.

Der eingestellte EPROM-Typ wird im Feld 'Typ' der Kopfzeilen eingeblendet; er bleibt so lange unverändert erhalten, bis er durch eine Neueingabe oder nach einem Kaltstart verändert wird.

Parameter-Eintrag: SB:----
SE:----
DB:----

Hinweis: Es werden nur die beiden letzten Stellen der Eingabe ausgewertet, so daß folgende Zuordnung zwischen möglicher Eingabe und EPROM-Typ besteht:

16 = 2716
32 = 2732
64 = 2764
28 = 27128
56 = 27256

Quittung: o.k.

Error # 84: Typangabe unzulässig (korrekten Wert eingeben).

```
Typ: 2764          BB:----                                  M1,08.04.87:10:20:40h
Vop: 25 V          EE:----
Tim: 0'30          II:----
-------------------------------------------------------------------------------
MOPPEL-EPROM-Utilities V 16.0
Copyright (C) hms'86

 A: Adjust
 B: Blank Check
 C: Compare
 M: Monitor
 P: Program
 R: Read
 T: Typ
 V: Vpp

P:v 25                                   o.k.

P>v 125                                  o.k.

P>v 15                                   Error # 85

P>
```

10. V: Vpp (Programmierspannung einstellen)

Format: P>V vv<Ret>

Die eingestellte Programmierspannung wird im Feld 'Vpp' der Kopfzeilen eingeblendet; sie bleibt so lange unverändert erhalten, bis sie durch eine Neueingabe oder nach einem Kaltstart verändert wird.

Funktion: Programmierspannung einstellen bzw. ändern.

Parameter-Eintrag: SB:----
SE:----
DB:----

Hinweis: Es werden nur die beiden letzten Stellen der Eingabe ausgewertet, so daß folgende Zuordnung zwischen möglicher Eingabe und Programmierspannung besteht:

12 = 12,5 V (Standardwert für 27256)
21 = 21,Ø V (Standardwert für 2732A, 2764 und 27128)
25 = 25,Ø V (Standardwert für 2716 und 2732)

Quittung: o.k.

Error # 85: Spannungsangabe unzulässig (korrekten Wert eingeben).