

**KLEINCOMPUTER**

**KC compact**

**KC compact-Floppy** Handbuch  
für den Programmierer

KLEINCOMPUTER

---

# KC compact

KC compact-Floppy  
Handbuch für den Programmierer

*de-pro-ma* *electronic GmbH*  
Mühlhausen

Ohne Genehmigung des Herausgebers ist es nicht gestattet, das Buch oder Teile daraus nachzudrucken oder auf fotomechanischem Wege zu vervielfältigen.

Redaktionsschluß dieser Ausgabe: Juni 1990

## Inhaltsverzeichnis

Präambel .....	5
----------------	---

### Hardware

1. Das KC compact-Floppy-System .....	6
1.1. Speicheraufteilung .....	6
1.2. I/O-Adresse .....	6
1.3. Floppy-Disk-Interface .....	7

### Software

#### Teil A

#### MicroDOS-Betriebsart

1. Einführung .....	8
2. Systemdatenstrukturen .....	10
2.1. Systemparameter .....	10
2.2. Der Dateisteuerblock .....	10
2.3. Der Standardpuffer .....	11
2.4. Der Systemsteuerblock .....	12
3. Logische Geräte .....	13
3.1. Laufwerke .....	13
3.2. Konsole und Drucker .....	13
4. Kommandoeingabe .....	15
5. BDOS-Funktionen .....	16
5.1. Vorbemerkungen .....	16
5.2. Aufruf der BDOS-Funktionen .....	16
5.3. Liste der BDOS-Funktionen .....	17
6. BIOS-Funktionen .....	40
6.1. Beschreibung der BIOS-Schnittstelle .....	40
6.2. Initialisierung .....	40
6.3. Logische Ein/Ausgabekanäle .....	41
6.4. Liste der BIOS-Funktionen .....	43
6.5. Verwaltung der Diskettenlaufwerke .....	46

## Teil B

### BASDOS-Betriebsart

1.	Systemstart .....	53
2.	Schnittstellenbeschreibung .....	54
2.1.	Verschiebliche RAM-Variablen .....	54
2.2.	Nichtverschiebliche RAM-Variablen des BASDOS .....	56
2.3.	Im BASDOS genutzte Betriebssystemvektoren .....	57
2.4.	RSX-Erweiterungen des BASDOS .....	58
3.	Diskettenformate .....	64
4.	Dateiaufbau .....	65
4.1.	ASCII-Dateien .....	65
4.2.	Nicht-ASCII-Dateien .....	65

### Anlagen

Anlage 1:	Reservierte Speicherplätze (Seite 0) .....	66
Anlage 2:	Bildschirmsteuerzeichen .....	67
Anlage 3:	ESC-Funktionen .....	68
Anlage 4:	BDOS-Übersicht .....	71
Anlage 5:	Programmierbeispiele: .....	72
Anlage 5.1.	DUMP-Ausgabe .....	72
Anlage 5.2.	Datei in Speicher des Grundgerätes kopieren .....	76
Literaturverzeichnis	.....	78
Abkürzungsverzeichnis	.....	79

## **P r ä a m b e l**

Das Erarbeiten von Anwenderprogrammen für ein KC compact-Floppy-System hat vor allem dann Bedeutung, wenn nicht auf Standardsoftware zurückgegriffen werden kann, die Standardsoftware zu modifizieren (installieren) ist oder Sonderprobleme zu lösen sind. Voraussetzung für den Programmierer sind entsprechende Hardwarekenntnisse sowie Kenntnisse in der Assemblerprogrammierung /1/. In diesem Handbuch erfolgt die Beschreibung der Hardware des KC compact-Floppy-Systems aus der Sicht des Programmierers. Die Erläuterung der Systemgrundlagen erfolgt getrennt in zwei Teilen für die beiden möglichen Betriebsarten. Im Teil A erfolgt die Beschreibung der Komponenten des MicroDOS zur Nutzung in Maschinenprogrammen /2/ bzw. In höheren Programmiersprachen /3/, /4/, /5/, /6/. Im Teil B zur BASDOS-Betriebsart werden Hinweise zu deren Nutzung in Anwenderprogrammen und zum Aufbau der Diskettendateien gegeben.

## Hardware

### 1. Das KC compact-Floppy-System

#### 1.1. Speicheraufteilung

Die KC compact-Floppy-Elektronik enthält 64 KByte dynamischen RAM und 8 KByte ROM, die durch den KC compact verwaltet werden. Sie fügen sich folgendermaßen in die Speicherverwaltung des KC compact ein:

Durch die Ausgabe eines Datenwortes auf das Bankswitchregister wird eine von acht möglichen RAM-Konfigurationen eingestellt. Das Bankswitchregister hat die I/O-Adresse 7FXXCX (siehe KC compact-Systemhandbuch S.8f.). Wenn der RAM des KC compact und der KC compact-Floppy-Elektronik in 16-KByte-Bereiche eingestellt wird und diese Bereiche von 0 bis 7 durchnummeriert werden, kann die Speicheraufteilung übersichtlich dargestellt werden. Die Speicherbänke 0 bis 3 sind im KC compact und die Speicherbänke 4 bis 7 in der KC compact-Floppy-Elektronik enthalten. Die folgende Tabelle enthält die Zuordnung der Speicherkonfigurationen zu den auf das Bankswitchregister ausgegebenen Datenbytes.

Datenbyte	Speicherbank auf			
	0000FH-3FFFH	4000H-7FFFH	8000H-BFFFH	C000H-FFFFH
C0H	0	1	2	3
C1H	0	1	2	7
C2H	4	5	6	7
C3H	reserviert			
C4H	0	4	2	3
C5H	0	5	2	3
C6H	0	6	2	3
C7H	0	7	2	3

Tabelle 1: Zuordnung der Speicherkonfiguration zu den Steuerbytes

Das ROM-Select (siehe KC compact-Systemhandbuch S. 17) wird benutzt, um den Erweiterungs-ROM der KC compact-Floppy-Elektronik in den Speicherbereich des KC compact einzublenden. Der ROM hat die Nummer 7.

#### 1.2. I/O-Adressen

Die KC compact-Floppy-Elektronik enthält einen Floppy-Disk-Kontroller und ein Flipflop zum Schalten der Motoren der Diskettenlaufwerke, die im I/O-Adreßraum des KC compact liegen. Sie sind über das Doppelregister BC zu adressieren (siehe KC compact-Systemhandbuch S. 8). Mit

```
LD BC,0FA7EH
OUT (C),C
```

werden die Motoren der Diskettenlaufwerke ausgeschaltet, mit

```
LD BC,0FA7FH
OUT (C),C
```

eingeschaltet. Der Floppy-Disk-Kontroller hat die Adressen FB7EH für sein Hauptstatusregister und FB7FH für das Datenregister.

### 1.3. Floppy-Disk-Interface

Der externe Anschluß für Diskettenlaufwerke hat die in folgender Tabelle aufgeführte Belegung.

Anschluß	Signal	Bedeutung	
A1	/FLT		Masse
A2	SE3		Masse
A3	/IX	Index - Spuranfang	Eingang
A4	SE0	Select Drive 0 - Auswahl	Ausgang
A5	SE2	Select Drive 0 - Auswahl	n.c.
A6	GND		Masse
A7	/ST	Step - Schritt	Ausgang
A8	/WD	Write Data - Schreibdaten	Ausgang
A9	/WE	Write Enable - Schreibbefehl	Ausgang
A10	GND		Masse
A11	GND		Masse
A12	GND		Masse
A13	/RDY	Ready - Bereitschaftsmeldung	Eingang
B1	/MO	Motor On - Motor ein	Ausgang
B2	GND		Masse
B3	GND		Masse
B4			Masse
B5	SE1	Select Drive 0 - Auswahl	Ausgang
B6	/TS	Two Side - zweiseitig	n.c.
B7	/ST	Step Direction - Schrittrichtung	Ausgang
B8	GND		Masse
B9	GND		Masse
B10	/TR0	Track 0 - Spur 0	Eingang
B11	/WP	Write Protected - Schreibschutz	Eingang
B12	/RD	Read Data - Lesedaten	Eingang
B13	/SS	Side Select - Kopfauswahl	Ausgang

n.c. (non connected):

Die KC compact-Floppy-Elektronik benutzt diese Signale nicht.

*Tabelle 2: Anschlußbelegung des Diskettenlaufwerkes*

## Software

### Teil A

## MicroDOS-Betriebsart

### 1. Einführung

Dieser Teil des Handbuches beschreibt die Systemorganisation, die Speicherstruktur und die Eintrittspunkte des Betriebssystems MicroDOS. Es werden die notwendigen Informationen gegeben, um Programme zu schreiben, welche unter MicroDOS laufen.

MicroDOS ist logisch in drei Teile aufgeteilt:

- in das Basis-Ein/Ausgabe-System (BIOS),
- in das Basis-Platten-Betriebssystem (BDOS), welches auch den Kommandoprozessor enthält und
- in das Feld für zeitweilig geladene Programme (TPA).

Das BIOS ist ein hardwareabhängiger Modul, welcher die Anbindung von peripheren Geräten auf einem niedrigen Niveau definiert. Das BDOS gewährleistet einen benutzerorientierten Zugriff zum Massenspeicher und den anderen peripheren Geräten. Als Besonderheit ist hierbei zu beachten, daß zum BIOS gehörende Programmteile im RAM des KC kompakt liegen, während alle anderen Teile des MicroDOS im RAM der KC compact-Elektronik abgearbeitet werden. Der TPA ist der Speicherbereich, in dem verschiedene nichtresistente Teile des Betriebssystems und Anwenderprogramme ausgeführt werden. Die unteren 256 Bytes des Speichers sind für Systemparameter reserviert.

Die Speicher im KC compact und in der KC compact-Floppy-Elektronik werden wie folgt aufgeteilt:

RAM der KC compact-Floppy-Elektronik

RAM des KC compact

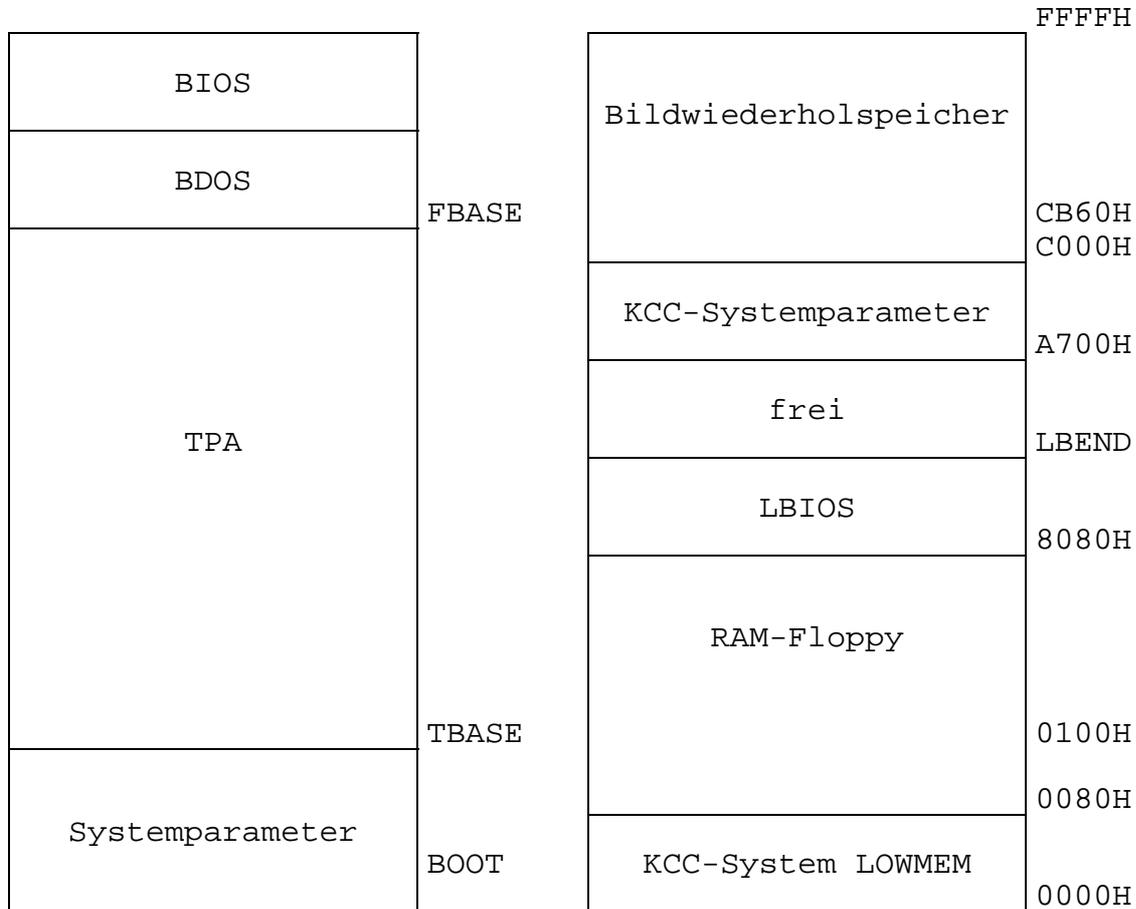


Bild 1: Speicheraufteilung im KC compact-Floppy-System

Die exakte Adresse von FBASE ist abhängig von der konkreten Version des MicroDOS. Der Maschinencode auf der Speicherzelle BOOT führt einen System-Warmstart durch, wobei alle Programme und Variablen geladen bzw. initialisiert werden, welche zur Übergabe der Steuerung an den Kommandoprozessor notwendig sind. Transiente Programme brauchen deshalb nur zur Position BOOT zu springen, um zum Kommandoniveau von MicroDOS zurückzukehren. Der prinzipielle Eintrittspunkt zum BDOS liegt bei BOOT +0005H, wo sich ein Sprung nach FBASE befindet. Die Zelle BOOT +0006H enthält den Wert von FBASE und kann zur Bestimmung des verfügbaren Speichers benutzt werden.

## 2. Systemdatenstrukturen

Im folgenden werden einige grundlegende Datenstrukturen des Betriebssystems erläutert.

### 1. Systemparameter

Auf den ersten 256 Bytes des Speichers befinden sich folgende vom System benutzte Datenfelder, die nachfolgend beschrieben werden.

Adresse	Bedeutung
0000H	Sprung zum Warmstart
0003H	I/O-Byte 0004H aktuelles Laufwerk
0005H	Sprung zum BDOS
0040H-0042H	Systemuhr
005CH	erster Dateisteuerblock
006CH	zweiter Dateisteuerblock
0080H	Standard-DMA-Puffer

Der Sprung auf Adresse 0000H führt direkt zum entsprechenden Anspring in der BIOS-Sprungtabelle. Es werden der Warmstart durchgeführt und die Kontrolle an den Kommandoprozessor übergeben. Durch die Sprungadresse kann die Adresse des CP/M-kompatiblen BIOS Sprungvektors ermittelt werden.

Das I/O-Byte dient der Aufspaltung der logischen Kanäle auf verschiedene physische Geräte.

Die Nummer des aktuellen Laufwerkes wird aus Kompatibilitätsgründen zu CP/M 2.2 auf der Adresse 0004H abgelegt.

Auf Adresse 0005H befindet sich der allgemeine Eintrittspunkt in das BDOS. Gleichzeitig gibt die Sprungadresse den ersten für das System reservierten Speicherplatz oberhalb des TPA an.

Ab Adresse 005CH werden vom CCP, entsprechend den im Kommando angegebenen Parametern, ein oder zwei Dateisteuerblöcke angelegt (siehe auch Abschnitt 2.2). Für die Benutzung im Anwenderprogramm muß der zweite Dateisteuerblock auf einen anderen Speicherplatz verschoben werden, da dieser sonst vom BDOS überschrieben würde.

Auf Adresse 0080H beginnt der Standard-DMA-Puffer, welcher vom System für die Diskettenarbeit benutzt wird. Beim Aufruf eine Kommandos wird hier zusätzlich die Kommandozeile abgelegt.

### 2.2. Der Dateisteuerblock

Der Dateisteuerblock ist eine Datenstruktur, die vom Dateisystem beim Zugriff auf die Dateien über das Verzeichnis verwendet wird. Alle Operationen mit Dateien benötigen als Ausgangsinformation diese Daten. Außerdem belegen die Funktionen des wahlfreien Zugriffs die drei Bytes hinter dem Dateisteuerblock. Beim Aufruf der Dateifunktionen adressiert das Registerpaar DE den Dateisteuerblock für die interessierenden Datei.

Der Dateisteuerblock (FCB) besteht aus einem Feld von 33 Bytes im Fall des sequentiellen Zugriffs und einer Folge von 36 Bytes bei wahlfreiem Zugriff. Der Standard-FCB auf Adresse 005CH kann für

wahlfreien Zugriff benutzt werden, wenn die drei Bytes ab 007DH für diesen Zweck zur Verfügung stehen.

Im folgenden ist das Format des Dateisteuerblockes aufgezeigt:

```
dr f1 f2 --- f8 t1 t2 t3 ex s1 s2 rc d0 --- dn cr r0 r1 r2
00 01 02 --- 08 09 10 11 12 13 14 15 16 --- 31 32 33 34 35
```

mit:

```
dr      Laufwerkscode 0-16
        0 - aktuell ausgewähltes Laufwerk
        1 - Laufwerk A
        2 - Laufwerk B usw.

f1...f8 Dateiname in ASCII-Großbuchstaben Bit 7=0
        t1...t3   Dateityp in ASCII-Großbuchstaben Bit 7=0
                t1: Bit 7=1 - schreibgeschützte Datei
                t2: Bit 7=1 - Systemdatei
        ex       aktuelle Erweiterungsnummer, normalerweise =0, bei Ein/Ausgabe im Bereich 0-31
        s1       für interne Systembenutzung reserviert
        s2       für interne Systembenutzung reserviert
                =0 bei Datei eröffnen, erstellen, suchen
        rc       Datensatzanzahl (von 0-128)
        d0...dn  wird vom Betriebssystem ausgefüllt
        cr       Datensatzzähler für sequentiellen Zugriff
                wird vom Benutzer auf 0 gesetzt
        r0...r2  Nr. für wahlfreien Datensatz von 0-65535 mit Überlauf in r2, niederwertiges Byte in r0, höherwertiges Byte in r1
```

Jede Datei, die mit Hilfe von MicroDOS benutzt wird, muß einen dazugehörigen FCB haben, welcher den Dateinamen und eine Sektorbelegung für alle folgenden Operationen bereitstellt. Wenn auf eine Datei zugegriffen werden soll, ist es Sache des Programmierers, die niederen 16 Bytes des FCB auszufüllen und das "cr"-Feld zu initialisieren. Normalerweise werden die Bytes 1-11 mit ASCII-Zeichen für Dateinamen und Typ belegt, während alle anderen Bytes mit 0 gefüllt werden.

Die Dateisteuerblöcke werden im Inhaltsverzeichnis der Diskette gespeichert und vor der Durchführung einer Dateioperation in den Hauptspeicher gebracht (siehe BDOS-Funktionen Datei eröffnen bzw. erzeugen im Abschnitt 5.3.) Die Speicherkopie des Dateisteuerblockes wird während der Dateioperationen korrigiert und nach Abschluß derselben auf der Diskette abgespeichert (siehe Funktion Datei schließen).

### 2.3. Der Standardpuffer

Der Standardpuffer ab Adresse 80H wird vom System als Zwischenspeicher für Diskettenoperationen genutzt. Auch Anwenderprogramme können diesen Puffer verwenden. Zusätzlich wird vom Kommandointerpreter der Teil der Kommandozeile, der auf den Programmnamen folgt, im Standardpuffer abgelegt (siehe Abschnitt 4.).

## 2.4. Der Systemsteuerblock

Der Systemsteuerblock ist eine Datenstruktur, die sich im Basisdiskettenbetriebssystem befindet. MicroDOS nutzt diesen Bereich für die Sicherung der Systemdaten und Verbindungen von BDOS und BIOS. Die Anwenderprogramme können auch einige Parameter, die sich in diesem Bereich befinden, nutzen. Programme, die den Systemsteuerblock nutzen, können nur mit MicroDOS oder SCP bzw. CP/M Version 3.0 arbeiten.

Im folgenden sind die Felder des Systemsteuerblocks aufgeführt, die den Nutzerprogrammen zugänglich sind. Die Position jedes Feldes wird durch den Versatz bezüglich des Systemsteuerblockanfangs angegeben.

00-04	reserviert
05	Versionsnummer
06-09	Anwenderflags
0A-0F	reserviert
10-11	Rückkehrcode
12-19	reserviert
1A	Konsolenbreite
1B	aktuelle Spalte des Cursors
1C	Seitenlänge
1D-21	reserviert
22-23	Zuordnungsvektor für Konsoleneingabe
24-25	Zuordnungsvektor für Konsolenausgabe
26-27	Zuordnungsvektor für Zusatzeingabe
28-29	Zuordnungsvektor für Zusatzausgabe
2A-2B	Zuordnungsvektor für DruckerAusgabe
2C	Seitenmodus
2D	reserviert
2E	Backspace (^H) aktiv (0: Zeichen wird im Puffer und auf dem Bildschirm gelöscht 1: Zeichen wird nochmals auf dem Bildschirm ausgegeben)
2F	Rubout (DEL) aktiv (Bedeutung wie bei Backspace)
30-34	reserviert
35-36	Pufferadresse für Kalt- und Warmstart
37	Begrenzer für Zeichenkette (Standard \$=24H)
38	Druckerflag
39-3B	reserviert
3C-3D	aktuelle DMA-Adresse
3E	aktuelles Laufwerk
3F-43	reserviert
44	aktuelle Nutzernummer
45-49	reserviert
4A	Multisektorzähler (=1)
4B	BDOS-Fehlermodus (siehe BDOS-Funktion 45)
4C-4F	Laufwerke für Dateisuche
50	Laufwerk für temporäre Dateien
51	Laufwerksnummer für Fehleranzeige
54	Flag für Diskettenwechsel
52-56	reserviert
57	Flag für Länge der Fehlermeldung
58-59	Datum (nicht verwendet)
5A-5C	Zeit (nicht verwendet)
5D-5E	Adresse des gemeinsamen Speicherbereichs
5F-61	Sprung zur Fehlermeldung
62-63	Adresse des TPA-Endes (wird auf 6,7 kopiert)

### 3. Logische Geräte

#### 3.1. Laufwerke

Die auf dem KC compact-Floppy-System implementierte Version des MicroDOS verwaltet maximal acht logische Laufwerke mit den Bezeichnungen A...H.

Laufwerk A ist stets das RAM-Floppy (RAM-Disk), dessen Kapazität 32 KByte beträgt. Laufwerk B ist beim Start das Systemlaufwerk. Es liegt immer auf dem physischen Laufwerk 0 und hat das Format 5\*1024\*80\*2 mit zwei Systemspuren (780K). Die Laufwerke C...H können beliebig installiert werden.

#### 3.2. Konsole und Drucker

Als Konsole werden die Bildschirmausgabe und die Tastatureingabe bezeichnet.

Die Standardbildschirmausgabe (siehe Abschnitt 6.3.) erfolgt mit 24 Zeilen zu 80 Zeichen. Anlage 2 enthält die auch im SCP bzw. CP/M üblichen Steuerzeichen. Darüber hinaus enthält die Konsolenschnittstelle eine Vielzahl zusätzlicher Funktionen zur optimalen Nutzung der Möglichkeiten des KC compact. Zu diesen Funktionen gehören unter anderem die Grafik und die Farbsteuerung. Alle diese Funktionen werden über ESC-Folgen gesteuert. Durch Ausgabe des ESC-Codes und eines Unterscheidungscode wird die Konsolenausgabe auf die Sonderfunktion umgeschaltet. Anschließend wird ggf. eine der Funktion entsprechende Anzahl von Parametern übertragen. Anlage 3 gibt einen Überblick über die ESC-Funktionen. Bei Benutzung der ESC-Funktionen (außer der auch im SCP üblichen Cursorpositionierung) sollte über die BIOS-Schnittstelle (Abschnitt 6.4.) gearbeitet werden, da die BDOS-Schnittstelle (Abschnitt 5.2.) die Codierung 09H in einer Parameterliste als Tabulator interpretiert und in Leerzeichen (20H) umsetzt.

Als Beispiel sei das Setzen eines Punktes auf die Position X=100, Y=50 angegeben:

```
LD    A,18H      ;ESC
CALL  BIOSOUT   ;Konsolenausgabe
LD    A,'A'     ;Funktionscode
CALL  BIOSOUT
LD    A,100     ;X-Wert low
CALL  BIOSOUT
LD    A,0       ;X-Wert high
CALL  BIOSOUT
LD    A,50     ;Y-Wert
CALL  BIOSOUT
```

Die Tastaturbelegung wurde gegenüber der Standardbelegung im KC compact geändert. Durch einige ESC-Folgen können anwenderspezifische Tastaturcodierungen erzeugt werden. Einige Tasten haben die sofortige Ausführung ihrer Funktion unter Umgehung des BDOS und des eventuell laufenden Anwenderprogramms zur Folge.

Einige Tasten und Tastenkombinationen werden im MicroDOS direkt ausgeführt. Auf der Ebene eines Anwenderprogrammes bewirken sie keine Reaktion. Sie dienen zum Einstellen des Bildschirms und der Tastatur. Folgende Funktionen werden ausgeführt:

- CAPS LOCK Die Tastaturbelegung wird gewechselt
- ^ COPY Es wird ein Hardcopy des Bildschirminhaltes ausgedruckt Die Steuerzeichen sind für EPSON-kompatible Drucker gültig.
- ^ Cursor ein/aus. Es wird nur die Taste "." in der Nähe der CTRL-Taste angenommen.
- ^ 7 Wechsel der Vordergrundfarbe. Die 27 Farben werden von 0 bis 9 und von A bis Q durchnummeriert. Zur Eingabe einer entsprechenden Taste wird am unteren Bildrand aufgefordert.
- ^ 8 Wechsel der Hintergrundfarbe (entspricht ^7).
- ^ 9 Wechsel des Zeichensatzes. Mit dieser Tastenkombination kann zwischen amerikanischem und deutschem Zeichensatz hin- und hergeschaltet werden.

Die Standardbelegung aller Tasten im MicroDOS wird in der Anlage des KC compact-Floppy-Benutzerhandbuches aufgelistet.

Die Standarddruckerausgabe des MicroDOS arbeitet über einen Druckertreiber im RAM des KC compact. Dieser wird als CENTRONICS-Treiber mit acht Datenbits installiert.

## 4. Kommandoeingabe

Beim Programmaufruf über die Kommandoeingabe analysiert der Kommandoprozessor die eingegebene Zeile und entscheidet, ob ein residentes Kommando eingegeben wurde oder ein Programm von Diskette geladen werden muß.

Der Kommandoprozessor konstruiert die ersten sechzehn Bytes von zwei möglichen Dateisteuerblöcken für ein transientes Programm durch Prüfung des Teils der Kommandozeile, der nach dem Namen des transienten Programms folgt. Dabei werden nichtspezifizierte Felder mit Leerzeichen gefüllt. Der erste Dateisteuerblock wird ab Adresse 005CH aufgebaut und kann in dieser Form für nachfolgende Dateioperationen genutzt werden. Der Anfang des zweiten Dateisteuerblocks liegt im d0...dn-Feld der ersten Dateisteuerung und muß vor der Benutzung in einen anderen Teil des Speichers umgeladen werden. Wenn z.B. der Benutzer folgende Zeile eingibt:

```
PROGRAMME B:DATEI1.XXX DATEI2.YYY
```

so wird die Datei "PROGRAMME" in den TPA geladen und der Standard-Dateisteuerblock ab Adresse 005CH mit dem Laufwerk 2, dem Dateinamen "DATEI1" und dem Dateityp "XXX" initialisiert. Der zweite Laufwerkscode bekommt den Wert ( ) zugewiesen, welcher ab Adresse 006CH geladen wird. Der Dateiname "DATEI2" wird ab Adresse 006DH abgelegt und der Dateityp "YYY" wird acht Bytes dahinter (Adresse 0075H) angeordnet.

Alle restlichen Felder bis "cr" werden auf Null gesetzt. Es muß nochmals darauf hingewiesen werden, daß es dem Programmierer obliegt, den zweiten Dateinamen und Dateityp in einen anderen Speicherbereich zu verschieben, bevor die Datei mit dem Steuerblock auf 005CH eröffnet werden kann, angesichts der Tatsache, daß die Eröffnungsfunktion den zweiten Dateinamen und -typ anderenfalls überschreibt.

Falls keine Dateinamen in dem Kommando angegeben wurden, enthalten die Felder ab 005CH und 006CH Leerzeichen. In jedem Fall werden Kleinbuchstaben in Großbuchstaben umgewandelt, um den MicroDOS-Dateinamenvereinbarungen zu entsprechen.

Für das oben angeführte Kommando würde der Puffer folgenden Inhalt aufweisen:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
18   B  :  D  A  T  E  I  1  .  X  X  X           D  A  T  E  I  2  .  Y  Y  Y
```

wobei das erste Byte die Anzahl der gültigen Zeichen angibt und dann folgen die Zeichen selbst, welche vom Kommandoprozessor gegebenenfalls in Großbuchstaben umgesetzt werden. Die Kommando-parameter müssen vom Programm ausgewertet werden, bevor der Puffer durch Diskettenoperationen überschrieben wird.

## 5. BDOS-Funktionen

### 5.1. Vorbemerkungen

Das Diskettengrundbetriebssystem (BDOS) stellt den Kern des MicroDOS-Betriebssystems dar. Es unterstützt die Ein- und Ausgabe auf den logischen Geräten (Konsole, Drucker, Zusatzkanäle) sowie die Dateiarbeit auf den Disketten. Über BDOS-Rufe können folgende Aufträge angefordert werden:

- zeichenorientierte Ein- und Ausgabe
  - \* Zuordnung der logischen zu den physischen Geräten
  - \* zeichen- u. zeichenkettenweise Ein- u. Ausgabe über Konsole
  - \* zeichenweiser Datenaustausch über die Zusatzkanäle
  - \* zeichenweise Ausgabe über den Druckerkanal
- Arbeit mit Disketten
  - \* Dateiverwaltung und -manipulation allgemein
  - \* sequentieller und direkter Dateizugriff
- Systemverwaltung
  - \* Initialisierung des Systems
  - \* Ermittlung des Systemzustandes
  - \* Zugriff auf Systemparameter

### 5.2. Aufruf der BDOS-Funktionen

Im folgenden werden die einzelnen Systemfunktionen mit ihren Eingangs- und Ausgangsparametern beschrieben. Der Aufruf der Systemfunktionen erfolgt durch einen CALL auf die Adresse 0005H, wobei die Funktionsnummer im Register C und notwendige Eingangsparameter im Register DE übergeben werden. Bei der Rückkehr werden die Ausgangsparameter im Akkumulator und im Registerpaar HL zurückgegeben. Feldeintragungen werden in einem Feld übergeben, dessen Anfang beim Aufruf im Registerpaar DE adressiert wird.

Es ist zu beachten, daß alle Register vom System verändert werden können. Deshalb ist es angebracht, wichtige Registerinhalte auf den Stack zu retten.

Das Betriebssystem benutzt einen eigenen Stapel, so daß das aufrufende Programm nur mit einer Stapelebene durch den BDOS-Aufruf belastet wird.

Bei der Benutzung der BDOS-Funktionen ist zu beachten, daß nur bei den BDOS-Rufen bis einschließlich Nummer 40 volle Kompatibilität zum SCPX gewährleistet ist. Das heißt, daß Programme die nur diese BDOS-Rufe verwenden, auch unter SCPX und CP/M 2.2 lauffähig sind. Programme, die die anderen BDOS-Rufe nutzen, können u.U. leistungsfähiger sein. Bei ihnen ist Kompatibilität, zu CP/M 3.0 und SCP 3.0 (PC1715W) gegeben.

MicroDOS entspricht einer CP/M-Version 2.6 und liegt damit im Funktionsumfang zwischen SCPX (CP/M 2.2) und SCP3 (CP/M 3.0).

### 5.3. Liste der BDOS-Funktionen

Die BDOS-Funktionen sind nach den Funktionsnummern geordnet.

#### **Funktion 0: System rücksetzen**

---

Eingangsparameter:  
Register C: 00H

Diese Funktion gibt die Steuerung an MicroDOS auf dem Kommando-Niveau zurück (Warmstart). Sie hat den gleichen Effekt wie ein Sprung auf die Adresse BOOT. Vor dem Aufruf der Funktion 0 kann durch Funktion 108 ein Rückkehrcode gesetzt werden.

#### **Funktion 1: Konsoleneingabe**

---

Eingangsparameter:  
Register C: 01H  
  
Ausgangsparameter:  
Register A: ASCII-Zeichen

Die Konsoleneingabe-Funktion liest das nächste Zeichen von der Konsole in Register A ein. Darstellbare Zeichen und die Steuerzeichen Wagenrücklauf, Zeilenvorschub und Rückwärtsschritt (CTRL-H) werden zur Konsole zurückgegeben. Tabulatorzeichen (CTRL-I) werden expandiert auf Spalten mit acht Zeichen. Es wird überprüft, ob Start/Stop der Bildschirmausgabe (CTRL-S) oder Start/Stop der Druckerausgabe (CTRL-P) auftreten. Das BDOS kehrt nicht eher zum aufrufenden Programm zurück, bevor nicht ein Zeichen auf der Konsole eingegeben wurde.

#### **Funktion 2: System rücksetzen**

---

Eingangsparameter:  
Register C: 02H  
Register E: ASCII-Zeichen

Das in Register E enthaltene Zeichen wird zur Konsole gesendet. Wie bei Funktion 1 wird die Überprüfung von Start/Stop der Bildschirm- und Druckerausgabe durchgeführt. Ist die parallele Druckerausgabe eingeschaltet, so werden alle Zeichen auch an den Drucker gesendet.

**Funktion 3: Zusatzeingabe**

---

Eingangsparameter:

Register C: 03H

Ausgangsparameter:

Register A: ASCII-Zeichen

Die Zusatzeingabe liest ein Zeichen von dem zusätzlichen Kanal in das Register A. Die Steuerung kehrt nicht zurück, bevor nicht ein Zeichen gelesen wurde.

**Funktion 4: Zusatzausgabe**

---

Eingangsparameter:

Register C: 04H

Register E: ASCII-Zeichen

Die Zusatzausgabefunktion sendet das im Register E enthaltene Zeichen zum zusätzlichen logischen Kanal.

**Funktion 5: Druckerausgabe**

---

Eingangsparameter:

Register C: 05H

Register E: ASCII-Zeichen

Die Druckerausgabefunktion sendet das im Register E enthaltene Zeichen zum logischen Drucker.

**Funktion 6: direkte Konsolen-Ein/Ausgabe**

---

Eingangsparameter:

Register C: 06H

E: 0FFH (Eingabe) oder  
ASCII-Zeichen (Ausgabe)

Ausgangsparameter:

Register A: Zeichen oder Status

Die direkte Ein/Ausgabe wird von MicroDOS unterstützt für solche speziellen Anwendungen, wo einfache Ein/Ausgabeoperationen gefordert sind. Im allgemeinen sollte die Benutzung dieser Funktion vermieden werden, da die normalen Steuerzeichenfunktionen von MicroDOS (d.h. CTRL-D und CTRL-P) umgangen werden. Bei Eintritt in die Funktion 6 enthält Register E entweder den hexadezimalen Wert 0FFH, was eine Konsoleneingabe kennzeichnet, kehrt die Funktion mit A=0 zurück, falls kein Zeichen von der Konsole bereitsteht. Ansonsten enthält Register A das nächste Eingabezeichen von der Konsole. Ist der Wert des Registers E nicht gleich 0FFH, so behandelt die Funktion 6 diesen Wert als gültiges ASCII-Zeichen und sendet dieses zur Konsole.

#### **Funktion 7: Status Zusatzausgabe**

---

Eingangsparameter:

Register C: 07H

Ausgangsparameter:

Register A: Status

Die Funktion 7 realisiert in der auf dem KC compact-Floppy-System vorliegenden Implementierung des MicroDOS die Abfrage des Zustandes der Druckerausgabe. Wenn freier Platz im Übergabepuffer zum Druckertreiber ist, wird im Akkumulator der Wert 0FFH übergeben, ansonsten der Wert 00H.

#### **Funktion 8: Status Zusatzausgabe**

---

Eingangsparameter:

Register C: 07H

Ausgangsparameter:

Register A: Status

Die Funktion 8 ist in der vorliegenden Implementierung des MicroDOS kurzgeschlossen und übergibt stets im Akkumulator den Wert 00H.

#### **Funktion 9: Zeichenkette ausgeben**

---

Eingangsparameter:

Register C: 09H

Registerpaar DE: Adresse der Zeichen

Die Zeichenkettenausgabefunktion sendet Zeichen aus dem Speicher, beginnend ab der in DE übergebenen Adresse, zur Konsole, bis in der Zeichenkette das im Systemsteuerblock gesetzte Endekenn-

zeichen erkannt wird. Standardmäßig wird '\$' als Endekennzeichen angenommen. Tabulatoren werden expandiert, wie in Funktion 2, und Start/Stop-Zeichen für Bildschirm- und Druckerausgabe werden ausgewertet. Bei paralleler Druckerausgabe werden alle Zeichen auf dem Drucker ausgegeben.

### **Funktion 10: Konsolenpuffer lesen**

Eingangsparameter:

Register C: 0AH  
Registerpaar DE: Pufferadresse

Ausgangsparameter:

Zeichen von der Konsole im Puffer

Diese Funktion liest eine Zeile von eingegebenen Zeichen in einen Puffer, der durch DE adressiert ist. Die Eingabe wird abgeschlossen, wenn der Eingabepuffer überläuft oder wenn ein RETURN eingegeben wird.

Der Puffer erhält folgende Form

```
DE +0 +1 +2 +3 +4 +5 +6 +7 ... +n
mx nc c1 c2 c3 c4 c5 c6 ... ??,
```

wobei mx die maximale Anzahl von einzulesenden Zeichen (1 bis 255) und nc die tatsächliche Anzahl von eingelesenen Zeichen (vom BDOS bei Rückkehr gesetzt) ist. Es folgen die eingelesenen Zeichen. Ist nc kleiner als mx, so folgen dem letzten Zeichen unbestimmte Informationen, die im obigen Beispiel mit "??" gekennzeichnet sind. Eine Reihe von Steuerzeichen wird während der Eingabe erkannt.

DEL	löscht das letzte Zeichen
CTRL-C	Neustart von MicroDOS (am Beginn der Zeile)
CTRL-E	gibt das physikalische Ende der Zeile an
CTRL-H	Rückwärtsschritt
CTRL-J	(Zeilenvorschub) beendet die Eingabe
CTRL-M	(RETURN) beendet die Eingabe
CTRL-R	schreibt die aktuelle Zeile neu
CTRL-U	löscht die aktuelle Zeile
CTRL-X	geht zurück zum Anfang der Zeile
CTRL-W	wiederholt den letzten Pufferinhalt
CTRL-^	löscht das letzte Wort

Die Erneuerung des Pufferinhalts bei Eingabe von CTRL-W erfolgt über den Pufferzähler. Wird dieser im Anwenderprogramm verändert, kann der regenerierte Inhalt falsch sein.

**Funktion 11: Konsolenstatus holen**

---

Eingangsparameter:  
Register C: 0BH

Ausgangsparameter:  
Register A: Konsolenstatus

Die Funktion Konsolenstatus überprüft, ob ein Zeichen auf der Konsole eingegeben wurde. Falls ein Zeichen eingegeben wurde, so wird im Register A ein Wert ungleich 0 übergeben, im allgemeinen der Wert 01H. Ist kein Eingabezeichen vorhanden, kehrt die Funktion mit A=0 zurück.

**Funktion 12: Versionsnummer holen**

---

Eingangsparameter:  
Register C: 0CH

Ausgangsparameter:  
Registerpaar HL: Versionsnummer

Die Funktion 12 übergibt einen 2-Byte-Wert, wobei H=00H anzeigt, daß es sich um ein CP/M 2.2-kompatibles System handelt (H=01H bei CP/M). Im Register L übergibt MicroDOS einen hexadezimalen Wert entsprechend der Versionsnummer des Systems, also z.B. 26H bei der Version 2.6. Die Verwendung dieser Funktion erlaubt das Schreiben von versionsabhängigen Programmen.

**Funktion 13: Diskettensystem zurücksetzen**

---

Eingangsparameter:  
Register C: 0DH

Diese Funktion wird benutzt, um vom Programm aus das Dateisystem in den Anfangszustand zu setzen, Dabei sind alle Laufwerke für Schreiben und Lesen zugelassen (siehe Funktionen 28 und 29), Laufwerk A ist angewählt und die DMA-Adresse ist auf den Standardwert 0080H festgelegt. Diese Funktion kann von Programmen verwendet werden, die während der Abarbeitung einen Diskettenwechsel erfordern. Siehe dazu auch Funktion 37.

**Funktion 14: Laufwerk anwählen**

---

Eingangsparameter:

Register C: 0BH

Register E: Laufwerkscode

Die Funktion kennzeichnet das in E bezeichnete Laufwerk als das aktuelle Laufwerk für die folgenden Dateioperationen, wobei E=0 für Laufwerk A steht, 1 für Laufwerk B usw. bis 7 für Laufwerk H in einem voll ausgebauten System mit acht logischen Laufwerken.

Das Laufwerk wird in einem "on line"-Status versetzt, welcher insbesondere das entsprechende Inhaltsverzeichnis bis zum nächsten Kaltstart, Warmstart oder Systemrücksetzen aktiviert. Dateisteuerblöcke, die den Laufwerkscode 0 aufweisen, beziehen sich automatisch auf das aktuelle Laufwerk. Laufwerkscodes von 1 bis 16 ignorieren die Standard-Anwahl und beziehen sich direkt auf ein Laufwerk von A bis H. War die Operation erfolgreich, wird im Akkumulator der Wert 00H zurückgegeben. Tritt jedoch ein Fehler auf, so wird bei der Fehlerbehandlung durch das System eine entsprechende Meldung ausgegeben und der Warmstart vollzogen. Bei Fehlerbearbeitung durch das Anwenderprogramm enthält Register A den Wert 0FFH und Register H einen der folgenden Fehlercodes:

01: Diskettenfehler

04: Auswahlfehler

**Funktion 15: Datei eröffnen**

---

Eingangsparameter:

Register C: 0FH

Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Die Dateieröffnung wird benutzt, um eine Datei zu aktivieren, die bereits auf der angewählten Diskette existiert. Das BDOS sucht im Inhaltsverzeichnis nach Übereinstimmung mit den Positionen 1 bis 14 des durch DE adressierten Speicherblocks (Byte s1 ist automatisch auf Null gesetzt), wobei ein Fragezeichen (3FH) in jeder dieser Positionen als Übereinstimmung gewertet wird. Normalerweise ist jedoch keine Fragezeichen eingefügt und weiterhin sind die Bytes "ex" und "s2" gleich Null. Wenn ein Eintrag des Inhaltsverzeichnisses übereinstimmt, wird die relevante Information in die Bytes "d0" bis "dn" des Dateisteuerblocks kopiert, wodurch der Zugriff auf diese Datei bei nachfolgenden Lese- und Schreiboperationen ermöglicht wird. Auf eine Datei darf nicht zugegriffen werden, bevor nicht eine entsprechende Eröffnung erfolgreich durchgeführt wurde. Die Funktion 15 übergibt in A einen "directory code" mit dem Wert von 0 bis drei, falls die Eröffnung erfolgreich durchgeführt wurde. Dieser Wert kennzeichnet, an welcher Stelle im DMA-Puffer der interessierende Eintrag liegt. Falls die Datei

nicht gefunden wurde, wird im Register A der Wert 0FFH übergeben. Der aktuelle Datensatz "cr" muß vom Programm auf Null gesetzt werden, wenn die Datei sequentiell vom ersten Datensatz an gelesen werden soll.

Bei Fehlerbehandlung im Anwenderprogramm wird einer der folgenden Fehlercodes im Register H übertragen:

00: Datei existiert nicht  
01: Diskettenfehler  
04: Auswahlfehler  
09: mehrdeutiger Name

#### **Funktion 16: Datei schließen**

---

Eingangsparameter:

Register C: 10H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Diese Funktion führt die Umkehrung der Dateieröffnung durch. Wurde der durch DE adressierte Dateisteuerblock durch eine vorhergehende Eröffnung oder Erzeugung (siehe Funktionen 15 und 22) aktiviert, so speichert die Dateischließung den Dateisteuerblock in das Inhaltsverzeichnis der Diskette. Der FCB-Suchvorgang ist derselbe wie bei der Eröffnungsfunktion. Der Ausgangsparameter im Akkumulator bei einem erfolgreichen Abschluß ist 0, 1, 2 oder 3, während 0FFH übergeben wird., falls der Dateiname im Inhaltsverzeichnis nicht gefunden wurde. Nach Leseoperationen braucht eine Datei nicht geschlossen zu werden. Nach Schreiboperationen ist in jedem Fall die Datei zu schließen, um den neuen Eintrag im Inhaltsverzeichnis abzuspeichern. Die Fehlercodes für die Fehlerbehandlung im Anwenderprogramm entsprechen denen der Funktion 15.

#### **Funktion 17: ersten Eintrag suchen**

---

Eingangsparameter:

Register C: 11H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Diese Funktion durchsucht das Inhaltsverzeichnis auf Übereinstimmung mit dem durch DE adressierten Dateisteuerblock. Der Wert 0FFH wird übergeben, wenn die Datei nicht gefunden wurde, andernfalls die Werte 0, 1, 2 oder 3, welche anzeigen, daß die Datei vorhanden ist. Der aktuelle DMA-Bereich wird mit dem Datensatz des Inhaltsverzeichnisses gefüllt, welcher dem FCB entspricht. Das Register A gibt in diesem Fall die Nummer des Eintrages im Datensatz an. Die relative Anfangsadresse kann folglich mit  $A*32$  berechnet werden. Obwohl es normalerweise für Anwenderprogramme nicht notwendig ist, kann der Inhaltsverzeichniseintrag von dieser Position geholt werden.

Ein Fragezeichen (3FH) in irgendeiner Position von "f1" bis "ex" bedeutet Übereinstimmung mit dem entsprechenden Feld im Inhaltsverzeichnis des angewählten Laufwerkes. Wenn das "dr"-Feld ein Fragezeichen enthält, ist die automatische Laufwerksauswahl unterbunden und das aktuelle Laufwerk wird auf Übereinstimmung geprüft, wobei die Suchfunktion jeden übereinstimmenden Eintrag übergibt, unabhängig von der Benutzernummer. Wenn das "dr"-Feld kein Fragezeichen enthält, wird das "s2"-Byte automatisch auf Null gesetzt.

Auch diese Funktion übergibt die oben genannten Fehlercodes an das aufrufende Programm, wenn die Fehlerbearbeitung durch das Anwenderprogramm eingestellt ist.

### **Funktion 18: nächsten Eintrag suchen**

---

Eingangsparameter:

Register C: 12H  
Registerpaar DE: FCB-Adresse

Register A: Verzeichniscode

Diese Funktion entspricht der Funktion 17 bis auf die Tatsache, daß die Suche im Inhaltsverzeichnis ausgehend vom letzten übereinstimmenden Eintrag fortgesetzt wird. Wie bei Funktion 17 übergibt die Funktion 18 den Wert (OFFH), wenn kein Eintrag mehr gefunden wurde.

Für die richtige Ausführung der Funktion 18 dürfen zwischen den Operationen 17 und 18 und zwischen aufeinanderfolgenden Aufrufen der Funktion 18 keine anderen Diskettenoperationen des BDOS folgen.

### **Funktion 19: Datei löschen**

---

Eingangsparameter:

Register C: 13H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Diese Funktion entfernt Dateien, die mit dem durch DE adressierten Dateisteuerblock übereinstimmen. Dateiname und Dateityp können mehrdeutig sein. (d.h., Fragezeichen in verschiedenen Positionen). Der Laufwerkscode darf nicht mehrdeutig sein, wie bei den Funktionen 17 und 18. Die Funktion 19 übergibt bei erfolgreicher Abarbeitung im Akkumulator den Verzeichniscode von 0 bis 3 und beim Auftreten eines Fehlers den Wert (OFFH). Im Modus der Fehlerbearbeitung durch das System erfolgen eine Fehlerermittlung und anschließend der Warmstart. Bei Fehlerbearbeitung durch das Anwenderprogramm werden die folgenden Codes übergeben:

- 00: Datei existiert nicht
- 01: Diskettenfehler
- 03: Datei geschützt (Schreibschutz oder Systemdatei)
- 04: Auswahlfehler.

**Funktion 20: sequentiell lesen**

---

Eingangsparameter:

Register C: 14H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Fehlercode

Wurde der durch DE adressierte Dateisteuerblock mittels der Funktion 15 aktiviert, so liest diese Funktion den nächsten 128 Bytes langen Datensatz in den Puffer mit der aktuellen DMA-Adresse. Der Datensatz wird von der Position "cr" der Erweiterung gelesen und das "cr"-Feld wird automatisch auf die nächste Position erhöht. Wenn das "cr"-Feld überläuft, wird die nächste logische Erweiterung automatisch eröffnet und das "cr"-Feld für die nächste Leseoperation auf Null gesetzt. Der Wert 00H wird im Register A übergeben, wenn die Leseoperation erfolgreich war, während ein Wert ungleich Null übergeben wird, falls bei der Operation ein Fehler auftrat. Dabei kennzeichnen der Wert 01H das Datenende und der Wert 0FFH das Auftreten eines physischen Fehlers. Im letzteren Fall enthält das Register H einen der folgende Fehlercodes:

01: Diskettenfehler  
04: Auswahlfehler**Funktion 21: sequentiell schreiben**

---

Eingangsparameter:

Register C: 15H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Wenn der durch DE adressierte Dateisteuerblock durch eine der Funktionen 15 oder 22 aktiviert wurde, schreibt diese Funktion den 128 Bytes langen Datensatz auf der aktuellen DMA-Adresse in die durch diesen FCB gekennzeichnete Datei.

Der Datensatz wird an der durch "cr" vorgegebenen Position abgelegt, und das "cr"-Feld wird automatisch auf die nächste Position erhöht. Läuft das "cr"-Feld über, wird automatisch die nächste logische Erweiterung eröffnet und das "cr"-Feld wird in Vorbereitung der nächsten Schreiboperation auf Null gesetzt. Schreiboperationen können in existierenden Dateien ausgeführt werden, wobei neu geschriebene Datensätze die schon existierenden überschreiben. Register A wird von einer erfolgreichen Schreiboperation mit dem Wert 00H übergeben, während ein Wert ungleich Null eine nicht ausgeführte Schreiboperation anzeigt. Dabei bedeuten die Werte im einzelnen:

01: kein Platz im Inhaltsverzeichnis  
02: kein Platz auf der Diskette  
FF: physischer Fehler (siehe Register H).

Beim Auftreten eines physischen Fehlers wird im Register H einer der folgenden Fehlercodes übergeben:

01: Diskettenfehler  
03: Datei geschützt  
04: Auswahlfehler.

### **Funktion 22: Datei erzeugen**

---

Eingangsparameter:

Register C: 16H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Diese Funktion gleicht der Eröffnungsfunktion mit dem Unterschied, daß der Dateisteuerblock eine Datei bezeichnen muß, die noch nicht in dem ausgewählten Inhaltsverzeichnis existiert (d.h., in demjenigen Verzeichnis, das explizit durch ein "dr"-Feld ungleich Null vorgegeben ist bzw. bei "dr"=0 in dem aktuellen Verzeichnis). Das BDOS erzeugt einen Eintrag für eine leere Datei im Inhaltsverzeichnis. Der Programmierer muß sicher sein, daß doppelte Dateinamen nicht auftreten, d.h., vorhergehende Löschoptionen sind erforderlich, wenn die Möglichkeit der Duplizität besteht. Bei Rückkehr enthält Register A die Werte 0, 1, 2 oder 3, wenn die Operation erfolgreich war oder ansonsten den Wert 0FFH. Im letzteren Fall wird im Register H die Fehlerursache übergeben:

01: Diskettenfehler  
04: Auswahlfehler  
08: Datei vorhanden  
09: mehrdeutiger Name.

Die Funktion aktiviert den FCB, so daß keine Eröffnung notwendig ist.

### **Funktion 23: Datei erzeugen**

---

Eingangsparameter:

Register C: 17H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Verzeichniscode

Diese Funktion benutzt den durch DE adressierten Dateisteuerblock, um alle Dateieinträge mit dem in den ersten 16 Bytes angegebenen Namen in den mit den zweiten 16 Bytes angegebenen Namen umzubenennen. Der Laufwerkscode "dr" auf Position 00 wird benutzt, um das Laufwerk anzugeben, während der Code auf Position 16 mit Null angenommen wird. Bei Rückkehr ist das Register A auf einen Wert von 0 bis 3 gesetzt, wenn die Umbenennung erfolgreich war oder 0FFH, falls keine Datei mit dem ersten angegebenen Namen vorhanden ist.

**Funktion 24: Abfrage Laufwerke im 'online'-Zustand**

---

Eingangsparameter:

Register C: 18H

Ausgangsparameter:

Register HL: Anwahlvektor

Der Anwahlvektor, der vom MicroDOS-System übergeben wird, ist ein 16-Bit-Wert in HL, bei dem das Bit 0 von Register L dem Laufwerk A entspricht und das Bit 7 des Registers H dem sechzehnten Laufwerk P (in der Implementierung auf KC compact-Floppy-System bis max. Laufwerk H) entspricht. Ein "0"-Bit zeigt an, daß das Laufwerk noch nicht angewählt wurde, während ein "1"-Bit ein durch eine explizite Anwahl bzw. durch eine Dateioperation mit einem "dr"-Feld ungleich Null angewähltes Laufwerk markiert. Kompatibilität mit älteren Versionen des Betriebssystems CP/M ist gesichert, da bei der Rückkehr Register A gleich Register L ist.

**Funktion 25: Datei erzeugen**

---

Eingangsparameter:

Register C: 19H

Ausgangsparameter:

Register A: aktuelles Laufwerk

Funktion 25 holt die Nummer des momentan angewählten Laufwerks in Register A. Die Laufwerksnummer liegt im Bereich von 0 bis 7 entsprechend den Laufwerken A bis H.

**Funktion 26: DMA-Adresse setzen**

---

Eingangsparameter:

Register C: 1AH

Registerpaar DE: DMA-Adresse

Im MicroDOS wird die Adresse, auf der vor einer Schreiboperation bzw. nach einer Leseoperation der 128 Byte lange Datensatz liegt, mit DMA-Adresse bezeichnet. Nach einem Kaltstart, einem Warmstart Rücksetzen des Diskettensystems ist die DMA-Adresse automatisch auf 0080H gesetzt. Die DMA-Adresssetzungsfunktion kann benutzt werden, um diesen Standardwert zur Adressierung eines anderen Speicherfeldes, wo die Daten liegen, zu ändern. Dadurch erhält die DMA-Adresse den in DE vorgegebenen Wert bis zu einer folgenden Adresssetzungsfunktion oder einem Neustart bzw. RESET. Einige Funktio-

nen des BDOS nutzen den DMA-Puffer für die Übergabe von Parametern, so übergibt z.B. die Funktion 46 die Größe des freien Diskettenplatzes im DMA-Puffer.

### **Funktion 27: Belegungsvektor holen**

---

Eingangsparameter:

Register C: 1BH

Ausgangsparameter:

Registerpaar HL: Belegungsvektor

Für jedes angewählte Laufwerk wird im Hauptspeicher eine Belegungstabelle angelegt. Verschiedene Programme nutzen die Information des Vektors, um die Größe des verbleibenden Speicherplatzes festzustellen. Funktion 27 übergibt die Basisadresse der Belegungstabelle für das gerade angewählte Laufwerk. Die Information kann jedoch ungültig sein, wenn die Diskette in diesem Laufwerk gewechselt wurde. Beim Auftreten eines Fehlers wird im Registerpaar HL der Wert 0FFFFH übergeben.

### **Funktion 28: Schreibschutz setzen**

---

Eingangsparameter:

Register C: 1CH

Diese Funktion gewährleistet einen vorübergehenden Schreibschutz für das gerade angewählte Laufwerk. MicroDOS führt die Diskette beim Schreiben automatisch in den Schreib/Lesezustand zurück. Funktion 28 wird nur zwecks Kompatibilität zu SCPX bzw. CP/M 2.2 ausgeführt.

### **Funktion 29: Schreibschutzvektor holen**

---

Eingangsparameter:

Register C: 1DH

Ausgangsparameter:

Registerpaar HL: Schreibschutzvektor

Funktion 29 übergibt im Registerpaar HL einen Bit-Vektor, der diejenigen Laufwerke angibt, bei denen ein vorübergehender Schreibschutz gesetzt ist. Wie bei Funktion 24 entspricht das Bit 0 vom Register L dem Laufwerk A, während das Bit 7 vom

Register H dem Laufwerk P zugeordnet ist. Das R/O wird durch die Funktion 28 gesetzt.

### **Funktion 30: Dateiattribute setzen**

---

Eingangsparameter:

Register C: 1EH  
Registerpaar HL: FCB-Adresse

Ausgangsparameter:

Registerpaar A: Verzeichniscode

Diese Funktion erlaubt die programmtechnische Manipulation von ständigen Indikatoren, die den Dateien zugeordnet sind. Im einzelnen werden die Schreibschutz- und Systemattribute (t1 und t2) gesetzt oder rückgesetzt. Das Registerpaar DE adressiert einen eindeutigen Dateisteuerblock mit den gesetzten Attributen. Funktion 30 sucht im Inhaltsverzeichnis nach Übereinstimmung und ändert den entsprechenden Eintrag, damit er die gewählten Attribute enthält. Die Indikatoren f1 bis f4 werden gegenwärtig nicht genutzt, können aber für Anwenderprogramme nützlich sein, da sie nicht vom Suchprozeß während einer Dateieröffnung oder einer Dateischließung berührt werden. Die Funktion 30 gibt bei erfolgreichem Abschluß im Register A den Verzeichniscode von 0 bis 3 zurück. Im Fehlerfall enthält Register A den Wert 0FFH und das Register H einen der folgenden Fehlercodes:

00: Datei existiert nicht  
01: Diskettenfehler  
04: Auswahlfehler  
09: mehrdeutiger Name.

### **Funktion 31: Parameteradresse holen**

---

Eingangsparameter:

Register C: 1FH

Ausgangsparameter:

Registerpaar HL: DPB-Adresse

Die Adresse des BIOS-residenten Diskettenparameterblocks (DPB) für das gerade aktive Laufwerk wird als Resultat dieser Funktion im Registerpaar HL übergeben. Diese Adresse kann für zwei Zwecke benutzt werden. Erstens können die Diskettenparameter für Anzeigzwecke oder zur Berechnung von Speicherplatz geholt werden oder transiente Programme können die Parameter ändern, wenn die Diskettenumgebung wechselt.

**Funktion 32: Benutzernummer holen/ersetzen**

---

## Eingangsparameter:

Register C: 20H  
Register E: 0FFH oder Benutzernummer

## Ausgangsparameter:

Register A: Benutzernummer oder kein Wort

Durch Aufruf der Funktion 32 kann ein Anwenderprogramm die aktuelle Benutzernummer ändern oder abfragen. Ist Register E=0FFH, so wird der augenblickliche Wert der Benutzernummer im Register A übergeben, wobei der Wert im Bereich von 0 bis 15 liegt. Ist Register E ungleich 0FFH, so wird die Benutzernummer auf den Wert von E gesetzt(modulo 16).

**Funktion 33: Parameteradresse holen**

---

## Eingangsparameter:

Register C: 21H  
Registerpaar DE: PCB-Adresse

## Ausgangsparameter:

Register A: Fehlercode

Die Funktion "wahlfrei lesen" ähnelt der Funktion "sequentiell lesen" mit dem Unterschied, daß die Leseoperation mit einer bestimmten Datensatznummer durchgeführt wird, welche durch den 24-Bit-Wert aus den drei dem FCB folgenden Bytes gebildet wird (Bytepositionen r0 bei 33, r1 bei 34 und r2 bei 35). Die Folge der 24 Bits ist mit dem niederwertigsten Byte (r0) zuerst, dem mittleren Byte (r1) in der Mitte und dem höherwertigsten Byte (r2) zuletzt abgespeichert. Das MicroDOS-System benutzt das r2-Byte nicht, ausgenommen bei der Berechnung der Dateigröße (Funktion 35). Byte r2 muß Null sein, da r2 ungleich Null einen Überlauf nach dem Dateiende anzeigt.

Das "r0,r1"-Bytepaar wird als Doppelbytewert behandelt, welcher den zu lesenden Datensatz angibt. Dieser Wert reicht von 0 bis 65535 und gewährleistet den Zugriff zu jedem einzelnen Datensatz einer 8-MByte-Datei. Um eine Datei im wahlfreien Zugriff zu verarbeiten, muß zuerst die Basiserweiterung 0 mit Funktion 15 oder 22 eröffnet werden. Unabhängig davon, ob die gewünschten Daten hierin enthalten sind oder nicht, wird dadurch gesichert, daß die Datei im Inhaltsverzeichnis eingetragen ist und entsprechenden Operationen zugänglich ist. Die ausgewählte Datensatznummer wird dann im Feld r0, r1 abgelegt und das BDOS wird aufgerufen, um den Datensatz zu lesen. Bei der Rückkehr enthält Register A einen Fehlercode, wie unten aufgeführt, oder den Wert 00H, wenn die Operation erfolgreich war. In diesem Fall enthält der momentane DMA-Bereich den gelesenen Datensatz. Im Gegensatz zum sequentiellen Lesen wird die Datensatznummer nicht erhöht. Deshalb würden nachfolgende Leseoperationen denselben Datensatz lesen.

Bei jeder wahlfreien Leseoperation werden die Werte für die logische Erweiterung und den momentanen Datensatz automatisch gesetzt. Deshalb kann die Datei sequentiell, ausgehend von der durch wahlfreien Zugriff

bestimmten Position, gelesen oder geschrieben werden. In diesem Fall wird der letzte Datensatz noch einmal gelesen beim Übergang vom wahlfreien zum sequentiellen Lesen bzw. der letzte Datensatz wird überschrieben beim Übergang vom wahlfreien zum sequentiellen Schreiben. Selbstverständlich kann die Datensatznummer nach jedem wahlfreien Zugriff erhöht werden, um den Effekt des sequentiellen Zugriffs zu erreichen. Es folgen die Fehlercodes, die im Register A übergeben werden:

- 01 Lesen von nicht existenten Daten
- 02 (wird nicht von wahlfreien Operationen übergeben)
- 03 die momentane Erweiterung kann nicht geschlossen werden
- 04 Suche nach nicht geschriebener Erweiterung
- 05 (wird nicht von Leseoperationen übergeben)
- 06 Suche über das physikalische Ende der Diskette hinaus
- FF physischer Fehler (siehe Register H)

Die Fehlercodes 1 und 4 treten auf, wenn eine wahlfreie Leseoperation auf einen Datenblock zugreift, welcher vorher nicht geschrieben wurde bzw. auf eine Erweiterung, die nicht erzeugt wurde. Fehlercode 3 tritt normalerweise bei richtiger Funktion des Systems nicht auf, kann jedoch durch einfaches Neulesen oder durch Neueröffnung der Erweiterung 0 gelöscht werden, solange die Diskette nicht physikalisch schreibgeschützt ist. Der Fehlercode 6 tritt auf, wenn das Byte r2 nicht Null ist. Normalerweise können auftretende Fehlercodes als Datenverlust angesehen werden. Der Wert 0FFH zeigt einen physischen Fehler an, dessen Ursache im Register H genauer spezifiziert wird. Der Rückkehrcode 00H zeigt vollständige Operationen an.

#### **Funktion 34: wahlfrei schreiben**

Eingangsparameter:

Register C: 22H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Fehlercode

Die Funktion "wahlfrei schreiben" wird auf die gleiche Weise eingeleitet wie die Funktion "wahlfrei lesen", nur werden hierbei Daten vom momentanen DMA-Bereich auf die Diskette geschrieben. Wenn die Erweiterung oder der Datenblock, auf welchen geschrieben werden soll, noch nicht belegt wurden, wird dies vor der Schreiboperation ausgeführt. Wie bei der Leseoperation wird im Ergebnis der Funktion die Datensatznummer nicht verändert. Die Nummer der Erweiterung und die Datensatznummer werden im Dateisteuerblock gesetzt in Übereinstimmung mit dem Datensatz, welcher geschrieben wurde. Wiederum können sequentielle Operationen folgen, wobei auch hier der momentan adressierte Datensatz noch einmal gelesen oder geschrieben wird, wenn die sequentielle Operation beginnt. Ebenso kann die Nummer des Datensatzes nach jeder Operation erhöht werden, um den Effekt des sequentiellen Schreibens zu erhalten. Allerdings erfolgt nach dem letzten Datensatz einer Erweiterung nicht ein automatisches Umschalten auf die nächste, wie beim sequentiellen Schreiben. Die übergebenen Fehlercodes sind die gleichen wie beim wahlfreien Lesen, wobei zusätzlich der Fehlercode 5 auftritt, welcher anzeigt, daß eine neue Erweiterung auf Grund eines Überlaufs des Inhaltsverzeichnisses nicht erzeugt werden kann.

**Funktion 35: Dateigröße berechnen**

---

Eingangsparameter:

Register C: 23H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Dateigröße in r0,r1,r2

Bei der Ermittlung der Dateigröße adressiert das Registerpaar DE einen Dateisteuerblock im Format für wahlfreien Zugriff, d.h., die Bytes r0 bis r2 sind vorhanden. Der Dateisteuerblock enthält einen eindeutigen Dateinamen, welcher im Inhaltsverzeichnis gesucht wird. Bei Rückkehr enthalten die Bytes für wahlfreien Zugriff die "virtuelle" Dateigröße, welche effektiv die Nummer des Datensatzes darstellt, der dem Dateieinde folgt. Wenn nach Aufruf der Funktion 35 das höchste Datensatzbyte r2 gleich 1 ist, dann enthält die Datei die maximale Anzahl von Datensätzen (65535). Anderenfalls bilden die Bytes r0 und r1 einen 16-Bit-Wert, welcher die Dateigröße darstellt (wie vorher ist r0 das niederwertigste Byte).

Daten können durch Aufruf der Funktion 35 an das Ende einer existierenden Datei angefügt werden, um die Positionen für wahlfreien Zugriff mit dem Ende der Datei zu belegen, und durch anschließendes wahlfreies Schreiben von Datensätzen, beginnend an der vorbelegten Datensatzadresse.

Die "virtuelle" Größe der Datei stimmt mit der realen Größe überein, wenn die Datei sequentiell geschrieben wurde. Ist die Datei jedoch im wahlfreien Modus geschrieben worden, dann existieren Lücken in der Platzierung der Datensätze und die Datei enthält weniger Datensätze, als durch die Größe angezeigt wird. Wenn z.B. nur der letzte Datensatz einer 8-MByte-Datei im wahlfreien Zugriff geschrieben wird (d.h., Datensatznummer 65535), wird die "virtuelle" Größe der Datei mit 65535 Datensätzen angegeben, obwohl nur ein Datenblock vorhanden ist.

**Funktion 36: Feld für wahlfreien Zugriff setzen**

---

Eingangsparameter:

Register C: 24H  
Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Feld für wahlfreien Zugriff setzen

Diese Funktion setzt automatisch das Feld für wahlfreien Zugriff einer Datei, die bis zu einem bestimmten Punkt sequentiell gelesen oder geschrieben wurde. Die Funktion kann auf zwei Arten nützlich sein. Erstens ist es oft notwendig, eine Datei zuerst sequentiell zu lesen und nach verschiedene "Schlüssel"-Informationen zu durchsuchen. Ist der Schlüssel gefunden, so wird Funktion 36 aufgerufen, um die Position des zugehörigen Datensatzes zu bestimmen. Die so gewonnene Datensatzposition kann in einer Tabelle für späteren Zugriff abgelegt werden. Nach-

dem die Datei durchsucht wurde und alle Schlüsseldatensätze in einer Tabelle abgelegt wurden, kann man auf einen bestimmten Datensatz über einen wahlfreien Zugriff, unter Benutzung der vorher abgespeicherten Datensatznummer, zurückgreifen. Dieses Schema kann leicht auf Dateneinheiten mit variabler Länge ausgedehnt werden, da dann zusätzlich zum Schlüssel und zur Datensatznummer nur die relative Position im Datensatz abgespeichert werden muß, um die exakte Startposition der Schlüsselinformation später zu finden.

Die zweite Verwendungsmöglichkeit der Funktion 36 tritt beim Umschalten vom sequentiellen Lesen oder Schreiben auf den wahlfreien Zugriff auf. Wenn eine Datei bis zu einem bestimmten Punkt sequentiell bearbeitet wurde, kann durch Aufruf der Funktion 36, welche die Datensatznummer setzt, die wahlfreie Bearbeitung ab dem bis dahin erreichten Punkt der Datei erfolgen.

### **Funktion 37: Diskette rücksetzen**

---

Eingangsparameter:

Register C: 25H

Registerpaar DE: Diskettenvektor

Ausgangsparameter:

Register A: 00H

Funktion 37 realisiert programmgesteuert das Rücksetzen der Disketten, die durch den im Registerpaar DE übermittelten Vektor angegeben werden. Beim Rücksetzen wird die Diskette logisch ausgeschaltet und in das Regime Lesen/Schreiben versetzt. Das niedrigste Bit des Diskettenvektors entspricht dem Laufwerk A, das höchste dem Laufwerk P. Der Wert 1 des Bits bedeutet, daß die entsprechende Diskette zurückgesetzt werden muß.

### **Funktion 40: Wahlfreies Schreiben mit Auffüllen durch Nullen**

---

Eingangsparameter:

Register C: 1EH

Registerpaar DE: FCB-Adresse

Ausgangsparameter:

Register A: Rückkehrcode

Register H: Fehlercode

Diese Funktion arbeitet analog zur Funktion des wahlfreien Schreibens 34 mit der Ausnahme, daß der zugewiesene Block vor dem Schreiben mit Nullen aufgefüllt wird. Wenn beim Anlegen der Datei diese Funktion genutzt wurde, enthalten die freien Sätze in den Blöcken Nullen. Wenn Operation 34 verwendet wurde, enthalten die freien Sätze nicht initialisierte Daten.

**Funktion 45: Festlegen des Regimes  
der Fehlerbereinigung**

---

## Eingangsparameter:

Register C: 2DH  
Register E: Regime der Bearbeitung

Funktion 45 legt das Regime der Fehlerbearbeitung des BDOS fest. Wenn im Register E der Wert 0FEH (254 dezimal) angegeben wird, dann kehrt das BDOS mit einem Fehlercode im Akkumulator zum aufrufenden Programm zurück. In allen anderen Fällen wird bei Auftreten eines Fehlers dieser auf der Konsole angezeigt, und der Benutzer hat die Möglichkeit, den Fehler zu ignorieren oder einen Warmstart auszuführen.

**Funktion 46: Abfragen des freien Speicherplatzes  
auf der Diskette**

---

## Eingangsparameter:

Register C: 2EH  
Register E: Diskette

## Ausgangsparameter:

Register A: Fehlerkennzeichen  
Register H: Fehlercode

Funktion 46 bestimmt die Anzahl der freien Sektoren (128 Byte lange Sätze) auf der Diskette, die durch Register E angegeben wird. Der Wert 0 entspricht der Diskette A, 1 der Diskette B usw. bis 15 für die Diskette P. Der Wert der Anzahl der freien Sektoren wird im Dual-Code in den ersten drei Bytes des aktuellen Puffers des Direktzugriffs (DMA) in folgendem Format zurückgegeben:

Byte 0 - niedrigstes Byte  
Byte 1 - mittleres Byte  
Byte 2 - höchstes Byte

Funktion 46 gibt bei erfolgreichem Abschluß im Register A den Wert 00H zurück. Beim Auftreten eines physischen Fehlers werden bei Fehlerbearbeitung durch das System die Fehlermitteilung auf der Konsole ausgegeben und der "Warmstart" vollzogen. Im Modus der Fehlerbearbeitung durch das Anwenderprogramm wird im Register A der Wert 0FFH zurückgegeben, dabei enthält Register H einen der folgenden Fehlercodes:

01: Diskettenfehler  
04: Auswahlfehler

**Funktion 47: Wechsel des Programms**

---

## Eingangsparameter:

Register C: 2FH

Funktion 47 ermöglicht den Aufruf eines anderen Programms aus dem laufenden Programm ohne Zusammenwirken mit dem Bediener. Das aufrufende Programm muß in den Puffer des Direktzugriffs bei Abbruch (ab Adresse 0080H) einen Kommandosatz einspeichern.

Die Funktion des Wechsels des Programms gibt dem aufrufenden Programm nicht die Steuerung zurück. Die festgestellten Fehler werden vom Kommandointerpreter bearbeitet.

#### **Funktion 49: Abfragen/Stellen der Parameter des Systemsteuerblocks**

---

Eingangsparameter:

Register C: 31H  
Registerpaar DE: Adresse der Parameter

Ausgangsparameter:

Register A: Parameterbyte  
Registerpaar HL: Parameterwort

Funktion 49 ermöglicht den Zugriff auf den Systemsteuerblock. Der Systemsteuerblock ist ein 100 Byte großer Datenbereich (Abschnitt 3.4.), der Marken und Daten beinhaltet, die vom System genutzt werden. Für die Anwendung dieser Funktion speichert das aufrufende Programm im Registerpaar DE die Adresse des Parameterblocks, der die Funktion bestimmt. Die Struktur des Parameterblocks kann folgendermaßen beschrieben werden:

```
SCBPB:  DEFB  OFFSET  ;Position im Systemsteuerblock
        DEFB  SET      ;0FFH-Setzen des Bytes
        DEFB  SET      ;0FEH-Setzen des Wortes
        DEFB  SET      ;00-Abfrage des Parameters
        DEFW  VALUE    ;Parameter zum Setzen
        DEFW  VALUE    ;(Byte oder Wort)
```

OFFSET bestimmt die Position des Parameters, der gestellt oder abgefragt werden muß, innerhalb des Systemsteuerblocks. SET bestimmt, ob der Parameter gelesen oder geschrieben wird und ob es sich um einen Byte- oder Wortparameter handelt. VALUE enthält das Wort oder das Byte für das Setzen.

Die Funktion 49 muß mit Vorsicht angewendet werden, da der Systemblock Systemvariablen enthält, deren Veränderung durch die Anwendungsprogramme zu Fehlern in der Arbeit des Systems führen kann. Funktion 49 ist zum Stellen der Parameter nur dann zu verwenden, wenn kein analoges Resultat durch eine andere Funktion erreicht werden kann.

#### **Funktion 50: Aufruf der BIOS-Funktionen**

---

Eingangsparameter:

Register C: 32H  
Registerpaar DE: Adresse der Parameter

Ausgangsparameter:

BIOS-Parameter

Die Funktion 50 realisiert den Aufruf der BIOS-Funktionen. Das aufrufende Programm übergibt im Registerpaar DE die Adresse des Parameterblocks, der die Nummer der BIOS-Funktion und die zu ihrer Ausführung notwendigen Parameter bestimmt. Dieser Parameterblock hat folgende Struktur:

```

BIOSBP:  DEFB  FUNC ;Nummer der Funktion
         DEFW  REGA ;Inhalt des Registers A
         DEFW  REGB ;Inhalt des Registerpaars BC
         DEFW  REGD ;Inhalt des Registerpaars DE
         DEFW  REGH ;Inhalt des Registerpaars HL

```

Die Nummer der Funktion im Feld FUNC für den Aufruf der BIOS-Funktion muß sich in den Grenzen von 00H...10H (0...16) bewegen. Es folgt eine Aufstellung der Funktionsnummern, der notwendigen Eingabedaten, der zurückgegebenen Werte und der Wirkungen dieser Funktionen.

#### Funktion Bedeutung

- |    |   |
|----|---|
| 0  | Kaltstart   |
| 1  | Warmstart   |
| 2  | Konsolenstatus<br>Wenn ein Zeichen bereit ist zur Eingabe, wird in Register A der Wert 0FFH zurückgegeben, sonst 00H.   |
| 3  | Konsoleneingabe<br>Das ein gegebene Zeichen wird in Register A zurückgegeben.   |
| 4  | Konsolenausgabe<br>Ausgabe des Zeichens in Register C auf die Konsole.  |
| 5  | Druckerausgabe<br>Ausgabe eines Zeichens aus dem Register C auf den Drucker.  |
| 6  | Zusatzausgabe<br>Ausgabe eines Zeichens in Register C auf den Zusatzkanal.  |
| 7  | Zusatzeingabe<br>Eingabe eines Zeichens vom Zusatzkanal in das Register A.  |
| 8  | Setzen des Diskettenkopfes auf Spur 0.  |
| 9  | Wahl des Laufwerks<br>Register C enthält 0 für Diskette A, 1 für B usw. bis 0FH für Diskette P. Die Funktion gibt im Registerpaar HL die Adresse des Parameterkopfes des Laufwerks zurück. Wird ein nicht existierendes Laufwerk angegeben, wird im Registerpaar HL der Wert 0000H zurückgegeben. |
| 10 | Setzen auf die durch das Registerpaar BC angegebene Spur.   |
| 11 | Setzen auf den durch das Registerpaar BC angegebenen Sektor.  |
| 12 | Stellen der Adresse des DMA-Puffers auf den durch Registerpaar BC angegebenen Wert.   |
| 13 | Lesen der Diskette mit den Parametern, die durch die Funktionen 9, 10, 11 und 12 festgelegt wurden. Bei erfolgreicher Ausführung der Funktion wird im Register A der Wert 00H zurückgegeben, sonst ein Wert ungleich 0.   |
| 14 | Schreiben auf die Diskette mit den Parametern, die durch die Funktionen 9, 10, 11 und 12 festgelegt wurden. Bei erfolgreicher Ausführung der Funktion wird im Register A 00H zurückgegeben, sonst ein Wert ungleich 0.  |
| 15 | Abfrage des Zustandes des logischen Gerätes LIST: Ist das Zeichen bereit zur Eingabe, wird in Register A der Wert 0FFH zurückgegeben, sonst 00H.  |

- 16 Wandelt die logische Nummer des Sektors, die im Registerpaar BC übergeben wurde, in die physische Nummer um. Die Adresse der Umwandlungstabelle wird im Registerpaar DE übergeben.

### **Funktion 108: Abfragen/Stellen des Rückkehrcodes**

Eingangsparameter:

Register C: 6CH

Registerpaar DE: Rückkehrcode oder FFFFH

Ausgangsparameter:

Registerpaar HL: Rückkehrcode

Funktion 108 ermöglicht dem Programm, vor dem Abschluß einen Rückkehrcode zu stellen. Der Rückkehrcode wird vor der bedingten Ausführung des Kommandos (Kennzeichen ':' im Kommandosatz) überprüft, und er kann auch von einem Programm, das mit Hilfe der Funktion 47 von einem anderen Programm aufgerufen wurde, überprüft werden. Das gewährleistet das Abarbeiten der Programme nur im Falle des erfolgreichen Abschlusses des vorhergehenden Programms.

Für das Abfragen des Rückkehrcodes übergibt das aufrufende Programm im Registerpaar DE den Wert 0FFFFH und für das Stellen den Rückkehrcode selbst. Die Werte des Rückkehrcodes werden unten angeführt.

Code	Bedeutung
0000-FEFF	erfolgreicher Abschluß
FF00-FFFE	kein erfolgreicher Abschluß
FF80-FFFC	Reservecodes, im Bereich "kein erfolgreicher Abschluß"
FFFD	Abschluß wegen eines Fehlers von BDOS
FFFE	Abschluß bei Eingabe von CTRL-C

### **Funktion 110: Abfragen/Stellen des Begrenzungszeichens einer Folge**

Eingangsparameter:

Register C: 6EH

Registerpaar DE: Begrenzer oder FFFFH

Ausgangsparameter:

Register A: Begrenzer

Funktion 110 gewährleistet das Abfragen oder das Stellen des Begrenzers einer Zeichenkette für die Funktion 9. Wenn im Registerpaar DE der Wert 0FFFFH übergeben wird, dann wird im Register A der aktuelle Wert des Begrenzers zurückgegeben, in allen anderen Fällen wird der Begrenzer durch das Zeichen im Register E ersetzt. Beim Warmstart wird das Zeichen "\$" als Begrenzer festgelegt.

### **Funktion 111: Ausgabe eines Puffers auf die Konsole**

---

Eingangsparameter:

Register C: 6FH  
 Registerpaar DE: CCB-Adresse

Funktion 111 gibt auf das logische Gerät CONOUT: den Inhalt eines Puffers aus, der durch den im Registerpaar DE angegebenen Zeichensteuerblock (CCB) bestimmt wird. Nachstehend wird das Format des Zeichensteuerblocks beschrieben:

Byte 0/1 : Anfangsadresse des Puffers  
 Byte 2/3 : Länge des Puffers

### **Funktion 112: Ausgabe eines Puffers auf den Drucker**

---

Eingangsparameter:

Register C: 70H  
 Registerpaar DE: CCB-Adresse

Funktion 112 gibt auf dem logischen Drucker den Inhalt eines Puffers aus, der durch den im Registerpaar DE angegebenen Zeichensteuerblock (CCB) bestimmt wird. Nachstehend wird das Format des Zeichensteuerblocks beschrieben:

Byte 0/1 : Anfangsadresse des Puffers  
 Byte 2/3 : Länge des Puffers

### **Funktion 152: Vorbereiten des Dateisteuerblocks**

---

Eingangsparameter:

Register C: 98H  
 Registerpaar DE: Adresse des Blocks PFCB

Ausgangsparameter:

Register A: Rückkehrcode

Funktion 152 gewährleistet die Vorbereitung eines Dateisteuerblocks aus dem Namen der Datei. Das aufrufende Programm übergibt im Registerpaar DE die Adresse eines Parameterblocks, der folgendes Format besitzt:

```
PFCB: DEFW STRING      ;Adresse des Dateinamens
      DEFW AFCB        ;Adresse des vorzubereitenden
                        ;Dateisteuerblocks
```

Die Dateibezeichnung muß folgendermaßen angegeben werden:

[D:]Dateiname[.Dateityp] ,

wobei die Felder in den eckigen Klammern nicht obligatorisch sind.

Die Länge der Folge, die den Namen enthält, darf 128 Bytes nicht überschreiten. Funktion 152 untersucht den angegebenen Dateinamen und bereitet den Dateisteuerblock vor. Leer- und Tabulatorzeichen vor dem Namen werden übergangen. Als Begrenzer für den Dateinamen dient eines der folgenden Zeichen:

Zeichen	Hexadezimalcode
-----	-----
Null	00H
CR	0DH
Tabulator	09H
Leerzeichen	20H
:	3AH
;	3BH
=	3DH
^	5FH
.	2EH
[	5BH
<<	3CH
>>	3EH
,	2CH

Wenn im angegebenen Namen Steuerzeichen mit dem Code von 0 bis 20H auftreten, die nicht in der Tabelle angegeben sind, gibt die Funktion 152 im Registerpaar HL den Code 0FFFFH zurück. Bei erfolgreicher Vorbereitung des Namens überprüft die Funktion das nächste Zeichen. Ist das nächste Zeichen 0 oder << Wagenrücklauf >> (0DH), wird im Registerpaar HL der Wert 0 zurückgegeben. Wenn das nächste Zeichen eines der Begrenzer ist, wird im Registerpaar HL die Adresse des Begrenzers zurückgegeben, sonst die Adresse des ersten folgenden Leer- oder Tabulatorzeichens.

## 6. BIOS-Funktionen

### 6.1. Beschreibung der BIOS-Schnittstelle

Das BIOS ist der von der Hardware abhängige Modul des Betriebssystems MicroDOS. Es beinhaltet die für die spezielle Hardware notwendigen Ein-/Ausgaberoutinen. Damit bildet es die Schnittstelle zwischen der Hardware und dem hardwareunabhängigen Teil des Betriebssystems bzw. dem Anwenderprogramm. Die im BIOS enthaltenen Ein-/Ausgaberoutinen können in drei Gruppen zusammenfaßt werden:

1. Systeminitialisierung
2. Zeichenein- und -ausgabe
3. Disketten ein- und -ausgabeoperationen

Die Routinen erreicht man über einen sogenannten "Sprungvektor". Im MicroDOS ist der Sprungvektor selbst Bestandteil des BDOS. Der Sprungvektor stellt eine zusammenhängende Folge von Sprungbefehlen dar. Nachfolgend sind die Routinen der oben genannten Gruppen angegeben, zu denen je ein Sprungbefehl im Sprungvektor enthalten ist:

Systeminitialisierung:	Kaltstartroutine Warmstartroutine
Zeichenein/ausgabeoperationen:	Status CONSOLE-Gerät Eingabe von CONSOLE-Gerät Ausgabe auf CONSOLE-Gerät Ausgabe auf LIST-Gerät Status LIST-Gerät Ausgabe auf PUNCH-Gerät Eingabe von READER-Gerät
Diskettenein/ausgabeoperationen:	Positionieren Spur Null Laufwerk auswählen Spur auswählen Transformation Sektornummer Sektor auswählen Datenpufferadresse setzen Selektierten Sektor lesen Schreiben selektierten Sektor

### 6.2. Initialisierung

Es gibt im BIOS zwei Routinen zur Initialisierung des Systems, den Kaltstart und den Warmstart.

Die Kaltstartroutine BOOT wird nur nach dem Umladen oder der Neuinstallation von MicroDOS im Speicher aktiviert. Sie führt eine grundlegende Systeminitialisierung sowohl des Betriebssystems als auch der Hardware durch und gibt einen System-Kaltstarttext auf den Bildschirm aus. Wenn eine INITIAL.SUB Datei auf dem Systemlaufwerk vorhanden ist, so wird diese gestartet. Die Initialisierung schließt mit der Übergabe der Steuerung an den CCP ab.

Die Warmstartroutine WBOOT wird immer dann aktiviert, wenn ein Nutzerprogramm zur Adresse 0000H verzweigt. Nach der Initialisierung der Systemparameter wird der CCP aufgerufen.

### 6.3. Logische Ein-/Ausgabekanäle

MicroDOS unterstützt vier logische Kanäle:

- einen Ein-/Ausgabekanal CON:
- einen Eingabekanal RDR:
- zwei Ausgabekanäle LST: und PUN:

Über die im Abschn. 5.3. beschriebenen BDOS-Funktionen sind diese Kanäle ansprechbar.

Die Unterprogramme, höhere Programmiersprachen und verschiedene Kommandos (z.B. PIP ,STAT,..., siehe dazu auch /8/) greifen über die BDOS-Funktionen auf die E/A-Kanäle zu.

Die Schlüsselbegriffe CON:, LST:, PUN: und RDR: stellen die Kanalbezeichnungen dar, wie sie u.a. in den Programmen PIP und STAT Verwendung finden. Die vier logischen E/A-Kanäle haben folgende Eigenschaften:

- CON: (CONSOLE)  
Dieser logische Kanal kann sowohl zur Ein- als auch zur Ausgabe von Zeichen genutzt werden.  
Eingaberoutinen:  
CONIN für zeichenweise Eingabe  
CONST zur Abfrage, ob ein Zeichen verfügbar ist  
Ausgaberoutine:  
CONOUT für zeichenweise Ausgabe
- LST: (LIST)  
Der logische Kanal dient zur Ausgabe einzelner Zeichen. Das LIST-Gerät ist normalerweise ein Drucker.  
Eingaberoutine:  
LISTST für Statusabfrage des LIST-Gerätes  
Ausgaberoutine:  
LIST für zeichenweise Ausgabe
- PUN: (PUNCH)  
Der logische Kanal dient zur zeichenweise Ausgabe.  
Ausgaberoutine:  
PUNCH für zeichenweise Ausgabe
- RDR: (READER)  
Der logische Kanal dient zur zeichenweisen Eingabe.  
Eingaberoutine:  
READER für zeichenweise Eingabe

Jedem logischen E/A-Kanal kann genau einem von vier möglichen logischen Subkanälen zugeordnet werden. Diese Zuordnung wird durch den Inhalt des I/O-Bytes auf Adresse 0003H bestimmt. Die Belegung des I/O-Bytes kann z.B. mittels des transienten Programmes STAT geändert werden.

Es gelten folgende Zuordnungen zwischen I/O-Byte, Subkanälen und physischen Gerätetreibern:

log.E/A-Kanal	Subkanal	Bits im I/O-Byte								physischer Treiber
		7	6	5	4	3	2	1	0	
CON:	TTY:							0	0	StandardbildschirmAusgabe
	CRT:							0	1	StandardbildschirmAusgabe
	BAT:							1	0	Standarddruckertreiber
	UC1:							1	1	Standarddruckertreiber
(Eingaben immer über Tastatur)										
RDR:	TTY:							0	0	Konsoleneingabe
	PTR:							0	1	(RET)
	UR1:							1	0	(RET)
	UR2:							1	1	(RET)
PUN:	TTY:							0	0	StandardbildschirmAusgabe
	PTP:							0	1	Standarddruckerausgabe
	UP1:							1	0	(RET)
	UP2:							1	1	(RET)
LST:	TTY:							0	0	Standarddruckerausgabe
	CRT:							0	1	StandardbildschirmAusgabe
	LPT:							1	0	Standarddruckerausgabe
	UL1:							1	1	(RET)

*Tabelle 7:*  
Zuordnung zwischen I/O-Byte, Subkanälen und physischen Gerätetreibern

Die physischen Gerätetreiber haben dabei folgende Bedeutung:

StandardbildschirmAusgabe: Ausgabe im Format 80\*24 Zeichen mit ESCape-Steuerung

Standarddruckerausgabe: Ausgabe über CENTRONICS (RET):  
Eingabe kehrt mit Kennzeichen ^Z zurück. Ausgabe kehrt sofort zurück.

Bei der Benutzung der Tastatur als READER muß die Eingabe mit einem ^Z abgeschlossen werden.

#### 6.4. Liste der BIOS-Funktionen

Im Betriebssystem MicroDOS hat der BIOS-Sprungvektor nach folgend beschriebenen Aufbau. Die symbolischen Sprungadressen dienen nur zum besseren Verständnis der nachfolgenden Beschreibungen.

Sprungnummer	Befehl	Funktion
0	JMP BOOT	;Kaltstartroutine
1	JMP WBOOT	;Warmstartroutine
2	JMP CONST	;CONSOLE-Status abfragen
3	JMP CONIN	;CONSOLE-Eingabe
4	JMP CONOUT	;CONSOLE-Ausgabe
5	JMP LIST	;LIST-Ausgabe
6	JMP PUNCH	;PUNCH-Ausgabe
7	JMP READER	;READER-Eingabe
8	JMP HOME	;Spur Null einstellen
9	JMP SELDSK	;Laufwerk auswählen
10	JMP SETTRK	;Spur auswählen
11	JMP SETSEC	;Sektor auswählen
12	JMP SETDMA	;Datenpufferadresse setzen
13	JMP READ	;Selektierten Sektor lesen
14	JMP WRITE	;Selektierten Sektor schreiben
15	JMP LISTST	;LIST-Status abfragen
16	JMP SECTRAN	;Umrechnen Sektornummer

Die angegebenen Routinen werden als Unterprogramme aufgerufen, enden also mit einem Rücksprung (mit Ausnahme der Warm- und Kaltstartroutine, für die eigene Regeln gelten). Dabei werden eventuell benötigte Parameter in folgenden Prozessorregistern übergeben:

- an das BIOS: 8-Bit-Werte in Register C,  
16-Bit-Werte im Registerpaar BC,  
(zweiter 16-Bit-Wert im Registerpaar DE);
- vom BIOS: 8-Bit-Werte in Register A,  
16-Bit-Werte im Registerpaar HL.

Ein Programm kann, neben dem Aufruf über die BDOS-Funktion 50, die BIOS-Routinen auch unmittelbar nutzen. Der Ausgangspunkt dazu ist der Sprung auf Adresse 0. Hier befindet sich ein Sprung zur Warmstartroutine (zweite Eintragung im Sprungvektor). Aus der Zieladresse dieses Sprungbefehls und der Nummer der benötigten BIOS-Routine läßt sich leicht die Adresse berechnen, die das Programm gegebenenfalls aufrufen muß:

$(\langle\langle\text{Sprungnummer}\rangle\rangle - 1) * 3 + \langle\langle\text{Zieladresse des Sprungs auf Adresse 0}\rangle\rangle$

Wenn in einem Programm beispielsweise der Zustand des CONSOLE-Gerätes (Sprungnummer = 2) gebraucht wird, dann kann das mit dem folgenden Unterprogramm geschehen:

```
LD  DE, 3      ; (Nummer der BIOS-Routine)-1)*3
LD  HL, (1)    ; Adresse Warmstart
ADD HL, DE     ; Adresse der BIOS-Routine
JP  (HL)       ; Sprung zur BIOS-Routine
```

Nachfolgend werden die zu den BIOS-Eintrittspunkten gehörenden Routinen beschrieben.

**KBOOT** Kaltstart  
 Aufrufparameter: -  
 Rückkehrparameter: -  
 Die Routine erhält die Steuerung vom Urlader und ist verantwortlich für die grundlegende Systeminitialisierung.

**WBOOT** Warmstart  
 Aufrufparameter: -  
 Rückkehrparameter: -  
 Die Systemparameter werden wie folgt initialisiert:

Speicheradresse	Inhalt
-----	-----
0, 1, 2	Sprung zu WBOOT für Warmstart
5, 6, 7	Sprung zum BDOS-Anfang
	Zieladresse des Sprungs gibt das Ende des TPA an

Der Inhalt der Speicheradressen 3 und 4 bleibt unverändert. Nach vollständiger Initialisierung verzweigt die Routine für den Systemneustart zum CCP.

**CONST** Abfrage Status Kanal CON:  
 Aufrufparameter: -  
 Rückkehrparameter: A - Status

Diese Routine untersucht den Status der Tastatur und liefert im Register: A = 0 falls kein Zeichen bereitsteht  
 A = 0FFH ein Zeichen

**CONIN** Empfangen Zeichen vom Kanal CON:  
 Aufrufparameter: -  
 Rückkehrparameter: A - Empfangenes Zeichen  
 Diese Routine liest das nächste Zeichen von CON: in das Register A. Steht kein Zeichen bereit, wird bis zur Zeicheneingabe gewartet.

**CONOUT** Senden Zeichen auf Kanal CON:  
 Aufrufparameter: C - Sende-Zeichen  
 Rückkehrparameter: -  
 Das in Register C bereitgestellte Zeichen wird auf Kanal CON: ausgegeben.

**LIST** Senden Zeichen auf Kanal LST:  
 Aufrufparameter: C - Sende-Zeichen  
 Rückkehrparameter: -  
 Das in Register C bereitgestellte Zeichen wird auf den Kanal LST: ausgegeben.

**PUNCH** Senden Zeichen auf Kanal PUN:  
 Aufrufparameter: C - Sende-Zeichen  
 Rückkehrparameter: -  
 Die Routine gibt das in Register C bereitgestellte Zeichen auf den Kanal PUN: aus.

**READER** Empfangen Zeichen vom Kanal RDR:  
 Aufrufparameter: -  
 Rückkehrparameter: A - Empfangenes Zeichen

Die Routine liest das nächste Zeichen von Kanal RDR: in Register A.

Die EOF-Bedingung wird durch 1AH (CTRL-Z) geliefert.

**HOME**      **Positionieren Spur 0**

Aufrufparameter:    -

Rückkehrparameter: -

Die Routine positioniert den Kopf des selektierten Laufwerkes auf die Spur 0.

**SELDSK**    **Selektieren Laufwerk**

Aufrufparameter:    C - Laufwerksnummer

Rückkehrparameter: HL - Diskettenparameterkopf DPH

Die Routine wählt das im Register C angegebene Laufwerk für den nächsten Datenzugriff aus. Dabei enthält das Register C eine 0 für Laufwerk A, eine 1 für Laufwerk B, eine 2 für Laufwerk C bzw. eine 3 für Laufwerk D. Bei jedem Aufruf der Routine wird in Hl die Adresse eines Speicherbereiches (DPH siehe Abschn. 7.5.) bereitgestellt, indem die Merkmale der betreffenden Diskette enthalten sind und Platz für Zwischenergebnisse beim Diskettenzugriff vorhanden ist. Dieser Speicherbereich ist als Vorspann für die BDOS-Arbeit vor allem deshalb wichtig, da hierüber die Informationen zur Diskettenstruktur und andere wichtige Daten erreicht werden.

Wird SELDSK für ein nicht vorhandenes Laufwerk aufgerufen, dann wird in HL 0000H zurückgegeben.

Die physische Laufwerksauswahl wird erst bei einem tatsächlichen Datenzugriff (READ oder WRITE) ausgeführt.

**SETTRK**    **Spurpositionierung**

Aufrufparameter:    BC - Spurnummer

Rückkehrparameter: -

Die Routine nimmt die Spurpositionierung entsprechend dem Inhalt des Registerpaars BC für den nächsten Zugriff auf das ausgewählte Laufwerk vor.

Im BIOS wird lediglich die Spurnummer gemerkt und die eigentliche Positionierung bis zu einem Lese- oder Schreibbefehl zurückgestellt. Dadurch lassen sich Verwaltungsaufgaben vereinfachen. Wichtig ist lediglich, daß beim Datenzugriff der Kopf auf der richtigen Spur steht. Die Spuradresse bleibt bis zum erneuten Aufruf der Routine erhalten.

**SETSEC**    **Sektorpositionierung**

Aufrufparameter:    BC - Sektornummer

Rückkehrparameter: -

Entsprechend der im Registerpaar BC eingestellten Sektornummer wird auf dem ausgewählten Laufwerk der Sektor für den nächsten Diskettenzugriff positioniert. Die Sektornummer wird zunächst intern gemerkt und der eigentliche Zugriff bis zu einem Schreib- oder Lesebefehl verschoben.

Die Sektoradresse bleibt erhalten bis zu einem erneuten Aufruf der Routine.

**SETDMA**    **Einstellen Pufferadresse**

Aufrufparameter:    BC - Pufferadresse

Rückkehrparameter: -

Die Routine erhält in BC die Anfangsadresse eines 128 Bytes umfassenden Speicherbereichs, der als Datenpuffer für alle nachfolgenden Schreib- und Leseoperationen dient.

Vom Betriebssystem wird standardmäßig ein Datenpuffer der Länge 128 Bytes ab Adresse 80H angelegt.

**READ** Lesen Sektor  
 Aufrufparameter: -  
 Rückkehrparameter: A - Fehlercode

Unter der Voraussetzung, daß Laufwerk, Spur, Sektor und Adresse des Datenpuffers festgelegt wurden, versucht die Routine, den durch diese Parameter bestimmten Sektor zu lesen.

Im Register A wird folgender Fehlercode zurückgegeben:

```
0 fehlerfreies Lesen
1 Lesen nicht möglich
```

Ist der Wert im Register A gleich 0, dann wird vom Betriebssystem der Diskettenzugriff als erfolgreich abgeschlossen. Tritt jedoch ein Fehler auf, dann versucht das BIOS durch mehrmaliges Wiederholen festzustellen, ob der Fehler behebbar ist.

**WRITE** Schreiben Sektor  
 Aufrufparameter: -  
 Rückkehrparameter: A - Fehlercode

Die Routine schreibt die Daten aus dem vorher festgelegten Datenpuffer auf das zuvor ausgewählte Laufwerk, Spur und Sektor. Die in Register A zurückgegebenen Fehlerbedingungen sind analog der READ-Routine.

**LISTST** Abfrage des Status von Kanal LST:  
 Aufrufparameter: -  
 Rückkehrparameter: A - Status Kanal LST:

Die Routine übermittelt dem Nutzer eine Statusinformation über den Kanal LST:

```
Register A = OFFH, wenn der Kanal LST: bereit ist, ein
                  Zeichen zu übernehmen.
Register A = 0     Kanal LST: nicht bereit
```

**SECTRAN** Umwandeln der Sektornummer  
 Aufrufparameter: BC - umzuwandelnde Sektornummer  
 (00,...)  
 DE - Adresse Umwandlungstabelle  
 Rückkehrparameter: HL - umgewandelte Sektornummer

Die Routine erhält die logische Sektornummer in Registerpaar BC und die Adresse einer Umwandlungstabelle in Registerpaar DE. Die logische Sektornummer (relativ zu Null angegeben) wird als ein Index in der Umwandlungstabelle verwendet. Die durch die Umwandlung bestimmte physische Sektornummer wird im Registerpaar HL zurückgegeben.

Hier wird noch nichts darüber ausgesagt, ob überhaupt ein und, wenn ja, welcher Sektor beim nächsten Datenzugriff tatsächlich gelesen oder geschrieben wird. Es wird lediglich die Sektornummer bestimmt.

## 6.5. Verwaltung der Diskettenlaufwerke

Auf Grund der Vielfalt von Diskettenlaufwerken und Diskettenformaten schließt das BIOS die Möglichkeit der Anpassung an verschiedene Laufwerke und Diskettenformate ein. Deshalb enthält das BIOS Tabellen, die dem Nutzer die Disketten- und Laufwerkseigenschaften mitteilen.

### Diskettenparameterkopf DPH

Jedem Laufwerk ist ein 16 Bytes großer Diskettenparameterkopf (DPH-Disk Parameter Header) zugeordnet, der Informationen über das Diskettenlauf-

werk enthält und Arbeitsbereiche für bestimmte BDOS-Operationen einschließt. Durch die BIOS-Routine SELDSK wird das Laufwerk ausgewählt und außerdem die Adresse des zugehörigen DPH im Registerpaar HL zurückgegeben.

Ein DPH hat folgenden Aufbau:

Byte	Name	Bedeutung
0, 1	<b>XLT</b>	Adresse der Übersetzungstabelle für die Sektornummer Ist die Adresse gleich 0, dann stimmen logische und physische Sektornummer überein.
2-7		Arbeitsbereich für BDOS reserviert
8, 9	<b>DIRBUF</b>	Adresse eines 128-Byte-Verzeichnispuffers Alle DPH enthalten die gleiche Adresse.
10, 11	<b>DPB</b>	Adresse des Diskettenparameterblockes (DPB) Jedes Laufwerk hat einen eigenen DPB.
12, 13	<b>CSV</b>	Adresse eines Puffers, der für das Speichern eines Prüfsummenvektors zur Prüfung auf Diskettenwechsel erforderlich ist. Jedes Laufwerk hat einen eigenen Puffer.
14, 15	<b>ALV</b>	Adresse eines Vektors, der die Diskettenbelegung widerspiegelt. Bit n des Vektors gleich 1 bedeutet, daß der Block n der Diskette von einer Datei belegt ist. Bit n gleich 0 bedeutet, daß der Block unbelegt ist. Die ersten Blöcke, und damit die ersten Bits, sind durch das Verzeichnis belegt. Jedes Laufwerk hat einen eigenen Vektor.

Die für die verschiedenen Laufwerke zuständigen DPH stehen lückenlos hintereinander.

Die im DPH erfaßten Daten und Speicherbereiche werden für jedes Laufwerk getrennt bereitgestellt.

Eine Ausnahme ist der 128-Byte-Puffer für die Verzeichnisauswertung. Er kann nur einmal im System vorhanden sein, da das BDOS immer nur ein Laufwerk zur Zeit erfassen kann und bei jeder Laufwerksumschaltung das Verzeichnis neu abfragt.

### Diskettenparameterblock DPB

Der Diskettenparameterblock (DPB) für jedes Laufwerk ist wesentlich umfangreicher. In diesem Block sind alle Informationen zusammengefaßt, die zur Verwaltung der betreffenden Diskette notwendig sind.

Dies umfaßt:

- Informationen zur Speicherkapazität und
- Informationen zur Speicherorganisation.

Der DPB enthält unter anderem:

- Angaben zur Anzahl von Sektoren pro Spur,
- Angaben zur Anzahl von Sektoren pro Block,
- Angaben zur Größe und Lage des Verzeichnisses sowie dazu, ob die Verzeichniseinträge bei jedem Zugriff auf Diskettenwechsel überprüft werden sollten und schließlich
- eine Angabe zur Anzahl der auf der betreffenden Diskette für das Betriebssystem reservierten Spuren.

Diese Informationen sind im DPB wie folgt festgehalten:

Byte	Name	Bedeutung																		
0, 1	<b>SPT</b>	Sektoren pro Spur																		
2	<b>BSH</b>	Blockverschiebungsfaktor Darin ist die Blockgröße verschlüsselt als: $\text{Log}_2(\text{Blockgröße}/128)$ Dieser Wert stellt ein Maß für die Anzahl der Sektoren pro Block dar.																		
3	<b>BLM</b>	Blockmaske widerspiegelt ebenfalls die Blockgröße Für die Blockmaske gilt: $2^{\text{BSH}} - 1$ Zwischen Blockgröße, Blockverschiebungsfaktor und Blockmaske bestehen folgende feste Beziehungen																		
		<table border="1"> <thead> <tr> <th>Blockgröße</th> <th>BSH</th> <th>BLM</th> </tr> </thead> <tbody> <tr> <td>1024</td> <td>3</td> <td>7</td> </tr> <tr> <td>2048</td> <td>4</td> <td>15</td> </tr> <tr> <td>4096</td> <td>5</td> <td>31</td> </tr> <tr> <td>8192</td> <td>6</td> <td>63</td> </tr> <tr> <td>16384</td> <td>7</td> <td>127</td> </tr> </tbody> </table>	Blockgröße	BSH	BLM	1024	3	7	2048	4	15	4096	5	31	8192	6	63	16384	7	127
Blockgröße	BSH	BLM																		
1024	3	7																		
2048	4	15																		
4096	5	31																		
8192	6	63																		
16384	7	127																		
4	<b>EXM</b>	Extentmaske Anzahl der Blöcke pro Diskette Ihre Größe hängt von der Organisation des Verzeichniseintrages ab. Dieser enthält als wesentlichsten Teil für die Speicherverwaltung die Nummern der jeweils belegten Blöcke: 16 Einträge zu je 1 Byte bei weniger als 256 Blöcken pro Diskette oder 8 Einträge zu je 2 Bytes bei mehr als 255 Blöcken pro Diskette.																		

Im einzelnen bestehen die Beziehungen:

Blockgröße	Extentmaske für	
	mehr als 255 Blöcke	weniger als 256 Blöcke
1024	-	0
2048	0	1
4096	1	3
8192	3	7
16384	7	15

5, 6	<b>DSM</b>	Anzahl der Blöcke pro Diskette minus 1 (einschließlich des Verzeichnisses, aber ohne Systemspuren)
7, 8	<b>DRM</b>	Anzahl der Verzeichniseintragen minus 1 Die Größe einer Verzeichniseintragung beträgt 32 Byte.
9, 10	<b>AL0,AL1</b>	16-Bit-Vektor, in dem die vom Verzeichnis belegten Blöcke vermerkt sind. Dieser Vektor wird beim ersten Laufwerkszugriff an den Anfang der Belegungstabelle kopiert und dient so zur Reservierung der Verzeichnisblöcke. Er ist aus diesem Grund umgekehrt als sonst üblich organisiert: Die Zählung beginnt mit dem höchstwertigen Bit, so

hat der Vektor beispielsweise bei vier Verzeichnisblöcken den Wert F000H.

Byte	Name	Bedeutung
11, 12	<b>CKS</b>	Größe des Verzeichnis-Prüfvektors (Anzahl zu prüfender Verzeichniseintragen dividiert durch 4)
13, 14	<b>OFF</b>	Anzahl der Systemspuren
14	<b>PSH</b>	physischer Sektorverschiebungsfaktor, darin ist die physische Sektorengröße verschlüsselt als: $\text{Log}_2 (<\text{Sektorgröße}>/128)$ (außer Laufwerk A)
15	<b>PHM</b>	physische Sektormaske widerspiegelt ebenfalls die physische Sektorgröße $2^{\text{PSH}} - 1$ (außer Laufwerk A)

Unmittelbar an den DPH schließt sich an:

#### Diskettendefinitionsblock DDB

Die beiden folgenden Tabellen sind nur für die Laufwerke B bis H vorhanden. Der DDB beschreibt physische Kennwerte der Diskette:

Byte	Name	Bedeutung
1	<b>EOT</b>	Nr. des letzten Sektors der Spur
2	<b>GAP</b>	GAP3 Lücke
3	<b>NTR</b>	Anzahl der Spuren

#### Laufwerkparameterblock DRPB

Byte	Name	Bedeutung
1	<b>PUN</b>	physische Gerätenummer (0...3)
2	<b>DTYP</b>	Drivetyp
3	<b>PSO</b>	physischer Sektoroffset
4	<b>TSS</b>	Schrittzeit
5	<b>HLT</b>	Kopfladezeit
6	<b>CUR</b>	erste Spur mit Schreibstrombegrenzung

#### Laufwerksteuerung

Die Laufwerksteuerung umfaßt drei Schritte, die zum Adressieren eines Sektors auf der Diskette notwendig sind. Mit Sektor wird im weiteren ein 128 Byte großer Aufzeichnungsabschnitt auf der Diskette bezeichnet.

1. Auswahl des gewünschten Laufwerkes (mittels der Routine SELDSK)
2. Schreib-Lese-Kopf auf die Spur setzen, in der sich die Information befindet. (mittels der Routine SETTRK)
3. Das Einstellen der Sektornummer erfolgt in zwei Schritten. Im ersten Schritt erfolgt über die Routine SECTAN die Umwandlung der logischen Sektornummer in die physische Sektornummer. Diese Umwandlung ermöglicht eine Diskettenorganisation, die einen zeitoptimalen Zugriff auf die gewünschte Information gewährleistet. Im zweiten Schritt wird über die Routine SETSEC die Adressierung des physischen Sektors vorgenommen.

## Datenverkehr

Der Datenverkehr umfaßt neben dem Lesen und Schreiben von Informationen weiterhin die Festlegung der Adresse des Datenpuffers, jenes 128-Byte-Bereiches im Speicher, der die Daten von der Diskette übernimmt bzw. von dem sie kommen. Liegt der Ort der Aufzeichnung auf der Diskette fest, und ist der Ort des Datenpuffers im Speicher bestimmt, dann können die Daten gelesen oder auf die Diskette geschrieben werden. Dazu bietet das BIOS die Routinen:

### - **SETDMA**

Festlegen des Datenpuffers als Ziel oder Herkunft der Daten.

### - **READ**

Lesen eines 128-Byte-Sektors von der Diskette und Übertragen in den Datenpuffer.

### - **WRITE**

Schreiben der im Datenpuffer vorliegenden Information in den adressierten 128-Byte-Sektor.

## RAM-Floppy (auch RAM-Disk)

Der Zugriff auf das RAM-Floppy geschieht nutzerseitig wie auf jedes andere Laufwerk. Es besitzt eine Spurgröße von 16 KByte und zwei Spuren. Das RAM-Floppy wird im RAM des 'KC compact' verwaltet. Das Verzeichnis befindet sich auf Spur 0, ab Sektor 1. Pro Eintrag stehen 32 Bytes zur Verfügung. Ein Sektor kann vier Verzeichniseinträge enthalten. Die Einträge beginnen auf den Adressen 00H, 20H, 40H... Die folgende Tabelle enthält die

Bedeutung der einzelnen Bytes in einem Verzeichniseintrag.

Byte	Bedeutung
0	0E5H -> Datei gelöscht 00H bis 0FH User-Bereich (von 0 bis 15)
1-8	Dateiname (ASCII-Zeichen) zusätzlich kann Bit 7 gesetzt sein. Bedeutung des Bits 7 bei einigen Dienstprogrammen für Byte 1: Kennzeichnung der Datei als Quelle beim Kopieren (optional) Beim Auflisten des Verzeichnisses erscheint zwischen Name und Typ anstelle des Punktes ein >-Zeichen. für Byte 2: Kennzeichnung der Datei als Ziel beim Kopieren (optional) Beim Auflisten des Verzeichnisses erscheint zwischen Name und Typ anstelle des Punktes ein <-Zeichen (auch, wenn Bit 1 gesetzt ist). Byte 3 - 8: ohne Bedeutung, können beliebig genutzt werden
9-0BH	Dateityp (ASCII-Zeichen) zusätzlich kann Bit 7 gesetzt sein. Bedeutung des Bits 7 für Byte 9 : Schreibschutz für Byte 0AH : Systemfile für Byte 0BH : ohne Bedeutung, kann beliebig genutzt werden

- 0CH Extentnummer; Belegt eine Datei mehr als acht Blöcke, sind mehrere Verzeichniseinträge nötig. Diese werden bei 0 beginnend durch die Extentnummer festgelegt.
- 0DH-0EH 00H
- 0FH Anzahl der Sektoren, die durch diesen Verzeichnis-Eintrag belegt werden (maximal 80H, je 10H pro Block)  
Der letzte Block muß nicht voll belegt sein.
- 10H-1FH Blockadressen; In der angegebenen Reihenfolge ist die Datei auf dem RAM-Floppy verteilt. Je nach Kapazität werden ein oder zwei Bytes je Block belegt.

Der Verzeichnisaufbau ist auf den anderen Laufwerken prinzipiell gleich, wobei sich jedoch das Verzeichnis installationsabhängig meist auf Spur 2 befindet. Die Anzahl der Verzeichniseinträge und deren maximale Anzahl sind vom Diskettenformat abhängig (siehe DPB).

### Die Schnittstellen der MicroDOS-Implementation auf dem 'KC compact' zu den Ressourcen des Rechners:

Bei der Implementation des Betriebssystems MicroDOS auf dem 'KC compact' ist es gelungen, unterschiedliche, sich z.T. widersprechende Forderungen zu erfüllen. Die Routinen des Betriebssystems des 'KC compact' sollten auch im MicroDOS verfügbar sein, besonders die komfortable Interruptbehandlung und Eventgenerierung. Dennoch sollten in der MicroDOS-Betriebsart alle Register der CPU zur freien Verfügung des Anwenders stehen.

Die Behandlung von Interrupts wurde folgendermaßen gelöst. In der MicroDOS-Betriebsart wird der RAM der Diskettenerweiterung benutzt. Der RAM des Grundgerätes wird als RAM-Floppy, für das BIOS und das System des Grundgerätes, als Bildwiederholtspeicher und als Anwenderspeicher benutzt.

In der MicroDOS-Betriebsart arbeitet die CPU im Interruptmode 2. Da der Datenbus beim Einlesen des Lowteiles des Interruptvektors immer 0FFH liefert, wurde die Adresse der Interruptbehandlungsroutine auf 0FEFFH abgelegt. Diese Adresse ist fest und wird sich auch bei eventuell nötigen Revisionen des MicroDOS nicht ändern. Die folgenden Adressen sind ebenfalls reserviert. Die Beschreibung ihrer Funktion ermöglicht es dem Anwender, alle Ressourcen des Rechners zu nutzen.

FEFE LAST:JP IHAND ;Adr Interruptroutine

Durch den Aufruf von 0FEFEH kann ein Interrupt simuliert werden.

FF01 JP HANDLER

Durch den Aufruf von 0FF01H, gefolgt von der Adresse (DEFW ADR), kann eine Routine im RAM des 'KC compact' ausgeführt werden. Die Erstregister der CPU können zur Übergabe von Parametern genutzt werden.

FF04 DEFW BIOSEND

Dieses Wort beschreibt den ersten freien Speicherplatz im RAM der Floppy-Erweiterung. Bis zu 0FFFFH steht Platz zur Erweiterung des BIOS durch den Anwender zur Verfügung. Bei Nutzung dieses freien Speichers sollte das Wort auf 0FF04H durch den Anwender aktualisiert werden.

FF06 DEFW LBIOSEND

Dieses Wort hat eine ähnliche Funktion, wie das auf 0FF04H. Es beschreibt aber den ersten freien Speicherplatz des RAM im Grundgerät. Dieser RAM kann durch die Ausgabe einer ESCape-Folge beschrieben

werden. Im RAM des Grundgerätes stehen dem Anwender alle Routinen des Betriebssystems voll zur Verfügung.

Die folgende Tabelle enthält die Adressen der Routinen, die über das I/O-Byte angewählt werden. Hier können vom Anwender eigene Routinen eingebracht werden, indem die Adresse an der passenden Stelle in dieser Tabelle eingetragen wird.

Je zwei Bits des I/O-Bytes sind für die Auswahl eines Gerätes zuständig. Diese Geräte werden durch die BIOS-Routinen CONSTAT, CONIN, CONOUT, LSTOUT, PUNOUT, RDRIN und LSTSTAT bedient. Die folgende Tabelle enthält die Zuordnung der Geräte zu diesen BIOS-Routinen. Sie sind entsprechend der Zählung der zwei zuständigen Bits im I/O-Byte in der Tabelle geordnet. Das I/O-Byte wird mit 00H initialisiert.

```

;
;I/O Byte Tab f. Behandlungsroutinen
;
;Konsolenstatus abfragen (Bit 0,1 im I/O-Byte)

FF08  CSTAT:      DEFW  CHRST      ;Status Tastatureingaben
FF0A                      DEFW  CHRST
FF0C                      DEFW  CHRST
FF0E                      DEFW  CHRST
;
;Druckerstatus abfragen (Bit 6,7 im I/O-Byte)
FF10  LSTAT:      DEFW  DRUSTA     ;Status CENTRONICS
FF12                      DEFW  DUMMYST ;Dummy-Routine (immer bereit)
FF14                      DEFW  DRUSTA
FF16                      DEFW  DUMMYST
;
;Konsoleneingabe (Bit 0,1 im I/O-Byte)
FF18  CIN:        DEFW  CHRIN      ;nächstes Byte von Tastatur
FF1A                      DEFW  CHRIN ;holen
FF1C                      DEFW  CHRIN
FF1E                      DEFW  CHRIN
;
;Readereingabe (Bit 2,3 im I/O-Byte)
FF20  RIN:        DEFW  CHRIN      ;Tastatureingabe
FF22                      DEFW  DUMMYIN ;Dummyeingabe, liefert
FF24                      DEFW  DUMMYIN ;immer 1AH
FF26                      DEFW  DUMMYIN
;
;Konsolenausgabe (Bit 0,1 im I/O-Byte)
FF28  COUT:       DEFW  CHROUT     ;Ausgabe auf Bildschirm
FF2A                      DEFW  CHROUT
FF2C                      DEFW  DRUOUT ;Ausgabe auf Drucker
FF2E                      DEFW  DRUOUT
;
;Ausgabe auf List-Gerät (Bit 6,7 im I/O-Byte)
FF30  LOUT:       DEFW  DRUOUT     ;Ausgabe auf Drucker
FF32                      DEFW  CHROUT ;Ausgabe auf Bildschirm
FF34                      DEFW  DRUOUT
FF36                      DEFW  DUMMYOUT ;es geschieht nichts
;
;Ausgabe auf Punch-Gerät (Bit 4,5 im I/O-Byte)
FF38  POUT:       DEFW  CHROUT     ;Ausgabe auf Bildschirm
FF3A                      DEFW  DRUOUT ;Ausgabe auf Drucker
FF3C                      DEFW  DUMMYOUT ;Dummy-Routine (macht nichts)
FF3E                      DEFW  DUMMYOUT
;
FF40  BIOSEND:    NOP

```

## **Teil B**

### **BASDOS-Betriebsart**

#### **1. Systemstart**

Nach dem Einschalten oder dem Rücksetzen des 'KC compact' wird vom Kernel ein ROM-walk durchgeführt. Dabei werden externe ROMs festgestellt und Speicherplatz für ihre RAM-Variablen wird reserviert. Bei angeschlossener 'KC compact'-Floppy-Elektronik ist der BASDOS-ROM mit der ROM-Nummer 7 im System eingebunden. Er legt bei seiner Initialisierung zwei Variablenbereiche im RAM fest, die im folgenden beschrieben werden. Außerdem werden die Vektoren für die Kassettenarbeit gepatcht und einige RSX-Kommandos dem Betriebssystem angemeldet. Sie stellen die softwareseitige Schnittstelle zwischen BASDOS und dem Betriebssystem des 'KC compact' dar und werden ebenfalls im weiteren beschrieben.

## 2. Schnittstellenbeschreibung

### 2.1. Verschiebliche RAM-Variablen

Die verschieblichen RAM-Variablen werden beim Initialisieren des ROMs an das obere Ende des freien RAMs gelegt. Je nach Vorhandensein und Reihenfolge der externen ROMs können sie verschiedene Basisadressen haben. Ein Zugriff auf diese Variablen sollte immer über die Basisadresse erfolgen. Die Basisadresse der RAM-Variablen des BASDOS ist normalerweise 0A700H. Sie ist auf 0BE7DH zu finden. Nachfolgend sind alle RAM-Variablen des BASDOS aufgeführt.

0	angemeld. Drive
1	angemeld. User
2	aktives Drive
3	Zeiger auf DPH (210H 220H) des aktiven Drives
5	Flag, ob OPEN auf angemeldetes Drive aktiv ist
6	Zwischensp. für SP für alle logischen Routinen des BASDOS
-----	
8 - 28	erweiterter FCB für OPENIN
8	Flag für OPENIN
	0FFH kein OPENIN aktiv
	00 OPENIN auf Drive A
	01 OPENIN auf Drive B
9	USER-Nr. für OPENIN
A	Filename
15	Extend-Nummer
16,17	00
18	Anzahl Records in diesem Extend
19-28	Blocknummern
-----	
29-2B	Anzahl der bisher gelesenen Records bei INPUT
2C-4F	erweiterter FCB für OPENOUT
2C	Flag für OPENOUT
	0FFH kein OPENOUT aktiv
	00 OPENOUT auf Drive A
	01 OPENOUT auf Drive B
2D	USER-Nr. für OPENOUT
2E	Filename
39	Extend-Nummer
3A,3B	00
3C	Anzahl Records in diesem Extend
3D-4C	Blocknummern
4D-4F	Anzahl der bisher geschriebenen Records bei OPENOUT
-----	
50-99	Fileheader OPENIN
50	1-Disc in Char
	2-Disc in Direct
51	Zeiger auf Anfang des 2K-OPENIN-Buffers
53	Zeiger auf aktuelles Zeichen im OPENIN-Buffer
55	USER-Nr. des Files
56-64	Filename für den Fileheader, mit Nullen aufgefüllt
65	Blocknummer
66	last Block
67	File Type (INPUT)
68	Data length

---

6A	Data location
6C	first Block
6D	logical length
6F	Entry Adress
71-94	User field, frei für Anwender
95-97	3-Byte Zähler, Anzahl gelesener Zeichen
98-99	2-Byte Checksumme von 55H bis 97H
-----	
9A-E3	Fileheader OPENOUT
9A	1-Disc out Char
	2-Disc out Direct
9B	Zeiger auf Anfang des 2K-OPENOUT-Buffers
9D	Zeiger auf aktuelles Zeichen im OPENOUT-Buffer
9F	USER-Nr. des Files
A0-AE	Filename für den Fileheader, mit Nullen aufgefüllt
AF	Blocknummer
B0	last Block
B1	File Type (OUTPUT)
B2	Data length
B4	Data location
B6	first Block
B7	logical length
B9	Entry Adress
BD	Länge des Datenblocks bei DISC OUT DIRECT
BF	Entry Adress bei DISC OUT DIRECT
C1-DE	User field, frei für Anwender
DF-E1	3-Byte Zähler, Anzahl gelesener Zeichen
E2-E3	2-Byte Checksumme von 9FH bis E1H
-----	
E4-173	temporärer Puffer (Record Buffer und zum Expand.)
174-18A	Puffer für ursprüngliche TAPE-Vektoren
18B-18D	Far adress für gepatchte TAPE-Vektoren (CD30H ROM 7)
190-1A8	Extended DPB Drive A
190,191	SPT (Rec/Track, 24H)
192	BSH (Blockshift, 3)
193	BLM (Blockmasc, 7)
194	EXM (Extend Masc, 0)
195,196	DSM (max. Blocknr., AAH)
197,198	DRM (max. DIR-Eintr.-1, 3FH)
199,19A	AL0,1 (Verzeichnisgröße, binär codiert, C000H)
19B,19C	CKS (16 Einträge Prüfsumme)
19D,19E	OFF (Spuroffset, 2)
19F	FSC (Sektoroffset, 41H)
1A0	PST (physische Sektoren/Track, 9)
1A1	GPS (Gap3 r/w, 2AH)
1A2	GPT (Gap3 format., 52H)
1A3	FLB (fill Byte format., E5H)
1A4	BPS (Logar. Byte/Sect, 2)
1A5	RPS (Rec/Sect, 4)
1A6	aktuelle Track-Nummer
1A7	Flag für read, write und recalibrate
1A8	Flag, ob bei jedem Diskettenzugriff Login erfolgen soll
1A9-1B8	CSA 16 Byte für Checksumme
1B9-1CE	ALTA Allocation Table A

1D0-1E8	extended DPB Drive B
1E9-1F8	CSB 16 Byte für Checksumme
1F9-20E	ALTB Allocation Table B
210-21F	DPH Drive A
	210,211 XLT Skew Faktor Table (nicht genutzt)
	212,213 TRACK BIOS Speicher in DIRNUM
	214,215 SECTOR BIOS Speicher
	216,217 DIRNUM
	218,219 DIRBUF (zeigt auf 230H)
	21A,21B DPB (zeigt auf 190H)
	21C,21D CSV (zeigt auf Speicher für Checksumme, 1A9H)
	21E,21F ALV (zeigt auf Allocationtable, 1B9H)
220-22F	DPH Drive B
230-2AF	DIR REC, 128 Byte Verzeichnisbuffer
2B0-4AF	SECBUF, 512 Byte Sektorpuffer

## 2.2. Die nicht verschieblichen RAM-Variablen des BASDOS

Einige häufig benutzte Variablen des BASDOS sind auf festen Adressen abgelegt. Der Bereich dieser nicht verschieblichen RAM-Variablen liegt oberhalb der Systemvariablen und unterhalb des Stackbereiches des 'KC compact'. Im folgenden werden die Adressen und der Inhalt der nicht verschieblichen RAM-Variablen des BASDOS aufgelistet:

Adresse	Inhalt
BE40H	Adresse Disk Parameter Header Drive A
BE42H	Adresse Disk Parameter Block Drive
BE44H	Wartezeit zum Hochlaufen des Drivemotors in 1/50 s
BE46H	Zeit vom letzten Zugriff bis zum Abschalten des Drivemotors in 1/50 s (Nachlaufzeit)
BE48H	Zeitkonstante für Spurformatieren
BE49H	Zeitkonstante für lange Verzögerungsschleife
BE4BH	Anzahl Bytes beim Auslesen des Interrupt-Status des Floppy-Disk-Controllers
BE4CH	Puffer zum Ablegen der Resultatbytes des Floppy-Disk-Controllers
BE53H	Drive (HS/US) (logisch)
BE54H	Track (logisch)
BE55H	Sector (logisch)
BE56H	Drive (HS/US) (physisch)
BE57H	Track (physisch)
BE58H	Sector (physisch)
BE59H	Anzahl Records/Track
BE5AH	Drive (HS/US) (unallocated)
BE5BH	Track (unallocated)
BE5CH	Sector (unallocated)
	In einem geblockten BIOS, wie im BASDOS, wird nicht jeder Lese- und Schreibzugriff sofort ausgeführt, es werden größere Bereiche in Blöcken verwaltet und mehrere physische Zugriffe zusammengefaßt. In einigen Fällen ist aber ein Zugriff direkt nötig, auf einen Record, der durch SELDSK, SETTRK, SETSEC und SETDMA des BIOS noch nicht vollständig beschrieben (alocated) wurde. Dazu dienen die letzten drei Variablen.
BE5EH	Flag für Sektorlesen/schreiben
BE5FH	Flag für Drivemotor an/aus
BE60H	Adresse des augenblicklichen Recordpuffers (128 Bytes) wird durch BIOS-Routine SETDMA gesetzt.
BE62H	Adresse des Puffers für einen physischen Sektor (512 Bytes)
BE64H	Zwischenspeicher für Stackpointer

	BASDOS benutzt häufig einen eigenen Stack.
BE66H	Anzahl der Leseversuche beim Auftreten eines Fehlers BE67H Ticker Block für Motor on/off Counter
BE6DH	Eventblock
BE74H	gewünschte Spurnummer
BE75H	Merkzelle für Kommando an den Floppy-Disk-Controller
BE76H	Adresse des Puffers für physischen Sektor
BE78H	Flag für Fehlermeldungen enabled/disabled
BE7DH	Anfangsadresse des Bereiches für die verschieblichen RAM-Variablen
BE7FH	Vektor zur Manipulation der unter BASDOS genutzten Betriebssystemvektoren

### 2.3. Die im BASDOS genutzten Betriebssystemvektoren

Die BASDOS-Routinen haben weitgehend die gleichen Schnittstellen, wie die entsprechenden Kassettenroutinen. Unterschiede bestehen, wenn Fehler erkannt wurden. Fehler, die gleichermaßen bei Kassetten- und Diskettenarbeit auftreten können werden durch rückgesetztes Carry- und Zeroflag angezeigt, Fehler, die nur bei Diskettenarbeit auftreten durch rückgesetztes Carry- und gesetztes Zeroflag. Fehler werden durch die Übergabe eines Fehlercodes im Register A der CPU genauer spezifiziert.

Folgende Fehlercodes sind möglich:

0EH	Die Datei ist nicht eröffnet.
0FH	Ende der Datei (hard End)
10H	Ende der Datei (soft End)
20H	Der Dateiname ist fehlerhaft.
21H	Die Datei existiert bereits.
22H	Die Datei existiert nicht.
23H	Das Verzeichnis ist voll.
24H	Die Diskette ist voll.
25H	Die Diskette wurde gewechselt, wobei Dateien noch nicht geschlossen waren.
26H	Die Datei ist nur lesbar.

Folgende Routinen des Betriebssystems des 'KC compact' werden vom BASDOS-ROM übernommen:

BC77H	CAS IN OPEN	Eingabedatei eröffnen
BC7AH	CAS IN CLOSE	Eingabedatei schließen
BC7DH	CAS IN ABANDON	Eingabedatei vergessen
BC80H	CAS IN CHAR	ein Zeichen aus der Eingabedatei holen
BC83H	CAS IN DIRECT	Eingabedatei mit einem Mal lesen
BC86H	CAS RETURN	Zurückgeben des letzten gelesenen Zeichens an die Eingabedatei
BC89H	CAS TESTEOF	Abfrage, ob Ende der Eingabedatei erreicht wurde
BC8CH	CAS OUTOPEN	Ausgabedatei eröffnen
BC8FH	CAS OUTCLOSE	Ausgabedatei schließen
BC92H	CAS OUTABANDON	Ausgabedatei vergessen
BC95H	CAS OUTCHAR	ein Zeichen in die Ausgabedatei schreiben
BC98H	CAS OUTDIRECT	Ausgabedatei mit einem Mal schreiben
BC9BH	CAS CATALOG	Inhaltsverzeichnis erstellen

Eine genaue Beschreibung dieser Routinen ist im 'KC compact'-Systemhandbuch enthalten.

## 2.4. Die RSX-Erweiterungen des BASDOS

Im Gegensatz zu den vorher beschriebenen Routinen zur Dateiarbeit gibt es eine ganze Reihe von Programmen im BASDOS, die nicht über Vektoren in einer Sprungleiste aufrufbar sind. Sie sind aber über einen Namen erreichbar. Ihre Namen werden dem Betriebssystem bei der Initialisierung des BASDOS als RSX-Kommandos bekanntgemacht. Im folgenden werden alle RSX-Routinen des BASDOS mit den dazugehörigen Ein- und Ausgabeparametern beschrieben sowie zwei Beispiele für ihre Anwendung gezeigt. Dabei gelten folgende Abkürzungen:

PE: ... Parameterübergabe an das Programm,  
PA: ... Parameterübernahme vom Programm,  
UR: ... unveränderte Register.

**CPM**           Laden und Starten eines neuen Betriebssystems von Diskette

PE: keine  
PA: keine, Start neues System

**DISC**           alle Ein- und Ausgabebefehle beziehen sich auf die Diskette

PE: A=0 .... kein Parameter folgt, sonst Fehler  
PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
   CY=1 ... OK  
UR: IX,IY

**DISC.IN**       alle Eingabebefehle beziehen sich auf die Diskette

PE: A=0 .... kein Parameter folgt, sonst Fehler  
PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
   CY=1 ... OK  
UR: IX,IY

**DISC.OUT**     alle Ausgabebefehle beziehen sich auf die Diskette

PE: A=0 .... kein Parameter folgt, sonst Fehler  
PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
   CY=1 ... OK  
UR: IX,IY

**TAPE**           alle Ein- und Ausgabebefehle beziehen sich auf die Kassette

PE: A=0 .... kein Parameter folgt, sonst Fehler  
PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
   CY=1 ... OK  
UR: IX,IY

**TAPE.IN** alle Eingabebefehle beziehen sich auf die Kassette

PE: A=0 .... kein Parameter folgt, sonst Fehler  
 PA: C Y=0 ... Fehler, A=4 ... Fehlercode Bad Command  
 CY=1 ... OK  
 UR: IX,IY

**TAPE.OUT** alle Ausgabebefehle beziehen sich auf die Kassette

PE: A=0 ... kein Parameter folgt, sonst Fehler  
 PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
 CY=1...OK  
 UR: IX,IY

**A** Laufwerk A wird aktuelles Laufwerk

PE: keine  
 PA: keine  
 UR: IX,IY

**B** Laufwerk B wird aktuelles Laufwerk

PE: keine  
 PA: keine  
 UR: IX,IY

**DRIVE** Festlegen des aktuellen Laufwerks

PE: A Anzahl der Parameter (=1, sonst Fehler)  
 (IX),(IX+1) Variablenpointer  
 PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
 CY=1 ... OK  
 UR: IX,IY

Der Variablenpointer selbst enthält die Adresse des Stringdeskriptors, der folgenden Aufbau hat:

DESCRIB: DEFB Länge der Zeichenkette  
 DEFW Adresse, bei der die Zeichenkette beginnt

Die Zeichenkette besteht aus ASCII-Zeichen. Ihr Ende ist nicht gekennzeichnet. DRIVE erwartet "A" oder "B" als String (Länge 1).

**USER** Festlegen des Nutzerbereiches

PE: A Anzahl der Parameter (=1, sonst Fehler)  
 (IX),(IX+1) Userbereich als 16-Bit-Wert (von 0 bis 15)  
 PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
 CY=1 ... OK  
 UR: IX,IY

**DIR** Anzeigen des Verzeichnisses

PE: A Anzahl der Parameter (0 oder 1)  
 (IX),(IX+1) Variablenpointer  
 PA: CY=0 ... Fehler, A=4 ... Fehlercode Bad Command  
 CY=1 ... OK  
 UR: IX,IY

Der Variablenpointer wird nur benötigt, wenn das DIR-Kommando mit einer Dateimaske ausgeführt werden soll (A:=1) und verweist auf den String, der die Dateimaske enthält (siehe Kommando DRIVE).

**ERA** Löschen eines Dateieintrags

PE: A Anzahl der Parameter (1)  
 (IX),(IX+1) Variablenpointer  
 PA: CY=0 ... Fehler, A Fehlercode  
 CY=1 ... OK  
 UR: IX,IY

**REN** Umbenennen einer Datei

PE: A Anzahl der Parameter (2)  
 (IX),(IX+1) Variablenpointer für alten Namen  
 (IX+2),(IX+3) Variablenpointer für neuen Namen  
 PA: CY=0 ... Fehler, A Fehlercode  
 CY=1 ... OK  
 UR: IX,IY

Die folgenden Befehlsnamen enthalten keine druckbaren ASCII-Zeichen. Sie sind aber auch normale RSX-Befehle.

**01H** Fehlermeldungen enable/disable

PE: A =0 Meldungen ein, ansonsten Meldungen aus  
 PA: A alter Status der Fehlermeldungsangabe  
 UR: BC,DE,HL,IX,IY

**02H** Drive-Parameter ändern

PE: HL Adresse Parametertabelle  
 PA: keine  
 UR: IX,IY

Die Parametertabelle hat folgendes Aussehen:

PARTAB:	DEFW	Einschaltzeit in 1/50 s für Laufwerksmotor
	DEFW	Nachlaufzeit des Laufwerksmotors nach letztem Zugriff auf die Diskette
	DEFB	0AFH
	DEFW	Wartezeit nach Spurwechsel
	DEFB	Head unload Time für FDC
	DEFB	Head load Time für FDC

**03H** Diskettenformat einstellen

PE: A Kennzeichen des gewünschten Formates  
 PA: keine  
 UR: IX,IY

BASDOS enthält eine automatische Formaterkennung. In einigen Fällen ist aber die Voreinstellung eines der drei möglichen Formate sinnvoll. Das gewünschte Format wird in Bits 6 und 7 von A verschlüsselt. Es gilt:

Bit 7	Bit6	Format
0	0	9 * 512 * 80 * 2, Sektoroffset 0
0	1	9 * 512 * 40 * 1, Sektoroffset 40H
1	1	9 * 512 * 40 * 1, Sektoroffset C0H

**04H** Sektor lesen

PE: C Sektornummer (mit Offset)  
 D Spurnummer  
 E Laufwerk (0 oder 1)  
 HL Anfangsadresse für 512 Byte Sektorpuffer  
 PA: CY=0 ... Fehler, A Fehlercode  
 CY=1 ... OK  
 UR: BC,DE,HL,IX,IY

Beim zweiseitigen Diskettenformat sind Sektornummern von 1 bis 18 möglich. Die höheren Sektornummern werden auf der zweiten Diskettenseite abgelegt.

**05H** Sektor schreiben

PE: C Sektornummer (mit Offset)  
 D Spurnummer  
 E Laufwerk (0 oder 1)  
 HL Anfangsadresse für 512 Byte Sektorpuffer  
 PA: CY=0 ... Fehler, A Fehlercode  
 CY=1 ... OK  
 UR: BC,DE,HL,IX,IY

**06H** Spur formatieren

PE: C Nummer des ersten zu formatierenden Sektors  
 (mit Offset)  
 D Spurnummer  
 E Laufwerk (0 oder 1)  
 HL Anfangsadresse für Parametertabelle  
 PA: CY=0 ... Fehler, A Fehlercode  
 CY=1 ... OK  
 UR: BC,DE,HL,IX,IY

Die Tabelle enthält pro zu formatierendem Sektor vier Byte. Ein solcher Eintrag hat folgendes Aussehen:

TABx:      DEFB          Tracknummer für Sektor-ID  
           DEFB          Kopfnummer  
           DEFB          Sektornummer  
           DEFB          Sektorgröße für ID (2)

Insgesamt werden 9 Einträge erwartet.

#### 07H          Kopf über eine bestimmte Spur fahren

PE:   D            Spurnummer  
       E            Laufwerk (0 oder 1)  
 PA:   CY=0 ... Fehler, A Fehlercode  
       CY=1 ... OK  
 UR:   BC,DE,HL,IX,IY

Obwohl von allen vorher beschriebenen Routinen die Spur selbständig angefahren wird, ist diese Routine u.U. nötig.

#### 08H          Test Drive

PE:   A            Laufwerk (0 oder 1)  
 PA:   CY           =0 Laufwerk verfügbar  
               =1 Laufwerk nicht verfügbar  
       A            Inhalt des Statusregisters 0 des FDC  
 UR:   IX,IY

#### 09H          Wiederholungen im Fehlerfall festlegen

PE:   A            neuer Wert  
 PA:   A            alter Wert  
 UR:   BC,DE,HL,IX,IY

Ein Wert von 0 wird als 256 interpretiert.

Die folgenden Beispielpprogramme demonstrieren die Anwendung der RSX-Befehle. Im ersten Beispiel soll die Datei TEST.BAK von der Diskette entfernt werden.

```

LD   HL,KOM1       ; Das Kommando ERA wird übergeben
CALL  0BCD4H       ; KL FIND COMMAND
                  ; Jetzt steht die Adresse der entspre-
                  ; chenden Routine in HL und die ROM-
                  ; Nummer in C
RET   NC           ; Kommando wurde nicht gefunden
LD   A,1           ; ein Parameter wird übergeben
LD   IX,VARPTR     ; und mit IX genau beschrieben
CALL  001BH        ; LOW KL FAR PCHL
                  ; Jetzt wird die Routine mit der Adresse
                  ; HL und der ROM-Nummer C ausgeführt,
                  ; d.h., der Befehl ERA

KOM1:  RET
      DEFM        'ER'
      DEFB        'A'+80H
VARPTR:  DEFW       DESCRIP       ; Der Variablenpointer zeigt auf
                  ; einen Stringdeskriptor
DESCRIP:  DEFB       8            ; Der String ist 8 Zeichen lang
          DEFW       NAME         ; und beginnt bei der Adresse NAME
NAME:   DEFM       'TEST.BAK'
```

Im zweiten Beispiel soll der Beginn des Verzeichnisses auf Spur 1 Sektor 1 in einen Puffer gelesen werden. Die RSX-Routine muß anders gestartet werden als die im ersten Beispiel, da das C-Register zur Parameterübergabe benötigt wird.

```

SEK      EQU      1
TRK      EQU      1
DRV      EQU      0      ; Laufwerk A
;
;
LD       HL, KOM2      ; Befehl 04H
CALL    0BCD4H        ; KL FIND COMMAND
; Jetzt steht die Adresse der entsprechenden Routine in HL und die
; ROM-Nummer in C
RET      NC           ; Kommando wurde nicht gefunden
LD       A, C         ; Faradresse im RAM ablegen
LD       (FARADR), HL
LD       (FARADR+2), A
LD       C, SEK      ; Parameter einstellen
LD       D, TRK
LD       E, DRV
LD       HL, BUFFER
RST     18H          ; Aufruf der Routine, die durch
DEFW    FARADR       ; FARADR beschrieben wird bei
; Erhalt aller Registerinhalte der CPU
; zur Parameterübergabe
RET
FAEADR:  DEFS      3
KOM2:    DEFB      84H      ; Bit 7 ist gesetzt, da die Zeichen-
; kette schon zu Ende ist.
BUFFER:  DEFS      256

```

### 3. Diskettenformate

In der BASDOS-Betriebsart ist ein anderes Diskettenformat Standardformat, als in der MicroDOS-Betriebsart. Damit wird einem Vermischen von Dateien aus beiden Betriebsarten auf einer Diskette und den damit verbundenen Fehlermöglichkeiten entgegen gewirkt. In der BASDOS-Betriebsart können keine MicroDOS-Disketten gelesen werden, wohl aber umgekehrt. Damit stehen dem Nutzer des BASDOS alle Möglichkeiten der Dateimanipulation und eventuell -reparatur zur Verfügung, die unter CP/M-ähnlichen Betriebssystemen lauffähig sind. BASDOS unterstützt ein Diskettenformat für 5,25"-Disketten (9 Sektoren zu 512 Bytes auf 80 Spuren, beidseitig, eine Systemspur und ein physischer Sektoroffset von 0) und zwei Diskettenformate, die mit einem 5,25"-Laufwerk ein 8"-Laufwerk simulieren (9 Sektoren zu 512 Bytes auf 40 Spuren, einseitig, keine Systemspur und ein physischer Sektoroffset von 0H bzw. zwei Systemspuren und ein physischer Sektoroffset von 40H). Die Blockgröße beträgt in allen drei Formaten 4 KByte. Es wird dem Anwender empfohlen, nur das 80-Spuren-Format mit 708 KByte Diskettenkapazität zu nutzen. In der MicroDOS-Betriebsart wird standardmäßig das Laufwerk C mit diesem Format belegt.

## 4. Dateiaufbau

In der BASDOS-Betriebsart besitzen die Dateien prinzipiell den gleichen Aufbau wie in der MicroDOS-Betriebsart, um einen Dateiaustausch zu realisieren. Die Verzeichnis-Einträge entsprechen denen des MicroDOS. Es bestehen jedoch einige Besonderheiten gegenüber Dateien in der MicroDOS-Betriebsart, die nachfolgend beschrieben werden.

### 4.1. ASCII-Dateien

ASCII-Dateien entsprechen genau denen in der MicroDOS-Betriebsart. Sie werden mit einem ^Z (1AH) abgeschlossen. Alle anderen Dateien beginnen mit einem 128 Byte langen Block (dem Header), in dem weitere Informationen zur Datei abgelegt sind.

### 4.2. Nicht-ASCII-Dateien

BASDOS-Dateien, die keine ASCII-Dateien, sind enthalten einen Header mit folgendem Inhalt:

Byte	Inhalt
0	USER-Nummer
1 bis 8	Name ggf. mit Leerzeichen gefüllt
9 bis 11	Typkennzeichnung
18	Status-Byte =0 normal abgespeichert =1 geschützte Datei (z.B. kein List möglich) =2 Binärfile
21, 22	Ladeadresse (Low-, Highbyte)
24, 25	Länge der Datei aussch. Header (Low-, Highbyte)
26, 27	Startadresse (Low-, Highbyte)
64, 65	Länge der Datei aussch. Header (Low-, Highbyte)
67, 68	Summe über die Bytes 0 bis 66 des Headers
69 - 127	beliebig

Alle anderen Bytes des Headers = 0!

---

## Anlagen

### Anlage 1: Reservierte Speicherplätze (Seite 0)

Speicherplätze von		Inhalt bis
0000H	0002H	Sprung zum BIOS-Eintrittspunkt WBOOT Damit ist ein einfacher programmierter Neustart (Sprung zur Adresse (0) möglich.
0003H		Enthält das IOBYTE
0004H		Nummer des aktuellen Laufwerkes und Benutzernummer
0005H	0007H	Enthält eine Sprunganweisung zum BDOS Die Sprunganweisung liefert einmal den Haupteintrittspunkt in das BDOS und zum anderen stellt die Sprungadresse die niedrigste vom Betriebssystem verwendete Speicheradresse dar. Debugger verändern die Sprungadresse, um den durch sie reduzierten Speicher zu kennzeichnen.
0008H	0037H	RST 1 bis RST 6 von MicroDOS nicht verwendet
0038H	003AH	RST 7 Enthält während Debug-Mode eine Sprunganweisung in den Debugger für programmierte Haltepunkte, wird vom Betriebssystem aber nicht benutzt
003BH	005BH	reserviert
0040H	0042H	Systemuhr
005CH	007FH	durch CCP erzeugter Standard-FCB
0080H	00FFH	Standard-Datenpuffer

## Anlage 2: Bildschirmsteuerzeichen

Bei der Ausgabe eines Zeichens auf den Bildschirm wird untersucht, ob ein druckbares Zeichen oder ein Steuerzeichen ausgegeben werden soll. Liegt ein Steuerzeichen vor, werden spezielle Aktionen veranlaßt.

Die folgende Tabelle enthält die Steuerzeichen und ihre Wirkung im MicroDOS.

Hex. code	Dez. code	Wirkung des Steuerzeichens
01	1	Cursor in linke obere Ecke setzen (HOME)
07	7	Ausgabe eines kurzen Tones (BEEP)
08	8	Cursor um eine Position nach links setzen
0A	10	Cursor um eine Zeile nach unten setzen
0C	12	Bildschirm löschen, Cursor home
0D	13	Cursor auf Zeilenanfang setzen
0E	14	amerikanischen Zeichensatz einstellen
0F	15	deutschen Zeichensatz einstellen
14	20	ab Cursorposition bis Bildschirmende löschen
15	21	Cursor um eine Position nach rechts setzen
16	22	ab Cursorposition bis Zeilenende löschen
18	24	Cursorzeile löschen, Cursor auf Zeilenanfang
1A	26	Cursor um eine Zeile nach oben setzen
1B	27	Einleiten einer ESCape-Steuersequenz
7F	127	Cursor ein Zeichen nach links, Zeichen löschen
82	130	Cursor einschalten
83	131	Cursor ausschalten

Die folgenden Steuerzeichen bewirken neben ihrer Ausführung auch die Ausgabe eines Leerzeichens auf den Bildschirm

84	132	normale Zeichendarstellung
85	133	inverse Zeichendarstellung
86	134	inverse Zeichendarstellung (für intensiv)
87	135	inverse Zeichendarstellung (für invers, intensiv)

**Anlage 3: ESCape-Funktionen**

Nach der Ausgabe von ESCape werden mehrere Parameter erwartet.

ESC-Folge	Wirkung
1BH Zeile+20H	Direkte Cursorpositionierung
1BH 41H xx y	Punkt setzen; xx ist 2 Byte lang (niederwertiger Teil zuerst). Wertebereich $0 \leq xx \leq 319$ , $0 \leq y \leq 191$
1BH 42H xx y	Punktlöschen
1BH 43H m	Grafikmodus einstellen; wirkt auf Punkt setzen, Linie und Kreis. Werte für m: 0 = Löschen, 1 = xor-Verknüpfung von zu setzenden Pixeln mit vorhandenem Bildinhalt, 2 = and-Verknüpfung, 3=or-Verknüpfung
1BH 44H xx1 y1 xx2 y2	Linie auf dem Bildschirm darstellen
1BH 45H xx y r	Kreis auf dem Bildschirm darstellen
1BH 48H v h	Farbe einstellen; v, h BASIC-Farbwerte für Vordergrund und Hintergrundfarbe
1BH 4EH v	Vordergrundfarbe einstellen
1BH 4FH h	Hintergrundfarbe einstellen
1BH 54H aa nn +nn Bytes	Füllen des Speichers des Grundgerätes mit nn Bytes beginnend ab Adresse aa
1BH 5EH 40H	norm. amerik. Zeichen ein + Leerzeichen
1BH 5EH 41H	inv. amerik. Zeichen ein + Leerzeichen (=intensiv)
1BH 5EH 42H	inv. amerik. Zeichen ein + Leerzeichen (=blinkend)
1BH 5EH 43H	inv. amerik. Zeichen ein + Leerzeichen (=blk.+int.)
1BH 5EH 44H	norm. deutsche Zeichen ein + Leerzeichen
1BH 5EH 45H	inv. deutsche Zeichen ein + Leerzeichen (=intensiv)
1BH 5EH 46H	inv. deutsche Zeichen ein + Leerzeichen (=blinkend)
1BH 5EH 47H	inv. deutsche Zeichen ein + Leerzeichen (=blk.+int.)
1BH 5EH 50H	inv. amerik. Zeichen ein + Leerzeichen
1BH 5EH 51H	inv. amerik. Zeichen ein + Leerzeichen (=inv.+int.)
1BH 5EH 52H	inv. amerik. Zeichen ein + Leerzeichen (=inv.+blk.)
1BH 5EH 53H	inv. amerik. Zeichen ein + Leerz. (=inv.+blk.+int.)
1BH 5EH 54H	inv. deutsche Zeichen ein + Leerzeichen
1BH 5EH 55H	inv. deutsche Zeichen ein + Leerzeichen (=inv.+int.)
1BH 5EH 56H	inv. deutsche Zeichen ein + Leerzeichen (=inv.+blk.)
1BH 5EH 57H	inv. deutsche Zeichen ein + Leerz. (=inv.+blk.+int.)
1BH 5FH 30H	amerik. Zeichensatz ein, kein Leerzeichen
1BH 5FH 31H	deutscher Zeichensatz ein, kein Leerzeichen
1BH 5FH 32H	eigener Zeichensatz ein (ist mit kyrillischem Alphabet initialisiert). Druckereinstellung so, daß Zeichen im Grafikmode gedruckt werden. Druck nur mit LF und FF möglich
1BH 5FH 33H	Rückgängig machen der Einstellungen von 1BH 5FH 32H
1BH 5FH 34H +760 Byte	Ablegen eines eigenen Zeichensatzes. Alle Zeichen von 20H bis 7EH werden als 8*8 Pixelfond erwartet. Erst danach erfolgen Bildausgaben. Der kyrillische Zeichensatz wird überschrieben.

ESC-Folge	Wirkung
1BH 5FH 35H	Bild-in-reset; Ausgehend von der Position des Bild-in-Pointers werden 16 Characterpositionen mit den Bytes gefüllt, die von Adresse 0FF80H bis Adresse 0FFFFH im Speicher der Floppy-Erweiterung stehen. Der Bild-in-Pointer wird auf das nächste noch nicht beschriebene Zeichen gesetzt.
1BH 5FH 36H	Bild-in-reset; Der Bild-in-Pointer wird auf die linke obere Ecke gesetzt.
1BH 5FH 37H	Bild-out; Ausgehend von der Position des Bild-out-Pointers wird der Bereich von Adresse 0FF80H bis Adresse 0FFFFH im Speicher der Floppy-Erweiterung mit den Bytes der 16 nächsten Characterpositionen gefüllt. Der Bild-out-Pointer wird auf das nächste noch nicht ausgegebene Zeichen gesetzt.
1BH 5FH 38H	Bild-out-reset; Der Bild-out-Pointer wird auf die linke obere Ecke gesetzt.
1BH 5FH 39H	Hardcopy des Bildschirminhaltes auf Drucker ausgeben. (Drucker EPSON-kompatibel)
1BH 5FH 40H	linke obere Ecke normal
1BH 5FH 41H	linke obere Ecke invers (=intensiv)
1BH 5FH 42H	linke obere Ecke invers (=blinkend)
1BH 5FH 43H	linke obere Ecke invers (=int.+blk.)
1BH 5FH 44H <sup>^</sup>	rechte obere Ecke normal
1BH 5FH 45H	rechte obere Ecke invers (=intensiv)
1BH 5FH 46H	rechte obere Ecke invers (=blinkend)
1BH 5FH 47H	rechte obere Ecke invers (=int.+blk.)
1BH 5FH 48H	linke untere Ecke normal
1BH 5FH 49H	linke untere Ecke invers (=intensiv)
1BH 5FH 4AH	linke untere Ecke invers (=blinkend)
1BH 5FH 4BH	linke untere Ecke invers (=int.+blk.)
1BH 5FH 4CH	rechte untere Ecke normal
1BH 5FH 4DH	rechte untere Ecke invers (=intensiv)
1BH 5FH 4EH	rechte untere Ecke invers (=blinkend)
1BH 5FH 4FH	rechte untere Ecke invers (=int.+blk.)
1BH 5FH 50H	oberer Intersect normal
1BH 5FH 51H	oberer Intersect invers (=intensiv)
1BH 5FH 52H	oberer Intersect invers (=blinkend)
1BH 5FH 53H	oberer Intersect invers (=int.+blk.)
1BH 5FH 54H	rechter Intersect normal
1BH 5FH 55H	rechter Intersect invers (=intensiv)
1BH 5FH 56H	rechter Intersect invers (=blinkend)
1BH 5FH 57H	rechter Intersect invers (=int.+blk.)
1BH 5FH 58H	linker Intersect normal
1BH 5FH 59H	linker Intersect invers (=intensiv)
1BH 5FH 5AH	linker Intersect invers (=blinkend)
1BH 5FH 5BH	linker Intersect invers (=int.+blk.)
1BH 5FH 5CH	unterer Intersect normal
1BH 5FH 5DH	unterer Intersect invers (=intensiv)
1BH 5FH 5EH	unterer Intersect invers (=blinkend)
1BH 5FH 5FH	unterer Intersect invers (=int.+blk.)
1BH 5FH 60H	horizontale Linie normal
1BH 5FH 61H	horizontale Linie invers (=intensiv)
1BH 5FH 62H	horizontale Linie invers (=blinkend)
1BH 5FH 63H	horizontale Linie invers (=int.+blk.)

ESC-Folge	Wirkung
1BH 5FH 64H	vertikale Linie normal
1BH 5FH 65H	vertikale Linie invers (=intensiv)
1BH 5FH 66H	vertikale Linie invers (=blinkend)
1BH 5FH 67H	vertikale Linie invers (=int.+blk.)
1BH 5FH 68H	Kreuz normal
1BH 5FH 69H	Kreuz invers (=intensiv)
1BH 5FH 6AH	Kreuz invers (=blinkend)
1BH 5FH 6BH	Kreuz invers (=int.+blk.)
1BH 60H t z	Tastenbelegung ändern; t Tastennr., muß <<80H sein z neues Zeichen außer 0FEH und 0FFH. z sollte kein Erweiterungszeichen (80H-9FH) sein
1BH 61H t z	Tastaturbelegung für Taste+Shift ändern
1BH 62H t z	Tastaturbelegung für Taste+Control ändern
1BH 63H t l	l*Byte Der Taste t wird ein String von l Zeichen zugeordnet. Die Länge des Strings ist max. 32.
1BH 64H t l	l*Byte Der Taste t+Shift wird String zugeordnet.
1BH 65H t l	l*Byte Der Taste t+Control wird String zugeordnet.
1BH 66H t	Autorepeat für Taste t ein
1BH 67H t	Autorepeat für Taste t aus
1BH 68H t1 t2	Festlegen der Verzögerungszeit beim Repitieren von Tasten. t1 Zeit bis zum ersten Repitieren t2 Zeit zwischen weiteren Repeats Zeiten in 1/50s, Standardwerte: t1=30 (0,6s), t2=2 (0,04s)
1BH Zeile+80H Spalte+80H	direkte Cursorpositionierung

**BDOS-Übersicht**

Nr.	Name	Eingangsparameter	Ausgangsparameter_
0	System rücksetzen	-	-
1	Konsoleneingabe	-	A = Zeichen
2	Konsolenausgabe	E = Zeichen	-
3	Zusatzeingabe	-	A = Zeichen
4	Zusatzausgabe	E = Zeichen	-
5	Druckerausgabe	E = Zeichen	-
6	Direkte Konsolen-E/A	siehe Definition	
7	Status Zusatzeingabe	-	A = 00/FF
8	Status Zusatzausgabe	-	A = 00/FF
9	String ausgeben	DE = Puffer	-
10	String einlesen	DE = Puffer	siehe Definition
11	Konsolenstatus	-	A = 00/FF
12	Versionsnummer	-	HL = Versionsnr.
13	Disk rücksetzen	-	siehe Definition
14	Laufwerk wählen	E = Laufwerk	siehe Definition
15	Datei eröffnen	DE = FCB	A = Verzeichniscode
16	Datei schließen	DE = FCB	A = Verzeichniscode
17	erste Datei suchen	DE = FCB	A = Verzeichniscode
18	nächste Datei suchen	-	A = Verzeichniscode
19	Datei löschen	DE = FCB	A = Verzeichniscode
20	sequentiell lesen	DE = FCB	A = Fehlercode
21	sequentiell schreiben	DE = FCB	A = Fehlercode
22	Datei erzeugen	DE = FCB	A = Verzeichniscode
23	Datei umbenennen	DE = FCB	A = Verzeichniscode
24	Anwahlvektor holen	-	HL = Vektor
25	aktuelles Laufwerk	-	A = aktuelles Laufw.
26	DMA-Adresse setzen	DE = DMA	-
27	Belegungsvektor holen	-	HL = Vektoradresse
28	Schreibschutz setzen	-	siehe Definition
29	Schreibschutzvektor holen	-	HL = Schreibschutz- vektor
30	Dateiattribute	DE = FCB	siehe Definition
31	Parameteradresse holen	-	HL = DPB-Adresse
32	Benutzercode	siehe Definition	siehe Definition
33	wahlfrei lesen	DE = FCB	A = Fehlercode
34	wahlfrei schreiben	DE = FCB	A = Fehlercode
35	Dateigröße berechnen	DE = FCB	r0, r1, r2
36	Datensatz setzen	DE = FCB	r0, r1, r2
37	Disketten rücksetzen	DE = Disk.vektor	A = 0
40	wahlfrei schreiben mit Nullen auffüllen	DE = FCB	A = Fehlercode
45	Fehlermodus setzen	DE = Modus	-
46	Disk. Platz bestimmen	E = Laufwerk	Sektorenanzahl
47	Programm wechseln	Kommando	-
49	Systemparameter	DE = Parameter-	HL = Parameteradr.
50	BIOS-Funktionen	DE = Parameteradr.	BIOS-spezifisch
108	Rückkehrcode	DE = Rückkehrcode	Rückkehrcode
110	Begrenzer	DE = FFFH/Begrenzer	Begrenzer
111	Pufferanzeigen	DE = Parameteradr.	-
112	Puffer drucken	DE = Parameteradr.	-
152	FCB erzeugen	DE = Paramteradr.	FCB

## Anlage 5: Programmbeispiele

Im folgenden sollen einige Beispiele die Nutzung der Systemfunktionen verdeutlichen. Die Quelldateien werden mit Hilfe des Textprozessors erstellt und dann mit dem Assembler (ASM in /2/) in REL-Dateien übersetzt. Mit dem Linker (LINK) können sie dann in ablauffähige COM-Programme umgesetzt werden, welche direkt unter MicroDOS arbeiten. Besonderen Wert wurde in den Beispielen auf die Nutzung der ESCape-Funktionen der Bildschirmausgabe gelegt.

### 5.1. DUMP-Ausgabe

Das erste Beispielprogramm ist allgemeiner Natur und läuft auch unter allen SCP- bzw. CP/M-Systemen. Dieses Ausgabeprogramm liest eine Eingabedatei, welche im CCP-Kommando angegeben wird, und gibt diese auf der Konsole in hexadezimaler Form aus. Das Programm rettet den CCP-Stackpointer beim Eintritt, setzt den Stackpointer auf einen lokalen Stack und stellt den CCP-Stackpointer vor der direkten Rückkehr zum CCP wieder her. Das heißt, es wird kein Warmstart am Ende des Programms durchgeführt.

```

;
; Dump-Ausgabeprogramm
;
; Z80
0005 BDOS EQU 0005H ; BDOS-Eintritt
0001 CONS EQU 1 ; Konsole lesen
0002 TYPEF EQU 2 ; Konsole schreiben
0009 PRINTF EQU 9 ; Puffer ausgeben
000B BRKF EQU 11 ; Konsolenstatus
000F OPENF EQU 15 ; Datei eröffnen
0014 READF EQU 20 ; Daten lesen

;
005C FCB EQU 005CH ; FCB-Adresse
0080 BUFF EQU 0080H ; Eingabepuffer

;
; Steuerzeichen
000D CR EQU 0DH ; Wagenrücklauf
000A LF EQU 0AH ; Zeilenvorschub

;
; FCB-Definitionen
005C FCBDN EQU FCB+0 ; Diskettenname
005D FCBFN EQU FCB+1 ; Dateiname
0065 FCBFT EQU FCB+9 ; Dateityp
0068 FCBR1 EQU FCB+12 ; Erweiterung
006B FCBRC EQU FCB+15 ; Datensatzanzahl
007C FCBRCR EQU FCB+32 ; Datensatzzähler
007D FCBLN EQU FCB+33 ; FCB-Länge

;
; Stack setzen
0000' 21 0000 LD HL, 0
0003' 39 ADD HL, SP
; Stackpointer des CCP in HL
0004' 22 00F5' LD (OLDSP), HL
; SP auf lokalen Wert setzen
0007' 31 0137' LD SP, STKTOP
; Puffer einlesen und ausgeben
000A' CD 00C1' CALL SETUP ; Datei eröffnen
000D' FE FF CP 255 ; Datei vorhanden?
000F' C2 001B' JP NZ, OPENOK ; Springen wenn ok

;
; Datei nicht vorhanden, Fehlermeldung
0012' 11 00DD' LD DE, OPNMSG
0015' CD 009C' CALL ERR

```

```

0018'   C3 0051'           JP    FINIS      ; Rückkehr
;
; Eröffnung ok, Pufferzeiger auf Ende setzen
001B'   3E 80           OPENOK: LD    A, 80H
001D'   32 00F3'        LD    (<IBP>, A) ; Zeiger=80H
; HL auf nächste Adresse setzen
0020'   21 0000'        LD    HL, 0 ; bei Null beginnen
;
GLOOP:  PUSH    HL ; Zeile retten
0023'   E5           CALL    GNB
0024'   CD 00A2'        LD    A, B
0027'   E1           POP    HL ; Zeile wiederholen
0028'   DA 0051'        JP    C, FINIS ; Carry wird von gnb
; bei Dateende gesetzt
002B'   47           LD    B, A
; Hexzeichen ausgeben, Zeile testen
002C'   7D           LD    A, L
002D'   E6 0F'        AND    0FH ; 4 Bit testen
002F'   C2 0044'        JP    NZ, NONUM
; Zeilennummer ausgeben
0032'   CD 0072'        CALL   CRLF
; Konsole auf Abbruch testen
0035'   CD 0059'        CALL   BREAK
; Akkumulator-LSB=1, falls Zeichen bereit
0038'   0F           RRC    A ; Carry-Bit setzen
0039'   DA 0051'        JP    C, FINIS ; Abbruch
;
003C'   7C           LD    A, H
003D'   CD 008F'        CALL   PHEX
0040'   7D           LD    A, L
0041'   CD 008F'        CALL   PHEX
;
NONUM:  INC    HL
0044'   23           LD    A, ''
0045'   3E 20'        CALL   PCHAR
0047'   CD 0065'        LD    A, B
004A'   78           CALL   PHEX
004B'   CD 008F'        CALL   PHEX
004E'   C3 0023'        JP    GLOOP
;
; Ende der Ausgabe, zurück zum CCP
;
0051'   CD 0072'        FINIS: CALL   CRLF
0054'   2A 00F5'        LD    HL, (OLDSP)
0057'   F9           LD    SP, HL
; Stackpointer enthält CCP-Stackadresse
0058'   C9           RET ; zurück zum CCP
;
; Unterprogramme
;
; Konsole auf Abbruch testen (Zeichen beliebig)
0059'   E5           BREAK: PUSH   HL
005A'   D5           PUSH   DE
005B'   C5           PUSH   BC ; Register retten
005C'   0E 0B'        LD    C, BRKF
005E'   CD 0005'        CALL   BDOS
0061'   C1           POP    BC
0062'   D1           POP    DE
0063'   E1           POP    HL
0064'   C9           RET
;
; Zeichen ausgeben
0065'   E5           PCHAR: PUSH   HL
0066'   D5           PUSH   DE
0067'   C5           PUSH   BC ; Registerretten
0068'   0E 02'        LD    C, TYPEF
006A'   5F           LD    E, A
006B'   CD 0005'        CALL   BDOS
006E'   C1           POP    BC
006F'   D1           POP    DE
0070'   E1           POP    HL ; Register zurück
0071'   C9           RET

```

```

;
; neue Zeile
0072' 3E 0D      CRLF: LD A, CR
0074' CD 0065'   CALL PCHAR
0077' 3E 0A      LD A, LF
0079' CD 0065'   CALL PCHAR

007C' C9                RET

;
; Hex-ZeicheninAausgeben
007D' E6 0F      PNIB: AND 0FH ; untere 4 Bits
007F' FE 0A      CP 10
0081' D2 0089'   JP NC, P10
; kleiner gleich 9
0084' C6 30      ADD A, '0'
0086' C3 008B'   JP PRN
; größer oder gleich 10
0089' C6 57      P10: ADD A, 'A'-10
008B' CD 0065'   PRN: CALL PCHAR
008E' C9                RET

;
; Register A in Hex-Format ausgeben
008F' F5                PHEX: PUSH AF
0090' 0F                RRCA
0091' 0F                RRCA
0092' 0F                RRCA
0093' 0F                RRCA
0094' CD 007D'   CALL PNIB
0097' F1                POP AF
0098' CD 007D'   CALL PNIB
009B' C9                RET

;
; Fehlermeldung ausgeben, DE adressiert die
; Meldung, welche mit "$" endet
009C' 0E 09      ERR: LD C, PRINTF
; Puffer ausgeben
009E' CD 0005    CALL BDOS
00A1' C9                RET

;
; nächstes Byte holen
00A2' 3A 00F3'   GNB: LD A, (IBP)
00A5' FE 80      CP 80H
00A7' C2 00B3'   JP NZ, G0

;
; neuen Datensatz lesen
00AA' CD 00CE'   CALL DISKR
00AD' B7                OR A
; 0 wenn lesen ok
00AE' CA 00B3'   JP Z, G0

; nächstes Byte
; Ende der Daten, Carry setzen für EOF
00B1' 37                SCF
00B2' C9                RET

;
; Byte auf Adresse buff+<a> lesen
00B3' 5F                G0: LD E, A ; puffer index
00B4' 16 00          LD D, 0 ; 16-Bit-Wert
00B6' 3C                INC A ; index erhöhen
00B7' 32 00F3'   LD (IBP), A ; und abspeichern
; Zeiger ist erhöht und gerettet
00BA' 21 0080      LD HL, BUFF
00BD' 19                ADD HL, DE
; HL enthält absolute Adresse des nächsten
; Zeichens
00BE' 7E                LD A, (HL)
; Byte ist im Akkumulator
00BF' B7                OR A ; carry rücksetzen
00C0' C9                RET

;
; Eingabedatei eröffnen
00C1' AF                SETUP: XOR A ; Akku löschen
00C2' 32 007C      LD (FCBCR), A ; "cr" löschen

```

```

;
00C5' 11 005C          LD    DE,FCB
00C8' 0E 0F          LD    C,OPENF
00CA' CD 0005        CALL BDOS
; Akkumulator enthält 255 bei Fehler

00CD' C9            RET

;
; Datensatz lesen
DISKR:
00CE' E5            PUSH  HL
00CF' D5            PUSH  DE
00D0' C5            PUSH  BC
00D1' 11 005C        LD    DE,FCB
00D4' 0E 14          LD    C,READF
00D6' CD 0005        CALL  BDOS
00D9' C1            POP   BC
00DA' D1            POP   DE
00DB' E1            POP   HL
00DC' C9            RET

; Feld für konstante Zeichenketten
00DD' 6E 6F 20 69   OPNMSG: DEFM 'no inputfile present$'
00E1' 6E 70 75 74
00E5' 20 66 69 6C
00E9' 65 20 70 72
00ED' 65 73 65 6E
00F1' 74 24

;
; Variablenfeld
00F3' IBP:  DEFS  2      ; eingabepuffer zeiger
00F5' OLDSP: DEFS  2    ; ccp stack pointer

;
; lokaler Stack
00F7' STKTOP: DEFS  64   ; 32 niveaus reserviert
0137'
;

```

## 5.2 Datei in Speicher des Grundgerätes kopieren

Dieses Programmbeispiel zeigt, wie eine Datei von der Diskette in den Speicher des KC compact-Grundgerätes kopiert werden kann. Der Aufruf erfolgt über den Programmnamen (z.B. MOVE) mit nachfolgender Dateiabezeichnung. An diese anschließend kann eine hexadezimale Zieladresse (vierstellig) eingegeben werden. Fehlt diese, wird als Vorzugsadresse (0FF06H) angenommen. In (0FF06H) wird nach Abschluß des Einlesens der Datei die Adresse der ersten freien Speicherzelle im RAM des 'KC compact' eingetragen. Der Anwender muß sich darum kümmern, daß die übertragenen Dateien nicht so lang sind, daß die Systemzellen ab 0A700H im RAM des 'KC compact' überschrieben werden.

```

;
; Übertragen Datei in KC-Speicher
; ab eingegebener Adresse oder ab 4000H
.Z80
0005 BDOS EQU 5
005C FCB EQU 5CH ; Name
006C FCB1 EQU 6CH ; Adresse hex.
0080 DMA EQU 80H
001B ESC EQU 1BH
FF06 ANFAD EQU 0FF06H ; Anfangsadressefreier
; Speicher im KCC
;
0000' 31 0120' START: LD SP, STACK ; Lokaler Stack
0003' 2A 0001 LD HL, (1) ; Warmstart
0006' 11 0009 LD DE, 9
0009' 19 ADD HL, DE
000A' 22 00B4' LD (CALAD+1), HL
000D' 21 006D LD HL, FCB1+1
0010' 11 0000 LD DE, 0 ; erfassen Adresse
0013' 06 04 LD B, 4
0015' 7E ST1: LD A, M
0016' 23 INC HL
0017' D6 30 SUB 30H
0019' 38 1E JR C, ST2
001B' FE 0A CP 0AH
001D' 38 06 JR C, ST3
001F' D6 07 SUB 7
0021' FE 10 CP 16
0023' 30 14 JR NC, ST2
0025' CB 23 ST3: SLA E
0027' CB 12 RL D
0029' CB 23 SLA E
002B' CB 12 RL D
002D' CB 23 SLA E
002F' CB 12 RL D
0031' CB 23 SLA E
0033' CB 12 RL D
0035' 83 ADD A, E
0036' 5F LD E, A
0037' 10 DC DJNZ ST1
0039' 78 ST2: LD A, B
003A' FE 04 CP 4
003C' EB EX DE, HL
003D' 20 03 JR NZ, ST4
003F' 2A FF06 LD HL, (ANFAD) ; Standardadresse
0042' 22 00DE' ST4: LD (MERAD), HL ; Merken--
0045' 0E 0F LD C, 15 ; OPEN
0047' 11 005C LD DE, FCB
004A' CD 0005 CALL BDOS
004D' 3C INC A
004E' CA 00BA' JP Z, S7
0051' AF XOR A ; cr=0
0052' 32 007C LD (FCB+32), A
0055' 21 0120' LD HL, PUFFER
0058' 0E 14 S1: LD C, 20 ; Sequ. lesen

```

```

005A' 11 005C          LD      DE,FCB
005D' E5              PUSH   HL
005E' CD 0005        CALL   BDOS
0061' E1              POP    HL
0062' A7              AND    A           ;Dateiende?
0063' 20 0C          JR     NZ,S20
0065' 11 0080        LD     DE,DMA
0068' EB              EX     DE,HL
0069' 01 0080        LD     BC,128
006C' ED B0          LDIR   ;umladen
006E' EB              EX     DE,HL
006F' 18 E7          JR     SI

;
S20: 0071' 3E 1B          LD     A,ESC      ;HL=Ende
0073' CD 00AF'       CALL   OUT
0076' 3E 54          LD     A,'T'     ;Uebertragung in KC
0078' CD 00AF'       CALL   OUT
007B' 3A 00DE'       LD     A,(MERAD) ;Adresse LOW
007E' CD 00AF'       CALL   OUT
0081' 3A 00DF'       LD     A,(MERAD+1);Adresse HIGH
0084' CD 00AF'       CALL   OUT
0087' 22 00DE'       LD     <MERAD>,HL ;neuen freien Speicher-
;platz eintragen

008A' 11 0120'       LD     DE,PUFFER
008D' A7              AND    A
008E' ED 52          SEC    HL,DE      ;Berechnung der Laenge
0090' 7D              LD     A,L
0091' CD 00AF'       CALL   OUT      ;Laenge LOW
0094' 7C              LD     A,H
0095' CD 00AF'       CALL   OUT      ;HIGH
0098' 1A              LD     A,(DE)
SL:  0099' CD 00AF'       CALL   OUT
009C' 2B              DEC    HL
009D' 13              INC    DE
009E' 7D              LD     A,L
009F' B4              OR     H
00A0' 20 F6          JR     NZ,SL
00A2' 3E 0A          LD     A,0AH     ;LF
00A4' CD 00AF'       CALL   OUT
00A7' 3E 0D          LD     A,0DH     ;CR
00A9' CD 00AF'       CALL   OUT
00AC' C3 0000        WARM:  JF     0           ;Warmstart

;
OUT: 00AF' E5              PUSH   HL
00B0' D5              PUSH   DE
00B1' C5              PUSH   BC
00B2' 4F              LD     C,A
00B3' CD 0000        CALAD: CALL  0           ;BIOS
00B6' C1              POP    BC
00B7' D1              POP    DE
00B8' E1              POP    HL
00B9' C9              RET
00BA' 11 00C4'       LD     DE,TX2
00BD' 0E 09          LD     C,9
00BF' CD 0005        CALL   BDOS
00C2' 18 E8          JR     WARM

;
TX2: 00C4' 20 44 61 74   DEFM   'Datei nicht gefunden!'
00C8' 65 69 20 6E
00CC' 69 63 68 74
00D0' 20 67 65 66
00D4' 75 6E 64 65
00D8' 6E 20 21
00DB' 0A 0D          DEFW  0A0DH
00DD' 24              DEFM  '$'
00DE' 0000          MERAD: DEFW 0000H   ;Merkzelle
00E0'              DEFS  64
0120'              STACK EQU $
0120'              PUFFER EQU $   ;Ladepuffer
0120'              END
    
```

---

**Literaturverzeichnis**

- /1/ Kieser, Meder: "Mikroprozessortechnik"  
VEB Verlag Technik Berlin
- /2/ "Systemhandbuch SCP - Anleitung für den Programmierer"  
Teil 2 - Assemblerprogrammierung  
VEB Robotron Büromaschinenwerk "E. Thälmann" Sömmerda
- /3/ "Bedienungsanleitung und Sprachbeschreibung BASI"  
VEB Robotron Büromaschinenwerk "E. Thälmann" Sömmerda
- /4/ "Programmtechnische Beschreibung REDABAS"  
VEB Robotron Büromaschinenwerk "E. Thälmann" Sömmerda
- /5/ "Bedienungsanleitung und Sprachbeschreibung PASCAL 880/S"  
VEB Robotron Büromaschinenwerk "E. Thälmann" Sömmerda
- /6/ "Bedienungsanleitung BASC"  
VEB Robotron Büromaschinenwerk "E. Thälmann" Sömmerda
- /7/ Beschreibung zum Modul M027 DEVELOPMENT  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /8/ D004 Handbuch für den Bediener  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /9/ Beschreibung zum Modul M003 V24  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /10/ Beschreibung zur Programmkassette C0171/1  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /11/ Systemhandbuch KC85/3
- /12/ Systemhandbuch KC85/4  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /13/ Beschreibung zum Modul M001 DIGITALIN/OUT  
VEB Mikroelektronik "W. Pieck" Mühlhausen
- /14/ Böhl, E.: Integrierte Floppy-Disk-Controller-Schaltungen  
U8272 D08 und U8272 D04, rfe, Berlin 36 (1987) 11, S.703
- /15/ Mikroprozessoren der II. Leistungsklasse  
Hefte CPU und CTC  
VEB Mikroelektronik "K. Marx" Erfurt
- /16/ "Anwendungsbeschreibung und Bedienungsanleitung DIENST" VEB  
Robotron Büromaschinenwerk "E. Thälman" Sömmerda

**Abkürzungsverzeichnis**

ASCII	American Standard for Information Interchange (international standardisierter Code zur digitalen Verschlüsselung von Texten)
BDOS	Basic Disk Operation System (Basissystem zur Diskettenverwaltung)
BIOS	Basic Input Output System (Basis Ein-/Ausgabesystem)
CAOS	Cassette Aided Operating System (die Kassettenarbeit unterstützen des Betriebssystem)
CCP	Console Command Processor
CP/M	Warenzeichen der Firma Digital Research, USA, für ein Betriebssystem
CS	ChipSelect (Schaltkreisauswahlsignal)
CTC	Counter Timer Circuit (Zähler Zeitgeber Schaltkreis)
DAK	DMA-AcKnowledge (DMA-Bestätigungs-Signal)
DEP	Disketten-Erweiterungs-Programm
DMA	Direct Memory Access (direkter Speicherzugriff, hier auch Bezeichnung für einen Pufferspeicher)
DDB	Disk Definition Block (Diskettendefinitionsblock)
DPB	Disk Parameter Block (Diskettenparameterblock)
DPH	Disk Parameter Header (Diskettenparameterkopf)
dRAM	dynamischer RAM (s. RAM)
DRPB	Drive Parameter Block (Laufwerksparameterblock)
DRQ	DMA-ReQuest (DMA-Anforderung)
FCB	Floppy Control Block (Dateisteuerblock)
FD	Floppy Disk
FDC	Floppy Disk Controller (Systemschaltkreis zur Floppy-Disk- Ansteuerung)
GND	GrouND (Masseleitung)
IDX	InDeX (Signalleitung für Spuranfang)
I/O	Input/Output (Ein/Ausgabe)
INT	INTerrupt (Unterbrechungsanforderung)
KC	KleinComputer
LED	Light Emitting Diode (Leuchtdiode)
MFM	Modifizierte Frequenz-Modulation (Diskettenaufzeichnungsverfahren)
MP/M	Warenzeichen der Firma Digital Research, USA, für ein Mehrnutzer-Betriebssystem
NMI	Nicht Maskierbarer Interrupt
RAM	Random Access Memory (Schreib-/Lesespeicher)
RDY	ReaDY (Bereitschaftsmeldung)
ROM	Read Only Memeory (Nur-Lese-Speicher)
SCB	System Control Block (Systemparameterspeicher)
SCP	Betriebssystem des VEB Kombinat Robotron
TC	Terminal Count (Signal des FDC zur Beendigung des DMA- Verkehrs)
TPA	Transient Program Area (Anwenderprogrammspeicher)

