BENUTZERHANDBUCH FÜR DAS DISKETTENBETRIEBSSYSTEM VDOS 2.0

Alle Änderungen vorbehalten

Stand November 1985

vortex Computersysteme
Klingenberg 13
D-7106 Neuenstadt 5

vortex
COMPUTERSYSTEME

INHALTSVERZEICHNIS

Einleitung ..... Seite 1
In eigener Sache ..... Seite 3
Relative Dateien unter Basic ..... Seite 4
Die Basic-Befehle zur relativen Dateiverwaltung ..... Seite 5
FILES ..... Seite 6
OPEN ..... Seite 6
FIELD ..... Seite 7
GET ..... Seite 8
PUT ..... Seite 8
CLOSE ..... Seite 8
Anwendungsbeispiele zur relativen Dateiverwaltung ..... Seite 9
Der romresidente Monitor ..... Seite 11
A(ssemble) ..... Seite 11
B(reakpoints) ..... Seite 12
D(ump) ..... Seite 12
F(ill) ..... Seite 12
G(o) ..... Seite 13
I(input) ..... Seite 13
L(ist) ..... Seite 13
M(ove) ..... Seite 14
O(utput) ..... Seite 14
P(rinter) ..... Seite 14
R(egister) ..... Seite 14
S(ubstitute) ..... Seite 14
T(race) ..... Seite 15
Die verschnellerte Bildschirmausgabe ..... Seite 15
FAST ..... Seite 16
SLOW ..... Seite 16
FRAME ..... Seite 17
GCHAR ..... Seite 17
GPAPER ..... Seite 18
GPEN ..... Seite 18
MASK ..... Seite 19
UNMASK ..... Seite 19
Der Schneider CPC als "Turn-Key-System" ..... Seite 20
128 Directory-Einträge ..... Seite 20
Abfangen von Diskettenfehlern ..... Seite 20
DERROR ..... Seite 21
ANHANG A - VDOS 2.0 Fehlercodes und ihre Bedeutung ..... Seite 22
ANHANG B - Fehlermeldungen unter CP/M ..... Seite 26
ANHANG C - Bildschirmverschnellerung unter CP/M ..... Seite 26
ANHANG D - Der Monitor unter CP/M ..... Seite 27
ANHANG E - Der Para-Patch ..... Seite 29
ANHANG F - Der Graphic-Master-Patch ..... Seite 32
Die Frage der Kompatibilität ..... Seite 33

DAS ERWEITERTE DISKETTENBETRIEBSSYSTEM VDOS 2.0

Das Betriebssystem VDOS 2.0 - eine Erweiterung des bereits außerordentlich mächtigen Betriebssystems VDOS 1.0 - läßt, was die Anforderungen an ein Diskettenbetriebssystem betrifft, wohl kaum mehr Wünsche offen.

Sowohl der BASIC-Programmierer, als auch der Maschinensprach-Profi kann jetzt in völlig neue Regionen der diskettengestützten Programmentwicklung auf dem Schneider Computer vorstoßen.

Zusätzlich zum vollen VDOS 1.0 Befehlsumfang wurden folgende völlig neuen Features eingebaut:

- **relative Dateien unter BASIC**
- **romresidenter Maschinensprach-Monitor**
- **schnellere Bildschirmausgabe**
- **neue BASIC-Befehle**
- **'Turn-Key-System'**
- **128 Directory-Einträge in BASIC und CP/M**
- **Abfangmöglichkeit von Diskettenfehlern.**
- **direkte Stringeingabe bei allen RSX-Kommandos (der lästige @ entfällt.)**

Diese schaffen, vor allem durch die Möglichkeit der relativen Dateiverwaltung und der damit möglichen Datenbankstrukturen, erstmals unter BASIC eine absolut professionelle Programmierumgebung. Unter Einbeziehung der Fehlerabfangmöglichkeit und der 'Turn-Key-Option' (d.h. Gerät einschalten und automatisches Durchstarten zu jeder gewünschten Betriebssystemebene) stößt der CPC mit einer vortex Floppy Diskstation jetzt entgültig in den Small Business Sektor vor.

Um ein möglichst breites Verständnis der zusätzlichen VDOS 2.0 Features zu erhalten, sollen diese nun der Reihe nach erläutert und auch anhand von Beispielen erklärt werden. Voraussetzung für eine erfolgreiche Benutzung ist hierbei nicht unbedingt die absolute Kenntnis von VDOS 1.0.

Testen Sie am Besten unser VDOS 2.0 direkt vor Ort am Gerät, denn nur so werden Sie am schnellsten mit dessen ungeahnten Möglichkeiten vertraut. Sie werden erstaunt sein, wie schnell auch Sie, selbst wenn Sie nur über durchschnittliche Programmierkenntnisse verfügen, professionelle Software erstellen können.

WICHTIGE HINWEISE: Die meisten BASIC-Diskettenoperationen sind unter VDOS 2.0 etwas langsamer als unter VDOS 1.0. Dies liegt daran, daß VDOS 2.0 128 Directoryeinträge zu verwalten hat, und daher das Absuchen der Directoryspur z.B. bei einem LOAD Kommando naturgemäß länger dauert.

Da relative Dateien auf der Diskette völlig anders abgelegt werden als sequentielle Dateien, stimmt die Dateigrößenangabe bei CAT unter BASIC nicht mit der eigentlichen Dateigröße überein.

Relative, wie auch sequentielle Dateien werden blockweise auf der Diskette abgelegt. Bei einer Blockgröße von 4KB werden in den meisten Fällen mehrere Records in einen Block gehen. Dies ist mit einer der Hauptgründe, weshalb das Dateiende nicht über den Fehler 55 (siehe Anhang) abgefragt werden kann.

Der vortex Diskmanager PARA ist in der ursprünglichen Form unter VDOS 2.0 nicht lauffähig, und muß deshalb geändert (gepatcht) werden. Hierzu befindet sich im Anhang ein kleines Programm, welches Sie mithilfe eines Debuggers (z.B. DDT, dieser befindet sich auf der CP/M Systemdiskette) eingeben müssen.

Dasselbe gilt für den "GRAPHIC MASTER 2.0". Siehe Anhang.

Muß an einen externen Befehl (RSX) eine Variable vom Typ String übergeben werden, so mußte dies bisher in der schwerfälligen Form ...,@a$,.. erfolgen. Dies ist auch weiterhin gültig. Zusätzlich könnnen Strings nun aber auch direkt übergeben werden ...,"string",...

Beispiel: a$="datei.ext":|era,@a$ aber auch
|era,"datei.ext"

Dies gilt im Übrigen für alle RSX-Kommandos, nicht nur für die aus dem Floppy-Controller, sondern z.B. auch für die externen Befehle unserer RAM-Erweiterung usw. .

IN EIGENER SACHE

Wie Sie sich sicher vorstellen können, handelt es sich bei einem Diskettenbetriebssystem der Kategorie unseres VDOS 2.0 um eine außerordentlich komplizierte Angelegenheit - Fehler sind praktisch nicht vermeidbar.

Sollten Ihnen bei der Benutzung Unregelmäßigkeiten oder Fehler auffallen, so teilen Sie uns dies bitte doch in einem kleinen Fehlerbericht möglichst umgehend mit. Wir werden dann so schnell wie möglich Abhilfe schaffen und Sie natürlich auch belohnen.

Vielen Dank für Ihr Verständnis

Ihr vortex Team

Neuenstadt im November 1985

VDOS ist eingetragenes Warenzeichen der vortex Computersysteme Vertriebs GmbH.

CP/M ist eingetragenes Warenzeichen der Digital Research Inc.

VDOS 2.0 ist urheberrechtlich geschützt (Copyright). Alle Rechte an diesem Programm liegen bei der Firma vortex Computersysteme GmbH.

Vervielfältigung oder Weitergabe des Programms und des Handbuchs - auch nur auszugsweise - sind untersagt und werden strafrechtlich verfolgt.

vortex übernimmt keinerlei Garantie, was die Eignung von VDOS 2.0 für bestimmte Anwendungen betrifft.

Alle Änderungen vorbehalten.

DIE VERWALTUNG VON RELATIVEN DATEIEN UNTER BASIC

Bisher konnten unter dem BASIC des CPC's maximal zwei sequentielle Diskettenkanäle geöffnet werden. (Sequentiell heißt soviel wie hintereinander und bedeutet in diesem Zusammenhang, daß Daten nur hintereinander abgespeichert und wieder ausgelesen werden können.)
Eine weitere Einschränkung resultiert daraus, daß jeder Diskettenkanal entweder zum Laufwerk hin oder nur vom Laufwerk weg betrieben werden kann - ein gleichzeitiges Einlesen und Abspeichern auf nur einem Kanal ist nicht möglich.
Diese Dateistruktur können Sie unter VDOS 2.0 aus Kompatibilitätsgründen auch weiterhin verwenden. Sie können nun aber auch noch bis zu 16 zusätzliche Disketten-Ein-/Ausgabekanäle öffnen, wobei auf jedem Kanal ein völlig freier Datenzugriff möglich ist. (d.h. Sie können gleichzeitig mit 16 relativen Dateien arbeiten.) Sie haben damit z.B. die Möglichkeit, in einer Lagerverwaltungsdatei den Bestand eines Artikels direkt abzufragen, können diesen dann ändern, einen beliebigen anderen Artikel und seinen Bestand anschauen, dann wieder zum letzten Artikel zurückkehren etc. und dies alles mit nur einmaligem öffnen einer einzigen relativen Datei. Die Funktionsweise solcher relativen Dateien soll nun erklärt werden.

Relative Dateien bestehen aus unabhängigen Datensätze, oft auch Records genannt. Records können in beliebiger Reihenfolge gelesen und/oder beschrieben werden. Die Länge eines Records ist weitgehend frei wählbar und wird nur durch den verfügbaren RAM-Speicher begrenzt, da ein sogenannter Recordpuffer angelegt wird. Während eine relative Datei geöffnet ist kann die Recordlänge nicht geändert werden.

Jede geöffnete relative Datei benötigt für ihren Recordpuffer, für ihren FCB (File Control Block) und etliche Pointer/Counter Speicher im RAM (alles in allem 90 Bytes + Puffer pro Kanal). Dieser Speicher wird bei der Definition des Kanals (siehe FILES) im oberen Speicherbereich durch dynamisches Herabsetzen des BASIC HIMEMs reserviert. Wurde ein Kanal definiert, so kann eine relative Datei auf ihm geöffnet werden (siehe OPEN). Es ist zu beachten, daß, sowohl bei der Kanaldefintion (FILES), als auch beim öffnen einer Datei auf diesem Kanal (OPEN), Recordlängen angegeben werden müssen. Dies hat folgenden leicht einsehbaren Grund: Es ist durchaus möglich auf einem Kanal hintereinander mehrere relative Dateien zu öffnen und zu schließen, ohne daß jedesmal ein Kanaldefinition (FILES) erfolgen muß. Man gibt nur bei der einmaligen Kanaldefinition die größte zu erwartende Recordlänge an und öffnet dann später die einzelnen Dateien (OPEN) mit ihren jeweils individuellen Recordlängen.

Nachdem eine Datei geöffnet (OPEN) wurde, muß der Recordpuffer vor dem ersten Zugriff (GET, PUT) auf die Datei strukturiert werden (FIELD), d.h. man muß angeben, welche Teile des Records beim Zugriff in Variablen eingelesen werden sollen. Jetzt ist die relative Datei für den Datentransfer bereit und man kann bis zum Schließen der Datei wahlfrei Records lesen und schreiben. Es ist

zu beachten, daß ausschließlich Stringvariablen in Records geschrieben und wieder ausgelesen werden können. Will man numerische Variablen oder Felder (auch Stringfelder!) in relativen Dateien speichern, so muß man diese zuerst in Stringvariablen umwandeln bzw. einer Stringvariablen zuweisen (siehe Benutzerhandbuch zum CPC, Kapitel 8 Seite 46, BASIC-Befehl STR$) und diese dann abspeichern. Will man z.B. wieder einen numerischen Wert rückgewinnen, so muß er nach dem Auslesen aus der Datei, mittels des BASIC-Befehls VAL (siehe Benutzerhandbuch zum CPC, Kapitel 8 Seite 49) wieder umgewandelt werden.

Ein Record kann, nachdem eine Datei geöffnet wurde, durchaus mehrmals neu segmentiert (FIELD) werden, ohne daß die Datei jedesmal geschlossen und wieder neu geöffnet werden müßte.

DIE BEFEHLE ZUR RELATIVEN DATEIVERWALTUNG

Die vortex BASIC-Befehlserweiterungen beginnen alle mit dem senkrechten Strich "|", welcher durch gleichzeitiges Drücken der SHIFT-Taste und der Taste mit dem "Klammeraffen" erzeugt wird. In den Zeichen / ... / eingeschlossene Teile der Befehlserklärung sind optional, können also je nach Bedarf weggelassen werden.

Folgende Abkürzungen werden bei den Befehlsbeschreibungen verwendet:

n	Numerischer Wert.
var$	String-Variable (die Variable muß zuvor als Dummy angelegt werden, z. B. var$="").
chn	Kanalnummer (der Wert darf nur zwischen 0 und 127 je einschließlich liegen!).
rec	Record-Nummer.
rl	Record-Länge.
vl	Variablen-Länge

WICHTIG: Muß an einen externen Befehl (RSX) eine Variable vom Typ String übergeben werden, so mußte dies bisher in der schwerfälligen Form ...,@a$,.. erfolgen. Dies ist auch weiterhin gültig. Zusätzlich könnnen Strings nun aber auch direkt übergeben werden ...,"string",...
Beispiel:

```
a$="datei.ext":|era,@a$        aber auch
|era,"datei."
```

Dies gilt im Übrigen für alle RSX-Kommandos, nicht nur für die aus dem Floppy-Controller, sondern z.B. auch für die externen Befehle unserer RAM-Erweiterung usw..

FILES

Format: !FILES /,chn1,rl1 /,chn2,rl2 /,...,...///

Funktion: Reservieren von Arbeitsspeicher für eine/mehrere relative Dateien.

Bemerkung: Bevor mit einer relativen Datei gearbeitet werden kann, muß hierzu ein Kanal und ein Recordpuffer definiert werden.
Das öffnen einer relativen Datei ohne vorhergehendes Reservieren eines Puffers für die Kanalnummer mit dem FILES-Befehl ergibt die Fehlermeldung FELD NICHT DEFINIERT. Aus diesem Grund sollte (muß aber nicht!) der FILES-Befehl am Anfang eines BASIC-Programmes stehen und für alle später verwendeten Kanalnummern ein ausreichender Recordpuffer reserviert werden.
Der FILES-Befehl arbeitet mit Zahlen-Doubletten, d.h. zu jeder angegebenen Kanalnummer muß eine maximale Recordlänge angegeben werden. Die Kanalnummer darf Werte von 0 bis 127 annehmen, die maximale Recordlänge kann beliebig gewählt werden und ist nur durch den zur Verfügung stehenden Arbeitsspeicher begrenzt.
Jeder weitere FILES-Befehl macht den zuvor erfolgten hinfällig. Wird nur FILES ohne Zahlen-Doubletten übergeben (d.h. !FILES und ENTER), so werden alle Arbeitsbereiche von relativen Dateien im RAM gelöscht.

Beispiel: !FILES,1,128,2,64,12,20 reserviert einen 128 Byte großen Puffer für Kanalnummer 1, einen 64 Byte großen Puffer für Kanalnummer 2 und einen 20 Byte großen Puffer für Kanalnummer 12. Es können jetzt die Kanalnummern 1, 2 und 12 als Dateien geöffnet werden.

OPEN

Format: !OPEN,"datei.ext",rl,chn
!OPEN,@name$,rl,chn

Funktion: öffnen einer relativen Datei.

Die Variable name$ muß bereits vor dem öffnen der Datei deren Namen enthalten. Danach muß man noch eine aktuelle Recordlänge und die der Datei zuge-

wiesene Kanalnummer angeben. Die Datei wird dann auf der Diskette geöffnet.
Bei der aktuellen Recordlänge und der Kanalnummer ist Folgendes zu beachten:
Die aktuelle Pufferlänge kann sich von der im FILES-Befehl angegebenen maximalen Pufferlänge insofern unterscheiden, daß sie kleiner, aber nie größer als diese sein darf. Der Kanalnummer muß vorher ein maximaler Puffer mit dem FILES-Befehl zugewiesen werden und die Kanalnummer darf keiner schon zuvor geöffneten Datei zugeordnet sein.

Beispiel: name$="TEST.REL":|OPEN,@name$,128,1 oder
|OPEN,"datei.ext"128,1 |ffnet auf Kanal 1 eine relative Datei TEST.REL mit einer aktuellen Pufferl{nge von 128 Bytes. Der Kanal und sein zugeh|riger Puffer m}ssen zuvor mit dem FILES-Befehl definiert worden sein.

****
* FIELD *
****

Format: |FIELD,chn1,vl1 / ,vl2 / ,... //

Funktion: Strukturierung des Recordpuffers.

Bemerkung: Nachdem eine relative Datei geöffnet wurde, muß deren Puffer segmentiert werden, d.h. man muß angeben wieviele Variablen mit welcher Länge in einem Record der Datei stehen sollen.

Beispiel: Man will in einen Record folgende drei Variablen ablegen: name$ mit der maximalen Länge von 12 Bytes, ort$ mit der maximalen Länge von 10 Bytes und tel$ mit der maximalen Länge von 15 Bytes. Den vorher im OPEN-Befehl angegebenen aktuellen Puffer strukturiert man mit dem FIELD-Befehl nun wie folgt: |FIELD,12,10,15. Die aktuelle Pufferlänge im OPEN-Befehl müßte 12+10+15=37 oder größer gewesen sein.
Beachten Sie bitte, daß eine Stringvariable im BASIC des CPC maximal 255 Stellen lang sein darf und daher vl bei der Segmentierung nur einen Wert zwischen 1 bis 255 annehmen kann/darf.

****
* GET *
****

Format: |GET,chn,rec,@Var1 /,@Var2 /,...//

Funktion: Lesen eines Records.

Bemerkung. Dieser Befehl liest Ihnen den Record mit der Nummer rec aus dem zugewiesenen Kanal in die angegebenen Variablen, die vorher definiert sein müßen. Wurden die Variablen nicht definiert, so meldet BASIC ein 'improper argument'.

Beispiel: |GET,1,173,@name$,@ort$,@tel$ liest Record 173 aus Kanal 1 in die drei Variablen name$, ort$, tel$ nach Maßgabe der Recordsegmentierung ein. FILES-, OPEN- und FIELD-Befehl müssen zuvor ordnungsgemäß angewandt und die drei Variablen müßen zuvor als Dummys definiert worden sein (z.B. durch name$="":ort$="":tel$="").

****
* PUT *
****

Format: |PUT,chn,rec,"aaa" /,"bbb" /,...//
|PUT,chn,rec,@Var1 /,@Var2 /,...//

Funktion: Schreiben eines Records.

Bemerkung: PUT arbeitet genau wie der GET-Befehl, nur schreibt er die angegebenen Variablen in den Record der Kanalnummer.

Beispiel: |PUT,1,142,"datei.ext","ort","tel" oder
|PUT,1,142,@name$,@ort$,@tel$ schreibt die Variablen name$, ort$ und tel$ in den 142. Record des Kanals Nr.1 .

****
* CLOSE *
****

Format: |CLOSE,chn

Funktion: Schließt die Datei auf dem angegebenen Kanal und dieser wird frei für eine andere Datei.

Beispiel: |CLOSE,1 schließt die Datei mit der Kanalnummer 1.

Anwendungsbeispiele zu den relativen Datei-Befehlen.

Als nächstes folgen nun die Listings zweier kleiner Programme, die Ihnen das Arbeiten mit relativen Dateien nochmals verdeutlichen sollen. Es wird bei diesem Beispiel die Datei TEST.REL geöffnet und es wird ein selektierbarer Record gelesen und angezeigt oder es wird ein Record mit eingebbarem Inhalt beschrieben.

HINWEIS: Überall, wo String-Variablen definierten Inhalts übergeben werden, kann man diese auch direkt in das externe Kommando einsetzen, z.B. siehe unten Zeile 30

statt 30 !OPEN,@a$,520,1 können Sie auch
30 !OPEN,"test.rel",520,1 schreiben.

Listing	Kommentar
10 !FILES,1,600	Einrichten eines Puffers von 600 Bytes Länge für den Kanal Nummer 1.
20 a$="TEST.REL"	Name der Datei in a$.
30 !OPEN,@a$,520,1	Eröffnen der Datei TEST.REL mit der aktuellen Recordlänge von 520 Bytes.
40 !FIELD,1,250,250	Es werden nur zwei Variablen in einen Record geschrieben, d. h. der Puffer muß entsprechend strukturiert werden.
50 x$="":y$=""	Die verwendeten Variablen müßen deklariert werden.
60 ?"(L)esen oder (S)chreiben eines Records oder (E)nde?"	Bei Eingabe von L oder S wird ein Record gelesen oder geschrieben, bei E wird die Datei geschlossen.
70 k$=upper$(inkey$):if k$<>"L" and k$<>"S" then 70	Warten auf den richtigen Tastendruck.
80 input "Recordnummer",rec	Eingabe der Recordnummer.
90 if k$<>"L" then 120	Der Fall des Lesens wird behandelt.
100 GET,1,rec,@x$,@y$	Lesen des Records in die beiden
110 ?x$:?y$:goto 60	Variablen und ausdrucken.
120 if k$<>"S" then 170	Der Fall des Schreibens wird behandelt.
130 input "1. Variable",x$	Eingabe der zwei Variablen.
140 input "2. Variable",y$	
150 !PUT,1,rec,@x$,@y$	Schreiben der zwei Variablen in den selektierten Record.
160 goto 60	Weiter mit der Eingabeschleife.
170 !CLOSE,1:end	Schließen der Datei und Programmende.

Das zweite Beispiel schreibt nacheinander 1000 Records in die relative Datei ZUFALL.REL, liest danach einen zufälligen Record und zeigt dessen Inhalt an.

Listing	Kommentar
10 !FILES,10,120	Reservieren von 120 Bytes Puffer für Kanalnummer 10.
20 a$="zufall.rel"	öffnen der relativen Datei ZUFALL.REL mit der Record-
30 !OPEN,@a$,25,10	länge von 25 Bytes.
40 !FIELD,10,25	Segmentierung des Puffers. Es wird nur ein String abgelegt.
50 for rec=1 to 1000	Es werden 1000 Records mit dem
60 x$="Record Nr."+str$(i)	Inhalt "Record Nr.1" bis
70 !PUT,10,rec,@x$	"Record Nr.1000" abgelegt.
80 next rec	
90 rec=int(1000*rnd(1))+1	Die Variable rec hat einen zufälligen Wert zwischen 1 und 1000.
100 !GET,10,rec,@x$	Dieser zufällige Record wird
110 ?x$:goto 90	gelesen und angezeigt.

DER ROM-RESIDENTE MONITOR

Der eingebaute Monitor, mit dem Maschinensprachprogramme geladen, getestet und wieder abgespeichert werden können, ist ein leistungsfähiges Werkzeug für Programmierer, das sofort und ohne wesentlichen Speicherplatzverlust zur Verfügung steht. Er beinhaltet vielerlei Funktionen, wie z. B. Disassemblieren, Anzeigen und Ändern von Speicher- und Registerinhalten, Setzen von Breakpoints, Einzelschrittausführung, etc. Diese werden desweiteren ausführlich erklärt. Der Monitor wird mit dem RSX-Befehl |M aus BASIC und mit den transienten Befehl M aus CP/M heraus gestartet (siehe Anhang D) und meldet sich mit dem Zeichen * und einem einem blinkendem Cursor. Das *-Zeichen wird im folgendem als Prompt bezeichnet. Sie können den Monitor durch Drücken der ESC-Taste immer dann verlassen, wenn dieser Prompt sichtbar ist und befinden sich danach an der Stelle nach dem Aufruf (d.h. der Monitor kann auch aus einem BASIC-Programm heraus aufgerufen werden und das BASIC-Programm läuft nach der Rückkehr weiter).

Die Befehle des Monitors

Nachfolgend werden in alphabetischer Reihenfolge die Befehle des Monitors erklärt. Zu beachten ist dabei, daß geforderte oder angezeigte Werte grundsätzlich in hexadezimaler Form dargestellt werden, wobei 8-Bit-Werte zweistellig und 16-Bit-Werte vierstellig sind, bzw. sein müßen.

A adresse - Zeilenassembler

Mit dem Zeilenassembler können Sie Z80-Befehle direkt in lesbaren Mnemonics eingeben, der Befehl wird dann sofort übersetzt und ab der angegebenen Adresse in den Speicher eingetragen. Sie können dann ab der nächsten Adresse nach dem letzten übersetzten Befehl den nächsten Z80-Befehl eingeben, können aber auch mit der ESC-Taste wieder zum Monitor-Prompt * zurückkehren. Bei der Benutzung des Zeilenassemblers ist allerdings zu beachten, daß keine Labels (wie bei größeren Assemblern) verwendet werden können, sondern nur fixe Adressen und daß Zahlenwerte immer in hexadezimaler Form mit führendem &-Zeichen eingegeben werden müssen.

Beispiele:

```
A2400 <ENTER>          Benutzung des Assemblers ab Adresse 2400H
2400 LD A,&65          in Adressen 2400 u. 2401 steht 3E und 65
2402 LD H,A <ENTER>    in Adresse 2402 steht 67.
2403 EX DE,HL <ENTER>  in Adresse 2403 steht EB.
2404 JR &2400 <ENTER>  in Adressen 2404 u. 2405 steht 18 und FA.
2405 <ESC>             Abbruch des Zeilenassemblers und
*                      Rückkehr zum Monitor-Prompt.
```

B - Breakpoints ansehen/ändern

Der Monitor bietet die Möglichkeit bis zu acht Breakpoints (=Unterbrechungspunkte) beliebig zu setzen. Trifft der Monitor beim Ablauf eines Programmes (siehe G-Befehl) auf einen dieser Breakpoints, so wird der Programmablauf unterbrochen und der Monitor-Prompt * ausgegeben. Die Ausgabe der Breakpoints erfolgt derart:

```
-B1-   -B2-   -B3-   -B4-   -B5-   -B6-   -B7-   -B8-
0000   0000   0000   0000   0000   0000   0000   0000
```

Sie können dann den ersten Breakpoint ändern oder mittels <ENTER> zum nächsten Breakpoint gelangen, usw. bis nach dem achten Breakpoint oder durch Drücken von <ESC> der Monitor-Promt * wieder erscheint.

D adresse1,adresse2 - Auflisten in hexadezimalem und ASCII-Format

Der D-Befehl gibt Ihnen die Möglichkeit Speicherinhalte in folgendem Format anzuzeigen:

```
0100  00 00 41 00 33 57 00 00 00 51 00 35 00 00 00 00  ..).19...3.#....
```

Am Anfang der Zeile wird Ihnen die Adresse der momentanen Speicherzelle angezeigt und darauf die Inhalte dieser und der fünfzehn folgenden Speicherzellen in hexadezimaler Form. Danach erfolgt eine Interpretation der Werte als ASCII-Zeichen; ist der Wert kleiner als 32 oder grösser als 127, so wird ein Punkt, ansonsten aber das entsprechende ASCII-Zeichen ausgegeben. Geben Sie nach Drücken der Taste D nur eine Adresse ein, so werden die nachfolgenden 100H Speicherinhalte ausgegeben, drücken Sie allerdings nach der ersten Adresse nicht <ENTER>, sonder die Komma-Taste, so können Sie eine zweite Adresse eingeben. Es werden dann von der ersten Adresse bis zur zweiten Adresse die Speicherinhalte aufgelistet. Eine solche Auflistung können Sie durch Drücken einer beliebigen Taste kurzfristig anhalten. Drücken Sie nun <ESC>, so erscheint wieder der Monitor-Prompt *, drücken Sie aber eine beliebige andere Taste, so läuft das Listing weiter.

F adresse1,adresse2,wert - Auffüllen eines Speicherbereichs mit einem Wert

Oft kommt es vor, daß bestimmte Speicherbereiche z.B. gelöscht werden sollen. Dies ist mit dem F-Befehl möglich, wobei von adresse1 bis adresse2 die Speicherinhalte mit der 8-Bit-Konstanten wert belegt werden.

Beispiel:

```
FC000,FFFF,FF <ENTER>   füllt den Bildschirmspeicher (Adressen
*                       C000 bis FFFF mit dem Wert FF.
```

G adresse - Ausführen eines Maschinensprachprogrammes

Dieser Befehl führt ein ab der eingegebenen Adresse befindliches Maschinen-sprachprogramm aus. Wird hierbei ein Breakpoint (s.o.) erreicht, so geht die Kontrolle an den Monitor zurück.

Beispiel:

```
GBD19 <ENTER>        führt das Maschinensprachprogramm ab der
                     Adresse BD19 aus.
```

I filename - Laden einer Binärdatei von Diskette/Cassette

Wurde der Monitor aus BASIC heraus aufgerufen, so kann mit diesem Befehl vom selektierten Eingabegerät (Diskette oder Cassette) eine Binärdatei geladen werden. Sie wird dabei in den originalen Speicher, d.h. in den Speicherteil aus dem sie erstellt wurde, geladen. Es werden die Ladeadresse, die Länge, sowie eine eventuell vorhandene Startadresse angezeigt.
Beispiel:

```
ITEST.BIN <ENTER>          lädt die Datei TEST.BIN in den Speicher.
<2000  0380  2140>         Die ermittelte Anfangsadresse ist 2000,
*                          die Länge 380 und die Startadresse 2140.
```

Wollen Sie eine COM-Datei unter CP/M bearbeiten, so müßen Sie diese Datei schon bei dem Aufruf des Monitors in den Speicher laden. Die geladene COM-Datei befindet sich dann ab Adresse 100H.

Beispiel:

```
A>M TEST <ENTER>           lädt die Datei TEST.COM ab Adresse 100H
*                          in den Speicher und startet den Monitor
```

L adresse1,adresse2 - Disassemblieren eines Speicherbereiches

Die Eingabe und Bedeutung der Adressen entspricht denen des D-Befehls (s.o.), wobei allerdings der Speicherinhalt in Form von Mnemonics dargestellt wird.

Beispiel:

```
L0100,0106 <ENTER>
0100 3E64       LD    A,64H
0102 ED5B3204   LD    BC,(0432H)
0106 E3         EX    (SP),HL
*
```

Listet den Bereich von 100 bis 106. Am Anfang steht die Adresse, dann deren Inhalt in hexadezimaler Form und dann der lesbare Z80-Mnemonic.

M adresse1,adresse2,adresse3

Der Inhalt des Speicherbereiches zwischen adresse1 und adresse2 wird in den Speicherbereich beginnend mit adresse3 kopiert.

Beispiel:

M0000,3FFF,C000 <ENTER> kopiert den Bereich von 0 - 3FFF in den
* Speicherbereich ab Adresse C000 (=Bildschirm).

O filename <ENTER> adresse1,länge,adresse2
- Abspeichern eines Speicherbereiches

Wenn Sie nach dem O einen zulässigen Filenamen eingegeben haben, wird endsprechend dem BASIC-Befehl SAVE "name",B nach Anfangsadresse des Speicherbereiches, dessen Länge und einer Startadresse gefragt. Danach wird dieser Speicherbereich auf dem selektierten Ausgabegerät (Diskette oder Cassette) abgespeichert.

Beispiel:

OTEST.BIN <ENTER> C000,4000,0000 Es wird in die Binärdatei
* TEST.BIN der Speicherbereich von C000 bis FFFF ohne Startadresse abgespeichert.

Wenn Sie unter CP/M mit dem Monitor arbeiten und ein ab Adresse 100H stehendes Programm als COM-Datei abspeichern wollen, so müssen Sie den Monitor verlassen und mit dem CP/M-Befehl SAVE den entsprechenden Speicherbereich auf Diskette abspeichern.

P - Ein-/Ausschalten des Druckers

Wenn ein Drucker angeschlossen ist, wird die Bildschirmausgabe durch einmaliges Drücken der Taste P auf diesem mitdokumentiert. Nochmaliges Drücken der Taste P schaltet den Drucker wieder ab.

R - Anzeigen/Ändern der Registerinhalte

Die Registerinhalte für die Abarbeitung eines Maschinensprachprogrammes (mit G oder T) sind mit diesem Befehl änderbar. Sie werden ähnlich den Breakpoints dargestellt und genauso editiert. Es ist nur der Erst-Registersatz änderbar.

S adresse - Speicherinhalt anzeigen/ändern

Nach Eingabe von S und einer Adresse sehen Sie deren Inhalt und können diesen in einer dritten Spalte entsprechend ändern. Durch Drücken von <ENTER> gehen Sie weiter zur nächsten Speicherzelle,

der eventuell geänderte Wert der alten Adresse wird in diese eingetragen. Durch drücken von <ESC> kommen Sie wieder zum Monitor-Prompt *.

Beispiel:

```
S4000 <ENTER>          ändern ab Speicherzelle 4000.
4000 $ 24 34 <ENTER>   Die alten Werte der Adresse 4000, 4001
4001 C 43 32 <ENTER>   und 4002 (24, 43, und 58) werden in die
4002 : 58 44 <ENTER>   neuen Werte 34, 32 und 44 abgeändert.
4003 <ESC>
*
```

T adresse,anzahl - Maschinenprogramm schrittweise ausführen.

Ein Maschinenprogramm kann mit diesem Befehl schrittweise ausgeführt werden, um eventuell vorhandene Fehler zu entdecken. Es wird dabei von dem mit dem R-Befehl eingegebenen Registerwerten ausgegangen und die Kontrolle wird nach jedem Schritt an den Monitor zurückgegeben, d.h. Sie sehen die eventuell geänderten Register und können die Abarbeitung auch unterbrechen, wenn Sie eine optionelle Schrittanzahl angegeben haben. Diese Unterbrechung erreichen Sie wie gewohnt mit ESC. Beachten Sie bitte, daß Programme im ROM oder Befehle die mit dem Zweitregistersatz des Z80 arbeiten nicht schrittweise abgearbeitet werden können. Desweiteren sollte der Stackpointer SP nicht unter einem ROM liegen, was zum Absturz des Systems führen kann.

Beispiel:

```
T4000,0007 <ENTER>
```

führt sieben Schritte ab Adresse 4000 aus. Die Registerinhalte werden immer angezeigt und der Ablauf kann wie beim D-Befehl angehalten oder abgebrochen werden.

DIE VERSCHNELLERTE BILDSCHIRMAUSGABE

Ein wesentliches Manko des CPC-Rechners war bislang die relativ langsame Bildschirmausgabe von Texten. Diese Textausgabe haben wir für den Bildschirm-Mode 2 neu geschrieben und in das VDOS 2.0 implementiert. Die Ausgabe als solche wurde ca. um den Faktor 4 verschnellert (siehe Monitor!), unter BASIC bleibt ca. Faktor 2 und unter CP/M ca. der Faktor 1.7 übrig.

Da die verschnellerte Bildschirmausgabe keine Windowprogrammierung zulässt, startet das CP/M defaultmäßig mit der normalen Bildschirmausgabe. Hierdurch wird z.B. die Funktionsfähigkeit des Graphik Turbo Pascals usw. gewährleistet.

Die Verschnellerung wird unter CP/M durch das kleine Programm FAST.COM ein- bzw. auch ausgeschaltet. (siehe Anhang C).
Für die Benutzung der Verschnellerung unter BASIC gibt es die beiden folgenden RSX-Befehle.

****
* FAST *
****

Format: |FAST

Funktion: verschnellerte Bildschirmausgabe im Mode 2

Bemerkung: Wenn Sie sich in der 80-Zeichen-Bildschirmausgabe (Mode 2) befinden, haben Sie die Möglichkeit, die Ausgabe von Text um den Faktor 2 zu beschleunigen. Der einzige Nachteil, den Sie in Kauf nehmen müssen, ist eine nicht mehr einwandfrei funktionierende Window-Technik. Bei normaler Textausgabe ohne definierte Windows arbeitet die verschnellerte Bildschirmausgabe vollgültig wie die normale Ausgabe. Ein Wechseln des Bildschirm-Modes bei eingeschalteter Fast-Routine führt allerdings zu einer inkorrekten Bildschirmausgabe!

Beispiel: Als Demonstration für die Verschnellerung können Sie folgenden "Einzeiler" eingeben:

```
10 z=z+1:?z:goto 10
```

Starten Sie das Programm mit RUN, so sehen Sie die normale Ausgabe. Unterbrechen Sie das Programm nun, tippen |FAST <ENTER> und starten das Programm wieder mit RUN <ENTER> , so werden Sie die drastische Geschwindigkeitserhöhung sofort feststellen können.

****
* SLOW *
****

Format: |SLOW

Funktion: Abschalten der verschnellerten Textausgabe.

Bemerkung: Wurde mittels des |FAST-Befehls die verschnellerte Textausgabe eingeschaltet, so kann mit diesem Befehl die normale Textausgabe wieder selektiert werden.

DIE ZUSÄTZLICHEN BASIC-GRAPHIKBEFEHLE

FRAME

Format: |FRAME

Funktion: erzeugen von flackerfreier bewegter Graphik.

Bemerkung: Dadurch das die Ausgabe auf dem Bildschirm normalerweise nicht mit dem physikalischem Bildschirmaufbau des Monitors synchronisiert wird, kann es in bestimmten Fällen zu einem flackerndem Bildschirmaufbau kommen. Um dies zu vermeinden ist der |FRAME-Befehl zu verwenden. Er entspricht im Übrigen dem Befehl CALL &BD19.

Beispiel: Als Beispiel geben Sie bitte folgendes kurzes Programm ein:

```
10 mode 2
20 for i=1 to 80
30 locate i,12:?"o":locate i,12:?" "
40 next
50 goto 20
```

Wenn Sie das Programm nun starten, wird das Zeichen o relativ ruckhaft von links nach rechts bewegt. Fügen Sie aber die folgende Zeile ein, so werden Sie eine vollkommen fließende Bewegung sehen. Die Zeile lautet:

```
35 |FRAME
```

GCHAR

Format: |GCHAR,x,y,@var

Funktion: Lesen eines Zeichens aus dem Bildschirm.

Bemerkung: Wenn Sie beispielsweise feststellen wollen, welches Zeichen sich an der Position 12,14 auf dem Bildschirm befindet, so können Sie dies mit diesem Befehl ermitteln. Sie müßen dazu zunächst eine Variable definieren, in die der ASCII-Wert des ermittelten Zeichens dann eingetragen wird. Diese Variable muß eine Integer-Variable sein, d.h. bevor Sie den |GCHAR-Befehl benutzen, müßen Sie

beispielsweise die Variable A wie folgt definieren:

10 defint a:a=0

Benutzen Sie danach den Befehl !GCHAR,x,y,@a dann wird der ASCII-Wert des Zeichens an den Koordinaten x/y in die Variable a eingetragen. Diesen Wert können Sie dann mit PRINT A ansehen. Bitte beachten Sie, daß unzulässige Koordinaten (y zwischen 1 und 25, x zwischen 1 und dem Maximalwert im jeweiligen Mode) oder durch das Anwenden von Graphikbefehlen nicht mehr identifizierbare Zeichen den Wert 0 zurückmelden.

GPAPER

Format: !GPAPER,wert

Funktion: Einstellen der Graphik-Hintergrundfarbe.

Bemerkung: Mit dem Befehl !GPAPER weisen Sie der Graphik-Hintergrundfarbe einen neuen Farbstift zu. Eine ausführliche Erklärung finden Sie beim !MASK-Befehl.

Beispiel: !GPAPER,3

Der Farbstift für die Graphik-Hintergrundfarbe ist der mit der Nummer 3.

GPEN

Format: !GPEN,wert

Funktion: Einstellen der Graphik-Vordergrundfarbe.

Bemerkung: Mit dem Befehl !GPAPER weisen Sie der Graphik-Vordergrundfarbe einen neuen Farbstift zu. Eine ausführliche Erklärung finden Sie beim !MASK-Befehl.

Beispiel: !GPEN,7

Der neue Farbstift für die Graphik-Vordergrundfarbe ist der Farbstift Nr. 7

MASK

Format: |MASK,wert

Funktion: Definition einer Maske für die Graphikausgabe.

Bemerkung: Punkte und Linien werden am CPC normalerweise nur in einer Farbe ausgegeben. Mit Hilfe des |MASK-Befehls ist es nun möglich, diese Ausgabe auch zweifarbig zu gestalten. Eine entsprechend mit diesem Befehl übergebene Maske wird dazu benutzt, Linien mit Vorder- und Hintergrundfarbe (siehe |GPEN und |GPAPER) zu mustern. Dabei wird der übergebene Wert als binäres Muster interpretiert, ein gesetztes Bit ergibt einen Punkt in der Vordergrundfarbe und ein nicht gesetztes Bit ergibt einen Punkt in der Hintergrundfarbe.

Beispiel: |MASK,&55

Es wird die binäre Maske 01010101 als Wert übergeben und damit werden im folgenden gezeicnete Linien abwechselnd mit Vorder- und Hintergrundfarbe gemustert.

UNMASK

Format: |UNMASK

Funktion: Aufheben der Graphik-Maskierung.

Bemerkung: Nach Eingabe des |UNMASK-Befehls ist die Graphikausgabe des CPC wieder in ihrer ursprünglichen Form, d.h. jegliche Maskendefinition mittels des |MASK-Befehls wird aufgehoben.

SLOW

Format: |SLOW

Funktion: Aufheben der FAST-Routine.

Bemerkung: Mittels des Befehls |SLOW wird die Textausgabe auf die normale Geschwindigkeit zurückgesetzt und die Window-Technik kann wieder benutzt werden.

Der Schneider CPC als "Turn-Key-System"

VDOS 2.0 bietet Ihnen die Möglichkeit, bei jedem Aus/Einschalten, bei jedem Reset mittels CTRL/SHIFT/ESC und bei jeder Rückkehr aus CP/M ins VDOS automatisch ein Programm zu laden und zu starten.

Dieses Programm muß auf der Diskette unter dem Namen HELLO.BAS oder HELLO.BIN abgespeichert worden sein. Wollen Sie diesen automatischen Ladevorgang unterbrechen, so müßen Sie genügend lange bevor der Recner die Meldung BASIC 1.0 bringt nur eine beliebige Taste drücken (am Besten die ENTER Taste) und der Rechner bricht die Hello-Sequenz ab und meldet sich gleich im READY-Mode. Die Hello-Sequenz wird ebenso übersprungen, wenn sich keine Diskette im Laufwerk befindet.

Als einfaches aber nützliches Beispiel sei folgendes Programm genannt:

```
10 |CPM
```

Dieses müßen Sie als HELLO.BAS auf eine Diskette speichern, mit der Sie nur unter CP/M arbeiten wollen und schon startet der Rechner automatisch das CP/M-Betriebssystem nach jedem Reset.

128 DIRECTORY-EINTRÄGE

Mit VDOS 2.0 haben Sie unter BASIC und unter CP/M 128 Directory-Einträge (VDOS 1.0 hatte nur 64!). Der freie RAM-Speicherplatz verringert sich deshalb um ca. 50 Bytes unter BASIC und ohne Speichererweiterung ist das größtmögliche CP/M eine Seite kleiner (178 Pages).
Bitte beachten Sie, daß beim CAT-Befehl unter BASIC die vier vom Inhaltsverzeichnis belegten KByte auch unter der Sparte 'Belegt' geführt werden, d.h. wenn Sie sich den Katalog einer leeren Diskette anzeigen lassen sind vier KByte belegt!

ABFANGMÖGLICHKEIT VON DISKETTENFEHLERN

Wer hat sich nicht schon geärgert, wenn sein BASIC Programm mittendrin mit der Meldung

```
Laufwerk A: Diskette voll
```

abbracht und sich anschließend der CPC wieder im READY Mode befand. Vor allem für Profianwendungen ist dieser Umstand geradezu das Aus. VDOS 2.0 schafft auch hier Abhilfe und bietet eine Möglichkeit Diskettenfehler direkt abzufangen und das Programm abhängig von der Art und Weise des aufgetretenen Fehlers zu verzweigen.

Hierzu dient der Befehl

```
**************************************
*                DERROR              *
**************************************
```

Format: !derror/,n/

Funktion: Ein/Ausschalten der Diskettenfehlermeldungen
Anlegen einer Fehlervariable

Bemerkung: Nach Eingabe von !DERROR,0 werden keine Diskettenfehler mehr auf dem Bildschirm ausgegeben. !DERROR,1 schaltet die Fehlerausgabe wieder ein. Dies ist auch der Defaultzustand nach dem Einschalten des Rechners.

Sind die Fehlermeldungen ausgeschaltet, so kann durch Eingabe von !DERROR (also ohne Parameter) die Errorcode-Variable DERR angelegt werden. Alle Fehlercodes sind im Anhang aufgeführt.

Beispiel:

```
10 !DERROR,0
20 CAT
30 !DERROR
40 IF DERR=72 THEN INPUT "Bitte Diskette einlegen
und irgendeine Taste druecken ",a:GOTO 20
50 ...
```

Achtung: Bestimmte Fehler können nicht alleine durch !DERROR,0 abgefangen werden, sondern es muß noch zusätzlich mit ON ERROR GOTO gearbeitet werden.

Beispiel: Wir nehmen an, daß sich im folgenden keine Diskette im Laufwerk befindet. Ein Zugriff auf das Laufwerk würde also ohne Fehlerabfang die Meldung

```
Laufwerk d: Diskette fehlt
...
```

erzeugen. Fangen wird diesen Fehler mit DERROR,Ø ab, so meldet BASIC trotzdem einen Fehler. Lassen Sie folgendes Programm einfach einmal laufen.

```
20 !DERROR,0
30 LOAD "TEST"
40 ...
```

Sie werden folgende Fehlermeldung erhalten:
File already open in 30
und das Programm bricht ab.

Testen Sie nun einmal folgendes Programm:

```
10 ON ERROR GOTO 100
20 !DERROR,0
30 LOAD "TEST"
40 ...
100 !DERROR:PRINT DERR:RESUME 40
```

Nun kommt keine Fehlermeldung mehr und die Fehlerabfangroutine in 100 druckt ein 72 auf den Bildschirm. (72=&48 d.h. Bit 6 und Bit 3 sind gesetzt und dies wiederum bedeutet es handelt sich um den FDC-Fehler 'Drive not ready' d.h. es befindet sich keine Diskette im Laufwerk.)

ANHANG A

VDOS 2.0 - Fehlercodes und ihre Bedeutung

Definitionen:

d: steht für eine Laufwerkkenunng also z.b. A:=Laufwerk A usw.
<...> ist Platzhalter für einen Dateinamen
FDC Floppy Disc Controller

Alle Fehlernummern werden dezimal, der Meldetext wird jeweils im Fettdruck angegeben, danach kommt eine kurze Beschreibung der Fehlersymptome und getrennt durch '-' mögliche Fehlerursachen.

Errorcode Meldetext und Bedeutung

17 **Drucker nicht bereit**
Diese Fehlermeldung kann nur unter CP/M kommen und dort immer dann, wenn der Drucker ca. 15 sec nach dem Versuch, ein Zeichen an ihn abzusenden, immernoch 'busy' meldet, also nicht bereit ist.

- Druckerkabel nicht angesteckt
- Drucker nicht eingeschaltet
- Drucker nicht selektiert

10 **Laufwerk d: Diskette wurde gewechselt <...> wird geschlossen**
Es wurde eine Datei mit OPENOUT eröffnet und dann, ohne eine CLOSEOUT zu geben, die Diskette gewechselt.

21 **Laufwerk d: Diskette voll**
Die Diskette ist voll.

22 **Laufwerk d: Directory voll**
Das Inhaltsverzeichnis der Diskette ist voll. Beachten Sie, daß diese Meldung schon kommen kann, bevor Sie z.B. unter CAT auch wirklich 128 Einträge sehen. Grund: pro 64KB Daten/Programm wird ein Directoryeintrag benötigt. Eine Datei, die z.B. 129KB groß ist belegt bereits 3(!) Directoryeinträge.

23 **<...> wurde nicht gefunden**
Wird immer dann gemeldet, wenn versucht wird auf eine Datei zuzugreifen, die im Inhaltsverzeichnis nicht existiert. Der Fehler kann bei folgenden Befehlen auftauchen: LOAD, SAVE, RUN, OPENIN, OPENOUT, ERA, REN, ATTRIBUT.

24 **<...> existiert bereits**
Es wird versucht einer Datei einen neuen Namen zu geben, wobei bereits eine andere Datei mit diesem Namen existiert.

25 **<...> kann nur gelesen werden**
Es wird versucht eine Datei zu löschen, bzw. zu beschreiben, die ein 'Read only' Attribut trägt.

33 **falsche Eingabe**
Es wurde ein Syntax- oder Semantikfehler gemacht.

39 **Feld zu lang**
Die aktuelle Feldlänge ist größer als die im FILES Kommando definierte.

- aktuelle Feldlänge in OPEN zu groß
- Summe der Segmente in FIELD größer als die in OPEN definierte

40 **Feld nicht definiert**

- es wird versucht einen Kanal zu öffnen, ohne daß vorher ein FILES Kommando gegeben wurde

41 **Feld nicht segmentiert**
Es wird versucht ein GET oder PUT Kommando durchzuführen, ohne daß zuvor das Feld mit FIELD segmentiert wurde.

42 **Kanal bereits geoeffnet**
Es wurde versucht, einen bereits mit OPEN geöffneten Kanal nochmals zu öffnen.

43 **Kanal nicht geoeffnet**
Es wurde versucht, ein FIEL, CLOSE, GET oder PUT Kommando an einem nicht geöffneten Kanal durchzuführen.

44 **Kanalnummer falsch**
Es wurde innerhalb einer CLOSE, FIELD, OPEN, GET, PUT, FILES eine unerlaubte Kanalnummer angegeben. Kanalnummern dürfen zwischen 0 und 127 je einschließlich liegen.

45 **Recordnummer zu gross**
VDOS 2.0 kann maximal 65536 Records a 128 Bytes verwalten. Abhängig von der in OPEN definierten aktuellen Feldlänge ändert sich die maximale Recordnummer. z.B. Feldlänge=512 Bytes dann max. Recordnummer=16384

53 **Kanal nicht definiert**
Es wurde versucht einen Kanal mit OPEN zu öffnen, der zuvor im Kommando FILES nicht definiert wurde.

55 **Record enthält keine Daten**
Es wurde versucht einen Record zu lesen, der nicht mehr in der Blockbelegungstabelle ausgewiesen ist. Achtung: Dieser Fehler kann nicht unbedingt dazu benutzt werden, das Ende einer relativen Datei zu testen. Grund: Ist die aktuelle Recordlänge(=Feldlänge) kleiner als die Blocklänge (4KB beim vortex Laufwerk und 1KB beim 3" Laufwerk), so gehen mehrere Records in einen Block. Ist nun z.B. nur der erste Record im letzten in der Blockbelegungstabelle aufgeführte Block noch belegt, so kommt Fehler 55 erst, wenn man versucht einen Record im nächsten Block anzusprechen. Bei kleinen Feldlängen kann sich dessen Recordnummer u.U. stark von der des letzten tatsächlich belegten Records unterscheiden. Man sollte deshalb immer mit Indexdateien arbeiten, also zusätzlich zur Datendatei eine zweite sogenannte Indexdatei mitführen, in der Datenrecordnummern und sonstige Informationen verwaltet werden.

14 kein Meldetext; Datei nicht geöffnet
Es wurde versucht mit CLOSEIN oder CLOSEOUT eine sequentielle Datei zu schließen, die nicht geöffnet war.

15 kein Meldetext; physikalisches Ende einer sequentiellen Datei wurde erreicht.

Errorcodes, die vom FDC direkt kommen haben Bit 6 gesetzt, Bit 7 gelöscht und liegen deshalb zwischen 64 und 127. Die Bedeutung der einzelnen Bits ist folgende:

Bit:	Bedeutung (falls Bit gesetzt):
7	immer 0
6	immer 1
5	CRC Fehler im Daten- oder ID Feld
4	FDC wurde in einem Datenübertragungsprozeß mit dem Rechner innerhalb eines bestimmten Zeitraums nicht bedient.
3	Das Laufwerk meldet 'not ready'. Bedeutet meistens, daß keine Diskette im Schacht ist oder der Auswurfknopf nicht richtig arretiert wurde.
2	FDC kann während der Ausführung einer Leseoperation den gewünschten Sektor nicht finden.
1	FDC erhält während einer Schreib- oder Formatieroperation ein 'write protect' Signal vom Laufwerk. d.h. die Diskette hat einen Schreibschutz.
0	FDC kann nach dem zweiten Überfliegen des Indexloches das ID Feld nicht finden.

Ein häufiger Fehlercode dürfte wohl 72 sein, oder binär 01001000 und bedeutet, daß sich keine Diskette im Laufwerk befindet.

Wurden die Floppy Disk Fehlermeldungen nicht mit DERROR (siehe dort) abgeschaltet, so erfolgt eine Ausgabe auf den Bildschirm. Um dem Programmierer dies anzuzeigen, wird deshalb in diesem Fall das 7. Bit im Errorbyte gesetzt, d.h. 128 zu obigen Fehlercodes addiert.

Beispiel:

Errorcode	Bedeutung	Fehlermeldungen
21	Diskette voll	abgeschaltet
149	Diskette voll	eingeschaltet

WICHTIG: Floppy Fehler, die in der Folge keinen BASIC Fehler generieren können nicht mit ON ERROR GOTO abgefangen werden, sondern es muß jeweils nach dem Kommando auf Fehler abgefragt werden.

Beispiel:

```
10 ON ERROR GOTO 100
20 !DERROR,0 'MELDUNGEN UNTERDRÜCKEN
30 CAT
40 !DERROR 'ANLEGEN DER FEHLER VARIABLE DERR
50 IF DERR=....
....
100 'FEHLER BEHANDLUNG
```

Befindet sich in 30 keine Diskette im Laufwerk, so geht CAT ins "Leere", generiert hierbei aber keine BASIC Fehlermeldung, d.h. verzweigt nicht nach 100. Abgefangen wird der Fehler in Zeile 40, 50.

ANHANG B

Fehlermeldungen unter CP/M

Gegenüber VDOS 1.0 hat sich an der Struktur der Fehlermeldungen unter CP/M nichts wesentliches geändert, mit dem einen Unterschied, daß der Laufwerksmotor während der Meldung/Abfrage ausgeschaltet wird. So ist es jetzt z.B. möglich, wenn des Betriebssystem meldet

Laufwerk A: Diskette hat Schreibschutz - Wiederholen J/N ?

die Diskette herauszunehmen, den Schreibschutz zu entfernen, die Diskette wieder einzulegen und die Frage mit J(a) zu beantworten.

ANHANG C

Bildschirmverschnellerung unter CP/M

Erstellen Sie sich zum Ein/Aussschalten der Bildschirmverschnellerung die Kommandodatei FAST.COM wie folgt:

- Starten des 8080 Debuggers DDT.COM
- Aufruf des Zeilenassemblers: Kommando A
- Eingabe des Programms
- Verlassen des Debuggers und Absichern der Datei durch SAVE 1 FAST.COM

Das Protokoll würde folgendermaßen aussehen:

```
A>DDT<ENTER>
DDT VERS 2.2
-A100<ENTER>
0100  MVI A,0FE<ENTER>
0102  JMP 0BE86<ENTER>
0105  <ENTER>
-GO<ENTER>
A>SAVE 1 FAST.COM<ENTER>
```

Das Programm FAST wirkt wie ein Schalter, d.h. war die Verschnellerung ausgeschaltet, so wird sie eingeschaltet und umgekehrt. Der Defaultzustand nach dem Booten ist Verschnellerung ausgeschaltet.

Anmerkung für den Maschinensprach-Programmierer: Die Verschnellerung kann über ein Flagbyte bei BE7C gesteuert werden.

```
(BE7C)=00  Verschnellerung ausgeschaltet
(BE7C)=FF  -------"------- eingeschaltet
```

ANHANG D

Der Monitor unter CP/M

Unter CP/M wird der Monitor durch den transienten (auf der Diskette befindlichen) Befehl M.COM aufgerufen.

Nachfolgend finden Sie ein Sourcelisting der Datei M.COM. Dieses können Sie gegebenenfalls mit einem Editor eingeben und z.B. mit dem M80/L80 assemblieren/linken. Sollten Sie diese Möglichkeit nicht besitzen, so finden Sie anschließend an das Sourcelisting einen HEX-Dump, den Sie im DDT über den S-Befehl eingeben können.

```
MONIT   EQU     0E780H          ; ENTRY POINT IN MONITOR
BUFF    EQU     80H             ; KEYBOARD BUFFER
FCB     EQU     5CH             ; DEFAULT FCB
BDOS    EQU     5
ZIEL    EQU     0BF00H
LEN     EQU     STARTE-START
CPMFLG  EQU     0BEDDH
;
;
; LADER FÜR ROM-RESIDENTEN Z80 MONITOR UNTER VDOS 2.0
;
;
        LD A,0FFH
        LD (CPMFLG),A           ; SETZE MONITOR FLAG

        LD A,(BUFF)             ; WURDE NAME ANGEGEBEN
        OR A
        LD BC,0
        JP Z,MONIT              ; NEIN, DANN EINSPRUNG IN MONITOR
;
; HIER WEITER, FALLS NAME ANGEGEBEN WURDE
;
        LD C,15                 ; VERSUCHE DATEI ZU ÖFFNEN
        LD DE,FCB
        CALL BDOS
        INC A                   ; DATEI GEFUNDEN
        LD BC,0
        JP Z,MONIT              ; SPRING IN MONITOR, FALLS NICHT
;
; HIER WEITER, FALLS DATEI GEFUNDEN WURDE
;
        LD HL,START
        LD DE,ZIEL
        LD BC,LEN
        LDIR
        LD BC,0
        LD DE,100H              ; START OF TPA
        JP ZIEL
;
```

```
START:  .PHASE ZIEL
        PUSH BC
        PUSH DE
        LD C,26                 ; SET DMA ADRESS
        CALL 5                  ; DMA ADDRESSE EINRICHTEN
        LD C,20                 ; READ SEQUENTIAL
        LD DE,FCB
        CALL 5
        OR A
        POP HL
        POP BC
        JP NZ,MONIT
        LD DE,128
        ADD HL,DE
        EX DE,HL
        ADD HL,BC
        LD C,L
        LD B,H
        JR ZIEL
        .DEPHASE
STARTE:
        END
```

HEX-Dump obigen Sourcelistings:

```
A>DDT M.COM<ENTER>
NEXT  PC
0180 0100
-D100,160<ENTER>
0100 3E FF 32 DD BE 3A 80 00 B7 01 00 00 CA 80 E7 0E >.2..:..........
0110 0F 11 5C 00 CD 05 00 3C 01 00 00 CA 80 E7 21 32 ..ö....<......!2
0120 01 11 00 BF 01 1F 00 ED B0 01 00 00 11 00 01 C3 ................
0130 00 BF C5 D5 0E 1A CD 05 00 0E 14 11 5C 00 CD 05 ............ö...
0140 00 B7 E1 C1 C2 80 E7 11 80 00 19 EB 09 4D 44 18 .............MD.
0150 E1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0160 00 .
```

Sie können diesen Dump wie folgt eingeben

```
A>DDT<ENTER>
DDT VERS 2.2
-S100<ENTER>
0100 01 3E<ENTER>
0101 BC FF<ENTER>
0102 0F 32<ENTER>
0103 C3 DD<ENTER>
usw.
```

Den S-Modus verlassen Sie wieder durch die Eingabe eines "." oder eines unerlaubten Zeichens. Habens Sie alles eingegeben, so verlassen Sie den Debugger über

```
-GO<ENTER>
```

und speichern mit SAVE 1 M.COM<ENTER> ab.

Wollen Sie den Monitor unter CP/M nun benützen, so reicht die bloße Eingabe von M<ENTER> und schon meldet dieser sich mit

```
<0100  0000  0100>
*
```

hierbei gibt die erste Adresse den Start des TPAs, die zweite Adresse die Länge der augenblicklich geladenen Datei und die dritte Adresse die Startadresse (bei CP/M immer gleich TPA) an.

Möchten Sie eine Datei im Monitor bearbeiten, so müssen Sie diese beim Aufruf mitangeben:

```
M filename.ext<ENTER>
```

ein Aufruf vom Monitor aus ist nicht möglich. (I und O Kommando sind unter CP/M abgeschaltet.)

Wichtig: Genau wie unter BASIC wird der Monitor auch unter CP/M durch das Drücken der ESC-Taste verlassen. Verlassen Sie den Monitor durch den G(o)-Befehl mit GOOOO<ENTER>, so werden sie unter Umständen bemerken, daß Striche auf dem Bildschirm erscheinen, der Rechner aber trotzdem richtig ins CP/M zurückkehrt. Diese Striche hängen mit der Breakpoint-Initialisierung zusammen. Setzen sie doch einfach einmal alle Breakpoints auf den Wert BFOO, also in den Stack (das ist harmlos) und verlassen Sie dann den Monitor. Sie werden sehen, daß Sie dann nichts mehr sehen.

ANHANG E

Der PARA-Patch CP/M

Der PARA-Patch CP/M

VDOS 2.0 unterscheidet sich strukturell wesentlich vom bisherigen VDOS 1.0. Anwenderprogramme werden in ihrer Lauffähigkeit davon meist nicht betroffen, da sie nur von standardisierten Softwareschnittstellen des Betriebssystems (CP/M) Gebrauch machen.

Hochspezialisierte Systemsoftware allerdings, und hierzu gehört auch der vortex Diskmanager PARA, sind im Normalfall unter VDOS 2.0 nicht ungeändert lauffähig. Aber keine Angst, Sie müssen Ihr PARA jetzt nicht wegwerfen. Im Folgenden finden Sie den HEX-Dump eines kleinen Programmes, welches für Sie die Änderungen am PARA vornimmt und hierbei auch den Kopierschutz entfernt. Befolgen sie einfach alle Schritte der Reihe nach und ihr PARA läuft auch unter VDOS 2.0.

(1) Formatieren sie zuerst eine neue Diskette und übertragen sie mit PIP oder FILECOPY die Dateien von Ihrer PARA-Originaldiskette (diese sollte sicherheitshalber mit einem

Schreibschutz versehen sein) auf die frischformatierte Diskette. (Falls Sie nicht mehr genau wissen sollten, wie man formatiert bzw. kopiert, so lesen Sie dies bitte im Benutzerhandbuch zur Floppy Diskstation nach.)
Verwahren Sie jetzt Ihr Original PARA wieder an einem sicheren Ort- wir benötigen es jetzt nicht mehr.

(2) Kopieren sie auf die so erstellte Backup-Diskette mithilfe von PIP oder FILECOPY die Datei DDT.COM. (Diese befindet sich auf der CP/M-Systemdiskette, welche Ihrer Floppy Diskstation beilag.). Auch die CP/M-Systemdiskette benötigen wir vorerst nicht mehr.

(3) So jetzt geht's los. Starten Sie den 8080 Debugger durch Eingabe von DDT<ENTER>. Dieser meldet sich nach kurzer Zeit mit

```
DDT VERS 2.2
-
```

(Der Bindestrich "-" ist das Prompt des Debuggers. Wir befinden uns nun also auf der Kommandoebene des DDT's.)

Wir starten jetzt den Substitute-Modus des DDT und geben die unten aufgeführten Bytes der Reihe nach ein. Nur Mut, es sind zwar relativ viele Bytes, aber das Programm hat eine eingebaute Testroutine und meldet Ihnen, wenn sie Eingabefehler gemacht haben.

```
-S100<ENTER>
0100 01 31<ENTER>
0101 BC 00<ENTER>
usw.
0260 6A 5D<ENTER>
0261 00 .
-g0<ENTER>
```

So, Sie müssten sich jetzt wieder auf der CP/M Kommandoebene befinden und müssen jetzt nur noch Ihr Werk auf Diskette abspeichern:

```
SAVE 2 PPATCH.COM<ENTER>
```

Die Datei, die Ihr PARA ändern kann heißt also PPATCH.

```
31 00 03 21 00 01 01 5F 01 11 00 00 D5 5E 23 E3
19 E3 0B 79 B0 20 F6 D1 2A 5F 02 B7 ED 52 11 A5
01 C2 96 01 0E 0F 11 AF 01 CD 05 00 11 00 11 06
5E D5 C5 0E 1A CD 05 00 0E 14 11 AF 01 CD 05 00
C1 D1 21 80 00 19 EB 10 E8 06 1B 21 F3 01 C5 5E
23 56 23 4E 23 46 23 EB 71 23 70 EB C1 10 EF 0E
13 11 D1 01 D5 CD 05 00 0E 16 D1 CD 05 00 11 00
11 06 5E D5 C5 0E 1A CD 05 00 0E 15 11 D1 01 CD
05 00 C1 D1 21 80 00 19 EB 10 E8 0E 10 11 D1 01
CD 05 00 11 9E 01 0E 09 CD 05 00 C3 00 00 0D 0A
4F 4B 0D 0A 24 0D 0A 43 4B 45 52 52 0D 0A 24 00
```

```
50 41 52 41 20 20 20 20 43 4F 4D 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 50 41 52 41 32 30 20 20 43 4F 4D 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 37 22 A6 AF 55 11 14 B0 EF 1A 14 B0 42
22 14 B0 65 11 2D B0 FA 1A 2D B0 4D 22 2D B0 9E
11 C6 AF 05 1B 2C B0 10 1B 45 B0 B1 39 3E 10 B2
39 10 00 B1 22 00 AE F2 3A 00 AE 1A 3E 11 19 6F
1B 00 00 70 1B 00 00 3A 3D 00 BF C0 3C 01 BF C8
3C 01 BF E3 3C 01 BF A5 1B AE C0 8D 38 AE C0 01
38 46 B0 06 38 46 B0 0B 38 46 B0 10 38 46 B0 F1
5D
```

(4) Starten Sie jetzt PPATCH durch Eingabe von

PPATCH<ENTER>

Wichtig: Es ist Ihre Aufgabe dafür zu sorgen, daß sich PARA.COM auf derselben Diskette befindet wie PPATCH.COM. PPATCH in obiger Form überprüft dies nicht. (Sonst hätten Sie noch wesentlich länger Bytes "einhacken" müssen.)
Sollten Sie unter Punkt (3) Eingabefehler gemacht haben, so meldet PPATCH

CKERR

und springt ins CP/M zurück. In diesem Fall müssen sie das Programm nochmals überprüfen (siehe weiter unten). Ansonsten erhalten Sie die Meldung

OK

und es ist geschafft. Auf der Diskette befindet sich jetzt eine weitere Datei namens PARA20.COM. Dieses geänderte PARA besitzt keinen Kopierschutz und Sie können sich nach Lust und Laune Sicherungskopien anlegen.

WICHTIG: Die Hilfsprogramme DIR64, DIR128 und DIR256 sind unter VDOS 2.0 nicht funktionsfähig. Sie brauchen diese aber auch nicht mehr, da VDOS 2.0 von vorneherein 128 Directoryeinträge zulässt und dies eigentlich ausreichen sollte.

(5) Was machen, wenn PPATCH die Fehlermedlung CKERR gebracht hat?

Starten sie erneut den DDT durch Eingabe von

DDT PPATCH.COM<ENTER>

Es kommt die Meldung:

```
DDT VERS 2.2
NEXT  PC
0300  0100
-
```

Sie haben jetzt PPATCH wieder in den Speicher geladen und können es sich mit dem D-Kommando anschauen:

-D0100<ENTER>

Auf dem Bildschirm erscheint ein sogenannter HEX-Dump der Struktur:

Adresse 16 Bytes ASCII-darstellung

Vergleichen Sie alle Bytes von 0100 bis 0260 mit den unter (3) angegebenen nocheinmal sorgfältig. Der Fehler müßte zu finden sein.
Korrigieren Sie diesen dann wieder mit dem S-Kommando.

Für den Fall, daß Sie überhaupt nicht zurechtkommen:
Wir bitten Sie von Telefonanrufen abzusehen, senden Sie uns einfach eine formatierte Diskette (Achtung: postgerechte Verpackung), eine ausreichend frankierte Rückversandmöglichkeit und wir senden Ihnen PPATCH kostenlos zu.

ANHANG F

Der GRAPHIC MASTER PATCH

Die Erweiterung der Directory auf 128 Einträge auch unter BASIC führt zwangsläufig zu einer Herabsetzung des HIMEM's um ca. 40 Bytes. Aus diesem Grund wird unter VDOS 2.0 leider auch der "GRAPHIC MASTER V2.0" nicht mehr laufen. Sie beheben diesen Mißstand wie folgt:
Resetieren Sie Ihren Rechner und geben Sie die Befehle

POKE &AC01,&AF <ENTER>
POKE &AC02,&32 <ENTER>
POKE &AC03,&45 <ENTER>
POKE &AC04,&AE <ENTER>

ein. Danach laden Sie von Ihrer Original GRAPHIC MASTER Diskette mit LOAD "GRAMA" <CR> das Programm GRAMA.BAS. In Zeile 30 müssen Sie nun den Teil mit dem MEMORY-Befehl eliminieren und das Programm mit SAVE "GRAMA",P <ENTER> wieder abspeichern.

Die Frage der Kompatibilität

Die Schnittmenge von VDOS 1.0 und 2.0 ist an der Benutzeroberfläche (BASIC und CP/M) voll kompatibel. Ja mehr, es wurden gewisse Unzulänglichkeiten von VDOS 1.0 beseitigt.

Auf der Maschinenebene (ROM/RAM Adressen) sind beide Betriebssysteme von wenigen Ausnahmen abgesehen weitgehend inkompatibel. Vor allem beim VDOS 2.0 wurde wesentlich mehr Wert auf Softwareschnittstellen gelegt. So findet der versierte Maschinenprogrammierer z.B. ab BE80H sogenannte DOS Indirections, diese werden immer dann angesprungen, wenn vom DOS aus BIOS Routinen angefordert werden:

BE80	Select Disk
BE83	Set Track
BE86	Set Sector
BE89	Set Dma Address
BE8C	Read Sector (logical)
BE8F	Write Sector (logical)

Dadurch ist es außerordentlich einfach möglich, zusätzliche Drives ins DOS einzubinden.

AMSDOS-Kompatibilität: An der Benutzeroberfläche (BASIC und CP/M) sind VDOS (1.0/2.0) praktisch AMSDOS kompatibel. Wenn's allerdings mit PEEK's und POKE's losgeht sollte man vorsichtig sein, da nur sehr wenige RAM-Adressen praktisch aber keine ROM-Adressen übereinstimmen.