Tandy
Radio Shack

VORLÄUFIGES

INSTRUKTIONS - HANDBUCH

DISK BASIC VERSION 1.1

TRSDOS VERSION 2.0 + 2.1

KATALOG NR. 26 - 9406

7. JULI 1978

Übersetzt: 3.2.79

TANDY RADIO SHACK LUISENSTRASSE 98
4000 DÜSSELDORF 1

I N H A L T S V E R Z E I C H N I S

ANHANG

ANGABEN ZUM TRS-80-MINI-DISK-SYSTEM

Die TRS-80-MINI-DISK ist eine kleine Ausgabe einer Diskette. Sie bietet größere Speicherkapazität und viel schnellere Zugriffszeit als die Kassettenband-Speicherung. Das Diskettensystem verwendet eigene Befehle, die den Umgang mit den Disketten ermöglichen.

Disketten-Format

Die Information wird auf Sektoren der Diskette geschrieben, bzw. von Sektoren der Diskette gelesen. Die Diskette besitzt 35 Spuren (konzentrische Kreise mit Daten- und Positionsinformationen). Die Spur 00 (Null) befindet sich auf dem äußersten Kreis, die Spur 34 auf dem innersten Kreis. Jede Spur ist in 10 gleiche Sektoren mit je 256 Bytes eingeteilt, das sind 89 600 Bytes je Diskette. Die Diskette Nr. 0 nach Katalog-Nr. 26-1160 besitzt nur 80 000 Bytes Speicherplatz. Von dem Speicherplatz stehen aber nur 85 KB bzw. 55 KB (Diskette Ø) zur freien Verfügung, der übrige Speicherplatz wird für die Programm- und Dateienorganisation des TRSDOS benötigt.

Weitere Angaben zur Diskette

Datenübertragungsrate:	12 500 Bytes pro Sekunde
Diskettengeschwindigkeit:	300 U/min
Ansprechzeit:	100 msek im Mittel
Spurzugriffszeit:	200 msek im Mittel
	600 msek über alle 35 Spuren hinweg
Sicherungskopien:	in 1/10 min, Mittelwert einschl. Formatierung
Speicherbelegung:	4,2 KB RAM für TRSDOS
	5,8 KB RAM für DISK BASIC
Durchmesser der Diskette	5 1/4 Zoll = 13,3 cm
Geräteabmessung:	16,5 x 9 x 33,7 cm
Versorgung:	220 V~, 50 Hz, 35 W
Freier Speicherplatz für Dateien:	Diskette Nr. 0 55 K Bytes
	Disketten Nr. 1 - 3 je 85 K Bytes

Geräte-Anordnung

Zur Anwendung des Diskettensystems benötigen Sie ein TRS-80-Gerät mit BASIC LEVEL II und 16 KB RAM-Hauptspeicher, außerdem eine TRS-80-Erweiterungs-Schnittstelle (mit 0, 16 oder 32 KB RAM). Sie können insgesamt 4 Diskettengeräte (Nr. Ø bis 3) an die TRS-80-Erweiterungs-Schnittstelle anschließen.

Das Bandkabel, das Sie mit dem Diskettengerät Ø (Katalog-Nr. 26-1160) erhalten haben, ist für die Verbindung von insgesamt vier Diskettengeräten an die Erweiterungs-Schnittstelle vorgesehen. Die Kontaktstecker am Kabel sind im allgemeinen kodiert (d. h. bestimmte Kontaktstifte sind entfernt). Damit ist jeder Stecker nur für ein bestimmtes Gerät bestimmt (Stecker Ø für Gerät Ø usw.).

Das Diskettengerät Ø (26-1160) muß immer im System vorhanden sein. Alle übrigen Geräte (26-1161) können wahlfrei in jeder gewünschten Kombination eingesetzt werden. Die Anordnung der Geräte am Kabel ist im allgemeinen wie folgt:

Erweiterungs-Schnittstelle ÷ Diskettengerät Ø ÷ Diskettengeräte 1 - 3,

so daß Gerät Ø der Schnittstelle am nächsten, Gerät 3 von der Schnittstelle am weitesten entfernt liegt.

Übrigens ist auch eine andere Anordnung möglich, die im Anhang C/1, Punkt 1 beschrieben ist. Für diesen Anschluß der Geräte sollte man ein Band mit unkodierten Kontaktsteckern bestellen.

Jedes Diskettengerät muß zusätzlich an eine Spannungsquelle von 220 V~, 50 Hz, 35 W angeschlossen werden. Der Netzschalter ist der Kippschalter auf der Rückseite des Gerätes. Die Spannung ist eingeschaltet, wenn der Kippschalter nach oben zeigt.

Diskettenbetrieb

Während der Operation werden alle Motoren der Diskettengeräte gleichzeitig an- und ausgeschaltet. Nur mit einem Gerät kann zu einem gegebenen Zeitpunkt geschrieben bzw. gelesen werden. Eine Schreib- und Leseoperation löst ein "Klicken" des Ladekopfes aus und wird außerdem durch Aufleuchten des LED (rote Kontrollampe auf der Vorderseite des Gerätes) bei dem betreffenden Gerät angezeigt.

Bevor irgendeine Lese- oder Schreiboperation beginnen kann, muß der Motor ca. 1 Sekunde laufen, damit sichergestellt ist, daß sich die Diskette mit der richtigen Geschwindigkeit dreht.

Nach der Operation werden die Motoren wieder abgestellt.

ALLGEMEINE EINFÜHRUNG
IN
LEVEL II, DISK BASIC und TRSDOS

Mit der Anschaffung des TRS-80-Disk-Operating-Systems (genannt TRSDOS) besitzen Sie nun drei verschiedene, jedoch verwandte Systeme für Ihren Micro-Computer.

1) LEVEL II BASIC

LEVEL II BASIC ändert sich, bis auf das Verfahren zum An- und Abschalten (siehe Seite 4), nicht gegenüber dem LEVEL-II-BASIC-Handbuch.

2) DISK BASIC

Mit den Informationen auf der TRSDOS-System-Diskette wird Ihr normales LEVEL II BASIC zum DISK BASIC ausgeweitet, das Dateien auf Diskette lesen/schreiben und Programme laden/speichern kann.

3) TRSDOS

Das Disketten-Betriebssystem überwacht die Operationen Ihres Diskettengerätes und enthält wirkungsvolle Hilfsprogramme, wie das Kopieren einer Diskette auf die andere oder das Auflisten aller Programme, die auf einer Diskette gespeichert sind usw.

Die Anwendung der Mini-Diskette erweitert die Möglichkeiten des TRS-80. Die Diskette bietet Zugriffsmöglichkeiten zu den Programmen, die bei Kassetten-Speicherung nicht gegeben sind. Sie ermöglicht auch eine bequeme Speicherung von Daten.

Daten sind im DISK BASIC entweder durch den wahlfreien Zugriff (RANDOM) oder durch die sequentielle Methode zugänglich. Der sequentielle Zugriff ist eine Methode, die dem Speichern der Daten auf Kassette in LEVEL II ähnlich ist. Für die Beherrschung der Zugriffs-Methode (RANDOM) benötigt man sicher etwas mehr Zeit, aber ihre Vielseitigkeit und die Einsatzmöglichkeiten sind dieser Mühen sicher wert.

Disketten-Betriebssystem und DISK BASIC sind auf der System-Diskette gespeichert. Diese Diskette ist mit TRSDOS bezeichnet und MUSS immer im Gerät 0 (das der Schnittstelle am nächsten liegt) vorhanden sein. Da man nicht alle von DISK BASIC oder TRSDOS im Hauptspeicher (RAM) zur gleichen Zeit benötigt, werden die erforderlichen Daten erst in den Hauptspeicher übertragen, wenn sie angefordert werden. Deshalb muß die TRSDOS-Diskette im Gerät 0 immer zur Verfügung stehen.

Schalten Sie den Strom für die Erweiterungs-Schnittstelle und für die Diskettengeräte ein und legen Sie die TRSDOS-Diskette sehr vorsichtig in das Gerät 0. Danach schalten Sie die TRS-80-Tastatur ein (Power-Knopf). Das Disketten-Betriebssystem wird nun automatisch vom Gerät 0 in den RAM-Speicher geladen. Wenn diese Folge ausgeführt ist, gibt der Computer nachstehende Meldung aus:

TRSDOS-DISK OPERATING SYSTEM-VER 2.0
DOS READY

Das ist der Befehlsmodus des Disk Operating Systems (DOS). Hiermit sind Sie in einem "System"-Modus und können die Funktionen, die in Abschnitt 3 dieses Handbuches beschrieben sind, ausführen. Wenn Sie später den RESET-Knopf zu irgendeinem Zeitpunkt betätigen, kommen Sie in diesen Modus zurück.

Um LEVEL II BASIC auszuführen (ohne die DISK-BASIC-Erweiterung), geben Sie einfach ein:

BASIC 2 und (ENTER)

Der Computer antwortet wie folgt:

MEMORY SIZE ?

Sie können jetzt mit LEVEL II BASIC so arbeiten, wie im LEVEL II BASIC-Handbuch beschrieben ist. Beim Drücken des RESET-Knopfes kehrt der Computer in den TRSDOS-Modus zurück; alle Programme sollten vorher gespeichert werden, da sie sonst verlorengehen. Sie werden jedoch auch das RADIO-SHACK-DISK-BASIC anwenden wollen. Dazu müssen Sie den Befehl BASIC nach der Anzeige DOS READY eingeben, damit das DISK BASIC in den Speicher geladen werden kann.

Anschließend müssen Sie die maximale Anzahl von Dateien angeben, die Sie gleichzeitig benutzen wollen.
DISK BASIC fragt:

HOW MANY FILES ? (wieviele Dateien ?)

Antworten Sie mit der maximalen Anzahl von Dateien, die Sie benötigen. Man kann nicht mehr als 15 Dateien auf einmal benutzen. Jede Datei, die Sie anfordern, braucht einen 256 Bytes langen Puffer. (Näheres über Puffer können Sie der Beschreibung über sequentielle Dateien und Dateien mit direktem Zugriff entnehmen.) Zum Beispiel reserviert die Anforderung von 4 Dateien 1 KB Speicherplatz. Wird keine Zahl eingegeben und die (ENTER)-Taste betätigt, reserviert der Computer Speicherplatz für 3 Dateien.

Die nächste Frage ist:

MEMORY SIZE ? (Größe des benötigten Speichers ?)

Antworten Sie mit der höchsten Adresse (dezimal), die für DISK BASIC zur Verfügung stehen soll, oder drücken Sie (ENTER), und DISK BASIC nimmt jeden freien Speicherplatz, den es finden kann.

Wenn Sie zu irgendeinem Zeitpunkt von DISK BASIC ins Disketten-Betriebssystem (Disk Operating System) zurückkehren wollen, wird folgende Eingabe benötigt:

CMD"S"

Der Computer antwortet mit DOS READY. Befinden Sie sich im DISK BASIC, achten Sie darauf, daß jedes Programm auf Band oder Diskette gespeichert wird, bevor Sie in den TRSDOS-Modus zurückkehren, da sonst Ihr Programm verlorengeht.

DISKETTEN-GERÄTE

Der TRS-80 erlaubt den Anschluß von maximal 4 Mini-Disketten-Geräten. Die Geräte sind von 0 bis 3 numeriert. Der Anschluß des Gerätes 0 auf dem Verbindungskabel liegt der Erweiterungsschnittstelle am nächsten. Werden noch andere Mini-Disketten-Geräte angeschlossen, sind diese nach ihrer Lage an dem Verbindungskabel nacheinander durchzunumerieren (von 1 bis 3). In diesem Handbuch werden die Disketten mit den Nummern ihrer Geräte gekennzeichnet. Eine andere gültige Anordnung der Geräte ist im Anhang C/1, Punkt 1, beschrieben.

EINSATZ UND PFLEGE DER DISKETTEN

Disketten sind Platten aus magnetischem Wiedergabematerial, ausschließlich für den Einsatz in Computersystemen hergestellt. Gegen unsachgemäße Behandlung sind Disketten besonders empfindlich und sollten daher sorgfältig behandelt werden. Wenn sich die Disketten nicht im Gerät befinden, sind sie in ihren Schutzhüllen aufzubewahren. Der Anwender sollte die Diskettenoberfläche, die in dem ovalen Fenster der Schutzhülle zu sehen ist, nicht berühren. Wie jedes Wiedergabematerial sind die Disketten vor Berührungen, Staub, Verformung, hohen Temperaturen und magnetischen Feldern zu schützen.

Durch eine äußerlich angebrachte "Schreibsperre" können Sie eine Diskette vor dem Überschreiben schützen. Das geschieht durch das Aufsetzen einer kleinen Klappe (write tab) auf die rechteckige Kerbe der Diskette. Damit wird jede weitere Ausgabe auf die Diskette unterbunden, bis die Kappe von der Kerbe entfernt wird.

Nur eine Seite der TRS-80-Mini-Disketten dient zur Aufnahme von Informationen. Wenn die Diskette in das Gerät eingeschoben wird, muß die rechteckige Kerbe (Schreibschutz) nach oben zeigen, und die Aufschrift muß der roten Kontrolleuchte gegenüberliegen.

AUFTEILUNG DER DISKETTE

Die TRSDOS-Diskette ist in 35 konzentrische Kreise aufgeteilt, die zur Speicherung der Daten dienen. Der Kreis wird "Spur" genannt. Die Spur ist in 10 gleich lange Abschnitte aufgeteilt, die "Sektoren" genannt werden. Jeder Sektor kann 256 Bytes an Information speichern. Jede Spur enthält 2560 Bytes, und jede Diskette hat 89 600 Bytes. TRSDOS überträgt die Daten zwischen Diskette und Hauptspeicher mit 12,5 K Bytes je Sekunde. Nicht alle 89 KB der Diskette stehen zur freien Verfügung. TRSDOS führt von jeder Diskette ein Verzeichnis, das die Sektoren für Programm- oder Datendateien enthält. Für den Anwender verbleiben ungefähr 85 KB an freiem Speicherplatz. Die TRSDOS-Systemdiskette im Gerät 0 besitzt 55 KB freien Speicherplatz.

SPEICHERGRÖSSE

TRSDOS	4.2 KB RAM	
DISK BASIC	5.8 KB RAM	(Da DISK BASIC nur in Verbindung mit TRSDOS verwendet werden kann, benötigt man für die Anwendung von DISK BASIC ca. 10 KB RAM.)
Dateien	Für jede Datei muß ein Puffer von 256 Bytes reserviert werden, dazu kommen noch Speicherplätze für die Organisation, so daß zusammen ungefähr 280 Bytes je Datei benötigt werden.	

DISK BASIC

DISK BASIC von Radio Shack erweitert den BASIC-Interpreter auch um verschiedene Anweisungen, die nicht mit der Diskette zusammenhängen. Die zusätzlichen LEVEL II-Anweisungen und Funktionen, die durch die Mini-Diskette Ø zur Verfügung stehen, sind:

MID$ (auf der linken Seite der Wertzuweisung)
INSTR
TIME$
USR (USRØ bis USR9)
DEFUSR
Hexadezimale und oktale Konstanten
LINE INPUT
DEF FN
CMD"D"
CMD"T"
CMD"R"

MID$

MID$ kann auf der linken Seite einer Zuweisung verwendet werden, um eine Unter-Zeichenkette an einer bestimmten Stelle in eine bestehende Zeichenkette einzufügen. Die Anwendung entspricht der Form, die für MID$ auf der rechten Seite einer Zuweisung verwendet wird.

MID$ (Zeichenkette #1, I,J) = Zeichenkette #2

Diese Anweisung ersetzt einen Teil der Zeichenkette#1 , beginnend an der Stelle I durch J Zeichen aus der Zeichenkette #2 . J ist frei wählbar. Wenn J fehlt, wird der Teil von Zeichenkette #1 genommen, der bei I beginnt und die Länge von Zeichenkette #2 hat oder bis zum Ende von Zeichenkette #1 geht, wobei jeweils die kleinere der beiden Ketten gewählt wird. Diese Begrenzung schließt aus, daß die Länge von Zeichenkette #1 verändert wird.

Beispiel:

```
10 INPUT "ZKETTE 1, ZKETTE  2, ANFANG, LAENGE";B$, L$, S,K
20 MID$ (B$, S, K) = L$
30 PRINT B$
40 GOTO 10
   RUN

ZKETTE 1, ZKETTE  2, ANFANG, LAENGE ? ABCD, XY, 2,2        (Eingabe)
   AXYD                                                    (Ausgabe von B$)
ZKETTE 1, ZKETTE  2, ANFANG, LAENGE ? 1901 DAKAR ST. W., RD, 12,2
   1901 DAKAR RD. W.
ZKETTE 1, ZKETTE  2, ANFANG, LAENGE ? GEHT BRONCOS GEHT, COWBOYS, 6,7
   GEHT COWBOYS GEHT
```

INSTR - ist eine Zeichenketten-Funktion, die nach dem Vorhandensein einer Zeichenkette innerhalb einer anderen Zeichenkette sucht. INSTR gibt den Anfang dieser Zeichenkette aus. (Diese Funktion ersetzt die Subroutine INSTRING, die im Handbuch LEVEL II beschrieben ist.) Die allgemeine Form von INSTR ist:

INSTR (I, Zeichenkette #1, Zeichenkette #2)

Hiermit wird nach dem ersten Auftreten von Zeichenkette #2 in Zeichenkette #1 gesucht, und wenn Zeichenkette #2 gefunden worden ist, zeigt der ausgegebene Wert ihre Anfangsposition in Zeichenkette #1 an.

I ist ein wahlfreier Parameter, der die Position in der Zeichenkette #1 angibt, bei der mit der Suche begonnen werden soll. Wenn I größer ist als die Länge der Zeichenkette #1, wenn die Zeichenkette Null ist oder wenn kein vergleichbarer Wert gefunden wird, ergibt INSTR den Wert Ø. Wenn Zeichenkette #2 Null ist, ergibt INSTR den Wert I, wenn I angegeben, sonst 1.

TIME$ - ist eine Zeichenkette mit 17 Zeichen, welche Datum und Uhrzeit enthält. Datum und Uhrzeit werden durch die DATUM-ZEIT-Routine des Disketten-Betriebssystems ermittelt (siehe Abschnitt über TRSDOS-Hilfsprogramme, Seite 47). Das Format von TIME$ ist "MM/DD/JJ HH:MM:SS". Um die Zeit über den Zeilendrucker auszugeben, ist folgendes einzutippen:

LPRINT RIGHT$ (TIME$, 8)

Um das Datum in der Mitte des Bildschirms anzuzeigen, muß man eintippen:

PRINT @540, LEFT$ (TIME$, 8)

BEMERKUNG: Wenn die Uhr durch das Laden der Kassette unterbrochen wird (siehe CMD"T"), wird TIME$ durch DOS nicht fortgeschrieben. CMD"R" startet die Uhr von neuem.

DEF FN - Funktionen, die vom Anwender definiert werden können.

DEF FN bestimmt eine Variable für die Funktion.

DEF FN Variablenname (Variablen-Liste) = Ausdruck.

Der Variablenname ist der Name der Funktion. Der Name kann aus einer beliebigen Anzahl von Zeichen bestehen, jedoch muß das erste Zeichen alphabetisch sein, und nur die ersten beiden Zeichen werden berücksichtigt (wie es auch bei Variablen ge-

schieht). Die "Variablenliste" setzt sich aus den Variablen, die in den Funktionen benötigt werden, zusammen. Es können beliebig viele zulässige Variable sein, getrennt durch Kommas und abhängig von der Anzahl der Argumente, die für den Ausdruck benötigt werden. Der Ausdruck ist die Funktion selbst. Diese darf nur eine Anweisung von einer Zeile Länge sein. (Anweisungen getrennt durch Doppelpunkte sind nicht erlaubt.)
Zum Beispiel:

```
10 DEF FNMLT (X,Y) = X*Y
20 INPUT A, B
30 C = FNMLT (A, B)
40 PRINT C
```

Die Funktion MLT multipliziert zwei Argumente. Zeile 30 nimmt die Werte von A und B, übergibt sie an die vom Benutzer definierte Funktion in Zeile 10, die sie multipliziert und das Ergebnis in C speichert.

Auch Zeichenketten können durch Funktionen bearbeitet werden.
Zum Beispiel:

```
10 DEF FNADD$ (A$, B$) = A$+" "+B$
20 INPUT "EINGABE VORNAME";X$
30 INPUT "EINGABE NACHNAME";Y$
40 Z$=FNADD$ (X$, Y$)
50 PRINT Z$
```

In diesem Beispiel wird ein Dollarzeichen ($) zu einem Funktions-Namen hinzugefügt, um eine Zeichenketten-Funktion anzuzeigen. Das ist notwendig, weil - genau wie bei Variablen - die Funktionen gekennzeichnet eingegeben werden müssen, um festzulegen, welchen Variablen-Typ sie ausgeben sollen - mit einfacher (!) oder doppelter Genauigkeit (#), ganzzahlig (%) oder als Zeichenkette ($). Das Weglassen des Zeichens ergibt einfache Genauigkeit.

Die DEF FN-Anweisung erweist sich als nützlich, wenn eine spezielle Funktion mehrmals in einem Programm auftritt. Sie spart bei wiederholt auszuführenden Operationen Zeit und Speicherplatz.

HEXADEZIMALE UND OKTALE KONSTANTEN - In einigen Funktionen ist es bequemer, hexadezimale (Basis 16) oder oktale (Basis 8) Konstante zu benutzen als Dezimal-Zahlen. Diese Zahlen-Basen sind durch die folgenden Symbole gekennzeichnet:

& (oktale Konstante)
& H (hexadezimale Konstante)

Zum Beispiel:

POKE &H42E9, &HFF

Hier würde POKE die hexadezimale Konstante FF (255 dezimal) auf die hexadezimale Speicherstelle 42E9 (17129 dezimal) bringen. Hexadezimale und oktale Konstante dürfen bei INPUT- oder DATA-Anweisungen nicht benutzt werden.

USR - Die USR-Funktion ist erweitert worden und erlaubt, bis zu 10 Benutzer-Routinen in Maschinensprache einzugeben. Die Routinen können durch das Editor/Assembler-Programm umgewandelt und durch den System-Befehl gespeichert werden, oder das Maschinen-Programm kann über die Tastatur eingegeben und mit Hilfe der POKE-Anweisung gespeichert werden.

DEFUSR teilt den Routinen die Start-Adressen zu, unter denen sie aufgerufen werden können, genauso wie diese Adressen durch POKE auf die vorgesehenen Speicherplätze gebracht werden können (siehe Handbuch LEVEL II, 8/8).

Das allgemeine Format, unter dem eine USR-Routine vom BASIC-Programm aufgerufen werden kann, ist:

USRn (arg)

n ist eine ganze Zahl mit einem Wert zwischen 0 und 9 und stellt die Routine dar, die mit der DEFUSR-Anweisung vereinbart wird (siehe DEFUSR). Diese Nummer ruft eine der 10 möglichen Benutzer-Routinen (USR) auf.
Zum Beispiel:

X=USR3 (0)

ruft das Benutzer-Programm 3 auf. Falls die Routine einen Wert ergibt, wird er hier der Variablen X zugewiesen.

Wie im normalen LEVEL-II-Betrieb werden die Benutzer-Routinen in DISK BASIC geschützt, indem man beim Einschalten die Frage MEMORY SIZE (Speichergröße) mit

dem höchsten Speicherplatz beantwortet, den man für BASIC verwenden will. Alle darüberliegenden Speicherplätze sind dann geschützt.

DEFUSR - weist der USR-Routine die Adresse zu, unter der sie aufgerufen werden kann. Diese Anweisung ersetzt das POKE-Verfahren, das im LEVEL-II-Handbuch (8/8) beschrieben ist. Das allgemeine Format dieser Funktion lautet:

DEFUSRn = addr

n darf wieder eine Nummer zwischen 0 und 9 sein, die eine der 10 möglichen USR-Routinen bedeutet. Addr ist ein ganzzahliger Wert, der die Start-Adresse der USR-Routine enthält. So würde

DEFUSR7 = &H70E9

der Routine USR7 die hexadezimale Start-Adresse 70E9 zuweisen.

Zum Beispiel: Nachfolgendes Programm druckt die Zahlen von 1 - 100 aus und ruft dann ein in Maschinensprache geschriebenes Unterprogramm auf (Zeile 100), das den Schirm weiß macht. Die Routine in Maschinensprache wird durch die POKE-Anweisung (Zeile 40) unter Benutzung der DATA-Daten (Zeile 140) in den Speicher gegeben.

```
10  DEFUSR1 = &H7D00
20  FORX=32000 TO 32013
30  READ A
40  POKE X,A
50  NEXT X
60  CLS
70  FOR X=1 TO 100
80  PRINT X;
90  NEXT X
100  X=USR1 (0)
110  FOR X=1 TO 1000
120  NEXT X
130  GOTO 60
140  DATA 33,0,60,54,255,17,1,60,1,255,3,237,176,201
```

EIN- UND AUSGABE VON ARGUMENTEN DER USR-ROUTINEN

Es gibt zwei Möglichkeiten, Argumente zu Unterprogrammen in Maschinensprache zu übertragen:

1. das Argument durch POKE auf feste Plätze im RAM zu übertragen und, nachdem das Unterprogramm ordnungsgemäß durchlaufen ist, das Ergebnis durch PEEK ins BASIC-Programm zurückzuholen.
2. das Argument als Teil einer USR-Funktion zu übergeben, indem man mit Hilfe von BASIC-Programmen die Zahlen-Umwandlungen durchführt.

Folgendes Beispiel übergibt einem Unterprogramm den Wert von X und bringt über Y einen Wert zurück:

BASIC

10 DEFUSR5= &H7D00	vereinbart die hexadezimale Start-Adresse 7ØØØ für das Benutzer-Unterprogramm 5.
20 INPUT X	
30 Y = USR5 (X)	An dieser Stelle wird die Kontrolle an das Unterprogramm übergeben.

UNTERPROGRAMM

CALL 0A9AH	Das ist eine Routine im ROM, welche den X-Wert (das Argument in der Zeile 30) in eine ganze Zahl umwandelt und im HL-Registerpaar speichert.
. . . "Rumpf" des Unter- . Programms . .	
JP 0A9AH	Das ist eine Routine im ROM, welche den Wert im HL-Register als neuen Wert von USR5 an das aufrufende BASIC-Programm übergibt. Im oberen Beispiel erhält Y den Wert des HL-Registerpaares.

CMD"T" - stellt die Uhr ab. Dieser Befehl muß entweder im Befehlsmodus oder innerhalb eines Programms gegeben werden, bevor man mit der Kassette arbeiten kann.

CMD"T" schaltet die Real-Zeit-Uhr ab, so daß die Uhr die empfindlichen Kassetten-Befehle nicht unterbrechen kann. Dabei handelt es sich um CLOAD, CSAVE, INPUT#-1 (oder -2), PRINT#-1 (oder -2), SYSTEM (Datei-Name).

CMD"R" - stellt die Uhr wieder an. Dieser Befehl wird nach einer Bandoperation benutzt, um die Uhr wieder in Betrieb zu nehmen.

Beispiel:

```
10    CMD"T"                  Ausschalten der Uhr
20    INPUT# - 1,  A, B, C
30    CMD"R"                  Wiedereinschalten der Uhr.
```

CMD"D" - Dieser Befehl lädt die DOS-Fehlerbehandlungsroutine DEBUG. Siehe DOS-Hilfsprogramme, Abschnitt DOS DEBUGGER (Seite 48).

CLOAD? - Diese Prüf-Routine zum Vergleichen von Programmen, die unter LEVEL II gespeichert sind, ist unter DISK BASIC nicht verfügbar. Eine "BAD"-Antwort wird stets das Ergebnis sein. Programme, die in DISK BASIC mit CSAVE gespeichert werden, sind auch auf Richtigkeit geprüft.

LINE INPUT - bewirkt, daß alle eingegebenen Zeichen einer Zeichenketten-Variablen zugewiesen werden. Diese Anweisung wird benötigt, wenn Kommas, Anführungszeichen oder andere Begrenzer eingegeben werden, wie in Namen, Adressen usw.

Line Input "Abfrage-Zeichenkette"; Variable

Beispiel:

```
10 LINE INPUT "GEBEN SIE IHREN NAMEN AN";N$
```

bewirkt, daß alle eingegebenen Buchstaben bis zum Drücken der ENTER-Taste an N$ übergeben werden, selbst wenn der Name vielleicht ein Komma enthält. Ein Fragezeichen wird vorher nur angezeigt, wenn es Teil der Abfrage-Zeichenkette ist.

CLOAD - darf unter DISK BASIC nicht mit einem Dateinamen benutzt werden. CLOAD"A" kann zum Beispiel bewirken, daß der Computer sich festfährt. Zum Laden von Kassetten wird die nachstehende Anweisungsfolge benötigt:

```
CMD"T"      Ausschalten der Uhr
CLOAD
CMD"R"      Wiedereinschalten der Uhr
```

CSAVE erfordert weiterhin einen Datei-Namen.

DISKETTEN-BEFEHLE

Folgende Befehle, die für das Disk-System gelten, ermöglichen die Steuerung des Disketten-Betriebs. Sie eröffnen und schließen Disketten-Operationen und kennzeichnen die Arten der benutzten Dateien. Sie kontrollieren auch Dateien in ganz ähnlicher Weise, wie LEVEL-II-BASIC-Befehle Programme kontrollieren.

Die LEVEL-II-Disketten-Befehle sind:

OPEN
CLOSE
SAVE
LOAD
MERGE
KILL
CMD"S"
RUN"Datei-Name"

DATEI-NAMEN

In diesem Handbuch sind überall Hinweise auf Datei-Namen gegeben. Diese Namen können immer aus bis zu 4 Teilen bestehen: der Name selbst, eine Dateiergänzung, ein Kennwort und eine Disketten-Gerätenummer, in dieser Reihenfolge. Alles, außer dem Namen selbst, ist wahlfrei.

Der Name kann sich aus 1 bis 8 alpha-numerischen Zeichen zusammensetzen. Das erste Zeichen muß ein Buchstabe sein.
Beispiel:

MASTERIN
PAYEMP 25
DOLLAR
Z
XYZ 123

Die Dateiergänzung setzt sich aus 1 bis 3 alpha-numerischen Zeichen zusammen und wird zur Bezeichnung eines Dateityps benutzt. Den Dateiergänzungen muß ein Schrägstrich (/) vorangehen. Wenn er nicht angegeben ist, verwendet das System für die Dateiergänzung Leerzeichen.
Beispiel:

/CMD	zeigt etwa eine Befehlsdatei an
/OBJ	zeigt etwa eine Objektdatei an
/SYS	Systemdatei
/DAT	Datendatei
/BAS	BASIC-Programm-Datei

Wenn eine Dateiergänzung einmal vereinbart worden ist, muß sie bei allen Diskettenoperationen, die diese Datei einschließen, angegeben werden.

Bei Aufruf des Befehls "DIR" wird die Dateiergänzung neben dem Datei-Namen angezeigt.

Das Kennwort dient zum Schutz der Datei. Es kann 1 bis 8 alpha-numerische Zeichen enthalten. Wenn ein Kennwort zugeteilt ist und eine Datei durch eine

OPEN-Anweisung eröffnet wurde, dann muß dieses Kennwort auch bei späteren Diskettenoperationen immer angegeben werden. Ohne das richtige Kennwort kann eine Datei nicht gelesen, kopiert, eröffnet oder gelöscht werden. VORSICHT !!! Wenn Sie das Kennwort vergessen, sind Sie niemals mehr in der Lage, diese Datei wieder zu benutzen.
Allen Kennworten geht ein Punkt voraus.
Beispiel:

.CRIMSON
.SKY
.XMASTREE

Die Diskettengerät-Nummer ist eine Ziffer zwischen 0 und 3, der ein Doppelpunkt vorangeht. Sie verweist auf ein bestimmtes Disketten-Gerät. (Disketten-Gerät 0 ist das nächste zum Interface, Disketten-Gerät 3 ist das am weitesten entfernte.) Wenn keine Disketten-Nummer für die neue Datei genannt ist, wird diese Datei auf die niedrigste Disketten-Nummer im System geschrieben, auf der noch Platz vorhanden und die nicht schreibgeschützt ist. Wenn zur Eingabe von Dateien keine Disketten-Nummer angegeben wurde, sucht das Betriebs-System jede Diskette nach der gefragten Datei ab, wodurch sich natürlich die Zugriffsgeschwindigkeit verringert.
Beispiel:

:3 Diskette 3, die letzte am Bandkabel
:2 Diskette 2, die vorletzte

Nachstehend finden Sie Beispiele für den Gebrauch von Datei-Namen mit verschiedenen Kombinationen der Namens-Bestandteile:

MATHTEST/BAS.LEHRER:2

Der Datei-Name ist MATHTEST, und es ist eine Basic-Programm-Datei. Das Kennwort ist LEHRER, und die Datei befindet sich auf dem Diskettengerät 2.

INVT:1

Der Dateiname ist INVT, und die Datei befindet sich auf dem Diskettengerät 1; Ergänzung und Kennwort sich nicht angegeben.

PAYROLL/DAT.IRS

Der Datei-Name ist PAYROLL, und es handelt sich um eine Daten-Datei. Das Kennwort ist IRS, und Sie wissen nicht, auf welchem Diskettengerät

sich die Daten befinden. Die Disketten werden in aufsteigender Reihenfolge durchsucht, bis eine gefunden wird, welche frei und nicht schreibgeschützt ist. Diese Diskette wird dann verwendet.

Sie dürfen zwei Dateien so lange mit dem gleichen Namen bezeichnen, wie wenigstens Teile des Datei-Namens (Ergänzung oder Disketten-Nummer) unterschiedlich sind.
Zum Beispiel sind alle folgenden Dateien verschieden.

FILE:1	FILE/BAS:1
FILE/BAS:0	FILE/SYS
FILE:0	FILE/OBJ

Das Kennwort allein verändert den Datei-Namen nicht. Wenn ein Programm unter FILE/BAS gespeichert ist, und dasselbe Programm soli zusätzlich unter FILE/BAS.PASS gespeichert werden, tritt ein FILE ACCESS DENIED-Fehler auf.
Das Kennwort eignet sich nicht, um Dateien zu unterscheiden. Ein Kennwort schützt nur die Datei, wenn sie gespeichert ist.

Die allgemeine Form eines Dateinamens mit den vier möglichen Beschreibungen lautet:

Datei-Name/Dateiergänzung. Kennwort : Disketten-Nr.

Die Bezeichnungen können bestehen aus:

(8 alphanum. Zeichen)/(3 alphanum. Zeichen).(8 alphanum. Zeichen):(1 Nummer)

Die Bezeichnungen sind:

(erforderlich)/(wahlfrei).(wahlfrei):(wahlfrei)

DISKETTEN-ANWEISUNGEN

Die OPEN-Anweisung wird benötigt, bevor mit irgendeiner Diskette Daten-Dateien gelesen oder geschrieben werden können. Beim Sichern und Laden von Programmen braucht diese Anweisung nicht verwendet werden. Das Format der OPEN-Anweisung lautet:

OPEN "Modus", #Datei-Nummer, "Datei-Name"

Der Modus kennzeichnet den Typ der Datei. Folgende Typen werden verwendet:

MODUS	BESCHREIBUNG
O	Ausgabe-Modus für das Schreiben sequentieller Dateien auf Diskette
I	Eingabe-Modus für das Lesen sequentieller Dateien von Diskette
R	Eingabe und/oder Ausgabe von Dateien mit direktem Zugriff.

Datei-Nummern werden zur Kennzeichnung einer eröffneten Datei verwendet. In späteren Eingabe/Ausgabe-Operationen wird diese Nummer zur Kennzeichnung der Datei und zur Wahl weitaus mehr verwendet als Name, Modus, Disketten-Nummer usw. Datei-Nummern sind ganze Zahlen und bewegen sich zwischen 1 und 16. Das bedeutet, daß 16 Dateien zur gleichen Zeit eröffnet und in verschiedenen Operationen verwendet werden können, indem man die Dateinummer der OPEN-Anweisung einsetzt.

Die Dateinummern müssen mit der Anzahl der Dateien übereinstimmen, die auf die Frage HOW MANY FILES ? (wieviele Dateien ?) eingegeben wurden. Wenn Sie zum Beispiel "6 Dateien" antworten, darf die Dateinummer nur zwischen den Nummern 1 bis 6 liegen. Wenn Sie auf die Frage nur mit ENTER geantwortet haben, können Sie entsprechend 3 Dateinummern verwenden: 1, 2 oder 3. Das Zeichen # kann in der OPEN-Anweisung wahlfrei verwendet werden. Zwei OPEN-Dateien dürfen nicht die gleiche Datei-Nummer tragen, aber Datei-Nummern dürfen gewechselt werden durch Schließen (CLOSE) der Datei und Wiedereröffnen mit anderer Datei-Nummer.

Der Datei-Name ist eine alpha-numerische Zeichenkette, die zur Kennzeichnung der Datei bei der Speicherung auf Diskette benutzt wurde, oder mit der Sie eine Datei bezeichnen, die Sie neu erstellen wollen.

Beispiele von OPEN-Anweisungen:

```
OPEN "O",1, "AUSGABE"
OPEN "I",2, "EINGABE"
OPEN"R",3,"DATEI:1"
OPEN D$, X, N$
```

Im vierten Beispiel werden in der OPEN-Anweisung Variable verwendet.

Es ist üblich, die OPEN-Anweisung mit einer Zeilennummer in einem Programm zu verwenden. Eine Datei kann zu einem gegebenen Zeitpunkt nur in einem Modus eröffnet werden. Versuchen Sie, eine Datei mit einer bereits benutzten Datei-Nummer zu eröffnen, erhalten Sie den Fehler FILE ALREADY OPEN (Datei bereits eröffnet).

CLOSE - CLOSE beendet den Zugriff auf eine bestimmte Datei, indem sie die Ein-/Ausgabe zu dieser Datei ausschließt.
Das Format ist:

```
CLOSE Dateinummer,...,Dateinummer
```

Beispiel:

```
CLOSE 1, 2, 6
```

schließt die Dateien, die durch 1, 2 und 6 gekennzeichnet sind.

Die Dateinummern sind frei wählbar. Wird CLOSE ohne Datei-Nummer benutzt, werden alle eröffneten Dateien geschlossen.

Ein CLOSE, das für eine sequentielle Datei bestimmt ist, leert den Puffer und schließt erst dann die Datei.

Eine NEW-Anweisung schließt automatisch alle eröffneten Dateien und löscht das Programm und die Variablen, die sich im Computer beffnden,

Alle Dateien müssen geschlossen sein, bevor man die Diskette eines bestimmten Gerätes wechselt.

KILL - KILL löscht eine Disketten-Datei. Damit wird Platz, der durch eine alte Datei besetzt war, freigegeben.

```
KILL"Dateiname"
```

Beispiel:

```
KILL"DATEN"
```

Eine Datei muß geschlossen sein, bevor Sie sie löschen können. Eine Datei, die geöffnet ist (siehe OPEN), kann nicht gelöscht werden. (Wenn es doch versucht wird, wird ein "FILE ALREADY OPEN" angezeigt.)

MERGE - Der MERGE-Befehl verbindet ein residentes (im Hauptspeicher vorhandenes) Programm mit einem Programm, das auf Diskette gespeichert ist. Das neu eingegebene Programm überlagert das residente Programm und ersetzt jede Zeile, die die gleiche Zeilen-Nummer trägt. Das verändert das vorhandene Programm, indem es die Zeilen durch diejenigen des Programmes ersetzt, das von der Diskette kommt.
Das allgemeine Format lautet:

MERGE"Datei-Name"

Beispiel:

MERGE"ASSPROG"

Die Datei muß mit "A" (ASC II Format) gespeichert sein, was unter SAVE beschrieben wird.

SAVE - Programme dürfen mit dem SAVE-Befehl auf Diskette gespeichert werden. Diese Operation wird unter Verwendung folgender Anweisung ausgeführt:

SAVE"Datei-Name"

Beispiel:

SAVE"PRGRM10"

Damit wird das Programm gespeichert, und alle anderen Programme mit dem gleichen Namen werden gelöscht. Wenn Sie ein Programm verbessert oder verändert haben, ersetzt das neueste Programm das ältere.

Programm werden normalerweise in einem komprimierten Format gespeichert. Dafür sorgt automatisch der Computer. Programme können aber auch im ASCII-Format, das durch "A" gewählt werden kann, gespeichert werden. Dieses Format wird gewählt, wenn Programmtext durch ein anderes Programm eingelesen werden soll (entweder mit MERGE oder LINE INPUT). Ein Beispiel eines solchen Programmes kann eine Routine sein, die die Zeilen in einem anderen Programm umnumeriert.
Das ASCII-Format wird folgendermaßen gewählt:

SAVE"Datei-Name",A

Beispiel:

SAVE"PRGRM 20", A

Programme, die in ASCII gespeichert werden, benötigen mehr Zeit zum Überspielen als Dateien im normalen binären Format.

LOAD - Dieser Befehl liest ein auf Diskette gespeichertes Programm in den Hauptspeicher ein. Der LOAD-Befehl wird in folgender Form benutzt:

LOAD"Dateiname",R(wahlweise)

Beispiel:

LOAD"PRGRM 30",R

Hiermit wird das genannte Programm von dem entsprechenden Diskettengerät eingelesen und die "R"-Option ausgeführt. (Ist kein Diskettengerät angegeben - wie im vorliegenden Fall -, wird das entsprechende Gerät mit Hilfe des Verzeichnisses gefunden.

Die "R"-Option bewirkt, daß das Programm nicht nur eingegeben, sondern auch mit RUN ausgeführt wird. Wird ein LOAD-Befehl ohne R-Option eingegeben, werden alle Dateien automatisch geschlossen und alle besetzten Speicher gelöscht (wie bei NEW). Verwendet man die R-Option, werden alle Dateien offen gelassen und nur die Programmzeilen und Variablen im Hauptspeicher gelöscht.

RUN"Dateiname"

Dieser Befehl führt die gleiche Funktion wie der LOAD-Befehl mit R-Option aus. Folgende Befehle sind also gleichwertig:

LOAD"PROGRAMM",R
RUN"PROGRAMM"

CMD"S" - Dieser Befehl gibt die Kontrolle an das Disketten-Betriebssystem zurück. Wenn dieser Befehl in DISK-BASIC angewandt wird, antwortet der Computer:

DOS READY.

SEQUENTIELLE DISKETTEN-DATEIEN

Die sequentielle Ein- und Ausgabe in die einfachste Art der Datenspeicherung auf Disketten. (Hierbei werden die Daten sequentiell = hintereinander aus- bzw. eingegeben. Die Anweisungen sind denen für die Kassettenspeicherung sehr ähnlich. Die für sequentielle I/O-Operationen (INPUT/OUTPUT = Ein/Ausgabe) verwendeten Anweisungen sind:

PRINT #	LINE INPUT #
PRINT USING	
INPUT #	EOF

PRINT # - Wird für das Schreiben von Daten auf eine sequentielle Ausgabe-Datei verwendet. Im Format ist sie der PRINT-Anweisung zum Aufbau von Kassetten-Dateien sehr ähnlich:

PRINT # Dateinummer, Variable oder Ausdruck;...;Variable oder Ausdruck

Diese Anweisung schreibt die Daten, die in der Liste von Variablen oder Ausdrücken angegeben sind, auf die Datei, die mit der Dateinummer gekennzeichnet ist. Die Dateinummer muß vorher mit einer OPEN-Anweisung für diese Datei vereinbart worden sein.

Beispiel: PRINT # 1, A; A$; Z; A$; LEFT$ (Z$,5)

Hierdurch wird die angegebene Information auf die Datei mit der Dateinummer 1 geschrieben. ASCII-Zeichen können dabei auch mit der Funktion CHR$ aus den entsprechenden ASCII-Kodes hergestellt und eingefügt werden. Dieses Verfahren wird angewandt, wenn eine Zeichenkette nicht-alphanumerische Zeichen oder Kontroll-Zeichen enthält, wie Kommas, Zeichenvorschub usw.

Beispiel: PRINT # 1, CHR$ (34); X$; CHR$ (34) ....

Wenn die Zeichenkette rein alphanumerisch ist, können Kommas auch wie folgt eingesetzt werden:

PRINT # 1, A$;",";B$;",";...

PRINT USING - Hiermit kann man eine sequentielle Datei in einem bestimmten Format schreiben. Die Formate sind dabei die gleichen, wie in der BASIC-Anweisung PRINT - USING:

PRINT # Dateinummer, USING"Format"; Liste von Variablen oder Ausdrücken

Beispiel:

```
PRINT#1, USING"**$#####.##"; 121.129
```

Diese Anweisung schreibt *$121.13 auf die Disketten-Datei.

INPUT# - Wird verwendet, um sequentielle Daten wieder einzulesen, die mit einer PRINT# -Anweisung auf die Diskette geschrieben wurden.

```
INPUT#Dateinummer, Variable, Variable ....
```

Diese Anweisung liest Daten von der Diskette in die angegebenen Variablen. Die Dateinummer wurde vorher mit der OPEN-Anweisung festgelegt.

Beispiel:

```
INPUT#3, A, B, C
```

Hiermit werden die Daten von der Diskette eingelesen und der erste gelesene Wert der Variablen A, der zweite der Variablen B usw. zugewiesen.

Es gibt einige wichtige Unterschiede zwischen Disketten- und Kassetten-Dateien. Wenn man die Zeichenketten mit PRINT# auf Diskette schreibt, muß man sie durch in Anführungsstrichen eingefaßte Kommas trennen (","), wie nachstehend in der Programmzeile 30

```
10 OPEN"O", 1, "NAMEN"
20 A$="HANS": B$="SCHMIDT
30 PRINT#1, A$; ","; B$
40 CLOSE:OPEN"I", 1, "NAMEN
50 INPUT#1, A$:PRINTA$
60 CLOSE: END
```

(die letzten Anführungsstriche in einer Zeile können entfallen)

Wenn dieses Programm läuft, wird HANS ausgedruckt. Wenn aber die Zeile 30 PRINT#1, A$, B$ lautet, würde das Programm HANSSCHMIDT ausgegeben. Die Kommas zwischen Zeichenketten sind erforderlich, damit DISK BASIC weiß, wo eine Zeichenkette aufhört bzw. beginnt. Wenn Sie die Kommas vergessen, bekommen Sie sehr schnell einen READ-PAST-END-FILE-Fehler gemeldet. Zwischen numerischen Variablen werden keine Kommas benötigt.

```
PRINT#1, A; DC; B
```

ist also eine gültige Anweisung.

Ein anderer Unterschied zur Kassettenspeicherung ist, daß die PRINT#-Anweisungen nicht genau mit den INPUT#-Anweisungen übereinstimmen müssen, Bei Kassettenspeicherung erfordert ein PRINT#-1, A, B, C ein INPUT#-1, A, B, C um die drei Variablen wieder einzulesen. Bei Disketten ist das anders. Verfolgen Sie folgendes Programm:

```
10 OPEN"O", 1, "DATA"
20 A=1:B=2:C=3:D=4
30 PRINT#1, A, B, C, D:CLOSE
40 OPEN"I", 1, "DATA"
50 INPUT#1, A, B, :PRINT A, B
60 INPUT#1, A, B:PRINT A, B
> RUN
1             2             3             4
```

Beachten Sie, daß PRINT in Zeile 30 nicht mit den beiden INPUT's in Zeile 50 und 60 zusammenpaßt. Trotzdem arbeitet das Programm korrekt.

Um die Wirkung der Anweisung RESTORE mit einer Disketten-Datei nachzuahmen, muß die Datei nach dem Schreiben mit CLOSE geschlossen und dann mit OPEN neu eröffnet werden. Damit werden alle Leseanweisungen INPUT# auf den Anfang der Datei zurückgesetzt.

LINE INPUT - Hiermit kann man jede Zeichenkette von Diskette oder Tastatur eingeben einschließlich Kommas oder Anführungszeichen, bis ein Wagenrücklauf (ENTER) gelesen wird oder 255 Zeichen eingelesen sind. Vom Computer wird Eingabe über Tastatur verlangt, ohne daß ein Fragezeichen angezeigt wird. (Nur der Cursor erscheint auf dem Bildschirm.) Sie können Anführungsstriche ("), Kommas (,) oder Zeilenvorschub (↓) eintippen, bis Sie ENTER drücken.

```
10 CLEAR 300
20 LINE INPUT A$
30 PRINT A$
```

Testen Sie vorstehendes Programm mit Kommas und Zeichen für Zeilenvorschub. Beachten Sie, daß BREAK das Programm während LINE INPUT anhält.

Das folgende Programm liest sich selbst von der Diskette ein und zeigt sich auf dem Bildschirm an.

```
10 CLEAR 500
20 OPEN"I",1,"PROG"
30 FORI = 1TO5
40 LINEINPUT#1,A$
50 PRINT A$: NEXT
> SAVE"PROG",A
> RUN
```

Sorgen Sie dafür, daß das Programm als ASCII-Datei gesichert ist (SAVE"PROG",A), bevor Sie es mit RUN starten. Die Formate für LINE INPUT sind:

```
LINE INPUT   Zeichenketten-Variable  (Tastatur)
LINE INPUT   "Zeichenkette";Zeichenketten-Variable  (Tastatur)
LINE INPUT   Dateinummer,Zeichenketten-Variable  (Diskette)
```

Beispiel:

```
LINE INPUT X$
LINE INPUT"DATEINAME EINGEBEN?";X$
LINE INPUT#2,X$
```

EOF - Bezeichnet das Ende einer Datei. Wenn man Daten von einer Disketten-Datei einliest (INPUT#), erkennt diese Funktion das Ende der Datei. EOF arbeitet als logische Funktion, um den letzten Satz einer Datei zu finden. Sie kann auf zweierlei Weise angewendet werden:

```
Variable = EOF (Dateinummer)
Beispiel:   X=EOF (1)
```

X erhält den Wert -1, bei Dateiende bzw. den Wert Ø wenn weitere Daten vorhanden sind. EOF kann auch in einer IF-Anweisung stehen:

```
IF EOF (1) GOTO 999
```

d.h., wenn das Ende der Datei gefunden ist, verzweige nach 999, wenn nicht, bearbeite die nächste Programmzeile. Beispiel:

```
10 OPEN"I",1,"DATEI 1"
20 FOR X = 1 TO 20
30 IF EOF (1) THEN 60
40 INPUT #1, A(X)
50 NEXT
60 PRINTX;"DATEISAETZE GELESEN"
70 CLOSE
```

Einer Datei werden automatisch 1 1/4 K (1280 Bytes) zugewiesen. Ist mehr Platz erforderlich, werden nochmals 1 1/4 K bereitgestellt usw. Eine Datei belegt also immer wenigstens 1280 Bytes im Hauptspeicher oder ein Vielfaches davon. Weniger ist nicht möglich.

DATEIEN MIT DIREKTEM ZUGRIFF (RANDOM FILES)

Dateien mit direktem Zugriff bieten zwei bestimmte Vorteile gegenüber sequentiellen Dateien. Bei der Datensuche ist man bei sequentiellem Zugriff gezwungen, Satz für Satz durchzugehen, während beim direkten Zugriff die erforderlichen Daten unmittelbar verfügbar sind. Sequentielle Daten sind im ASCII-Format gespeichert, wohingegen RANDOM-Daten in einem komprimierten binären Format gespeichert sind. Sequentielle Dateien können nur im Input- oder Output-Modus eröffnet werden, während bei einer Datei mit direktem Zugriff wahlweise geschrieben bzw. von der Datei gelesen werden kann. Daten können bei direktem Zugriff einfach modifiziert, d. h., die zu ändernden Daten können gelesen, geändert und auf die Diskette zurückgeschrieben werden, wodurch man die alten Informationen überschreibt.

Ein Satz von 256 Bytes ist das Grundelement in der Datei mit direktem Zugriff. Eine Datei kann bis zu 329 Sätze enthalten. Eine Satznummer zwischen 1 und 329 ist jedem Satz zugeordnet. Sie dient als Satzkennzeichen, und alle Schreib- bzw. Lese-Befehle müssen mit dieser Satznummer als Kennzeichen versehen sein. Selbstverständlich hat der erste Satz in einer Datei die Satznummer 1, der zweite die Nummer 2 usw. Sofern Sie irgendeinen Satz in einer Datei benötigen, brauchen Sie nur seine Satznummer zu kennen, um unmittelbar auf seine Daten zugreifen zu können.

Sätze mit direktem Zugriff werden in einen Puffer (BUFFER) gelesen oder darin zusammengestellt. Ein Puffer ist ein Bereich von 256 Bytes im Hauptspeicher, in dem Sätze aufgebaut werden, immer eine Variable nach der anderen. Der Pufferinhalt wird dann auf die Diskette geschrieben. Umgekehrt werden in dieser Form Sätze in direktem Zugriff eingelesen: Der Puffer wird von der Diskette ausgefüllt und die Variablen werden eine nach der anderen aus dem Puffer geholt.

Diese Aufteilung des Puffers in Variable und das Herausholen der Daten wird Feldaufteilung (fielding) genannt. Ein Beispiel soll es erklären. Wir nehmen an, daß eine Datei aus Namen und Adressen besteht. Der erste Satz enthält folgendes:

MEYER WOLFGANG LINDEMANNSTR. 100 46 DORTMUND

Woher weiß der Computer, wo Name bzw. Adresse beginnt, wenn der Satz von der Diskette in den Puffer geladen wird? Diese Information muß der Anwender dem Computer mitteilen.

Mit einer Feldanweisung FIELD wird der Computer über das Format der Daten informiert.

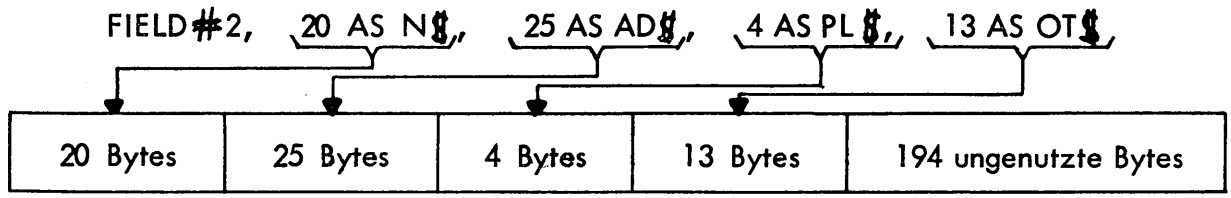
Beispiel:

```
OPEN"R",2,"POST"
FIELD#2, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
```

Diese Feldanweisung legt das Format für die Datei 2 fest, indem sie mit AS die ersten 20 Zeichen für den Namen N$, die nächsten 25 Zeichen für Straße AD$, die nächsten 4 Zeichen für die Postleitzahl PL$ und die nächsten 13 für den Ort OT$ festlegt. Im allgemeinen regelt die Feldanweisung die Länge aller Felder in einem Satz und bestimmt die Feldvariablen, der sie zugeordnet werden. Seien Sie vorsichtig beim Umgang mit Zeichenketten-Variablen in Feldanweisungen. Wie Sie sehen werden, müssen sie anders behandelt werden als gewöhnliche Variable.

Wenn man die Längen aller Felder in obenstehender Feldanweisung addiert, erhält man als Gesamtlänge der Daten 20 + 25 + 4 + 13 = 62 Zeichen. Aber der Puffer ist 256 Zeichen lang, was geschieht mit den übrigen 194 Zeichen? Sie werden vergeudet ! Jedesmal, wenn ein Name und eine Adresse auf die Diskette übertragen werden, bleiben 194 Zeichen ungenutzt. So gehen dreiviertel der Datei verloren. Später werden wir ein Verfahren kennenlernen, mit dessen Hilfe vier Adressen auf einmal in den Puffer gespeichert werden können. Damit gehen nur 256 - 4 x 62 = 8 Bytes des 256 Byte langen Puffers verloren.

Nachstehendes Bild zeigt, wie Puffer und Feldanweisung zusammenarbeiten.



Puffer für Datei 2 (256 Bytes insgesamt)

Alle Variable in dem Puffer werden als Zeichenketten gespeichert. In DISC BASIC gibt es spezielle Befehle, die ganze Zahlen, einfach genaue und doppelt genaue Zahlen in Zeichenketten und wieder zurück in Zahlen umwandeln können.

Ganze Zahlen im Bereich von - 32768 bis + 32767 sind als zwei Byte lange Dualzahlen gespeichert. Daher werden sie bei der Umwandlung als 2-Byte-Zeichenkette in der Datei mit direktem Zugriff gespeichert.

Zahlen mit einfacher Genauigkeit werden als 4-Bytes lange Dualzahlen gespeichert. Sie werden als 4-Bytes lange Zeichenketten auf der Diskette gespeichert.

Zahlen mit doppelter Genauigkeit erfordern 8 Bytes und benötigen daher auch 8-Bytes lange Zeichenketten zur Speicherung.

Bei der Einteilung des Puffers in Felder haben obenstehende Werte Bedeutung, daher wollen wir sie nochmals kurz zusammenfassen:

Zahlenart	benötigte Bytes bei der Feldeinteilung
Ganze Zahl (Integer)	2
Einfache Genauigkeit	4
Doppelte Genauigkeit	8

ANLEGEN VON DATEIEN AUF DISKETTEN MIT DIREKTEM ZUGRIFF

Dateien werden angelegt, indem man die entsprechenden Daten in den Puffer gibt und dann den Inhalt des Puffers als Satz auf die Diskette schreibt. Da allen Sätzen eine Nummer zugeordnet ist, kann man gezielt auf jeden Satz zugreifen, indem man sich auf diese Nummer bezieht.

Der erste Schritt ist, die Datei durch Angabe von R (RANDOM = direkter Zugriff) zu eröffnen:

OPEN "R", Dateinummer, "Dateiname"

Wie beim sequentiellen Zugriff wird die Dateinummer gewählt, um eine bestimmte Datei anzusprechen. Mit dieser Zahl wird der Datei auch ein Puffer zugeordnet, um vom Programm aus auf die Disketten-Datei zugreifen zu können.

FIELD - Der nächste Schritt ist, den RANDOM-Puffer in Segmente einzuteilen, denen die unterschiedlichen Datenelemente zugeteilt werden können. Jede FIELD-Anweisung bezieht sich auf einen besonderen Satz-"Typ". Wenn alle Sätze einer Diskette gleich sind, benötigt man nur eine FIELD-Anweisung für alle Sätze. Als besten Weg, die Felder für einen Puffer festzulegen, empfehlen wir, einen repräsentativen Satz zusammenzustellen und danach die maximale Länge für jedes Element zu bestimmen. Nehmen wir an, wir möchten Sätze in einer Datei speichern, die einen Namen, eine Adresse, Telefonnummer, Vorwahlnummer und einen DM-Betrag enthalten. Ein repräsentativer Satz könnte so aussehen:

DATEN	ANZAHL DER BYTES	MAXIMALE ANZAHL DER BYTES	DATEN-TYP
HANS KAMP	9	20	Zeichenkette
SCHNEIDERSTR. 123	18	20	Zeichenkette
25-2627	7	8	Zeichenkette
0231	2	2	Ganze Zahl
132.84	4	4	einfache Genauigkeit

Da die meisten Namen mehr als 9 Zeichen lang sind, muß das erste Feld lang genug sein, um auch die längsten Namen aufnehmen zu können (20 Zeichen sind üblich).

Das allgemeine Format für eine FIELD-Anweisung ist:

FIELD #Dateinummer, Feldlänge AS Puffer-Zeichenketten-Variable,...,
Feldlänge AS Puffer-Zeichenketten-Variable

Das Pfund-Zeichen (#) vor der Dateinummer kann wahlweise gesetzt oder weggelassen werden.

Die FIELD-Anweisung für unser Beispiel würde so lauten:
FIELD 1, 20 AS NAM $, 20 AS ADR $, 8 AS TEL $, 2 AS VW $, 4 AS DM $

In einer Datei können mehr als ein Feld-"Typ" verwendet werden. Dazu wird der Puffer mit einer anderen Feld-Anweisung neu aufgeteilt. Wenn mehrere verschiedene Feldaufteilungen in einer Datei oder in einem Programm benötigt werden, ist es üblich, Unterprogramme für die Feld-Anweisungen zu schreiben und sie aufzurufen, wenn ein bestimmter Datentyp übertragen werden soll. Das soll später noch gezeigt werden.

Bei der FIELD-Anweisung ist folgendes zu beachten: Der Puffer ist 256 Bytes lang, aber Sie erinnern sich, daß BASIC nur Zeichenketten von 255 Bytes Länge zuläßt. Daher sind nachstehende Anweisungen nicht zulässig:

FIELD 1, 256 AS Q $

oder auch:

FIELD 2, 128 AS Q $, 128 AS R $

Merke: Die Länge eines einzelnen Feldes darf 255 Bytes nicht überschreiten, aber auch die Summe der Längen aller Elemente darf 255 nicht überschreiten.

Datenübertragung in den Puffer - Nachdem der Puffer mit FIELD eingeteilt ist, können Sie die passenden Daten in den Feldern unterbringen. Mehrere Anweisungen führen diese Übertragung aus bzw. kontrollieren sie.

ZEICHENKETTEN - Je nachdem, ob Sie die Zeichenkette rechts- oder linksbündig wünschen, werden Zeichenketten durch zwei verschiedene Anweisungen RSET oder LSET übertragen. Eine linksbündige Zeichenkette beginnt links und füllt das Feld von links nach rechts auf. Wenn die Kette länger als das Feld ist, gehen die überlaufenden Zeichen auf der rechten Seite verloren.

Wenn die Kette kürzer als das zugeteilte Feld ist, wird es auf der rechten Seite mit Leerzeichen aufgefüllt. Eine rechtsbündige Zeichenkette ist genau das Gegenteil. Das Feld wird von rechts nach links aufgefüllt und auf der linken Seite mit Leerzeichen ergänzt, wenn die Kette kürzer ist bzw. abgeschnitten, wenn sie länger ist als das Feld. Das bewirken die Funktionen LSET und RSET:

```
LSET Puffer-Zeichenketten-Variable = Zeichenketten-Ausdruck
RSET Puffer-Zeichenketten-Variable = Zeichenketten-Ausdruck
```

Im Beispiel werden die Variablen NAM$ in der FIELD-Anweisung 20 Bytes zugeteilt. Dem Feld könnte mit LSET oder RSET in der folgenden Weise eine Zeichenkette zugewiesen werden:

```
A$ = "KAMP":B$ = "HANS"
LSET NAM$ = A$ + " , " + B$
```

Der Puffer würde dann wie folgt aussehen (ø entspricht einem Leerzeichen):

```
KAMP , øøHANS øøøøøøøøøø
RSET NAM$ = A$ + " , " + B$
```

würde dagegen folgendes bewirken: øøøøøøøøøø KAMP , øøHANS

LSET oder RSET müssen i m m e r verwendet werden, wenn für direkten Zugriff irgend etwas in den Puffer übertragen werden soll.

NUMERISCHE VARIABLE - MKI$, MKS$ und MKD$ -- Da Zahlen in einer Datei für direkten Zugriff als Zeichenketten gespeichert werden, stehen drei Funktionen zur Verfügung, um die Umwandlung durchzuführen und sie in die jeweiligen Felder im Puffer einzuordnen.

MKI$ - Diese Funktion wandelt eine ganze Zahl in eine 2-Bytes lange Zeichenkette um und schickt sie in das zugewiesene Feld im Puffer.
Die allgemeine Form lautet:

```
MKI$ (ganze Zahl oder ganzzahlige Variable)
```

Beispiel: Angenommen, die Variable VW$ ist mit einer FIELD-Anweisung als 2-Bytes lange Zeichenkette für eine ganze Zahl vereinbart worden. Um die ganze Zahl in eine 2-Byte-Zeichenkette umzuwandeln und sie dem Feld zuzuweisen, können Sie schreiben:

```
A % = 261
LSET VW$ = MKI$ (A %)
```

Damit wird der Inhalt der ganzzahligen Variablen A % in eine 2-Bytes lange Zeichenkette umgewandelt und der Puffer-Variablen VW$ zugewiesen. Der ganzzahlige Wert muß im Bereich - 32768 bis + 32767 liegen. Zusätzlich müssen i m m e r LSET und RSET verwendet werden, wenn irgend etwas in den Puffer übertragen werden soll.

MKS$ - Diese Funktion wandelt Werte mit einfacher Genauigkeit in Zeichenketten von 4-Bytes Länge um und speichert sie in das vorgesehene Feld im Puffer. Die allgemeine Form der Funktion ist:

MKS$ (Variable oder Wert mit einfacher Genauigkeit)

Wenn DM$ mit einer FIELD-Anweisung als 4-Bytes lange Puffervariable definiert wurde, könnten die Übertragungs-Anweisungen so aussehen:

```
A ! = 132.84
RSET DM$ = MKS$ (A !)
```

Damit wird die einfach genaue Zahl in der Variablen A ! in eine 4-Bytes lange Zeichenkette umgewandelt und in das Feld mit der Feldvariablen DM$ übertragen.

MKD$ - Diese Funktion wandelt eine doppelt genaue Variable in eine 8-Bytes lange Zeichenkette um und speichert sie in das für sie bestimmte Feld. Das Format ist:

MKD$ (Variable oder Wert mit doppelter Genauigkeit)

Beispiel:

```
A# = 3.141592625 DØ
LSET ND$ = MKD$ (A#)
```

Diese Anweisung wandelt den angegebenen doppelt genauen Wert in eine 8-Bytes lange Zeichenkette um und überträgt sie in das Pufferfeld, das durch die Feldvariable ND$ bezeichnet wird.

PUT - Bisher haben wir die Anweisungen beschrieben, die den Puffer bereitstellen und Daten in den Puffer übertragen. Wenn der Puffer die Informationen enthält, die Sie als Satz auf die Diskette speichern wollen, braucht nur eine einzige Anweisung ausgeführt werden, um den Inhalt des Puffers auf die Disketten-Datei zu speichern. Die PUT-Anweisung weist den Daten im Puffer eine Satznummer zu und speichert sie in der angegebenen Datei. Das Format ist:

PUT Dateinummer, Satznummer

Die Dateinummer bezeichnet die Datei, die eröffnet worden ist, um die Daten aufzunehmen, und die Satznummer ist die Nummer, die einem der 329 Sätze zugewiesen worden ist. Die Satznummer darf auch weggelassen werden. Wenn die Satznummer nicht angegeben wird, werden die Daten im Puffer zuerst in den Satz geschrieben, danach in den nächsten Satz usw., wobei sich die Satznummer jedes Mal um eins erhöht, wenn PUT ausgeführt wird.
Beispiel:

PUT 1

Wenn man einen Satz mit PUT überträgt, wird auf der Diskette Speicherplatz für alle Sätze von 1 bis zur angegebenen Satznummer reserviert. So bewirkt eine Anweisung PUT 1, 350 eine Fehlermeldung DISK FULL, auch wenn nur ein Satz mit richtigen Daten geschrieben wurde, weil Platz für die Sätze 1 bis 349 reserviert wurde.

LOF - Diese Funktion ergibt den Wert der letzten Satznummer einer bestimmten Datei. Die Anweisung ist besonders nützlich, wenn an eine bestehende Datei Sätze anzufügen sind. Das allgemeine Format der LOF - Funktion ist:

LOF (Dateinummer)

Beispiel:

PRINT LOF (1)

ergibt als Wert die letzte Satznummer der Datei 1. Das verhindert, daß man über das Ende der Datei hinausliest und sinnlose Zeichen in den Puffer geraten.

LESEN DER DATEN VON EINER DATEI MIT DIREKTEM ZUGRIFF

Das Lesen der Daten von einer Datei mit direktem Zugriff ist dem Verfahren zum Schreiben der Daten auf diese Datei sehr ähnlich. Allerdings läuft der Vorgang hier umgekehrt ab.

FIELD - Diese Anweisung wird eingesetzt, um den Puffer für die Daten, die von der Datei kommen, vorzubereiten. (Das Format für die Feld-Anweisung ist das gleiche, wie das zum Speichern der Daten.)

FIELD# Dateinummer, Feldgröße AS Puffer-Zeichenketten-Variable, ... ,
Feldgröße AS Puffer-Zeichenketten-Variable

Mit derselben FIELD-Anweisung, mit der die Daten auf die Diskette gespeichert wurden (oder mit einer gleich aufgebauten), können die Daten von der Diskette gelesen werden, sofern die Daten in derselben Weise eingelesen werden sollen, wie sie gespeichert worden sind. Sie können die Daten aber auch in einem anderen Format lesen, als sie geschreiben worden sind.

Sind zum Beispiel die ersten 40 Zeichen eines Satzes als 3 getrennte Zeichenketten geschrieben worden, können sie doch als eine Zeichenkette zurückgelesen werden:

FIELD 1, 40 AS THE$, ... usw.

Bei einigen Verarbeitungen wird nur ein Teil eines Satzes benötigt. Wenn die gewünschte Information am Anfang des Satzes steht, kann man mit FIELD den Satz sehr einfach so aufteilen, daß die gewünschte Anzahl von Zeichen in eine Puffervariable und der Rest in eine andere, eine Dummy-Variable, kommen.

Wenn jedoch die erforderliche Information mitten im Satz steht, gibt es mehrere Verfahren, die Daten zu holen. Das eine Verfahren weist die nicht benötigten Daten vor der Information mit einer FIELD-Anweisung einer Dummy-Variablen zu, dann wird die gewünschte Information einer zweiten Variablen zugewiesen und der Rest wieder einer Dummy-Variablen. Wenn Sie ein Feld vorliegen haben, das ungefähr so aussieht:

N$	SS$	VW$	TE$
NAME	PERSONAL NR.	VORWAHL	TELEFON-NR.

und Sie nur die Vorwahl-Nummer benötigen, können Sie mit FIELD N $ und SS $ in einer Zeichenkette, VW $ in einer zweiten und TE $ in einer dritten Zeichenkette unterbringen. VW $ steht dann in einem Feld für sich und kann allein gelesen werden. Die Dummy-Variable nach der Information kann in der FIELD-Anweisung auch weggelassen werden.

Wir nehmen an, daß N $ (Name) und SS $ (Personal-Nr.) insgesamt 25 Zeichen enthalten. Um nun die Vorwahl-Nr. im 5. Satz zu lesen, kann man folgendes Programm schreiben:

```
10 CLEAR 500 REM * ES WIRD PLATZ FUER MEHR ALS 50 ZEICHEN BENOETIGT *
20 OPEN"R", 2, "DATA"
30 FIELD 2, 25 AS DUMMY $ , 6 AS VW $
40 GET 2,5
50 PRINT VW $
```

Das Programm druckt die 6 Vorwahl-Zeichen aus dem 5. Satz aus. Die ersten 25 Zeichen sind der Dummy-Variablen zugewiesen und wurden nicht gebraucht. Sie teilen aber den Puffer mit auf, so daß auf die Zeichen 26 bis 31 zugegriffen werden kann.

Ein anderes Verfahren benutzt dieselbe Feld-Anweisung wie bei der Speicherung der Daten und liest dann nur die Zeichenkette ein, die die benötigten Daten enthält, während der Rest übergangen wird.

GET - Diese Anweisung wird angewandt, um einen Satz von der Diskette in den mit FIELD aufgeteilten Puffer für direkten Zugriff zu übertragen. Das Format ist:

GET Dateinummer, Satznummer #

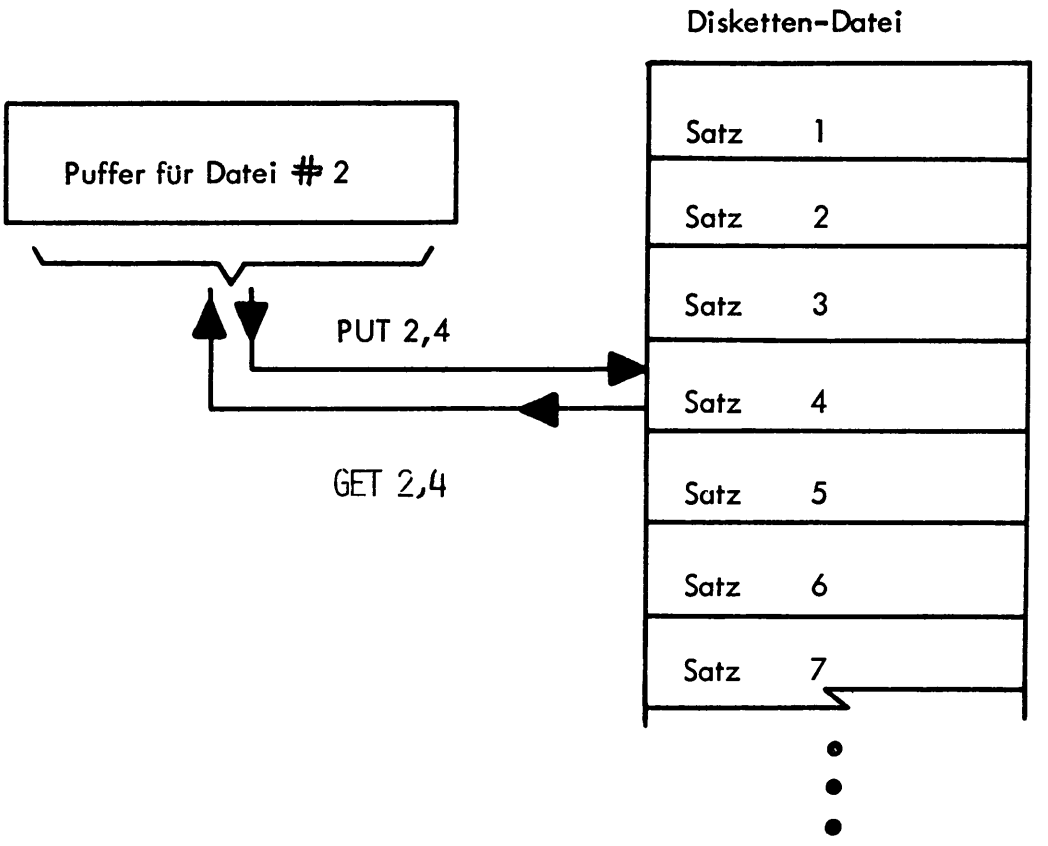
Beispiele:

GET 1, 100 oder GET X, R

Diese Anweisung überträgt den Satz von der angegebenen Datei in den zur Datei gehörenden Puffer.

Die Satznummer kann auch weggelassen werden. Wenn keine Satznummer angegeben ist, ist der erste Satz, auf den die GET-Anweisung zugreift, der Satz Nr. 1. Der nächste Satz ist Nr. 2 und so nach der Reihe fort bis zum Ende der Datei.

DATENFLUSS - DIAGRAMM FÜR PUT/GET



LESEN DER DATEN VOM PUFFER

ZEICHENKETTEN - Zeichenketten können vom Puffer gelesen werden, indem man sich einfach auf die Variable des bestimmten Feldes bezieht.
Zum Beispiel:

Wenn die Zeichenkette AA$ durch die nachstehende Feldanweisung von der Disketten-Datei in den Puffer eingelesen wurde:

```
FIELD 1, 10 AS AA$
```

kann man die Zeichenkette wie folgt vom Puffer holen:

```
B$ = AA$
```

Damit wird der Inhalt des Puffer-Feldes AA$ der Zeichenketten-Variablen B$ zugewiesen. Da die ursprüngliche Zeichenkette entweder als rechtsbündige (RSET) oder linksbündige (LSET) Zeichenkette gespeichert wurde, haben B$ und AA$ den gleichen Aufbau.

NUMERISCHE WERTE - Da numerische Werte als 2, 4 oder 8 Bytes lange Zeichenketten gespeichert sind (unter Verwendung von MKI$, MKS$ und MKD$), müssen sie wieder in Zahlen zurückverwandelt werden, wenn man sie aus dem Puffer holt. Hierzu gibt es wieder drei Anweisungen, welche die drei Zahlentypen bearbeiten:

Ganze Zahlen
Zahlen mit einfacher Genauigkeit
Zahlen mit doppelter Genauigkeit

CVI - Diese Anweisung holt eine Zwei-Byte-Zeichenkette vom Puffer in den Arbeitsspeicher und wandelt sie in eine ganze Zahl der Basis 10 um. Das allgemeine Format ist:

```
CVI (2-Byte-Zeichenkette)
```

Beispiel:

```
A % = CVI (NI$)
```

Damit wird die Zeichenkette, die im Feld durch NI$ vertreten ist, in die entsprechende ganze Zahl umgewandelt und der Variablen A % zugewiesen.

Ein Versuch, eine Zeichenkette, die kleiner als zwei Bytes ist, umzuwandeln, erzeugt eine FC-Fehlermeldung. Wenn die Zeichenkette länger als 2 Bytes ist, werden

die zusätzlichen Bytes unterdrückt, was in einigen Fällen zu einem unkorrekten Wert führt.

CVS - Diese Funktion holt eine 4-Byte-Zeichenkette aus dem Puffer und wandelt sie in den entsprechenden Zahlenwert einfacher Genauigkeit um. Das Format ist wie folgt:

CVS (4-Byte-Zeichenkette)

Beispiel:

A ! = CVS (NS$)

Hiermit wird die 4-Byte-Zeichenkette, die im Puffer durch die Feld-Variable NS$ vertreten ist, in eine Zahl mit einfacher Genauigkeit verwandelt und in der einfach genauen Variablen A ! gespeichert.

Wenn die Zeichenkette kleiner als 4 Bytes lang ist, wird ein FC-Fehler angezeigt. Ist die Zeichenkette länger als 4 Bytes, werden die zusätzlichen Bytes (auf der rechten Seite) nicht berücksichtigt, was einen unkorrekten Wert ergeben kann.

CVD - Diese Funktion holt eine 8-Byte-Zeichenkette aus dem Puffer und wandelt sie in einen doppelt genauen Zahlenwert um. Das Format ist:

CVD (8-Byte-Zeichenkette)

Beispiel:

A# = CVD (ND$)

Dieses Beispiel wandelt die 8-Byte-Zeichenkette, durch die Feld-Variable ND$ im Puffer vertreten, in ihren doppelt genauen Wert um und speichert ihn in der doppelt genauen Variablen A#.

DIE ANWENDUNG VON UNTER-SÄTZEN

Was soll man mit dem leerstehenden Speicherplatz auf einer Diskette anfangen? Denken wir z. B. an das Adressen-Problem zurück, bei dem wir 194 Bytes in einem Satz verschenkt haben. Da nur 62 Bytes benötigt wurden, haben wir Platz für vier (4) Sätze innerhalb eines Puffers. Viermal (4) zweiundsechzig (62) gibt 248 Zeichen. Damit verbleiben nur 8 Bytes je Satz unbenutzt.

Jeder physische Satz (entspricht dem Inhalt des Puffers) enthält vier (4) Unter-Sätze (auch logische Sätze genannt), numeriert von 0 bis 3. Am Anfang des physichen Satzes Nr. 1 findet am die 1. Adresse des 1. Untersatzes. Zu Beginn des zweiten physischen Satzes steht die 5. Adresse (des 5. Untersatzes) usw. Die 5. Adresse kann als der 5. "logische Satz" bezeichnet werden. Wenn die Nummer des logischen Satzes (d. h. des eigentlichen Datensatzes) gegeben ist, können die Nummer des physischen Satzes und die Nummer des Untersatzes berechnet werden.

LR = Nummer des logischen Satzes
physischer Satz = INT ((LR - 1)/4) + 1
Unter-Satz = LR - 4* ((LR - 1)/4) - 1

Es scheint, daß hier für die Feld-Aufteilung (fielding) vier gesonderte Feld-Anweisungen erforderlich sind. Sie lauten wie folgt:

```
FIELD 1, 0 * 62 AS D$, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
FIELD 1, 1 *62 AS D$, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
FIELD 1, 2 *62 AS D$, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
FIELD 1, 3 *62 AS D$, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
```

Hiermit sind die Feldanweisungen für die Unter-Sätze 0 bis 3 gegeben. Beachten Sie die Dummy-Variable D$, die jeweils den Anfang des logischen Satzes enthält, der übergangen werden soll.

Wenn man das Muster dieser Feld-Anweisung studiert, kommt man zu einem allgemeinen Format des Feldes, sofern die Nummer des Unter-Satzes (SR) bekannt ist.

```
FIELD 1, SR *62 AS D$, 20 AS N$, 25 AS AD$, 4 AS PL$, 13 AS OT$
```

Ein vollständiges Programm für die Bearbeitung der Adressendatei POST ist nachstehend wiedergegeben:

```
100 CLEAR 1000:CLS:CLOSE
110 PRINT:INPUT"EINGABE:1 FUER SCHREIBEN, 2 FUER LESEN";N
120 OPEN"R",1,"POST"
130 CLS:ON N GOTO 200,300
200 PRINT:INPUT"NUMMER DES LOGISCHEN SATZES ENGEBEN";LR
210 IF LR=0 THEN  100
220 GOSUB 500:PR=INT((LR-1)/4)+1
230 GET 1,PR:PRINT"PHYSISCHE SATZ-NR. =";PR:PRINT
240 PRINT"NAME";TAB(20);:INPUTA$:LSET N$=A$
250 PRINT"ADRESSE";TAB(20);:INPUTA$:LSET AD$=A$
260 PRINT"POSTLEITZAHL";TAB(20);:INPUTA$:LSET PL$=A$
270 PRINT"ORT";TAB(20);:INPUTA$:LSET OT$=A$
280 PUT 1,PR:GOTO 200
300 PRINT:INPUT"NUMMER DES LOGISCHEN SATZES EINGEBEN";LR
310 IF LR=0 THEN  100
320 GOSUB500:PR=INT((LR-1)/4)+1
330 GET 1,PR:PRINT"PHYSISCHE SATZ-NR. =;PR:PRINT
340 PRINT"NAME";TAB(20);N$
350 PRINT"ADRESSE";TAB(20);AD$
360 PRINT"POSTLEITZAHL";TAB(20);PL$
370 PRINT"ORT";TAB(20);OT$:GOTO 300
490 REM UNTERPROGRAMM ZUM EINORDNEN DES UNTERSATZES INNERHALB
495 REM     DES 256  BYTES  LANGEN  PHYSISCHEN SATZES
500 SR=LR-4*INT((LR-1)/4)-1
510 PRINT "UNTERSATZ-NR. =";SR
520 FIELD 1,SR*62 AS D$,20 AS N$,25 AS AD$,4 AS PL$,13 AS OT$
530 RETURN
```

Im vorstehenden Programm befindet sich die Feld-Anweisung in dem Unterprogramm (Zeile 500 bis 530). Wenn die Nummer des logischen Satzes (LR) gegeben ist (Zeile 200, INPUT LR), berechnet das Programm, in welchem physischen Satz (PR) der logische Satz gefunden werden kann (Zeile 220). Dieser Satz wird in den Puffer gelesen (Zeile 230, GET 1, PR). Als nächstes wird der gesamte Puffer mit FIELD in Felder aufgeteilt, und den Variablen N$, AD$, PL$ und OT$ werden die Werte des aus LR errechneten Unter-Satzes (SR) zugewiesen (Zeile 520).

Das Programm ist in der Lage, jeden logischen Satz einzugeben, wiederzufinden und zu ändern. Ist das Programm gestartet (RUN), wird der Anwender gefragt, ob er auf die Diskette schreiben oder von der Diskette lesen möchte (Zeile 110). Anschließend fragt der Computer nach der Nummer des logischen Satzes (Zeile 300), und hat der Anwender sich für Lesen (2) entschieden, werden die Daten auf dem Bildschirm angezeigt. Zum Schreiben werden die Daten für Name, Adresse, Ort und Postleitzahl abgefragt (Zeile 240 bis 270) und die neuen Daten auf Diskette geschrieben (Zeile 280). Ist die Ein- bzw. Ausgabe der logischen Sätze beendet, tippen Sie für die Nummer des logischen Satzes eine Null ein. Anschließend können Sie wieder zwischen Schreiben/Lesen neu wählen. Soll das Programm gestoppt werden, drücken Sie an dieser Stelle BREAK. Beachten Sie, daß das Programm einen Unter-Satz mitten in einem physischen Satz ändern kann, ohne die Daten in den anderen Unter-Sätzen zu beschädigen.

Beispiel für einen typischen Lauf:

```
>RUN

EINGABE:1 FUER SCHREIBEN, 2 FUER LESEN ?1

NUMMER DES LOGISCHEN SATZES EINGEBEN ?  1
UNTERSATZ-NR.  =  0
PHYSISCHE SATZ NR.  =  1

NAME                    ? HANS SCHMIDT
ADRESSE                 ? KAMPSTR. 32
POSTLEITZAHL            ? 4400
ORT                     ? MUENSTER

NUMMER DES LOGISCHEN SATZES EINGEBEN ?  6
UNTERSATZ-NR.  =  1
PHYSISCHE SATZ NR.  =  2

NAME                    ? WALTER MEYER
ADRESSE                 ? BRASSERTSTR. 15
POSTLEITZAHL            ? 4600
ORT                     ? DORTMUND

NUMMER DES LOGISCHEN SATZES EINGEBEN ?  0

EINGABE:1 FUER SCHREIBEN, 2 FUER LESEN ?2

NUMMER DES LOGISCHEN SATZES EINGEBEN ?  6
UNTERSATZ-NR.  =  1
PHYSISCHE SATZ-NR.  =  2

NAME                    WALTER MEYER
ADRESSE                 BRASSERTSTR. 15
POSTLEITZAHL            4600
ORT                     DORTMUND

NUMMER DES LOGISCHEN SATZES EINGEBEN ?  0

EINGABE: 1 FUER SCHREIBEN, 2 FUER LESEN ?

                                        <BREAK>

BREAK IN 110

READY

>
```

DISK - FEHLERMELDUNGEN

DISK BASIC gibt die Fehlermeldungen, die im LEVEL II nur aus 2 Buchstaben bestanden, als vollständige englische Texte aus. Der Fehler M O - ERROR heißt jetzt MISSING OPERAND ERROR usw. Eine weitere Gruppe von Meldungen zeigt Fehler beim Diskettenbetrieb an. Solche Fehler können u. U. Dateien zerstören, wenn sie häufig auftreten. Deshalb wird die Fehlerbehandlung der Diskettenfehler nicht unterstützt. Alle anderen Fehler können im LEVEL II mit "ON ERROR GOTO" behandelt werden.

FEHLERMELDUNGEN

Kode	Fehlermeldung	Erklärung
50	FIELD OVERFLOW	In einer FIELD-Anweisung wurden mehr als 255 Bytes vereinbart.
51	INTERNAL OVERFLOW	Im Disketten-Betriebssystem selbst oder bei der Ein-/Ausgabe mit Disketten ist ein Fehler aufgetreten.
52	BAD FILE NUMBER	Die angegebene Dateinummer wurde nicht oder mit anderen Parametern definiert.
54	FILE NOT FOUND	Eine nicht vorhandene Datei soll bearbeitet werden.
55	BAD FILE MODE	tritt auf, wenn eine wahlfreie Datei sequentiell bearbeitet werden soll oder umgekehrt.
57	DISK I/O ERROR	Bei der Datenübertragung zwischen System und Diskette trat ein Fehler auf.
58	FILE ALREADY EXISTS	tritt auf, wenn mit RENAME eine Datei umbenannt werden soll und der neue Name schon für eine Datei vergeben ist.
59	SET TO NON-DISK STRING	LSET oder RSET benutzen eine Zeichenkettenvariable, die nicht mit FIELD vereinbart wurde.
61	DISK FULL	Der gesamte verfügbare Speicherplatz einer bestimmten Diskette ist verbraucht.
62	INPUT PAST END	Von einer sequentiellen Datei sollten mehr Daten gelesen werden als vorhanden.
63	BAD RECORD NUMBER	Die Satznummer für einen wahlfreien Satz liegt außerhalb des zulässigen Bereichs (1-340).
64	BAD FILENAME	Ein unzulässiger Dateiname sollte vereinbart werden.
65	MODE MISMATCH	Eine wahlfreie Datei wurde wie eine sequentielle eröffnet oder umgekehrt.
66	DIRECT STATEMENT IN FILE	Beim Laden eines in ASCII gespeicherten Programms wurde eine direkte Anweisung gelesen.
67	TOO MANY FILES	Auf einer Diskette sollten mehr als 48 Dateien eingerichtet werden.

SYSTEM-HILFSPROGRAMME FÜR DEN TRSDOS-BETRIEB

Das Betriebssystem hat die Kontrolle, wenn die Meldung DOS READY auf dem Bildschirm erscheint. Der Rechner ist automatisch in diesem Modus, wenn er eingeschaltet oder wenn CMD"S" in DISK BASIC eingegeben wird. TRSDOS enthält mehrere Hilfsprogramme, die jederzeit zur Verfügung stehen, wenn DOS READY angezeigt wird. Keines dieser Dienstprogramme ist unter DISK BASIC verfügbar und keines der Hilfsprogramme arbeitet, wenn CMD"T" eingegeben wird (d. h. die Uhr ausgeschaltet ist).

Dateien werden nachfolgend in der Form - Dateiname 1, Dateiname 2 - dargestellt. Der Dateiname kann, wenn erforderlich, durch eine Ergänzung, eine Gerätenummer und ein Kennwort erweitert werden (wie es in dem Abschnitt DATEI-NAMEN (Seite 17) dieses Handbuches beschrieben ist).

AUTO Dateiname 1

lädt jede CMD-Datei oder jedes Hilfsprogramm nach Einschalten des Gerätes. (BASIC-Programme werden nicht geladen, da nach dem Einschalten BASIC selbst noch nicht eingelesen ist.) Dieses Programm ist eine sehr wertvolle Hilfe für bestimmte Einsätze des TRS-80. Wenn man AUTO und den Namen des Programmes eingibt, z. B. BASIC, wird jedesmal, wenn man den TRS-80 einschaltet, automatisch das Programm (z. B. BASIC) geladen. Nur ein Parameter ist zulässig.

HINWEIS: Wenn AUTO angewiesen wird ein fehlerhaftes Programm zu laden, bleibt der Computer scheinbar stehen, während er versucht das schlechte Programm zu laden. In diesem Fall ist ein Handverfahren anzuwenden. Man drückt hierbei die ENTER-Taste, während der TRS-80 eingeschaltet wird und unterdrückt somit die Durchführung des AUTO-Befehls. Dann wird AUTO eingetastet und ENTER gedrückt. AUTO ohne Argument bringt den TRS-80 in den normalen Zustand nach dem Stromeinschalten.

BACKUP (Ergänzung in Vers. 2.1 siehe S. 51)

Backup kopiert den Inhalt einer Diskette auf eine leere Diskette. Der Programmname BACKUP wird nach DOS READY eingegeben. Das Programm fragt zuerst nach dem Diskettengerät, in dem die zu kopierende Diskette eingelegt ist (SOURCE DRIVE). Antworten Sie mit der betreffenden Gerätenummer Ø bis 3. (Gerät Ø liegt dem Erweiterungs-Interface am nächsten.) Dann wird nach dem Gerät mit der leeren Diskette gefragt (DESTINATION DRIVE). Die leere Platte muß eingelegt sein - formatiert oder unformatiert; BACKUP wird dann die Formatierung automatisch vornehmen. Wenn die leere Diskette irgendwelche Daten enthält, bricht BACKUP den Lauf ab. Zur Wiederverwendung einer Diskette kann man entweder mit einem Magneten über die Diskette streichen oder besser einen Tonkopf-Entmagnetisierer, wie den von Radio Shack, Katalognummer 44 - 210, verwenden. Anschließend wird nach dem Datum der Übertragung gefragt, das in der Form MM/DD/YY (Monat, Tag, Jahr) eingegeben werden muß, BACKUP wird die Platte anschließend formatieren (wenn erforderlich), prüfen und kopieren.

Danach übergibt BACKUP dem DOS die Steuerung zurück. Haben Sie nur eine Diskette, beantworten Sie beide Fragen, SOURCE DRIVE und DESTINATION DRIVE, mit Ø (Null). Das System führt dann ein Ein-Disketten-BACKUP durch, das ein mehrfaches Austauschen der Disketten, vor und zurück, erforderlich machen kann.

COPY

Die Anweisung:

COPY Dateiname 1 TO Dateiname 2 erzeugt ein Duplikat der Datei 1 mit Namen und Erweiterungen des Dateinamens 2.
Beispiel:

COPY JOURNAL4 : 1.XYZ TO DOKUMENT:3

(Bei der Eingabe sind die Leerstellen zu beachten.)

Eine Datei mit dem Namen JOURNAL4 auf dem Diskettengerät 1 mit dem Kennwort XYZ wird hiermit auf die Diskette im Gerät 3 unter dem Namen DOKUMENT dupliziert. Ein BASIC-Programm können Sie kopieren, indem Sie es von einer Diskette mit LOAD laden und auf eine andere mit SAVE speichern.

DOS DEBUGGER (Fehlersuche) (Ergänzung in Vers. 2.1 siehe S. 51)

Die Fehlerroutine wird eingesetzt, indem man zur DOS-Zeit DEBUG eintippt. Die Fehlerroutine arbeitet, wenn 1.) die BREAK-Taste gedrückt wird und 2.) ein Programm geladen wird. Im DISK BASIC tippen Sie CMD"D", und die Fehlerroutine arbeitet sofort; mit G und ENTER kehren Sie in das BASIC zurück. CMD"D" funktioniert nicht, wenn die Uhr (REAL-TIME-CLOCK) bei Kassettenoperationen mit CMD"T" ausgeschaltet worden ist.

Die folgenden Befehle kontrollieren die Arbeit des Fehler-Hilfsprogrammes:

D nnnn	zeigt den Speicherinhalt der Adresse nnnn an.
M nnnn ⌀ xx	verändert den Wert an der Adresse nnnn zu xx; Leerzeichen erhöht die Adresse.
R rr ⌀ nnnn	lädt das Registerpaar rr mit nnnn.
X	Normaler Anzeigemodus (Register und Speicher), beendet auch jede unvollständige Eingabe.
S	Vollständiger Bildschirmanzeigemodus (nur Speicher).
A	zeigt den ganzen Speicher in ASCII an.
H	zeigt den ganzen Speicher hexadezimal an.
;	erhöht die Speicheranzeige um 1 Block.
-	setzt die Speicheranzeige um 1 Block zurück.
G nnnn (,bbbb,cccc)	verzweigt zur Speicheradresse nnnn, bei Bedarf mit Unterbrechungen bei bbbb und cccc. Wenn nnnn nicht angegeben ist, wird der Lauf bei der letzten Unterbrechung wieder aufgenommen.
I	führt eine Anweisung nach der anderen aus.
C	Einzelne CALL-Instruktionen werden vollständig ausgeführt, sonst wie I.
U	unterbricht die Anzeige durch Drücken von X oder BREAK
G (ENTER)	verzweigt zurück zum BASIC, wenn der Debugger mit CMD"D" aufgerufen wurde.

DIRØ: (GERÄTE-NUMMER) - zeigt alle Dateinamen (mit ihren Ergänzungen) an, die auf der Diskette mit der genannten Geräte-Nummer stehen. (Siehe Ergänzung, Seite 53.)
Beispiel:

DIRØ:1 zeigt die Namen aller Dateien auf dem Gerät 1 an.

FORMAT - formatiert eine neue oder magnetische gelöschte Diskette nach dem TRSDOS-Standard von 35 Spuren, 10 Sektoren mit je 256 Bytes (siehe auch Ergänzungen Seite 51). Disketten, die bereits genutzt wurden, müssen erst mit einem Magnet oder besser mit einem hochwertigen Tonkopf-Entmagnetisierer, wie z. B. von Radio Shack, Kat.-Nr. 44-210, gelöscht werden. Alle Spuren werden geprüft und Sektoren, die nicht mehr zu verwenden sind, als nicht anwendbar im System vermerkt. Von dem Hilfsprogramm FORMAT wird eine Gerätenummer verlangt; antworten Sie mit einer Nummer zwischen Ø und 3, unter der die entsprechende Diskette zu finden ist. Gerätenummer Ø ist die Diskette, die der Erweiterungs-Schnittstelle am nächsten liegt. Anschließend wird der Name der Diskette verlangt, und Sie können mit jedem Namen bis zu 8 Buchstaben Länge antworten. Dieser Name wird angezeigt, wenn man mit DIR die Liste der Dateien ausgeben läßt. Die nächste Eingabe ist das Datum in der Form MM/TT/JJ (Monat/Tag/Jahr). Schließlich muß noch das Haupt-Kennwort eingegeben werden. Dieses Kennwort kann jedes Wort mit bis zu 8 Zeichen sein. In einer zukünftigen Version von DOS soll es zur Suche von Dateien herangezogen werden, deren Kennwort in Vergessenheit geraten ist. Die nächste Frage ist für das TRACK LOCKOUT vorgesehen (Ausschluß von Spuren). Wenn Sie nicht wollen, daß die Spuren ausgeschlossen werden, müssen Sie mit N antworten. Wenn die Diskette in einem Ihnen bekannten Teil beschädigt ist, können Sie den intakten Teil verwenden und den schlechten Teil mit der Antwort Y ausschließen. Die nächste Frage lautet dann WHICH TRACK ? (welche Spur), darauf antworten Sie mit den

Nummern der beschädigten Spuren, durch Kommas getrennt, oder mit einer Folge von Nummern, durch Bindestriche (-) getrennt. Die letzte Frage lautet: FORMAT THE LOCKED OUT TRACKS?, die Sie mit Y oder N beantworten, wonach das Formatieren beginnt.

CLOCK

zeigt die Uhrzeit auf dem Bildschirm an.
Sie kann nur durch Abschalten des Stromes oder durch den Befehl CMD"T" in DISK BASIC abgestellt werden.

TIME␣hh:mm:ss	(Stunde/Minute/Sekunde) gibt die Tageszeit ein
DATE␣mm/tt/jj	(Monat/Tag/Jahr) gibt das Datum ein
TRACE	zeigt den Programmzähler auf dem Bildschirm an. Die Anzeige kann nur durch Abschalten des Stromes oder durch Reset gelöscht werden.

DOS-FEHLER MELDUNGEN

CRC-FEHLER - zeigt eine Beschädigung auf der Diskette an. Die Spur wird gesperrt und kann vom System nicht verwendet werden. Dadurch wird der Speicherplatz auf der Diskette eingeschränkt, doch nicht ihre Anwendung.

FILE ACCESS DENIED (Zugriff zur Datei wird verwehrt) - Die Datei besteht, aber das richtige Kennwort ist nicht angegeben worden.

FILE NOT FOUND (Datei nicht gefunden) - Datei gibt es nicht, oder der Na- ist nicht vollständig angegeben worden. Denken Sie daran, daß die Datei-Erweiterungen (/BAS/SYS usw.) mit angegeben werden müssen, wenn diese bei der Erstellung der Datei genannt worden sind.

ERGÄNZUNGEN FÜR DEN TRSDOS-BETRIEB, VERSION 2.1

Einige Anwender werden mit den Bezeichnungen "Version 2.1", "erste Ausgabe" usw. nichts anzufangen wissen. Sie werden diese Bezeichnungen öfters hören, weil wir das TRSDOS-System laufend verbessern wollen. Daher wollen wir diese Bezeichnung näher erläutern.

Eine neue Version ist eine umfassende Änderung und Erweiterung der vorhergehenden Version. Zum Beispiel können neue Hilfsprogramme, höhere Programmiersprachen usw. in der neuen Version enthalten sein. Die Versionen werden fortlaufend mit 1, 2, 3 usw. numeriert.

Eine neue Ausgabe (release) einer Version ist dagegen lediglich eine Verbesserung der vorhergehenden Ausgabe. Diese neue Ausgabe enthält im allgemeinen Lösungen früher aufgetretener Probleme, Verbesserungen und Erweiterungen von Befehlen usw. Die Ausgaben werden fortlaufend mit .1, .2, .3 usw. numeriert.

So steht "Version X.Y" als Kurzbezeichnung für Ausgabe Y von Version X.

> HINWEIS: Großbuchstaben, Leerzeichen (␣) und Satzzeichen müssen wie angegeben eingetippt werden. Kleinbuchstaben, bestimmte Parameter und wahlfreie Ergänzungen werden so eingesetzt, wie sie in der nachstehenden Befehlsbeschreibung angegeben sind. Beachten Sie auch die bereits vorliegenden Angaben über Dateibezeichnung, Gerätebezeichnung, Kennwort (password) usw. in der vorläufigen Erläuterung für TRSDOS 2.0 (Seiten 1 - 50 und B/1 - B/2).

Verbesserungen von Problemen in 2.0

1. Die Zuweisung von Datei-Speicherplatz auf Disketten und das Löschen (SAVE und KILL) funktionieren jetzt einwandfrei. Die Verwendung einer fast vollen oder vollen Diskette ist jetzt möglich. (Siehe B/2, Punkt 15.)

2. FORMAT- und BACKUP-Anweisungen sind jetzt so verbessert, daß sie nicht mehr automatisch das System nach der Ausführung in den alten Zustand versetzen. Das gibt Ihnen die Möglichkeit, nachzuschauen, ob irgendeine Fehlermeldung erzeugt wurde. Mit ENTER setzt man das System wieder in den alten Zustand.

3. Das DEBUG-Hilfsprogramm kann (und sollte auch) vom Kernspeicher (RAM) gelöscht werden, wenn Sie es benutzt haben, indem RESET gedrückt wird und dann folgende Befehle eingegeben werden:

   DEBUG Ø (OFF) (ENTER)
   DIR (ENTER)

   Diese Befehlsfolge verhindert den unkontrollierten Wiedereintritt in das DEBUG-Hilfsprogramm.
   HINWEIS: Um das DEBUG-Hilfsprogramm in TRSDOS auszugeben, tippen Sie:

   G402D (ENTER)

4. Der LIST-Befehl setzt den Cursor an die richtige Stelle.

APPEND␣Dateibezeichnung1␣TO␣Dateibezeichnung2

fügt die anfangs stehende Datei 1 an die Datei 2 an. Die erste Datei wird dabei nicht verändert. Dieser Befehl ist hauptsächlich für die Anwendung bei Daten-Dateien gedacht.

Beispiel:

APPEND␣DATA7␣TO␣DATA8

ATTRIB␣Dateibezeichnung␣(attrib,ACC=psw1,UPD=psw2,PROT=param)

Hierbei bedeuten:

attrib	=	I (für invisible, unsichtbar). Wahlfrei; wird meist weggelassen
psw 1	=	neues Zugriffs-Kennwort. Wahlfrei
psw 2	=	neues Veränderungs-Kennwort. Wahlfrei
param	=	eine der folgenden Parameter: KILL, RENAME, WRITE, READ, EXEC(ute). Wahlfrei

Dieser Befehl gibt Ihnen die Möglichkeit, der gekennzeichneten Datei zwei Kennworte zuzuweisen: ein Zugriffs (access)-Kennwort, das den Zugriff zur Datei gemäß der Festlegung im PROT-Parameter ermöglicht; und ein Veränderungs (update)-Kennwort, das den Gesamt-Zugriff zu der Datei ermöglicht. Beachten Sie, daß die Schutz-Ebenen eine Hierachie bilden, und daß jede Ebene den Zugriff zu allen niedrigeren Ebenen einschließt:

KILL (Löschen)	schließt jeden Zugriff ein.
RENAME (neu benennen)	schließt WRITE, READ, EXEC ein.
WRITE (schreiben)	schließt READ und EXEC ein.
READ (lesen)	schließt EXEC ein.
EXEC (ausführen)	ermöglicht nur die Ausführung (EXEC).

Die Angabe I (unsichtbar) ist wahlfrei. Sie macht die Datei für den normalen DIR-Befehl unsichtbar (siehe DIR, Seite 53).

Seien Sie bei der Zuweisung von Schutz-Ebenen an Dateien sehr vorsichtig. Wenn Sie das Kennwort vergessen, kann weder auf die Datei zugegriffen, noch kann sie verändert werden, außer über den PROT-Befehl und das Haupt-Kennwort (Master-password), das bei PROT (Seite 54) beschrieben ist.

Beispiel:

ATTRIBØPROG1/BAS.PSWØ(ACC=PSW1,UPD=PSW2,PROT=READ)

weist der Datei PROG 1 neue Kennworte zu

ATTRIBØPROG1/BAS.PSW2Ø(ACC=,UPD=)

löscht die Kennworte von PROG 1.

DEVICE

Dieser Befehl besitzt keine Argumente oder Parameter. Er listet einfach alle angeschlossenen I/O-Geräte (Ein-/Ausgabe-Geräte) auf: KI (Computer mit Tastatur), DO (Bildschirm) und PR (Drucker).

DIR - die Anweisung kann wie folgt erweitert werden:

DIR[0]:d[0] (param1,param2,param3)

Hierin ist:

d eine wahlfreie Nummer für das Diskettengerät (d = 0, 1, 2 oder 3).

Die Parameter sind wahlfrei und können einzeln, kombiniert oder insgesamt eingesetzt werden:

S zeigt alle System- und sichtbaren Dateien an.
I zeigt alle unsichtbaren und Nicht-System-Dateien an.
A gibt den Speicherplatz für alle angezeigten Dateien an.

Disketten-Speicherplatz wird wie folgt angegeben: LRL (logische Satz-Länge), EOF (Ende der Datei) und GRANS (Anzahl von Granulen, die belegt sind; jedes Granule = 1/2 Spur oder 1.25 K).
Beispiel:

DIR
zeigt die Benutzer-Dateien (keine System-Dateien, keine unsichtbaren Dateien) auf Diskettengerät 0 an.

DIR[0]:1[0] (I,S,A)
zeigt alle Dateien auf Diskettengerät 1 einschließlich der dazugehörigen Speicherplatzbelegung an.

HINWEIS: DIR zeigt neben allen Benutzer-Dateien, deren Kennwort nicht Null ist, ein P an.

Wenn die DIR-Liste nicht auf den Schirm paßt, werden nur die ersten 12 Zeilen angezeigt. Drücken Sie (ENTER), wenn Sie die nächsten 16 Zeilen sehen wollen.

DUMP

Allgemeine Form:

DUMP␣Dateibezeichnung␣(START=X'AAAA',END=X'BBBB',TRA=X'CCCC')

Hierin sind AAAA, BBBB, CCCC hexadezimale Adressen im Hauptspeicher (RAM), und AAAA, BBBB liegen über 6FFF (Speicherplatz dezimal 28671).

Dieser Befehl schreibt den Inhalt des Speichers von der angegebenen START-Adresse bis zur END-Adresse auf Diskette und gibt eine TRANSFER-Adresse an, von der an die Ausführung des Programms beginnt, wenn die Datei zur Ausführung aufgerufen wird.

Der Hauptzweck des DUMP-Befehls ist, Diskettendateien zu eröffen, die Programme in Maschinensprache enthalten (erstellt mit TBUG, Editor/Assembler, POKE-Anweisungen in BASIC usw., mit dem SYSTEM-Befehl von Band geladen).

Die meisten System-Kassetten in LEVEL II können nicht unter DOS ausgeführt werden, da Unterschiede in LEVEL II und DOS hinsichtlich der Hauptspeicher-Aufteilung bestehen. Weitere Angaben hierzu finden Sie unter Punkt 6 in Anhang C.

FREE

Diese Funktion erfordert keine Argumente oder Parameter. Sie zeigt die Anzahl der freien Speicherplätze an, die in allen augenblicklich in Betrieb befindlichen Disketten-Geräten zur Verfügung stehen, aufgeteilt nach Anzahl der freien Dateien und Anzahl der freien Granulen.

Jede Diskette kann bis zu 48 Benutzer-Dateien speichern.

LIB

Erfordert keine Parameter oder Argumente. LIB zeigt alle System-Befehle an, die in TRSDOS zur Verfügung stehen.

LOAD

LOAD Dateibezeichnung

lädt die angegebene Datei (die Z-80-Maschinensprache (Objekt-Kode) enthalten muß) in den RAM-Hauptspeicher und gibt dann die Kontrolle an TRSDOS zurück. Die angegebene Datei ist gewöhnlich mit einem DUMP- oder TAPEDISK-Befehl aufgebaut worden.

PROT

PROTØ:dØ(PW,param)

Hierin sind: d eine wahlfreie Disketten-Gerät-Nummer, PW ein wahlfreier bestimmter Parameter und "param" ein wahlfreier Parameter, entweder LOCK oder UNLOCK.

Dieser Befehl ändert den Sicherheitsstatus von allen Nicht-System-Dateien auf dem angegebenen Disketten-Gerät. Wenn kein Gerät angegeben ist, wird Gerät Ø angenommen.

Um PROT zu verwenden, müssen Sie das Haupt-Kennwort der Diskette kennen, das während FORMAT oder BACKUP angegeben wurde. (Ihre TRSDOS-Diskette hat das Kennwort PASSWORD.)

Um das Haupt-Kennwort zu ändern, müssen Sie PW angeben. Um alle Kennwörter von allen Benutzer-Dateien zu löschen, müssen Sie UNLOCK einsetzen. Um allen Benutzer-Dateien das Haupt-Kennwort zuzuteilen, müssen Sie LOCK eingeben.
Beispiel:

PROT␣:1␣ (UNLOCK)

Nachdem Sie diesen Befehl eingegeben haben, fragt TRSDOS nach dem Haupt-Kennwort. Wenn Sie das richtige Wort eingegeben haben, werden alle Kennworte von den Benutzer-Dateien auf der Diskette in Gerät 1 gelöscht.

PROT (PW,LOCK)

Nachdem Sie das Haupt-Kennwort richtig eingegeben haben, wird TRSDOS Sie auffordern, ein neues Haupt-Kennwort einzugeben. Dann wird dieses neue Kennwort der Diskette im Gerät 0 und allen auf ihr vorhandenen Benutzer-Dateien zugeordnet.

RENAME

RENAME␣Dateibezeichnung1␣TO␣Dateibezeichnung2

ermöglicht Ihnen, den Namen (und/oder die Erweiterungen) der zuerst angegebenen Datei durch den Namen der zweiten Datei zu ersetzen. Mit RENAME können Sie keine Kennworte ändern oder hinzufügen.

VERIFY

VERIFY␣(param)

In diesem Befehl ist param = ON oder OFF. VERIFY (ON) veranlaßt TRSDOS, alle Disketten-Ausgabe-Operationen (z. B. Dateien von BASIC ausgeben) zu überprüfen. VERIFY (OFF) schaltet diese Funktion aus.

VERIFY beeinflußt keine Disketten-Ausgaben auf System-Datein; diese werden immer überprüft. Beachten Sie auch, das TRSDOS sich beim Einschalten im Zustand VERIFY (OFF) befindet.

TAPEDISK

Hiermit besitzen Sie ein Hilfsprogramm, mit dem Sie die SYSTEM-Kassetten von Radio Shack in den RAM-Hauptspeicher laden und anschließend das Programm aus dem RAM auf eine bestimmte Disketten-Datei ausgeben können.

Versuchen Sie nicht, Kassetten-Dateien einzulesen, die unterhalb der hexadezimalen Adresse 54F4 (Speicherplatz 21748) gespeichert sind. Näheres finden Sie in Anhang C, Punkt 6.

TAPEDISK (ENTER)

lädt das TAPEDISK-Programm und zeigt ein eigenes Prompt-Zeichen an:

?_

Sie antworten mit einem der drei folgenden Befehle:

C (ENTER)

stellt den Kassetten-Recorder an (Recorder Nr. 1, wenn beide angeschlossen sind) und lädt eine SYSTEM-Kassette in den RAM-Hauptspeicher. Wenn die Datei eingelesen ist, kehrt das Prompt-Zeichen zurück. Sie können dann eine andere SYSTEM-Kassette ins RAM laden, indem Sie wieder C (ENTER) eintippen - oder einen anderen Befehl eingeben.

F/Dateibezeichnung/AAAA/BBBB/CCCC (ENTER)

ist der Befehl zum Ausgeben auf Diskette. Hierin bedeutet "Dateibezeichnung" den gewählten Dateinamen, der eine Disketten-Geräte-Nummer enthalten muß. AAAA ist die Start-Adresse im RAM, BBBB die End-Adresse und CCCC der Anfangspunkt für die Ausführung des SYSTEM-Programmes. Alle Adressen sind in hexadezimaler Form anzugeben. Nach dem Speichern kommt das Prompt-Zeichen zurück.

E (ENTER)

führt zum TRSDOS zurück.

DISKDUMP/BAS

Mit diesem besonderen Hilfsprogramm können Sie eine Prüfung Sektor für Sektor in jeder gewünschten Benutzer-Datei vornehmen. DISKDUMP/BAS ist ein BASIC-Programm. Wenn Sie es anrufen wollen, müssen Sie sich im DISK BASIC befinden und nach den Fragen FILES? und MEMORY SIZE? (ENTER) gedrückt haben.

Das Programm ist für die Ausgabe mit dem Zeilendrucker geschrieben. Wenn Sie keinen Drucker angeschlossen haben, können Sie alle LPRINT- in PRINT-Anweisungen ändern, und die Ausgabe läuft über den Bildschirm. Löschen Sie dann alles nach 160 GET 1,SN (in Zeile 160).

Das Programm fragt Sie nach dem Dateinamen und anschließend nach dem Sektor, den Sie zu prüfen wünschen. Sie geben die Antworten ein und drücken (ENTER). Sie können das zweite PROMPT auch einfach mit (ENTER) beantworten, dann erfolgt die Sektor-Prüfung nacheinander, angefangen von Sektor 1. Beachten Sie, daß bei dem Versuch, einen höheren Sektor zu prüfen als in der Datei vorhanden ist, keine Fehlermeldung erzeugt wird. Die Sektoren erscheinen als Bytes mit dem Wert Null.

Von den Sektoren werden jeweils 16 Bytes auf einmal ausgedruckt. Diese 16 Bytes werden zuerst in hexadezimaler Schreibweise gedruckt und dann in dem entsprechenden ASCII-Kode. Die ASCII-Darstellung ist in "!"-Zeichen eingeschlossen. Wenn der hexadezimale Kode keinem alphanumerischen Zeichen entspricht, wird in der ASCII-Darstellung ein Punkt für dieses Byte ausgegeben.

Dieses Programm kann für Ihr Verständnis der Datenspeicherung auf Diskette (direkt oder sequentiell) sehr hilfreich sein.

CLOCK (param)

Hierin ist param entweder ON oder OFF. Wenn kein Parameter eingegeben wird, nimmt der Computer ON an. Diese Funktion war schon in Version 2.0 vorhanden, doch ist hier die OFF-Funktion hinzugekommen.

TRACE (param)

Hierin ist param entweder ON oder OFF. Wird kein Parameter eingegeben, nimmt der Computer ON an. TRACE gab es schon in Version 2.0, doch ist die Möglichkeit hinzugekommen, OFF einzugeben.

A/ Zusammenfassung von DISK BASIC 1.1 und TRSDOS VERSION 2.1

TRSDOS VERSION 2.1

Hilfsprogramme	Wirkung	Beispiele
APPEND	fügt eine Datei an das Ende einer anderen (vor allem bei Daten-Dateien)	APPEND/DATA7/TO/DATA8
ATTRIB	vergibt oder ändert Kennworte in 5 Stufen zum Schutz von Dateien, um unbefugten Zugriff auf Dateien zu verhindern.	ATTRIB/PROG1/BAS.KENW/ (ACC=KEN1,UPD=KEN2, PROT=READ)
AUTO (Dateiname 1)	lädt beim Ausschalten automatisch jede gewünschte CMD-Datei oder jedes Hilfsprogramm	AUTO BASIC AUTO POST
BACKUP	kopiert eine ganze Diskette auf eine andere, leere Diskette. (Bei Systemen mit nur einer Diskettenstation müssen die Disketten dabei mehrfach ausgetauscht werden.)	BACKUP
CLOCK	zeigt in der oberen rechten Ecke des Bildschirmes die Uhrzeit an. Sie kann nur durch Ausschalten des Stromes oder den Befehl CMD"T" in DISK BASIC abgestellt werden.	CLOCK
COPY Dateiname 1 TO Dateiname 2	kopiert Datei 1 und gibt ihr den Namen (mit Erweiterungen) von Datei 2	COPY JOURNAL 4:1. XYZ TO DOKM:3 COPY ADR TO NAME

Hilfsprogramme	Wirkung	Beispiele
DATE␣mm/tt/jj	stellt das Datum ein, so daß mit TIME$ in DISK BASIC darauf zurückgegriffen werden kann.	DATE␣11/04/79
DEBUG	Testhilfe für Maschinenkode-Programme. Sie können damit Unterbrechungspunkte (Breakpoints) setzen, Instruktionen in Einzelschritten ausführen, Register während der Ausführung überwachen oder den Speicherinhalt hexadezimal oder in ASCII anzeigen lassen. (Siehe Seite 49)	DEBUG
DIR␣: Geräte-Nr.␣ (I, S, A)	zeigt ein Verzeichnis aller Dateien des angegebenen Diskettengeräts an, auf Wunsch mit der Länge der Sätze, dem Dateiende und dem belegten Platz.	DIR␣:1␣ (I,S,A) DIR␣:2␣ (A)
DUMP␣Dateibezeichnung	speichert den Maschinenkode-Programm aus dem Hauptspeicher auf Diskette. (Siehe Seite 51.)	DUMP␣SYS1␣(START=X6000, END=X6200, TRA=X6080)
FORMAT	formatiert und prüft eine leere Diskette. Beschädigte Sektoren werden markiert.	FORMAT
FREE	zeigt den freien Speicherplatz auf den Disketten aller Laufwerke an.	FREE

Hilfsprogramme	Wirkung	Beispiele
KILL	Löscht eine Datei, so daß deren Platz wieder neu genutzt werden kann.	KILL "ALTDAT/BAS. PSW1 (BASIC) KILL␣ALTDAT/BAS. KENNWORT (TRSDOS)
LIB	zeigt alle Systembefehle des TRSDOS an.	LIB
LIST	listet eine Datei über den Video-Bildschirm auf.	LIST␣PROG 1/TXT (TRSDOS)
LOAD Dateibezeichnung	lädt ein Maschinenkode-Programm von Diskette in den Speicher.	LOAD␣SYS 2
PRINT	listet eine Datei über den Zeilendrucker auf.	PRINT␣SEQCHEK/TXT (TRSDOS)
PROT␣d␣ (PW,param)	ändert den Schutzstatus aller Benutzer-Dateien auf einer Diskette.	PROT␣1:␣ (UNLOCK)
RENAME␣Dateibez. 1 ␣Dateibez. 2	ändert den Namen einer Datei.	RENAME␣DATEI1␣TO␣DATEI2
TAPEDISK	lädt Systemkassetten über den Hauptspeicher in die angegebene Disketten-Datei.	TAPEDISK E C F␣SYS3/SYS:0␣6000␣6200␣6080
TIME	stellt die Tageszeit für die CLOCK-Anzeige und die BASIC-Funktion TIME$ ein.	TIME 23:59:30
TRACE	zeigt den Programmzähler auf dem Video-Bildschirm an.	TRACE TRACE (OFF)

Bestellnummer für Disketten

TRSDOS - Diskette	Bestell-Nr.	26 - 310
Leere Disketten, einzeln	Bestell-Nr.	26 - 305
Leere Disketten, (Dreierpackung)	Bestell-Nr.	26 - 405

DISK BASIC von Radio Shack

TRSDOS lädt immer dann das DISK BASIC, wenn der Befehl BASIC eingegeben wird. Mit dem Befehl CMD"S" im BASIC wird die Kontrolle dem TRSDOS zurückgegeben. (Der Befehl "BASIC2" versetzt den TRS-80 in das BASIC LEVEL II, so daß der RAM-Hauptspeicher wieder voll verfügbar ist.) DISK BASIC erweitert den LEVEL II um einige neue Anweisungen und Funktionen.

DISK BASIC baut Dateien mit entweder direktem oder sequentiellem Zugriff auf. Programme können auf Dateien im ASCII- oder im verdichteten Format gespeichert werden. Alle Dateien erhalten Namen, die aus einem 8-Zeichen-Wort, einer Erweiterung von 3 Zeichen, einem Kennwort von 8 Zeichen und der Nummer eines bestimmten Disketten-Laufwerks bestehen können. (Alle außer dem eigentlichen Dateinamen sind wahlfrei.)

Erweiterungs-Anweisungen

Anweisung	Wirkung	Beispiele
CMD"D"	ruft DEBUG auf (s. TRSDOS).	CMD"D"
MID$	ermöglicht, MID$ auch auf der linken Seite einer Wertzuweisung anzuwenden.	MID$(B$,S,K) = L$
INSTR	ergibt die Lage der zweiten Zeichenkette innerhalb der ersten vom Zeichen Nr. I an.	INSTR(A$," ") INSTR(I,A$," ")
DEF FN	ermöglicht dem Benutzer, eigene Funktionen zu definieren.	DEF FNMLT(X,Y)=X*Y
hexadezimale Konstante	Mit "&H" bzw. "&" gekennzeichnet, können auch hexadezimale oder oktale Konstanten eingegeben werden.	POKE &H42E9,&77

Anweisung	Wirkung	Beispiele
DEFUSR	definiert einen Entry-Point für eine von 10 Benutzer-Routinen in Maschinenkode.	DEFUSR7 = &H7ØE9
USR n	ruft eine von höchstens 10 Routinen in Maschinenkode auf.	Y = USR7 (X)
CMD"T"	schaltet den 25-msec-Uhrtakt aus. Dies ist jeweils vor Kassetten-Eingaben oder -Ausgaben erforderlich.	CMD"T"
CMD"R"	schaltet den 25-msec-Uhrtakt wieder ein, was beim Einsatz von Disketten ratsam ist.	CMD"R"
TIME$	ergibt Datum und Uhrzeit als Zeichenkette. Datum und Uhrzeit werden mit den TRSDOS-Befehlen DATE und TIME eingestellt.	PRINT RIGHT$ (TIME$,8) PRINT LEFT$ (TIME$,8)

Datei-Anweisungen

Anweisung	Wirkung	Beispiele
OPEN	eröffnet eine Datei. Diese Anweisung legt sequentielle/direkte Eingabe oder Ausgabe, Dateinummer und Dateinamen fest.	OPEN "Ø", 1, "AUSGABE" OPEN "R", 3, "DATEI"
CLOSE Datei-Nr,...., Datei-Nr.	schließt eine oder mehrere offene Dateien ab. Vor einem neuen Einsatz muß die Datei wieder neu eröffnet werden.	CLOSE 1, 2, 6

Anweisung	Wirkung	Beispiele
SAVE	sichert ein BASIC-Programm auf Diskette. (Sie können vorher ein Programm von Kassette laden und es dann leicht auf die Diskette schreiben lassen.) Der Parameter A bewirkt, daß im ASCII-Format gespeichert wird.	SAVE "PRGRM1Ø" SAVE "PRGRM2Ø",A
LOAD	lädt ein BASIC-Programm von Diskette in den Hauptspeicher. Der Parameter R bewirkt, daß das Programm auch ausgeführt wird.	LOAD "PRGRM3Ø",R
MERGE	fügt ein BASIC-Programm mit einem anderen zusammen, das gerade im Hauptspeicher steht.	MERGE"ZUSATZ"
DISKDUMP	ist ein Programm, um eine Disketten-Datei über Zeilendrucker oder Video-Bildschirm sektorweise und hexadezimal oder in ASCII auszugeben, je 16 Bytes auf einmal. Es zeigt, wie die Daten auf der Diskette formatiert sind.	RUN"DISKDUMP/BAS"
KILL	löscht eine Datei	KILL"DATEN17"
LINE INPUT	liest eine ganze Datenzeile von Diskette in eine Zeichenketten-Variable.	LINE INPUT #4,"EINGABE?";N$
EOF	eine Funktion,um das Dateiende einer sequentiellen Datei abzuprüfen.	X = EOF (1)
PRINT # Dateinummer	schreibt in eine sequentielle Datei.	PRINT #1,CHR$(34);X$;CHR$(34) PRINT #1,A$;",";B$ PRINT #1,USING F$;C

Anweisung	Wirkung	Beispiele
INPUT # Dateinummer	liest aus einer sequentiellen Datei.	INPUT#3,A,B,C
FIELD	legt das Format für einen Satz einer Datei mit wahlfreiem Zugriff fest.	FIELD 1, 20 AS NAM$, 35 AS ADR$
LOF	ergibt die Nummer des höchsten physischen Satzes einer Datei.	PRINT LOF (1)
LSET/RSET	stellt Daten vor der Ausgabe auf Diskette in einen Puffer für Dateien mit dirketem Zugriff.	LSET NAM$ = A$+","+B$ RSET ADR$ = A$
PUT	schreibt Daten aus einem Puffer in den mit Nummern angegebenen Satz einer Datei mit direktem Zugriff.	PUT 1
GET	liest den angegebenen Satz von einer direkten Datei in den dazugehörigen Puffer der Datei.	GET DATEI1, SATZ

Die folgenden Funktionen sind verfügbar, um Zahlen in Zeichenketten und zurück umzuwandeln. Damit kann etwa eine fünfstellige ganze Zahl in 2 Bytes gespeichert werden, statt in 5 ASCII-Zeichen, so daß Diskettenplatz gespart wird.

Typ	Umwandlung in Text	Umwandlung in Zahlen	Bytes
ganzzahlig	MKI$	CVI	2
einfach genau	MKS$	CVS	4
doppelt genau	MKD$	CVD	8

B / DINGE, DIE SIE WISSEN SOLLTEN

Die Angaben beziehen sich auf Version 2.0, siehe auch Anhang C.

1. TRSDOS und DISK BASIC erfordern gemeinsam 10 KB RAM-Hauptspeicher.

2. Nach Ausführung einer INPUT#-Anweisung von der Kassette werden die Daten jedesmal durch die folgenden READ-Anweisungen automatisch mit RESTORE wiederhergestellt, wenn eine READ-Anweisung ausgeführt wird. Um das zu vermeiden, setzt man einfach die Anweisung POKE 16 553,255 ein, bevor man die erste READ-Anweisung ausführt. Das gilt nur für LEVEL II, nicht für DISK BASIC.

3. Bei Ausführung eines INPUT # von der Kassette beträgt die maximale Anzahl von Bytes, die gelesen werden können, 248. Das bezieht sich nicht auf Disketten-Operationen.

4. Wenn der RESET-Knopf gedrückt wird und die Erweiterungsschnittstelle an den TRS 80 angeschlossen ist, gehen alle Programme im Speicher verloren.

5. Wird ein BASIC-Programm während des Laufes angehalten und werden Änderungen in dem Programm vorgenommen oder der EDIT-Modus aufgerufen (auch ohne darauffolgende Änderung), werden ALLE VARIABLEN auf Null gesetzt. Das Programm muß danach wieder von vorn gestartet werden (RUN).

6. Wenn ein LPRINT oder LLIST ausgeführt wird, ohne daß ein TRS 80-Zeilendrucker angeschlossen ist, bleibt der Computer wie angefroren stehen. Der Anwender muß RESET drücken oder einen Zeilendrucker anschließen und anstellen.

7. Alle Funktionen in BASIC LEVEL II ergeben nur Werte mit einfacher Genauigkeit (6 - 7 Stellen genau). Neuerdings ist ein Kassettenprogramm lieferbar (Nr. 26 - 1704) mit dessen Hilfe man mathematische Funktionen auch mit doppelter Genauigkeit berechnen kann. Alle trigometrischen Funktionen benötigen bzw. ergeben Werte im Bogenmaß. Die Umrechnung von Winkeln in Bogenmaß können dem LEVEL II BASIC-Handbuch entnommen werden.

8. Alle Programme in Maschinensprache, die augenblicklich bei Radio Shack erhältlich sind, laufen im DISK BASIC nicht einwandfrei.

9. Mehrfaches Auftreten der SYNTAX-Fehlermeldung kann durch einen der folgenden, schlecht erkennbaren Fehler entstanden sein.

   a) Wenn ein Buchstabe oder das at-Symbol (@) mit der SHIFT-Taste eingegeben werden, erscheinen die Zeichen korrekt auf dem Bildschirm, sind tatsächlich aber fehlerhaft. Schreiben Sie die Zeile noch einmal und hüten Sie sich vor der

SHIFT-Taste.

b) Manchmal ist ein Leerzeichen in einer BASIC-Anweisung erforderlich. Alle nachfolgenden Zeilen sind fehlerhaft.

IFD<ØD=Ø

FIELD#1, 20ASC$

Die Zeichen "ØD" bedeuten eine doppelt genaue Null. "ASC" ist ein für BASIC reservierter Wert. Die richtigen Anweisungen heißen (beachten Sie das Leerzeichen und das THEN):

IFD <Ø THEN D=Ø

FIELD#1,2ØAS C$

10. Sind die Kassettenrekorder 1 und 2 über die Erweiterungs-Schnittstelle an den TRS-80 angeschlossen, müssen alle Befehle CSAVE und CLOAD mit "#-Kassettennummer" versehen werden. Zum Beispiel heißt das Format eines CLOAD?-Befehls zur Prüfung einer Kassette Nr. 2:

CLOAD#-2,?"Dateiname"

11. Wenn das Drücken einer Taste oft mehrere Buchstaben erzeugt (prellen) sollten die Kunststoffknöpfe abgenommen und die darunter liegenden Kontakte gereinigt werden. Anschließend sind die Knöpfe wieder aufzusetzen.

12. Besitzer des TRS 80 können Auskunft durch den Radio Shack Computer Service bekommer
Telefon: (817) 390 - 3583
Adresse für schriftliche Anfragen:

HUGH MATTHIAS
Radio Shack Computer Service
P.O.Box 185
Fort Worth, TX 76 102

13. Führt man eine Eingabe mit INPUT über die Tastatur durch, wird auf dem Bildschirm die Zeile unter der augenblicklichen Cursor-Position gelöscht.

14. Die folgende Tabelle faßt die Auswirkungen bei der Eröffnung einer Datei zusammen (siehe auch Seite 21):

Modus	wenn Datei vorhanden ist	wenn Datei nicht vorhanden ist
"I"	in Ordnung	FILE NOT FOUND - Fehlermeldung
"O"	Datei wird überschrieben	Datei wird erzeugt
"R"	in Ordnung	Datei wird erzeugt

15. Die Platzreservierung und das Löschen (SAVE und KILL) auf einer Diskette werden nicht immer einwandfrei vorgenommen, wenn auf der Diskette nur noch 5K Speicherplatz vorhanden sind. Die letzten 5K freien Speicherplätze auf jeder Diskette sollten

bis zur Freigabe von Version 2.1 TRSDOS nicht ausgenutzt werden.

BEMERKUNG: Bei Version TRSDOS 2.1 kann der Speicherplatz voll genutzt werden (siehe Seite 51).

C / WEITERE DINGE, DIE SIE WISSEN SOLLTEN

Ergänzungen bezüglich TRSDOS Version 2.1 . Damit werden die Angaben in Anhang B ergänzt bzw. abgeändert.

1. Beachten Sie immer folgende Regeln, wenn Sie Mini-Disketten-Geräte an den TRS-80 anschließen:
   a) Sie können ein bis vier Geräte mit dem TRS-80 verwenden. Bei Ihrem Geräte-Satz muß jedoch ein Gerät (aber nur eins) mit der Radio-Shack-Katalog-Nr. 26 - 1160 vorhanden sein. Alle anderen Geräte müssen der Katalog-Nummer 26 - 1161 entsprechen.
   b) Gerät 26 - 1160 muß immer das letze ("Terminal"-) Gerät sein (am weitesten von der Erweiterungs-Schnittstelle entfernt). Zwischen dem letzten Gerät und Schnittstelle darf keine Steckverbindung freibleiben. (Die auf Seite 1 beschriebene Anordnung ist ebenfalls gültig.)
   c) Denken Sie auch daran, daß in TRSDOS die Geräte mit den Nummern Ø, 1, 2, 3 bezeichnet werden. Hierin liegt Gerät Ø der Schnittstelle am nächsten, während 3 das am weitesten entfernt liegende Gerät ist.

2. Wenn nur ein Gerät (Gerät Ø) verfügbar ist, werden BASIC-Programm-Dateien von einer 2.0- auf eine 2.1-Diskette folgendermaßen kopiert: Zuerst sichert man die 2.0 Datei mit CSAVE auf eine Kassette, dann lädt man das System neu mit einer 2.1-Diskette, liest die Datei mit CLOAD von der Kassette in den Speicher und sichert die Datei mit SAVE auf die 2.1-Diskette.

3. Die maximale TAB-Druckstelle in der LPRINT-Anweisung ist 63. Der Zeilendrucker setzt keinen TAB-Tabulator nach Spalte 63.
   Mit der STRING -Funktion steht jedoch ein einfacher Weg zur Verfügung, um diese Begrenzung zu umgehen. Der Gedanke ist, mit der STRING$-Funktion eine Zeichenkette (String) von Leerzeichen auszudrucken, um den Schreibkopf an die gewünschte Stelle zu bringen. Dabei wird jedoch der Schreibkopf nur relativ zur augenblicklichen Schreibposition bewegt, so daß Sie einige wenige Berechnungen machen müssen, um den Schreibkopf an die gewünschte Position zu bringen.
   Im allgemeinen gilt:
   TAB(n) kann durch eine STRING$ (N-1-laufende Schreibkopf-Position,32) ausgeführt werden.

N-1-laufende Schreibkopf-Position gibt die gewünschte Spalte in Relation zur augenblicklichen Position an; 32 ist der ASCII-Kode für Leerzeichen (Blank).
Beispiel:
LPRINT TAB (5)"NAME"TAB(30)"ADRESSE"STRING$(63,32)"SALDO"

druckt "NAME" auf Spalte 5, "ADRESSE" auf Spalte 30 und "SALDO" auf Spalte 100

4. PRINT#n (Schreiben auf eine sequentielle Disketten-Datei) gibt die Information auf die Diskette im gleichen Format aus, wie ein PRINT (Anzeige) dieselben Informationen auf dem Bildschirm anzeigt. Es benutzt nicht das Format von PRINT#-n (Ausgabe auf Kassette). Sind zum Beispiel
A = 3.14159 und B = - 2.3, dann speichert
PRINT#1,A,B
folgendes in die Datei 1:
␣3.14159␣␣␣␣␣␣␣-2.3␣
Dagegen würde
PRINT#1,A;B
die Datei ohne all die Extra-Leerzeichen speichern:
␣3.14159␣-2.3␣
Wenn Sie numerische Daten auf eine Disketten-Datei mit PRINT#n schreiben, müssen Sie daher Semikolons als Trennzeichen verwenden.

5. Wenn Sie mit PRINT#n Zeichenketten sequentielle Dateien schreiben, müssen Sie Obacht geben, da Trennzeichen nach STRING-Daten nicht automatisch eingesetzt werden.
Wenn zum Beispiel A$= "HANS SCHMIDT" und B$ "KLAUS SCHMIDT" ist, ergibt
PRINT#1,A$,B$
folgende Speicherung auf der Diskette:
HANS SCHMIDTKLAUS SCHMIDT
Sie können dann die beiden Daten mit einer INPUT#n-Anweisung später nicht in zwei verschiedene Variable einlesen.
Um das zu vermeiden, kann man ein in Anführungsstriche (") eingeschlossenes Komma nach jeder Zeichenkette in die PRINT#-Anweisung einfügen. Zum Beispiel wird
PRINT#1, A$;",";B$
es ermöglichen, A$ und B$ getrennt mit einer
INPUT#1,A$,B$
zurückzuholen.

Um Zeichenketten zu schreiben, die Kommas und/oder Wagenrückläufe enthalten, setzt man die Variablen in Anführungsstriche, indem man das entsprechende ASCII-Zeichen "CHR$(34)" wählt.

Zum Beispiel:

PRINT#1,CHR$(34);A$;CHR$(34)

Wenden Sie diese Techniken an, und prüfen Sie mit dem DISKDUMP-Programm genau nach, was auf der Datei geschrieben steht.

6. Die einzige augenblicklich verfügbare System-Kassette von Radio Shack, die unter DISK BASIC oder DOS eingelesen und für spätere Anwendungen auf einer Diskette gespeichert werden kann, ist das RENUM-Programm zum Neu-Numerieren von BASIC-Programmen.

   Zuerst müssen Sie RENUM von der Kassette einlesen, indem Sie TAPEDISK verwenden. Wenn Sie dann RENUM einsetzen wollen, tippen Sie

   RENUM (ENTER)

   Mit dem Aufruf von LEVEL II

   BASIC 2 (ENTER) und

   MEMORY SIZE? Antwort 31819

   können Sie das RENUM-Programm auch in LEVEL II anwenden.

7. Wenn Sie ein bestimmtes, oft auftretendes Problem haben, das Sie nicht selbst lösen können, schreiben Sie genau die Umstände auf (Geräte-Konfiguration und Folge, bei der das Problem auftrat), und nachstehende Abteilung wird sich Mühe geben, die Lösung zu finden. Das gilt nicht für Programmfehler in selbstgeschriebenen Programmen, da wir hierfür heute noch keine Zeit haben.

   TRS - 80 Problem Desk
   1100 ONE TANDY CENTER
   Fort Worth, TX 76 102

   Ankündigungen neuer Versionen oder Ausgaben werden Ihnen automatisch zugesandt.

8. Die Real-Zeit-Uhr braucht nur unmittelbar vor der Ausführung einer Kassetten-Ein-/Ausgabe (I/O) abgeschaltet zu werden, und zwar mit dem DISK-BASIC-Befehl CMD"T" Denken Sie daran, die Uhr sofort wieder einzuschalten (mit dem DISK-BASIC-Befehl CMR"R"), sobald die Kassetten-Ein-/Ausgabe beendet ist. Sie könnten die TRSDOS-System-Dateien zerstören, wenn Sie Disketten-Ein-/Ausgaben zulassen, während die Uhr abgeschaltet ist. Beachten Sie diese Vorsichtsmaßnahme, um Schaden vorzubeugen.

D / SUCHREGISTER