Inhaltsverzeichnis zu Floppy-Erweiterungssatz

Allgemeines zu Floppy-Erweiterungssatz

OPEN-Variable

Um die Angaben, die beim Oeffnen eines Kanals gemacht werden müssen, zu speichern, wird bei jeder OPEN-Anweisung eine sogenannte OPEN-Variable angelegt. Diese wird wie eine Zweizeichen-Variable angelegt und benötigt 32 Bytes Speicherplatz. Die Funktion DSHOW$ liefert den Inhalt der OPEN-Variablen.
Die OPEN-Variable wird auch angelegt, wenn bei der Ausführung der OPEN-Anweisung ein Fehler auftritt (rote LED blinkt), aber der Rechner keinen ERROR meldet. In diesem Fall kann die OPEN-Variable mit CLOSE wieder aus dem Speicher gelöscht werden, obwohl der Kanal gar nicht geöffnet wurde. Um während dem Programmablauf sicher zu stellen, dass die OPEN-Anweisung fehlerfrei ausgeführt wurde, sollte unmittelbar nach der OPEN-Anweisung der Status des Laufwerkes mit DSTAT$ abgefragt werden.

*** ACHTUNG ***
*** ***
*** Die Anweisungen CLEAR und NEW löschen auch die OPEN-Variablen. ***
*** Diese beiden Anweisungen sollten deshalb nicht verwendet werden, ***
*** solange Kanäle zum Floppy offen sind. Wird ein Kanal nicht mit ***
*** CLOSE geschlossen, so kann dies zu Datenverlust führen. ***

Hinweise für TOOL3-Benützer

Die Anweisungen CLR DIM und PURGE DIM zerstören ebenfalls die OPEN-Variabeln und sollten deshalb nicht verwendet werden, solange Kanäle zum Floppy offen sind.
Die OPEN-Variabeln werden bei VLIST und VKEEP als on$*25 angezeigt, wobei n = logische File-Nummer.
Die OPEN-Variabeln und alle später angelegten Variabeln können mit VKEEP erst ab TOOL3 V1.6 (Auslieferung nach 1. Juni 1985) gerettet werden.

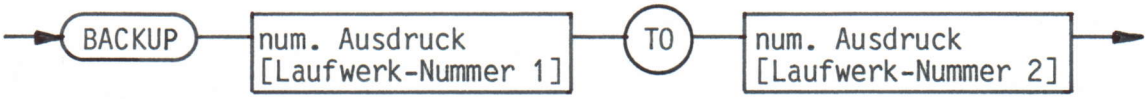
Speicherung der Daten auf Disk

Textausdrücke werden als Text aufgezeichnet, num. Ausdrücke werden beim Ausgeben mit PRINT # ebenfalls in Text umgewandelt. Daten können deshalb mit INPUT # immer in Textvariabeln eingelesen werden. Wird eine num. Variable angegeben, so werden die einzulesenden Daten in einen num. Ausdruck umgewandelt, dabei gelten die Regeln der Funktion VAL. Beim Umwandeln tritt deshalb nie eine Fehlermeldung auf.
Werden Daten an das Laufwerk übertragen, so werden diese vorerst nur in einem Buffer des Floppy gespeichert. Erst wenn dieser Buffer voll ist, oder wenn der Kanal zum File mit CLOSE geschlossen wird, werden die Daten auf die Disk übertragen.

BACKUP

```
**************************************************************
*                                                            *
*  Kopiert den gesamten Inhalt einer Disk von einem Lauf-    *
*  werk auf ein anderes.                                     *
*                                                            *
**************************************************************
```

Syntax



Parameter Beschreibung:

- [Laufwerk-Nummer 1]: Nummer des Laufwerkes, von dem die Daten kopiert werden sollen. Erlaubt ist ein Ausdruck im Bereich 8 ... 11.

- [Laufwerk-Nummer 2]: Nummer des Laufwerkes, nach dem die Daten kopiert werden sollen. Erlaubt ist ein Ausdruck im Bereich 8 ... 11.

Hinweise:

- Die Disk im Laufwerk 2 wird vor dem Kopieren automatisch neu formatiert. Sie hat nach dem Kopieren den gleichen Namen und die gleiche ID wie die Original-Disk im Laufwerk 1.

- Ein Backup mit einem COMMODORE VC-1541 dauert ca. 28 Minuten.

- Tritt während einem BACKUP ein Fehler auf, so müssen beide Laufwerke ab- und wieder eingeschaltet werden.

- Siehe auch Erläuterungen zu DEFDISK.

- BACKUP kann manuell oder programmkontrolliert ausgeführt werden.

CLOSE

****
*
* Schliesst den angegebenen Kanal.
*
****

Syntax

Parameter Beschreibung: - [log. File-Nr]: Logische File-Nummer des Files, dessen Kanal geschlossen werden soll. Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.

Besondere Fehlermeldung: - ERROR 6: Zur angegebenen log. File-Nummer existiert keine OPEN-Variable.

Hinweise:
- Beim Schliessen des Kanals wird die entsprechende OPEN-Variable aus dem Speicher gelöscht.
- Beim Schliessen des letzten offenen Kanals verlöscht die rote LED am entsprechenden Laufwerk.
- OPEN-Variabeln können auch mit CLOSE aus dem Speicher gelöscht werden, wenn der zugehörige Kanal nicht mehr offen ist (z.B. weil das Laufwerk ausgeschaltet wurde).
- CLOSE kann manuell oder programmkontrolliert ausgeführt werden.

DRECPOS

****
*
* Positioniert auf den angegebenen Datensatz in einem
* relativen File.
*
****

Syntax

Parameter Beschreibung:

- [log. File-Nr]: Logische File-Nummer des relativen Files, in welchem positioniert werden soll.
  Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.

- [Satz-Nummer]: Nummer des Satzes, auf welchen positioniert werden soll.
  Erlaubt ist ein num. Ausdruck im Bereich 1 ... 65535.

- [Byte-Nummer]: Nummer des Bytes, auf welches positioniert werden soll.
  Erlaubt ist ein num. Ausdruck im Bereich 1 ... 254.

Besondere Fehlermeldung:

- ERROR 6: Zur angegebenen log. File-Nummer existiert keine OPEN-Variable.

Hinweise:

- Siehe auch "Erläuterungen und Beispiele zu relativen Files", Seite FE-15.

- DRECPOS kann manuell oder programmkontrolliert ausgeführt werden.

DSHOW$

****
* Liefert den Inhalt der angegebenen OPEN-Variablen. *
****

Syntax

Ergebnis: Text

Parameter Beschreibung: - [log. File-Nr]: Logische File-Nummer der OPEN-Variablen, deren Inhalt ausgegeben werden soll. Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.

Ist zur angegebenen logischen Filenummer eine OPEN-Variable vorhanden, so hat der resultierende Textausdruck folgendes Format:

```
pp,ss,nnnnnnnnnnnnnnnn,t,m,lll
```

wobei: p = Primär-Adresse (2 Zeichen)
s = Sekundär-Adresse (2 Zeichen)
n = Filename (16 Zeichen)
t = Filetype (1 Zeichen)
m = Filemode (1 Zeichen)
l = Satzlänge (3 Zeichen)

Numerische Werte werden links, der Filename wird rechts mit "Spaces" aufgefüllt, so dass das Format unabhängig von den Parametern konstant bleibt. Fehlende Parameter werden vollständig durch "Spaces" ersetzt.

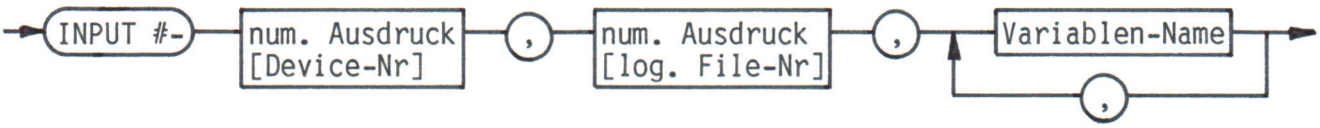
Existiert zur angegebenen logischen Filenummer keine OPEN-Variable, so liefert DSHOW$ einen Leerstring ("").

Hinweis: - DSHOW$ kann manuell oder programmkontrolliert ausgeführt werden.

INPUT #

Liest Daten über den angegebenen Kanal aus dem zugehörigen Buffer in die aufgelisteten Variabeln ein.

Syntax



Parameter Beschreibung:

- [Device-Nr]: Nummer des Peripherie-Gerätes, von dem die Daten eingelesen werden sollen. Die Device-Nummer des Floppy-Interfaces ist 15. Siehe auch Erläuterungen auf Seite vi und vii.
- [log. File-Nr]: Logische File-Nummer des Files, von dem die Daten eingelesen werden sollen. Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.
- Variablen-Name: Name der Variablen, in welche die Daten eingelesen werden sollen.

Hinweise:

- Wird eine num. Variable angegeben, so werden die zu lesenden Daten in einen num. Ausdruck umgewandelt. Für die Umwandlung gelten dieselben Regeln wie für die Funktion VAL.
- INPUT # kann manuell oder programmkontrolliert ausgeführt werden.

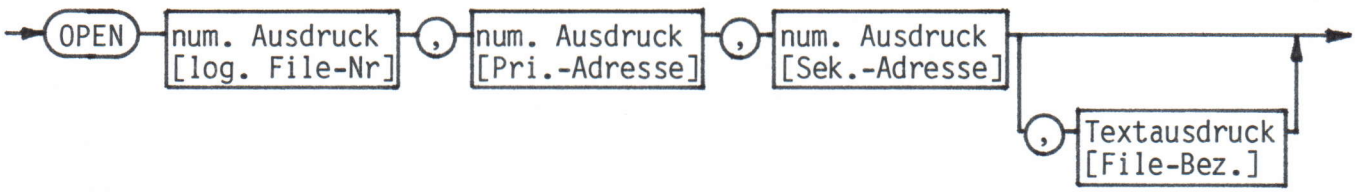
Besondere Fehlermeldungen:

- ERROR 6: Zur angegebenen log. File-Nummer existiert keine OPEN-Variable.
- ERROR 130: Es sind keine Daten vorhanden, die eingelesen werden können.

OPEN

****
* Oeffnet den angegebenen Kanal. *
****

Syntax



Parameter Beschreibung:

- [log. File-Nr]: Die Logische File-Nummer liefert den Namen der OPEN-Variablen.
  Sie dient allen weiteren Instruktionen als Referenz für das entsprechende File.
  Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.

- [Primär-Adresse]: Nummer des Laufwerkes, zu dem der Kanal geöffnet werden soll.
  Erlaubt ist ein num. Ausdruck im Bereich 8 ... 11.

- [Sekundär-Adresse]: Nummer des Kanals, der geöffnet werden soll.
  Erlaubt ist ein num. Ausdruck im Bereich 0 ... 15.
  Kanal 0 und 1 werden von DSAVE, bzw. DLOAD benutzt und sollten deshalb nicht verwendet werden.
  Kanal 15 ist der Befehlskanal. Ueber diesen Kanal können keine Daten, sondern nur Befehle an das Laufwerk übermittelt werden.

- [File-Bez.]: Die File-Bezeichnung ist ein Textausdruck und hat folgendes Format:

  - sequentielle Files: "Filename,Filetype,Filemode"
  - relative Files: "Filename,Filetype,Satzlänge"

  wobei: - Filename: Name des zu öffnenden Files.

  - Filetype: Es können folgende Filetypen angelegt werden:
    - "L" für relative Files
    - "P" für Programm-Files
    - "S" für sequentielle Files
    - "U" für User-Files

- Filemode: Legt fest, wie das File angesprochen werden kann (Zugriffsart).
  - "A" Append, d.h. die Daten werden am Ende des entsprechenden Files angefügt.
  - "R" Read, d.h. die Daten werden aus dem entsprechenden File gelesen.
  - "W" Write, d.h. die Daten werden in das entsprechenden File geschrieben.

- Satzlänge: Legt die Länge eines Records fest. Erlaubt ist ein num. Ausdruck im Bereich 1 ... 254.

Wird keine File-Bezeichnung angegeben, so wird der Kanal zum nächsten freien Buffer des Laufwerkes geöffnet.
Soll der Kanal zu einem bestimmten Buffer geöffnet werden, so muss anstelle der File-Bezeichnung die Zeichenfolge "#n" angegeben werden, wobei
n = Nummer des Buffers.
Siehe auch Anleitung zum Floppy.

Erläuterungen

Es können grundsätzlich zwei Arten von Files angelegt werden: Sequentielle Files und relative Files.

Sequentielle Files: Files der Typen "Program", "Sequentiel" und "User" sind sequentielle Files. Wird ein Kanal zu einem sequentiellen File geöffnet, so muss immer auch die Zugriffsart, also der Filemode, angegeben werden. Wird der Filemode "Append" oder "Read" angegeben, so muss das File bereits existieren, wird die Zugriffsart "Write" angegeben, so darf noch kein File mit dem entsprechenden Namen existieren.
Siehe auch "Erläuterungen und Beispiele zu sequentiellen Files", Seite FE-12.

Relative Files: Relative Files sind vom Filetype "Relative". Wird ein Kanal zu einem relativen File geöffnet, so müssen Filetype und Satzlänge nur angegeben werden, wenn das File noch nicht existiert. Relative Files werden immer zum Lesen und Schreiben geöffnet.
Siehe auch "Erläuterungen und Beispiele zu relativen Files", Seite FE-15.

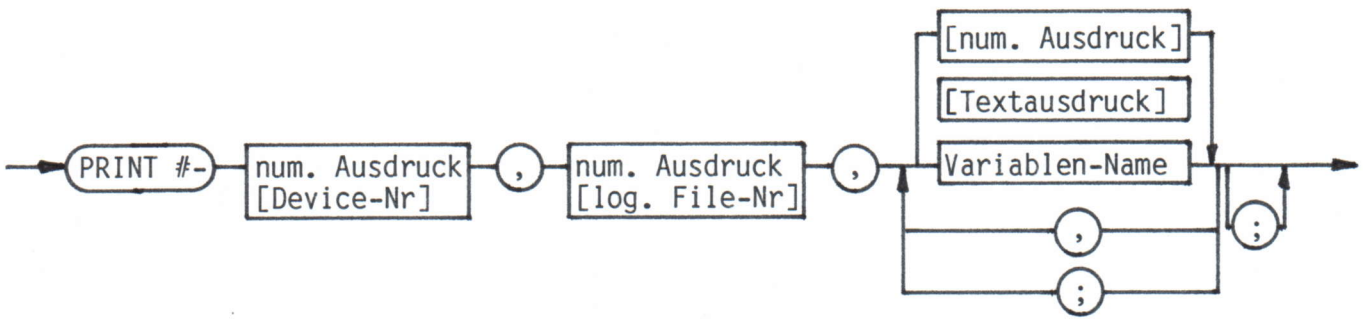
Besondere Fehlermeldungen:
- ERROR 5: Zur angegebenen logischen File-Nummer existiert bereits eine OPEN-Variable.
- ERROR 10: Zum Anlegen der OPEN-Variablen steht nicht genügend Speicherplatz zur Verfügung.

Hinweise:
- Die rote LED leuchtet, solange ein oder mehrere Kanäle offen sind.
- OPEN kann manuell oder programmkontrolliert ausgeführt werden.

PRINT #

Schreibt Daten über den angegebenen Kanal in den zugehörigen Buffer.

Syntax



Parameter Beschreibung:

- [Device-Nr]: Nummer des Peripherie-Gerätes, an das die Daten ausgegeben werden sollen. Die Device-Nummer des Floppy-Interfaces ist 15. Siehe auch Erläuterungen auf Seite vi und vii.
- [log. File-Nr]: Logische File-Nummer des Files, in das die Daten gespeichert werden sollen. Erlaubt ist ein num. Ausdruck im Bereich 0 ... 9.
- [num. Ausdruck] oder [Textausdruck]: Daten, die ins angegebene File geschrieben werden sollen.
- Variablen-Name: Name der Variablen, aus welcher die Daten abgespeichert werden sollen. DIM-Variabeln können nicht als Ganzes abgespeichert werden, sondern nur einzelne Elemente davon.

Besondere Fehlermeldung:

- ERROR 6: Zur angegebenen log. File-Nummer existiert keine OPEN-Variable.

Hinweise:

- Num. Ausdrücke werden vor dem Abspeichern in Textausdrücke umgewandelt.
- PRINT # kann manuell oder programmkontrolliert ausgeführt werden.

Erläuterungen und Beispiele zu sequentiellen Files

Sequentielle Files - auch sequentielle Dateien genannt - sind vom Filetype "P" (bzw. "PRG"), "S" (bzw. "SEQ") oder "U" (bzw. "USR"). Da sich diese Filetypen genau gleich verhalten, beschränken sich die nachfolgenden Ausführungen auf den Filetype "S" (bzw. "SEQ").

Sequentielle Files haben die Eigenschaft, dass Daten, die einmal ins File geschrieben wurden, zwar beliebig oft gelesen, aber nicht mehr verändert werden können. Es ist dagegen möglich, mit der Zugriffsart "Append" weitere Daten an ein bestehendes File anzufügen. Sollen die bestehenden Daten geändert werden, so müssen sie in ein neues File übertragen werden.

Wird ein neues File eröffnet (OPEN mit Filemode "Write"), so wird ein für den Benutzer unsichtbarer Zeiger auf den Filebeginn aufgesetzt. Mit jedem Eintrag in das File (PRINT #) wird der Zeiger vom Floppy nachgeführt. Die Daten werden durch den ASCII-Code "Carriage Return" (0D hex) getrennt. Dieser Trenncode wird immer ausgegeben, wenn ein Ausdruck, bzw. ein Variablen-Name nicht mit einem Strichpunkt (";") getrennt, bzw. abgeschlossen wird.

Wird ein bestehendes File mit der Zugriffsart "Append" geöffnet, so wird der Zeiger hinter den letzten Eintrag positioniert. Im übrigen gelten die gleichen Regeln wie für "Write".

Wird ein bestehendes File mit der Zugriffsart "Read" geöffnet, so wird der Zeiger auf den Filebeginn aufgesetzt. Mit jedem Einlesen von Daten (INPUT #) wird der Zeiger hinter das nächste "Carriage Return" positioniert.

Beispiele

1) Es soll ein neues sequentielles File mit dem Namen "Test1.DAT" angelegt werden. Die Primär-Adresse = Laufwerk-Nummer sei 8, die Sekundär-Adresse = Kanal-Nummer sei 5, die logische File-Nummer sei 2:

```
OPEN 2,8,5,"Test1.DAT,S,W"
```

Konnte die OPEN-Anweisung fehlerfrei ausgeführt werden, leuchtet nun die rote LED des Floppy.

DSHOW$ liefert den Inhalt der entsprechenden OPEN-Variablen:

```
DSHOW$ 2  liefert den Textausdruck " 8, 5,Test1.DAT       ,S,W,  0"
```

2) Nun können die Daten ins File übertragen werden:

```
T$ "SHARP PC-1500":DIM N(5):N(3)=1/7
PRINT #-15,2,"TRAMsoft",12345,T$,N(3)
```

Nach der Ausführung dieser Anweisungen befinden sich 4 Einträge in diesem File.

3) Der Kanal zum File "Test1.DAT" soll nun wieder geschlossen werden.

```
CLOSE 2
```

Die Daten werden auf die Disk geschrieben und der Kanal geschlossen. Die rote LED des Floppy verlöscht.

4) Das File "Test1.DAT" soll zum Lesen geöffnet werden. Die Primär-Adresse sei wiederum 8, die Sekundär-Adresse sei 12, die logische File-Nummer sei 0:

```
OPEN 0,8,12,"Test1.DAT,S,R"
```

Konnte die OPEN-Anweisung erfolgreich ausgeführt werden, leuchtet wieder die rote LED des Floppy.

5) Die Daten können nun eingelesen werden. Da alle Daten auf der Disk als Text aufgezeichnet werden (Siehe Seite FE-2), können alle 4 Einträge des Files in Textvariabeln eingelesen werden:

```
INPUT #-15,0,A$,B$,C$,D$
```

Danach enthält A$ "TRAMsoft", B$ "12345", C$ "SHARP PC-1500" und D$ "1.428571429E-01".

6) Die Daten können ebenfalls in num. Variabeln eingelesen werden. Da auf grund der INPUT-Anweisung in Beispiel 5) der Zeiger am Fileende steht, muss der Kanal zum File zuerst geschlossen und danach wieder geöffnet werden:

```
CLOSE 0
OPEN 0,8,12,"Test1.DAT,S,R"
INPUT #-15,0,A,B,C,D
```

Danach enthält A 0, B 12345, C 0 und D 1.428571429E-01.

Wird eine weitere INPUT-Anweisung ausgeführt, so tritt eine Fehlermeldung auf, da sich keine weiteren Daten im File befinden:

```
INPUT #-15,0,E$
```

führt zu einem ERROR 130.

7) Sollen weitere Daten in diesem File gespeichert werden, so muss der Kanal zum File zuerst geschlossen und dann mit der Zugriffsart "Append" wieder geöffnet werden:

```
CLOSE 0
A$="Dies ergibt ", B$="einen einzigen ", C$="Eintrag"
XY$="test-string", AZ=7
OPEN 4,8,3,"Test1.DAT,S,A"
PRINT #-15,4,A$;B$;C$,XY$,AZ
CLOSE 4
```

Mit dieser Anweisung werden drei weitere Einträge in das File "Test1.DAT" geschrieben. Da die Variabeln A$, B$ und C$ durch Strichpunkte getrennt sind, wird kein Trennzeichen übertragen. Beim Einlesen wird der Inhalt der Variabeln A$, B$ und C$ in eine einzige Variable eingelesen.

Erläuterungen und Beispiele zu relativen Files

Relative Files - auch relative Dateien genannt - sind vom Filetype "L".

Relative Files bestehen aus einer Anzahl Datensätze - auch Records genannt - die alle dieselbe Länge, d.h. die gleiche Anzahl Bytes, aufweisen. Dabei ist zu beachten, dass auch jedes Trennzeichen "Carriage Return" (OD hex) ein Byte im Record belegt. Dieses Trennzeichen wird jedesmal ausgegeben, wenn eine PRINT-Anweisung nicht mit Strichpunkt (";") getrennt, bzw. abgeschlossen wird.
Die Daten in einem relativen File können über eine sogenannte Satz-Nummer angesprochen werden. Zudem kann die Position innerhalb des Datensatzes durch eine Byte-Nummer angegeben werden, so dass jedes einzelne Byte in einer relativen Datei direkt adressiert werden kann.
Beim Oeffnen eines Kanals zu einem relativen File wird auf das erste Byte des ersten Datensatzes positioniert. Wird die Position nicht angegeben, so wird die Position bei jedem Datentransfer um einen Record erhöht. Die Anweisung DRECPOS ermöglicht das Positionieren auf ein beliebiges Byte in einem beliebigen Datensatz.

Beim Positionieren auf einen noch nicht existierenden Datensatz meldet der Floppy einen Fehler, wenn der Datensatz ausserhalb des im Buffer gespeicherten Bereichs liegt (rote LED blinkt). DSTAT$ liefert in diesem Fall die Fehlermeldung "50,RECORD NOT PRESENT". Wurde auf den Record positioniert, um Daten in diesen Record zu schreiben, so kann die Fehlermeldung ignoriert werden. Werden nach einer solchen Fehlermeldung die Daten an das Laufwerk übertragen, blinkt die rote LED zwar nicht, DSTAT$ liefert aber eine Fehlermeldung im Bereich 40 ... 49, welche in der Anleitung zum Floppy nicht dokumentiert sind. Diese Fehlermeldungen können ebenfalls ignoriert werden.

Beispiele

1) Es soll ein neues relatives File mit dem Namen "Test2.DAT" angelegt werden. Die Satzlänge soll 80 Bytes betragen. Die Primär-Adresse = Laufwerk-Nummer sei 8, die Sekundär-Adresse = Kanal-Nummer sei 10, die logische File-Nummer sei 9:

```
OPEN 9,8,10,"Test2.DAT,L,80"
```

Falls die OPEN-Anweisung fehlerfrei ausgeführt werden konnte, leuchtet die die rote LED des Floppy.

DSHOW$ liefert den Inhalt der entsprechenden OPEN-Variablen:

```
DSHOW$ 9  liefert den Textausdruck " 8,10,Test2.DAT       ,L, , 80"
```

Erläuterungen und Beispiele zu relativen Files (Fortsetzung)

2) Das File aus Beispiel 1) soll für eine Adressliste verwendet werden.
Ein Datensatz soll wiefolgt aufgeteilt werden:

- Position 1 ... 15: Vorname
- Position 16 ... 29: Name
- Position 30 ... 49: Strasse
- Position 50 ... 54: Postleitzahl
- Position 55 ... 69: Ort
- Position 70 ... 80: Telefonnummer

Die Daten seien wiefolgt vorhanden:

```
VN$="Peter", NA$="Muster", ST$="Langstrasse 15", PZ$="8899"
OT$="Aster", TN$="123.45.67"
```

Die Daten sollen nun in den 20. Datensatz eingetragen werden:

```
DRECPOS 9,20,1 :PRINT #-15,9,VN$
DRECPOS 9,20,16:PRINT #-15,9,NA$
DRECPOS 9,20,30:PRINT #-15,9,ST$
DRECPOS 9,20,50:PRINT #-15,9,PZ$
DRECPOS 9,20,55:PRINT #-15,9,OT$
DRECPOS 9,20,70:PRINT #-15,9,TN$
```

3) Der Datensatz soll nun in die Variablen A$ ... F$ eingelesen werden:

```
DRECPOS 9,20,1 :INPUT #-15,9,A$
DRECPOS 9,20,16:INPUT #-15,9,B$
DRECPOS 9,20,30:INPUT #-15,9,C$
DRECPOS 9,20,50:INPUT #-15,9,D$
DRECPOS 9,20,55:INPUT #-15,9,E$
DRECPOS 9,20,70:INPUT #-15,9,F$
```

4) Der Kanal zum File "Test2.DAT" soll geschlossen werden:

```
CLOSE 9
```

Die Daten werden nun aus dem Buffer auf Disk gespeichert und die rote LED des Floppy verlöscht.

Liste der Instruktionen

****

Anmerkung:

Mit dem Floppy-Erweiterungssatz kann die Anweisung DSAVE nicht mehr mit DS. abgekürzt werden, sondern es ist die Abkürzung DSA. nötig.