

G-DOS

Handbuch

M. Krueger / H. Scholz

Zur Beachtung:

TCS Computer GmbH behält sich das Recht vor, Änderungen und Verbesserungen der in diesem Handbuch beschriebenen Software zu jeder Zeit und ohne Ankündigung vorzunehmen.

**Copyright Software und Handbuch
(C) 1984 by TCS Computer GmbH**

Alle Rechte vorbehalten, insbesondere auch diejenigen aus der spezifischen Gestaltung, Anordnung und Einteilung des angebotenen Stoffes. Der auszugsweise oder teilweise Nachdruck sowie fotomechanische Wiedergabe oder Übertragung auf Datenträger zur Weiterverarbeitung ist untersagt und wird als Verstoß gegen das Urheberrechtsgesetz und als Verstoß gegen das Gesetz gegen den unlauteren Wettbewerb gerichtlich verfolgt. Für etwaige technische Fehler, sowie für die Richtigkeit aller in diesem Buch gemachten Angaben, übernehmen der Herausgeber und Autor keine Haftung.

Vorwort

Dieses Handbuch wurde zur Einweisung und Weiterführung in die diskettenunterstützte Programmierarbeit auf den Rechnern Genie I/II/IIIs/III/IIIs und Speedmaster unter den Betriebssystemen G-DOS 2.1a, 2.1b, 2.1c und 3.0 geschrieben.

Im ersten Teil soll vor allem all denjenigen geholfen werden, die durch den Erwerb eines Diskettenlaufwerks und des dazugehörenden Betriebssystems völliges Neuland betreten. Anhand von kurzen Beispielen werden Ihnen darin die wichtigsten DOS- und Disk-BASIC-Befehle Schritt für Schritt nahegebracht.

Zur perfekten Beherrschung des Betriebssystems ist es jedoch unumgänglich, auch den zweiten und dritten Teil dieses Handbuchs durchzuarbeiten. Diese haben die genaue Beschreibung aller DOS- und Disk-BASIC-Befehle zum Inhalt.

Im Anhang finden Sie die Anleitungen zu den auf der Systemdisk befindlichen Utility-Programmen.

Stellen Sie sich zu Beginn Ihrer Programmierversuche ein Duplikat Ihrer Systemdiskette her und legen Sie das Original an einen sicheren Ort. Arbeiten Sie nur mit dem Duplikat. Nur so vermeiden Sie die Zerstörung Ihrer Original G-DOS Diskette durch eine mögliche fehlerhafte Programmierung.

Anhand des Umfangs dieses Handbuchs mögen Sie ersehen, daß wir uns Mühe gegeben haben, alle Zusammenhänge möglichst ausführlich zu beschreiben. Wenn Sie trotzdem etwas nicht auf Anhieb verstehen oder der Rechner eingegebene Kommandos mit Fehlermeldungen quittiert, dann lesen Sie sich bitte den entsprechenden Abschnitt nochmals in Ruhe durch und geben alles nochmal neu ein. In der Regel funktioniert dann plötzlich alles.

Sollten Sie innerhalb der vorliegenden Beschreibung auf Punkte stoßen, die Ihnen unverständlich oder fehlerhaft dokumentiert scheinen, so sind wir für Verbesserungsvorschläge oder Anregungen selbstverständlich jederzeit dankbar.

Viel Erfolg bei der Arbeit mit Ihrem Genie und dem Genie-DOS wünschen Ihnen

Helmut Scholz Michael Krueger

Inhaltsverzeichnis

| | Seite |
|--|-------|
| Vorwort | 5 |
| Inhaltsverzeichnis | 7 |
| | |
| Teil I: Einführung ins G-DOS ===== | 15 |
| Vorteile des Floppy-Betriebs | 17 |
| Die Hardware | 19 |
| Die Floppy-Disk | 20 |
| Behandlung und Handhabung einer Diskette | 23 |
| Allgemeine Einführung (BASIC und G-DOS) | 25 |
| Vorbereitungen zur Inbetriebnahme | 26 |
| Einschalhinweise | 27 |
| DOS und BASIC | 29 |
| Das Kopieren einer Diskette | 31 |
| Das Formatieren einer Diskette | 34 |
| Das Inhaltsverzeichnis einer Diskette | 36 |
| Freier Speicherplatz einer Diskette | 38 |
| Befehlsliste des G-DOS aufrufen | 38 |
| Löschen von Dateien und Programmen | 39 |
| Uhrzeit und Datum | 40 |
| Umbenennung einer Datei oder eines Programms | 41 |
| Wiederholung eines Befehls | 41 |
| Das Abspeichern von Programmen | 42 |
| Das Laden von Programmen | 43 |
| Das Vereinigen von Programmen | 44 |
| AUTO-Start | 45 |
| Aufruf von BASIC ohne Speicheränderung | 46 |

| | Seite |
|---|-------|
| Programmierhilfen unter Disk-BASIC | 47 |
| Disk-BASIC Ergänzungen | 52 |
| Dateien (Files) | 55 |
| Öffnen und Schließen von Dateien | 57 |
| Sequentielle Dateien | 58 |
| Dateien mit direktem Zugriff (RANDOM FILES) | 64 |
| Das CMD-Kommando | 71 |
| DOS-Anpassungen | 77 |
| Noch einige Ratschläge | 80 |
| | |
| Teil II: DOS-Befehle ===== | 81 |
| | |
| Was ist ein Betriebssystem? | 83 |
| | |
| AIK | 86 |
| APPEND | 87 |
| ATTRIB | 88 |
| AUTO | 91 |
| B2 | 92 |
| BL | 94 |
| BOOT | 95 |
| BREAK | 96 |
| DDE | 98 |
| CLS | 101 |
| CONT | 102 |
| | |
| Der COPY Befehl | 105 |
| Syntax 1 | 107 |
| Syntax 2 | 109 |
| Syntax 3 | 110 |
| Syntax 4 | 111 |
| Syntax 5 | 112 |
| Syntax 6 | 116 |
| | |
| CREATE | 119 |

| | Seite |
|-------|-------|
| DATUM | 120 |
| DIR | 121 |
| DISK | 124 |
| DO | 126 |
| DR | 128 |
| DUMP | 129 |
| E | 131 |
| FREE | 132 |
| FORM | 133 |
| F# | 135 |
| HIMEM | 136 |
| I | 138 |
| INFO | 139 |
| JKL | 140 |
| KILL | 142 |
| LC | 144 |
| LF | 145 |
| LIB | 147 |
| LIST | 148 |
| LOAD | 149 |
| N | 150 |
| NDF | 151 |
| PAUSE | 154 |
| PD | 155 |
| PORT | 160 |
| PRINT | 161 |
| PROT | 162 |
| PURGE | 163 |
| R | 164 |

| | Seite |
|--------------------------|-------|
| S | 165 |
| SIO | 170 |
| STMT | 173 |
| UHR | 174 |
| V+ | 175 |
| V24 | 176 |
| Z | 177 |
| ZEIT | 178 |
| ZL | 179 |
| 0 | 181 |
| 64 | 183 |
| 80 | 184 |
| @ | 185 |
| > | 187 |
| & | 188 |
| ? | 189 |
| ## | 190 |
| Programmunterbrechungen | 191 |
| Das Mini-DOS | 192 |
| Das Programm DEBUG | 194 |
| | |
| Teil III: Disk-BASIC | 199 |
| ===== | |
| Allgemeine Informationen | 201 |
| Fehlermeldungen | 201 |
| BASIC | 202 |
| DEF FN | 204 |
| DEFUSR | 205 |
| ERROR | 206 |

| | Seite |
|-----------------------------------|-------|
| INSTR | 207 |
| KILL | 208 |
| LINE INPUT | 209 |
| LOAD | 210 |
| MID\$ | 211 |
| RUN | 212 |
| SAVE | 213 |
| TIME\$ | 214 |
| USR | 215 |
| Abkürzungen und Programmierhilfen | 216 |
| MERGE | 219 |
| REF | 221 |
| RENEW | 224 |
| RENUM | 225 |
| RENUM U | 226 |
| &H | 228 |
| &O | 229 |
| Die CMD-Kommandos | 230 |
| CMD"C" | 230 |
| CMD"E" | 231 |
| CMD"F",DELETE | 232 |
| CMD"F=KEEP" | 233 |
| CMD"F=SWAP" | 234 |
| CMD"F=POP N" | 235 |
| CMD"F=POP R" | 236 |
| CMD"F=POP S" | 237 |
| CMD"F=SAS Z" | 238 |
| CMD"F=ERASE" | 239 |
| CMD"F=SS" | 240 |

| | Seite |
|--|-------|
| CMD"J" | 241 |
| CMD"O" | 242 |
| CMD"R" | 246 |
| CMD"S" | 247 |
| CMD"T" | 248 |
| CMD"Dosbefehl" | 249 |
| Dateien | 250 |
| Sequentielle Dateien (PRINT/INPUT-Files) | 252 |
| OPEN | 252 |
| CLOSE | 254 |
| PRINT# | 255 |
| PRINT# USING | 256 |
| INPUT | 258 |
| LINEINPUT# | 259 |
| EOF | 260 |
| Anwendungsbeispiele | |
| PROGMOD/BAS | 261 |
| FORMLIST/BAS | 263 |
| LIST64/BAS | 265 |
| Allgemeine Dateien mit direktem Zugriff | 266 |
| OPEN | 267 |
| FIELD | 268 |
| LSET | 270 |
| RSET | 271 |
| MKI\$ | 272 |
| MKS\$ | 273 |
| MKD\$ | 274 |
| CVI | 275 |
| CVS | 276 |
| CVD | 277 |

| | Seite |
|--|---------|
| PUT | 278 |
| GET | 279 |
| CLOSE | 280 |
| LOF | 281 |
| EOF | 282 |
| LOC | 283 |
| Dateien mit direktem Zugriff und besonderer Dateistruktur | 284 |
| OPEN | 288 |
| GET bzw. PUT | 292 |
| LOC | 300 |
| Anhang | 301 |
| ===== | |
| Das Hilfsprogramm DIRCHECK/CMD | 303 |
| Das Hilfsprogramm VL/CMD | 305 |
| Das Hilfsprogramm JOB/CMD | 307 |
| Das Programm SUPER/CMD | 308 |
| Das Hilfsprogramm KEY/CMD | 310 |
| Das Hilfsprogramm ASM/CMD | 316 |
| Druckertreiberprogramme ITOH/CMD und STAR/CMD | 324 |
| Das Hilfsprogramm VIDEO/BAS | 325 |
| System-Files | 327 |
| Struktur des Inhaltsverzeichnisses | 332 |
| Systemeinsprungtabelle | 338 |
| Beschreibung der Restart-Befehle | 339 |
| Fehlercode-Tabelle | 240 |
| Vergleichsliste: Genie-DOS <-----> NEWDOS80.2 | 342 |
| Literaturhinweise | 343 |
| Register | 345 |

Teil I

Einführung

ins

G-DOS

Vorteile des Floppy-Betriebs

=====

Jeder Computer braucht einen externen Massenspeicher, um Programme und Daten dauerhaft und stromausfallsicher abzuspeichern. Beim Genie I ist hierfür ein Kassettenrekorder fest eingebaut, auch beim Genie II und Speedmaster läßt sich ein Rekorder leicht ohne weiteren Ausbau anschließen. Beim Genie III ist kein Rekorderanschluß vorhanden.

Die Arbeit mit Kassettenspeicherung ist 200mal langsamer als bei Diskettenspeicherung, und die Fehlerquote beim Lesen und Schreiben von Programmen und Daten liegt deutlich höher als beim Floppybetrieb. Aus diesem Grunde erlauben alle Genie-Modelle und der Speedmaster den Anschluß von Floppy-Disk Stationen, mit denen eine sehr viel schnellere, bequemere und sicherere Daten- und Programmspeicherung möglich ist.

Zusammengefasst sind es also zwei gravierende Vorteile, die eine Floppystation, auch Diskettenstation genannt, bieten kann:

* Schnelles Abspeichern und Laden

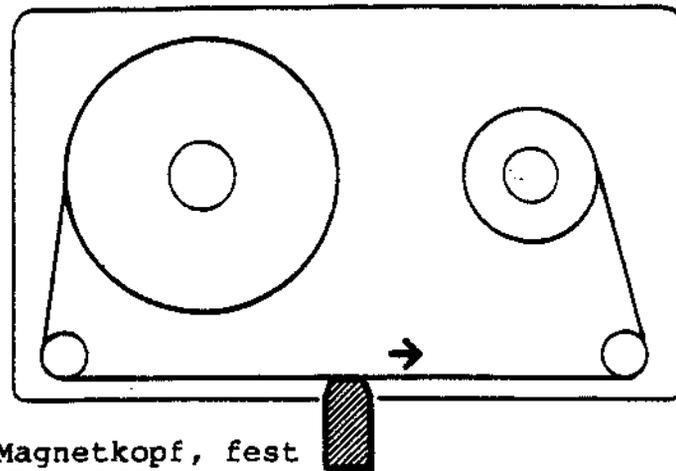
Während bei einem Kassettenrekorder die Datenübertragung mit 500 Bit pro Sekunde abläuft, das sind 62,5 Zeichen pro Sekunde, beträgt die Übertragungsgeschwindigkeit beim Floppy-Betrieb ca. 12500 Zeichen pro Sekunde.

* Wahlfreier Zugriff auf die gespeicherten Informationen

Auf der Kassette werden die Informationen eindimensional, hintereinander, in einer langen Reihe gespeichert. Für die Aufzeichnung von Daten oder Programmen muß man eine freie Bandstelle suchen und vor dem Lesen von das Band wiederum so einstellen, daß die gewünschte Information beim Anlauf des Antriebs sofort gefunden wird. Demgegenüber braucht man sich beim Floppy-Disk-Betrieb überhaupt nicht darum zu kümmern, auf welcher Stelle der Diskette die Speicherung erfolgt. Jede abzulegende Information erhält vor der Speicherung vom Programmierer einen Namen und kann dann durch Aufruf dieses Namens wieder von der Floppy in den Arbeitsspeicher des Computers geladen werden. Man hat also sofortigen, wahlfreien Zugriff auf alle gespeicherten Informationen.

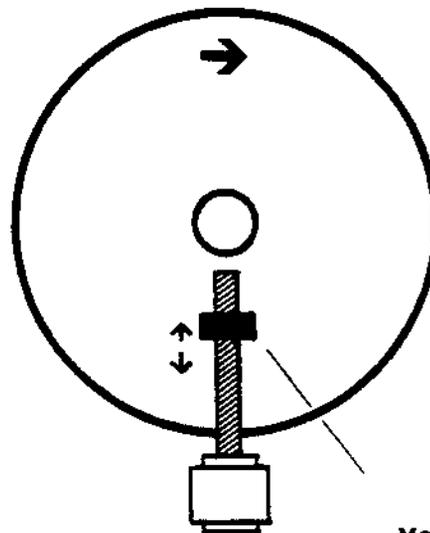
Die Vorteile, die Ihnen durch die Arbeit mit Floppystationen geboten werden, erfordern jedoch ausführliche Kenntnisse über die Handhabung und Behandlung der Disketten und die Funktionen des Disketten-Betriebssystems.

Lesen Sie daher zuerst dieses Handbuch genau durch.



Eindimensionale Aufzeichnung
auf Kassette

Um die gesuchte Aufzeichnung zu finden
muß das Band solange laufen bis der
Anfang der Aufzeichnung vor dem festen
Lesekopf steht.



Zweidimensionale Aufzeichnung
auf Diskette

Um die Aufzeichnung zu finden, wird
zunächst der Lesekopf radial in die
richtige Spur gebracht und dann wäh-
rend einer Umdrehung die Aufzeich-
nung gelesen.

Die Hardware

=====

Zur Anwendung des Diskettenbetriebssystems benötigen Sie entweder einen Speedmaster mit Floppystation, ein Genie III oder ein Genie I/II plus Floppylaufwerk mit eingebautem Controller.

Die Computer Genie I/II haben serienmäßig einen Arbeitsspeicher mit einer Kapazität von 48 K-Byte eingebaut. Diese Speicherkapazität reicht für den Floppy- und Kassetten-Betrieb in der Regel aus. In letzterem Fall haben Sie volle 48 K-Byte Arbeitsspeicher zur Verfügung.

Beim Speedmaster und Genie III liegt der BASIC-Interpreter nicht wie bei den Rechnern Genie I/II als ROM-Inhalt vor, sondern muß bei Bedarf von einer Diskette in den RAM geladen werden. Aus diesem Grunde ist der Betrieb dieser Rechner ausschließlich mit Diskettenlaufwerken möglich.

Alle oben erwähnten Computer können mit 1 bis 4 Floppy-Laufwerken, an einem Kabel hintereinander, verbundenen werden. Das erste Laufwerk am Kabel, vom Rechner aus gesehen, erhält die Nummer 0, die weiteren dann die Nummern 1, 2 und 3. Bei den TCS-Doppellaufwerken trägt jeweils das linke Laufwerk die Nummer 0. Beim Genie III erkennt man das Laufwerk 0 nach Betätigung der beiden RESET-Tasten: Das Laufwerk, bei dem sich der Antriebsmotor dreht, also die Leuchtdiode neben der Verschlussklappe aufleuchtet, hat die Nummer 0.

Es können verschiedene Laufwerkstypen an die Rechner angeschlossen werden.

Das Genie III ist serienmäßig mit doppelseitigen 80 Spur-Laufwerken bestückt und bietet dem Benutzer den schnellen Zugriff auf insgesamt 1,4 M-Byte.

Beim Genie I/II und Speedmaster stehen daneben auch noch einseitige 40 Spur-Laufwerke zur Verfügung.

Die Aufzeichnungen können außerdem mit doppelter Schreibdichte erfolgen, wobei die Informationen näher aneinandergerückt werden. (DOUBLE DENSITY).

Das Betriebssystem G-DOS ist für doppelt-dichte und doppelseitige Aufzeichnungen geeignet und die Hardware dazu ist sowohl in den FC-Laufwerken für Genie I/II, als auch auf der Floppykarte im Speedmaster, als auch im Genie III enthalten.

Die Mini-Floppys drehen sich im Laufwerk mit 300 Umdrehungen pro Minute, wobei eine Toleranz von 1,5 % zugelassen ist. Der Antriebsmotor läuft immer nur während der kurzen Zeit, in der geladen oder gespeichert wird.

Der Schreib/Lesekopf wird dabei durch einen weiteren Motor (Steppermotor) in kleinen Schritten radial bewegt.

Während einer Diskettenoperation werden alle Motoren der Laufwerke gleichzeitig ein- und ausgeschaltet. Ein Schreib- bzw. Lesevorgang ist jedoch zu einem gegebenen Zeitpunkt nur mit einem der angeschlossenen Laufwerke möglich.

Die Floppy-Disk

=====

Eine Floppy-Disk - auch Diskette oder nur Floppy genannt - ist eine dünne flexible Kunststoffscheibe, die beidseitig mit einer magnetisierbaren Schicht versehen ist. Auf dieser Schicht lassen sich, ähnlich wie bei einem Tonband, mit Hilfe eines beweglichen Magnetkopfes Aufzeichnungen machen, die dann später wieder gelesen oder gelöscht werden können.

Typisch für alle Disketten ist, daß sie in einer fest verklebten Hülle stecken und auch mit dieser Hülle in das Laufwerk geschoben werden. In der Mitte der Hülle befindet sich ein größeres Loch, in das die Antriebseinrichtung eingreift. Ein länglicher Schlitz ermöglicht den Kontakt des Magnetkopfes mit der Diskettenoberfläche. Das neben der Antriebsöffnung liegende, sogenannte Indexloch dient dem Floppy-Controller zu Synchronisationszwecken.

Mittels einer Auskerbung am Rand der Disketten erkennt das Laufwerk, ob Schreib- und Löschezugriffe auf die Disk erlaubt sind oder nicht. Wird dieser Schlitz mit einem undurchsichtigen Klebestreifen (Lichtschranke!) verschlossen, so ist die Diskette "schreibgeschützt" und ihr Inhalt kann auch selbst versehentlich nicht mehr geändert werden. Das Auslesen von Informationen ist immer möglich.

Beim Betriebssystem G-DOS werden nur 5 1/4" Disketten, auch Mini-Floppys genannt, verwendet.

Grundsätzlich erfolgt die magnetische Aufzeichnung bei einer Diskette nicht wie bei einer Schallplatte in einer langen Spirale, sondern in konzentrischen Ringen, die Spuren genannt werden. Jede Spur wiederum ist in einzelne Sektoren unterteilt. 5 Sektoren werden auch eine Einheit (= 1280 Byte) genannt.

Die außen liegende Spur wird immer mit 00 bezeichnet. Bezüglich der Anzahl der Spuren und der Sektoren pro Spur gibt es jedoch erhebliche Unterschiede.

Damit der Computer die verschiedenen Spuren und Sektoren einer Diskette erkennen kann, muß diese "formatiert" werden. Man versteht darunter das Einschreiben von bestimmten Informationen, aus denen der Rechner die gerade unter dem Magnetkopf vorbeilaufende Spur, sowie den Anfang bzw. die Nummer eines Sektors erkennen kann. Außerdem werden zu jedem Sektor Prüfzahlen aufgezeichnet, mit deren Hilfe unerwünschte Veränderungen bzw. Schreib- und Lesefehler festzustellen sind.

Unter G-DOS werden ausschließlich "softsektorierte" Disketten verwendet, die nur ein Indexloch zur Erkennung des ersten Sektors auf einer Spur besitzen.

Der zu verwendende Diskettentyp hängt vom Laufwerk ab. Es ist einleuchtend, daß bei dichter Aufzeichnung eine höhere Qualität erforderlich ist.

In diesem Zusammenhang werden Disketten mit folgenden Spezifikationen angeboten:

einfache / doppelte Schreibdichte
ein- / doppelseitige Aufzeichnung
40 / 80 (bzw. 77) Spuren

das heißt, der Hersteller garantiert für deren Einhaltung.

Auf Grund der hohen Qualitätsanforderungen der Hersteller an das Material der Disketten ist es aber möglich, daß z.B. auch eine nur für einseitige Aufzeichnung zertifizierte Diskette bei doppelseitiger Verwendung fehlerfrei arbeitet.

Kennbuchstaben für Disketten

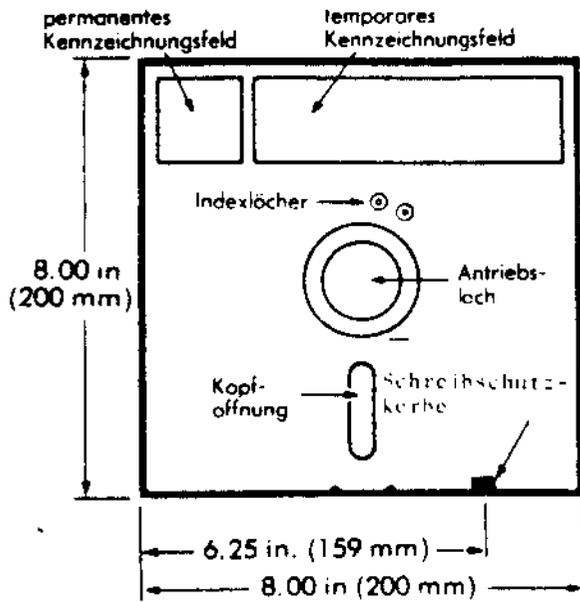
SD = SINGLE DENSITY = einfache Schreibdichte
DD = DOUBLE DENSITY = doppelte Schreibdichte

SS = SINGLE SIDED = einseitige Aufzeichnung
DS = DOUBLE SIDED = doppelseitige Aufzeichnung

1.3.1 Verschiedene Diskettenformatierungen

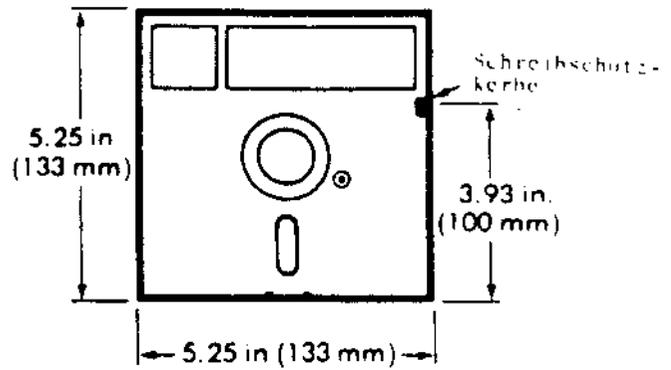
| Einseitige Aufzeichnung 5 1/4" | | | | |
|--------------------------------|--------|------------|-------------|------------|
| Schreibdichte | Spuren | Sekt./Spur | Sekt. Total | Kapazität |
| einfach | 40 | 10 | 400 | 102 K-Byte |
| doppelt | 40 | 18 | 720 | 184 K-Byte |
| einfach | 80 | 10 | 800 | 204 K-Byte |
| doppelt | 80 | 18 | 1440 | 368 K-Byte |

| Doppelseitige Aufzeichnung 5 1/4" | | | | |
|-----------------------------------|--------|------------|-------------|------------|
| Schreibdichte | Spuren | Sekt./Spur | Sekt. Total | Kapazität |
| einfach | 40 | 20 | 800 | 204 K-Byte |
| doppelt | 40 | 36 | 1440 | 368 K-Byte |
| einfach | 80 | 20 | 1600 | 408 K-Byte |
| doppelt | 80 | 36 | 2880 | 736 K-Byte |



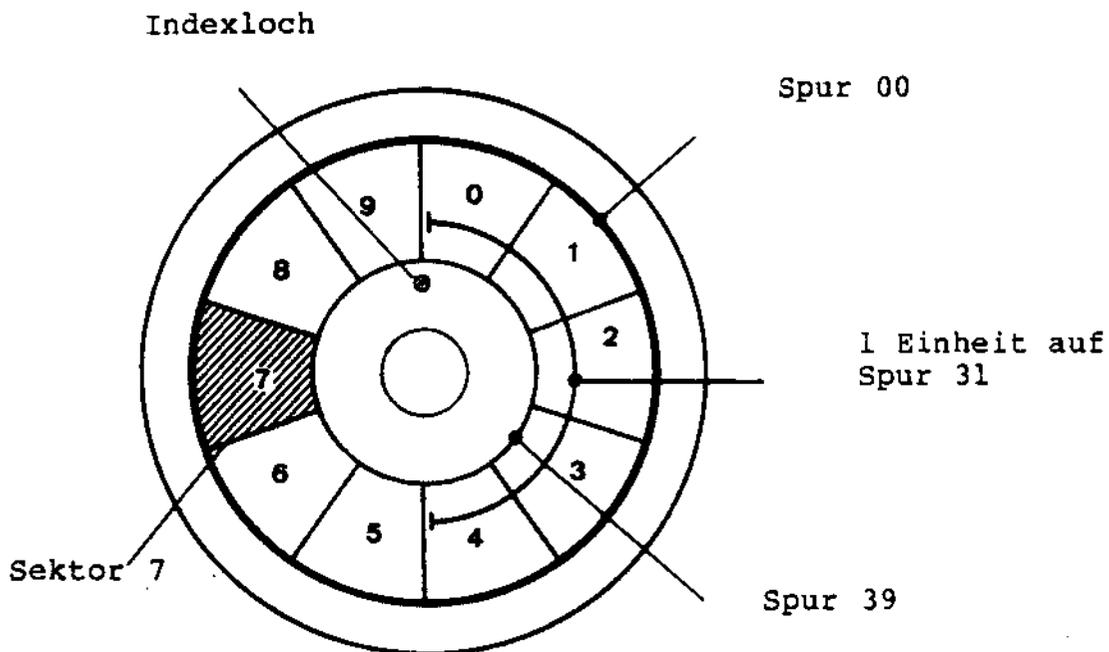
Normaldiskette 8"

Zum Schreiben muß Kerbe geschlossen sein.



Minidiskette 5 1/4"

Zum Schreiben muß Kerbe offen sein.



Mini Floppy = 5 1/4" Diskette mit 40 Spuren und normaler Schreibdichte (10 Sektoren)

Behandlung und Handhabung einer Diskette

=====

Obwohl Floppys verhältnismäßig unempfindlich sind, sollten Sie doch einige Regeln beim Umgang mit ihnen beachten, um keine Pannen beim Betrieb zu erleben. Sie müssen wissen, daß die Aufzeichnungen sehr gedrängt erfolgen. Auf einer Spurlänge von ca. 10 mm werden mehrere Tausend Einzelinformationen (Bits) gespeichert. Wenn aber auch nur ein Bit wegen Verschmutzung oder Beschädigung der Disk-Oberfläche falsch gelesen wird, kann ein ganzes Programm unbrauchbar werden.

* * * * *
* Die Floppy muß immer mit der Schreibe- *
* und dem länglichen Leseschlitz nach *
* hinten in die Öffnung *
* des Laufwerks eingeschoben werden. *
* Am besten wird sie dort angefaßt, *
* wo sich das Etikett auf *
* der Hülle befindet. Nach dem Einschie- *
* ben der Diskette muß *
* die Klappe am Laufwerk wieder geschlo- *
* ssen werden. *
* * * * *

Die folgenden Hinweise sollten Sie unbedingt befolgen:

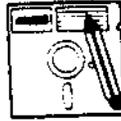
- * Niemals die Diskette am Leseschlitz berühren.
- * Diskette nach Gebrauch sofort mit dem Etikett nach oben in die Schutzhülle einschieben.
- * Diskette niemals falten, biegen oder knicken.
- * Niemals mit Kugelschreiber oder hartem Bleistift auf das Etikett schreiben, da Gefahr des Durchdrückens besteht.
- * Diskette niemals in der Nähe magnetischer Felder z.B. auf einem Netzgerät, Monitor, Telefonapparat, Lautsprecher u. ä. lagern, da dadurch Informationen auf ihr gelöscht werden können.
- * Diskette niemals einer Temperatur über ca. 50 Grad Celsius aussetzen. Vorsicht bei direkter Sonnenstrahlung, (z.B. im Auto)! Die schwarze Hülle kann sich dabei sehr stark aufheizen und verformen.

Es sei noch darauf hingewiesen, daß Disketten mit 80 Spuren und/oder doppelter Schreibdichte empfindlicher sind, als Disketten mit 40 Spuren und einfacher Dichte.

Behandlung von Disketten:



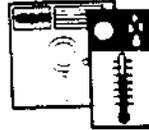
Magnetfläche
nicht berühren!



Nicht hart
beschriften!



Kein
Magnetismus !



Nicht feucht,
warm oder
kalt lagern !

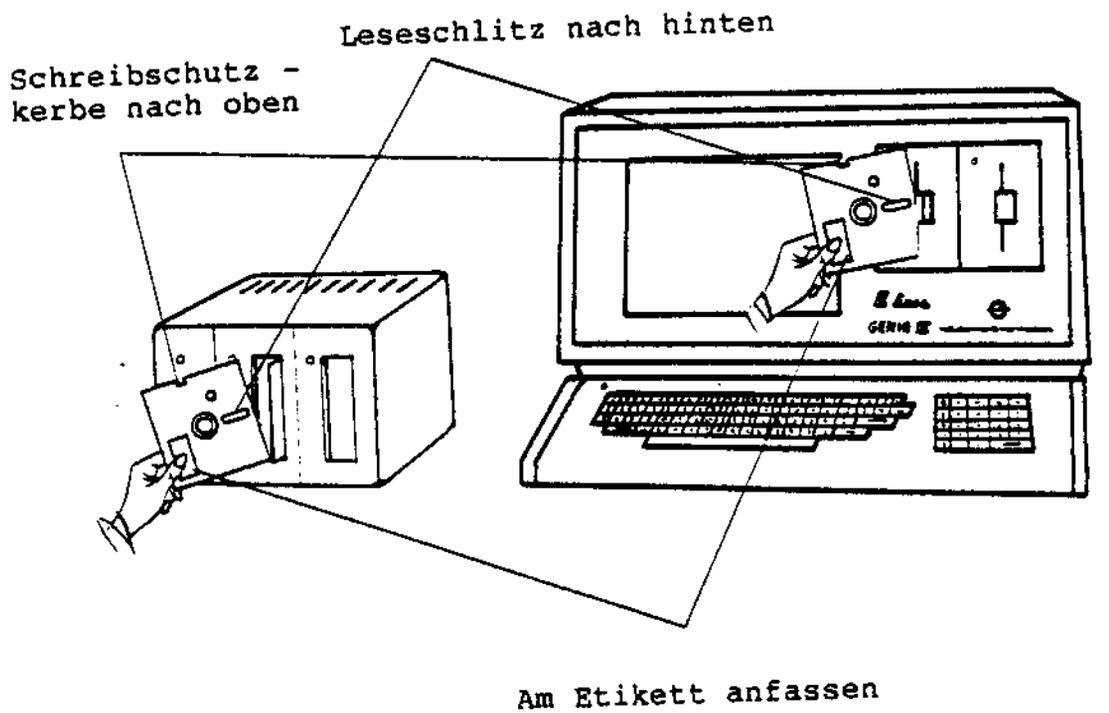


Nicht falten
knicken oder biegen!



Nach Gebrauch
zurück in die
Schutzhülle!

Richtiges Einschieben der Diskette:



Speedmaster, Genie I/II

Genie III

Als Besitzer des Genie-Disk-Operating-Systems (G-DOS), verfügen Sie über drei verschiedene, jedoch miteinander korrespondierende Betriebssysteme für Ihren Computer.

* Microsoft BASIC Level II

Die Arbeit mit dem Level II BASIC, welches beim Genie I/II als ROM-Inhalt vorliegt, ändert sich bis auf das Verfahren zum An- und Abschalten des Rechners nicht gegenüber einem Rechner ohne Diskettenstation.

* Disk-BASIC

Mit Hilfe des Disk-BASIC auf Ihrer System-Diskette wird das Level II BASIC noch um einige Befehle erweitert, insbesondere erhalten Sie die Möglichkeit, Dateien und Programme auf Diskette abzulegen und auch wieder einzulesen.

* G-DOS

Es gibt bereits eine große Anzahl von verschiedenen Betriebssystemen, die den Floppy-Disk Betrieb mit Kleincomputern ermöglichen. Sie unterscheiden sich zwar in vielen Einzelheiten, haben aber auf der anderen Seite auch viele Gemeinsamkeiten.

G-DOS ist aus dem Betriebssystem NEWDOS80.2 entstanden und bietet dem Programmierer eine Vielzahl von Programmiermöglichkeiten.

Ein genaues Studium des Handbuchs und einige Zeit praktischer Erfahrung sind jedoch unumgänglich, um das System perfekt zu beherrschen.

Mittels G-DOS können Disketten in verschiedenen Aufzeichnungsarten geschrieben und gelesen werden. So sind Aufzeichnungen bis zu 80 Spuren, mit doppelter Schreibdichte, für ein- oder zweiseitige Laufwerkstypen ohne weiteres möglich.

Mit dem Erwerb der G-DOS Diskette haben Sie also die Fähigkeiten Ihres Rechners um ein vielfaches erweitert. Sie können mit Programmen arbeiten, die für Kassettenspeicherung ungeeignet sind. Außerdem ist eine bequeme und schnelle Daten- bzw. Programmsicherung gewährleistet.

Dateien sind entweder durch wahlfreien Zugriff (RANDOM ACCESS) oder durch die sequentielle Methode zugänglich. Eine genauere Beschreibung dieser beiden Methoden erfolgt im weiteren Verlauf dieses Handbuchs.

Vorbereitungen zur Inbetriebnahme

Bevor Sie mit dem Floppy-Betrieb beginnen können, müssen Sie die Rechner Genie I/II und Speedmaster mit der Floppy-Station richtig verbinden (entfällt bei Genie IIs, Genie IIIs und IIIs).

Beim Anschluß der Kabel muß die
Stromzuführung aller Geräte
unterbrochen sein.

Beim Genie I/II wird das 50-adrige Kabel des FC-Adapters auf den 50-poligen Bus-Stecker an der Rückseite des Computers gesteckt. Es muß dabei so angebracht werden, daß es nach unten aus der Anschlußbuchse herausführt. Sollte es falsch angeschlossen sein, merkt man dies beim späteren Einschalten daran, daß das Laufwerk nicht anläuft und sich der Rechner nicht mit READY meldet, sondern auf dem Bildschirm nur wirre Zeichen anzeigt. In diesem Fall muß die Anlage wieder ausgeschaltet und das Kabel gedreht werden.

Die Karte FLOPPY 5/8 ermöglicht beim Speedmaster den Anschluß einer einfachen Floppystation (ohne Floppycontroller). Das Kabel der Floppystation muß beim Anschluß auf den 34-poligen Pfostenstecker der Floppykarte gesteckt werden.

Bevor Sie nun die Geräte einschalten, muß noch auf zwei wichtige Regeln hingewiesen werden.

Es dauert eine gewisse Zeit, bevor alle Baugruppen eines elektronischen Gerätes nach Einschaltung mit der Betriebsspannung versorgt werden und es ordnungsgemäß arbeiten kann.

Bei einem Radio ist dieser Vorgang z.B. für den meist im Lautsprecher zu hörenden "Einschaltknacks" verantwortlich. Was beim Radio der Lautsprecher, ist beim Laufwerk der Magnetkopf:

Es kann also passieren, daß ein unerwünschter Impuls auf eine im Laufwerk befindliche Diskette aufgezeichnet wird, der die dort gespeicherten Daten zerstört. Je nachdem, an welcher Stelle dies passiert, kann ein Sektor für die Datenaufzeichnung unbrauchbar werden, was nicht unbedingt sofort auffallen muß, ein Programm zerstört werden, womit es verloren ist oder schlimmstenfalls das Inhaltsverzeichnis der Disk in Mitleidenschaft gezogen werden, wodurch diese teilweise oder vollständig unlesbar wird, da der Rechner sich auf ihr nicht mehr zurechtfindet.

Regel 1: Floppys sollten nur ins Laufwerk gesteckt werden, wenn dieses eingeschaltet ist, bzw. aus ihm entnommen werden, bevor es ausgeschaltet wird.

Regel 2: Der Zugriff des Rechners auf die Diskette darf in keinem Fall unterbrochen werden, weder durch Ausschalten, noch durch Herausnehmen der Diskette aus dem Laufwerk.

Dies würde mit großer Wahrscheinlichkeit zu einem Datenverlust führen, da der Computer nicht mehr in der Lage ist, eine evtl. laufende Aufzeichnung ordnungsgemäß zu beenden, bzw. das Vorhandensein neuer Daten im Inhaltsverzeichnis zu vermerken. Ein Zugriff des Rechners auf die Diskette ist am Aufleuchten der Leuchtdiode unmittelbar neben der Verschlussklappe zu erkennen.

Einschalhinweise

=====

Nachdem Sie die Floppy-Station über das Anschlußkabel mit dem Rechner verbunden haben (entfällt bei Genie II, III und IIIs), vergewissern Sie sich noch einmal, ob alle Kabel, einschließlich Spannungsversorgung, richtig angeschlossen sind.

Beim Genie I/II und IIIs ist es im Gegensatz zum Genie III, IIIs und Speedmaster möglich, auch ohne Diskettenlaufwerke mit Level II BASIC zu arbeiten. Um in dieses Level II BASIC bei einer angeschlossenen Floppystation zu gelangen, müssen Sie entweder beim Einschalten des Rechners gleichzeitig die BREAK-Taste festhalten oder aber nach dem Einschalten gleichzeitig BREAK und RESET drücken. Auf dem Monitor erfolgt dann z.B. oben links die Meldung

```
READY?  
READY?
```

Nach Betätigung von NEW LINE befinden Sie sich im Level II BASIC, welches durch

```
READY  
>
```

unten links angezeigt wird.

Für die Arbeit mit Floppy-Laufwerken ist das Betriebsprogramm, das auch DOS (Disk Operating System) genannt wird, notwendig. Dieses Programm, oder zumindest Teile davon, müssen im RAM des Rechners stehen. Dadurch gehen rund 10 K-Byte Speicher verloren, sodaß Ihnen noch ca. 38 K-Byte freier Arbeitsspeicher unter Benutzung von Disk-BASIC zur Verfügung stehen.

Die Diskette, auf der sich das Betriebssystem G-DOS befindet, muß immer in Laufwerk Nummer 0 eingelegt werden. Auf ihr befinden sich alle für den Floppy-Betrieb notwendigen Programme. Sie werden von dort, je nach Bedarf, in den Arbeitsspeicher des Computers geladen.

Wenn Sie mit Ihrem G-DOS arbeiten wollen, muß die Diskette gebootet werden. Mit dem englischen Wort "booten" bezeichnet man das erste Laden des Betriebssystems von der System-Diskette in den Arbeitsspeicher. Wenn gemeinsam durch einen Hauptschalter eingeschaltet wird, müssen alle die Betriebsbereitschaft anzeigenden Lampen leuchten, und das Laufwerk Nummer 0 wird einige Sekunden lang laufen, was auch deutlich am surrenden Geräusch zu hören ist.

Beim Genie III erscheint kein Bild und beim Genie I/II ein zufälliges Zeichenmuster. Das Genie IIIs, IIIs und der Speedmaster geben eine dem Gerät entsprechende Meldung aus.

Jetzt kann die System-Diskette in den Schlitz des Laufwerks 0 eingelegt und die dazugehörige Klappe geschlossen werden.

Wenn Sie nun bei den Rechnern Genie IIs, III, IIIs und Speedmaster die beiden RESET-Tasten auf dem Keyboard drücken, wird das DOS in den Speicher des Rechners geladen (gebootet). Beim Genie I/II ist dazu der auf der Rückseite des Gerätes befindliche RESET-Knopf zu betätigen.
 Sie hören nun zunächst einige surrende und klickende Geräusche vom Laufwerk Nummer 0. Dann erscheint auf dem Bildschirm eine Ankündigung mit dem Titel des G-DOS, z.B.

G ENIE **DOS 2.1b** T C S Computer GmbH GENIE I/II

Befehlseingabe:

Ihr Gerät ist für den Floppy-Betrieb bereit, und Sie können nun direkt Daten von Ihren Disketten lesen oder darauf abspeichern.

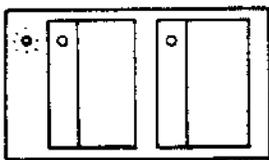
Wenn Sie mit fertigen Programmen arbeiten, kann die dazu gelieferte Diskette bereits das notwendige Betriebsprogramm enthalten, sodaß ein besonderes Booten nicht notwendig ist. Es gibt aber auch Programme, die ein vorheriges Booten erfordern und danach erst zu laden sind.

In jedem Fall müssen Sie das zum Programm gelieferte Handbuch zu Rate ziehen, da es sehr viele verschiedene Möglichkeiten gibt, mit einem Computer zu arbeiten.

Der Vorgang des Einschaltens:

Genie I/II

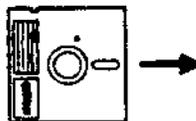
Hauptschalter einschalten



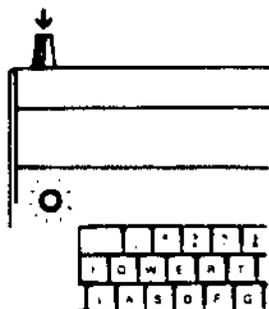
Hauptkontrolllampe und Betriebslampe leuchtet

Warten bis Betriebslampen dunkel sind

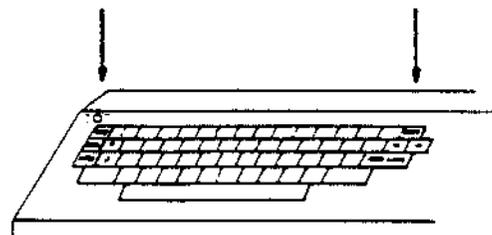
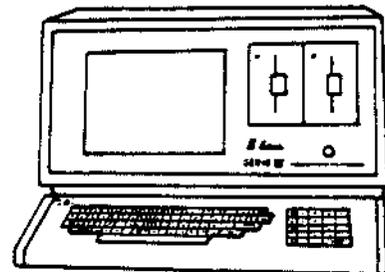
Systemdiskette einlegen



RESET drücken



Genie III



DOS und BASIC

=====

Wenn Sie Ihren Computer mit einer G-DOS Systemdiskette in Betrieb genommen haben, erscheint die Anzeige

Befehlseingabe:

Sie können dann alle im DOS zugelassenen Kommandos oder Aufrufe von Programmen eingeben.

Bekanntlich befindet sich auf allen G-DOS System-Disketten außer dem Floppy-Betriebssystem auch die Erweiterung des BASIC Level II zum Disk-BASIC.

Dies bedeutet, daß man immer darauf achten muß, ob sich der Rechner im DOS oder in BASIC befindet. Da normalerweise nach dem Booten zunächst das Betriebssystem aufgerufen wird, muß BASIC immer, wenn es gebraucht wird, durch Eingabe des Kommandos

BASIC

geladen werden. Der Computer arbeitet dann solange in dieser Programmiersprache bis durch Eingabe des Kommandos CMD"S" das DOS wieder aufgerufen wird.

Während Sie in BASIC arbeiten, können einzelne DOS-Kommandos auf einfache Weise angerufen werden, ohne daß das BASIC verlassen werden muß. Hierfür sind die DOS-Kommandos in Anführungsstrichen unter Voranstellung der Buchstaben CMD einzugeben. So gehören zum Beispiel die wichtigen Befehle I bzw. DIR, durch die die Anzeige des Inhaltsverzeichnisses einer Diskette veranlaßt wird, zu den DOS-Kommandos. Sie lassen sich in BASIC wie folgt eingeben:

```
CMD"DIR,0"   oder   CMD"DIR :0"   oder   CMD"DIR 0
CMD"I,0"     oder   CMD"I :0"     oder   CMD"I 0
```

In allen Fällen wird das Inhaltsverzeichnis des Laufwerks mit der Nummer 0, (also das mit der Systemdiskette), angezeigt. Die Anführungszeichen hinter dem DIR- bzw I-Befehl, sowie die Doppelpunkte können auch weggelassen und die Kommata durch Leerzeichen (Blanks) ersetzt werden.

Beispiel:

Das Disk-BASIC wird über das Kommando BASIC aus dem DOS heraus aufgerufen. Dabei erfolgt keine Speicherplatzreservierung und es können später nur 3 Dateien geöffnet werden. Nach dieser Befehlseingabe befindet sich das System im BASICmodus, was auf dem Bildschirm z.B wie folgt angezeigt wird:

```
GENIE I/II  B A S I C / erweiterte Version für Diskettenbetrieb
unter G-DOS 2.1b
```

```
READY
>
```

Wird BASIC jedoch z.B. mit dem Kommando

BASIC,60000,5

aufgerufen, erfolgt ein Speicherschutz oberhalb 60000 um dort eventuell Maschinenprogramme abzulegen.

Weiterhin können nach Eingabe des Befehls bis zu 5 Dateien gleichzeitig geöffnet werden.

Der Wert zur Platzreservierung von Dateien läßt sich auf maximal 15 einstellen.

Beispiel:

Abschalten vom DOS und erweiterten Disk-BASIC.

Es ist auch möglich ohne DOS und erweitertes Disk-BASIC zu arbeiten und nur das normale Microsoft Level II BASIC im Speicherbereich von 0000H bis 2FFFFH zu aktivieren.

Der DOS-Befehl dazu lautet:

B2

Je nach Rechner erscheint nach dieser Eingabe links oben die Meldung

READY?

BASIC?

Mem-Size?

Ein Rücksprung zum G-DOS ist dann mit der Eingabe

SYSTEM

/0

möglich.

Bei den Rechnern Genie I/II/IIs und Speedmaster muß jedoch zuvor noch POKE 14316,254 eingegeben werden, da sonst das System "hängt".

(Entfällt bei G-DOS Disketten in einfacher Schreibdicke).

Beispiel:

Booten Sie Ihre Systemdiskette und rufen Sie Disk-BASIC in der DOS-Befehlsebene mit dem Kommando BASIC auf.

Gehen Sie mittels des Kommandos CMD"S" aus dem BASIC-Modus in die DOS-Befehlsebene zurück.

Rufen Sie das Level II BASIC mit Hilfe des DOS-Befehls B2 auf.

Wenn Sie nun im Level II BASIC den Befehl SYSTEM und /0 eingeben, (beim Speedmaster und Genie I/II/IIs das POKEN vorher nicht vergessen !!!), befinden Sie sich wieder in der DOS-Befehlsebene, was an der Meldung "Befehlseingabe" zu erkennen ist.

Das Kopieren einer Diskette

Wenn Sie Ihre Diskette mit dem Betriebssystem ordnungsgemäß erworben haben, dürfen Sie für den eigenen Gebrauch so viele Kopien herstellen, wie Sie benötigen.

Von diesem Recht sollten Sie so bald wie möglich nach dem Kauf des Systems Gebrauch machen, da nie ausgeschlossen werden kann, daß eine System-Diskette beschädigt wird. Heben Sie also das Original gut auf und arbeiten Sie nur mit Duplikaten.

Es muß allerdings auch erwähnt werden, daß gewisse kommerzielle Programme nicht kopierbar sind. In diesem Fall verpflichtet sich der Lieferant jedoch im allgemeinen zur Ersatzlieferung, falls das Original beschädigt wird.

Das Kopieren einer ganzen Diskette wird durch ein DOS-Kommando gesteuert. Der Benutzer wird dabei durch Hinweise auf dem Bildschirm aufgefordert, bestimmte Disketten in die verschiedenen Laufwerke einzulegen.

Dazu müssen Sie wissen:

SYSTEMDISKETTE ist die Floppy mit dem Betriebssystem (G-DOS),
QUELLDISKETTE ist die Floppy, die kopiert wird,
ZIELDISKETTE ist eine leere Floppy, auf die kopiert wird.

Besonders einfach ist der Kopiervorgang, wenn zwei Laufwerke vorhanden sind. In diesem Fall kommt die System-Diskette in das Laufwerk 0 und eine leere Diskette in Laufwerk 1.

Das DOS-Kommando zum Kopieren lautet:

```
COPY,0,1,tt.mm.jj
```

Dieser Befehl sagt aus, daß von Laufwerk 0 nach Laufwerk 1 kopiert werden soll. Die Kommata zwischen den einzelnen Ausdrücken können auch durch jeweils eine Leerstelle ersetzt werden. Das Datum, wobei tt den Tag, mm den Monat und jj das Jahr bedeuten und die Eingabe immer in zweistelligen Zahlen erfolgen muß, ist nicht unbedingt erforderlich. Wichtig ist, daß die Zwischenräume und Satzzeichen, genau wie vorgeschrieben, verwendet werden.

Wenn eine System-Diskette mit Hilfe von 2 Laufwerken kopiert werden soll, ist kein Diskettenwechsel notwendig. Die Hinweise und Fragen auf dem Bildschirm zeigen Ihnen genau, was Sie zu tun haben. Bei einer Sicherungskopie Ihrer Systemdiskette legen Sie eine leere Floppy in Laufwerk 1, beantworten nach Eingabe des COPY-Befehls die Fragen, ob formatiert werden soll und ob System und Quelle identisch sind, mit "J" und drücken dann, um den Formatier- bzw. Kopiervorgang zu starten, die NEW LINE-Taste.

Die Laufwerke werden nun einige Zeit abwechselnd laufen und dabei Ihre DOS-Diskette kopieren. Der erfolgreiche Abschluß einer Kopieraktion wird Ihnen durch die Angabe ENDE auf dem Bildschirm übermittelt.

Etwas umständlicher ist der Kopiervorgang wenn nur ein Laufwerk zur Verfügung steht. In diesem Fall ist es notwendig die Disketten mehrfach zu wechseln, wobei die auf dem Bildschirm erscheinenden Hinweise sorgfältig befolgt werden müssen.

Zunächst wird die System-Diskette geladen, damit das dort aufgezeichnete Kopierprogramm nach Eingabe des Kopier-Kommandos in den Arbeitsspeicher geschrieben werden kann.

Ist das Programm geladen, erscheint auf dem Bildschirm die Frage ob formatiert werden soll. Wenn Sie mit "J" antworten, werden Sie vom Rechner aufgefordert, die Quelldiskette in Laufwerk 0 einzulegen.

Nach Betätigung der NEW LINE-Taste läuft das Laufwerk kurz an, und es erscheint die Aufforderung, die Zieldiskette einzulegen.

Nachdem Sie nun abermals NEW LINE gedrückt haben, beginnt die Formatierung der eingelegten Diskette. Anschließend fordert der Rechner Sie wieder auf, die Quelldiskette ins Laufwerk zu legen.

Erst jetzt beginnt nach Betätigung von NEW LINE der eigentliche Kopiervorgang. Der erste Teil der Aufzeichnungen des Originals wird in den Arbeitsspeicher übernommen. Danach werden Sie wieder aufgefordert, die Zieldiskette in Laufwerk 0 einzulegen. Da der Arbeitsspeicher nicht ausreicht, den ganzen Inhalt der Original-Diskette aufzunehmen, muß der Lade- und Schreibvorgang mit Diskettenwechsel mehrmals wiederholt werden, bis die Anzeige

ENDE

und die Aufforderung, eine Systemdiskette in Laufwerk 0 einzulegen erscheint. Damit ist dann der Kopiervorgang erfolgreich beendet.

Vereinfachung:

Beim G-DOS wurde zusätzlich zum Befehl COPY noch das Zeichen

>

als Kopierbefehl implementiert.

Demzufolge hat der Ausdruck

> 0 1 17.05.83

die gleiche Bedeutung wie

COPY,0,1,17.05.83

Beide Befehle kopieren den Inhalt der Diskette in Laufwerk 0 auf eine Diskette in Laufwerk 1.

Beispiel:

Kopiervorgang einer Diskette bei der Verwendung von zwei Laufwerken. Der Rechner muß sich in der DOS-Befehlsebene befinden. Folgende Eingaben sind durchzuführen, wobei das Datum nicht unbedingt notwendig ist.

```
Befehlseingabe: COPY 0 1 06.05.83
Diskette wird kopiert
Diskette formatieren? (Ja/Nein) J
Sind System und Quelle identisch? (Ja/Nein) J
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 1
```

Nach dem Drücken der NEW LINE-Taste erfolgt die Formatierung der Diskette in Laufwerk 0. Anschließend beginnt ohne Pause der eigentliche Kopiervorgang. Auf dem Bildschirm erscheint dabei

```
Formatiere
Kopiere
ENDE
Befehlseingabe:
```

ENDE zeigt Ihnen an, daß die Kopieraktion fehlerfrei durchgeführt wurde.

Beispiel:

Kopiervorgang bei Verwendung von nur einem einseitigen 40-Spur-Laufwerk, bei einfacher Schreibdichte der Diskette: Auch hierbei hat sich der Rechner in der DOS-Befehlsebene zu befinden.

```
Befehlseingabe: COPY 0 0 06.05.83
Diskette wird kopiert
Diskette formatieren? (Ja/Nein) J
" NEW LINE ", wenn Quelldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 0
```

```
Formatiere
Kopiere
```

```
" NEW LINE ", wenn Quelldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Quelldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Quelldiskette in Laufwerk Nr. 0
```

```
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 0
```

```
ENDE
```

```
" NEW LINE ", wenn ==> Systemdiskette in Laufwerk Nr. 0
```

```
Befehlseingabe:
```

Das Formatieren einer Diskette

Jede neue Diskette muß vor der Benutzung dem verwendeten Laufwerk entsprechend formatiert werden. Dabei werden auf der Diskette Daten von evtl. vorhergehenden Aufzeichnungen gelöscht. Das Kommando zum Formatieren einer Diskette lautet:

NDF, Laufwerknummer, Diskname, tt.mm.jj

Bei diesem Kommando folgt dem Befehlswort nach einem Komma die Nummer des Laufwerks, in dem die neue Diskette formatiert werden soll. Dann kann der Diskette bei Bedarf ein Name mit bis zu 8 Zeichen und nach einem weiteren Komma das Datum angegeben werden.

Ähnlich wie beim Kopiervorgang können Sie auch hier die Kommata gegen Leerzeichen austauschen und den Namen der Diskette, sowie das Datum weglassen.

Der Rechner gibt Ihnen auf dem Bildschirm an, welche Diskette Sie in welches Laufwerk einzulegen haben und was Sie sonst noch zu tun haben.

Bei einer Anlage mit 2 Laufwerken enthält Laufwerk 0 dabei die Systemdiskette und Laufwerk 1 eine Leerdiskette bzw. eine Diskette, deren Inhalt gelöscht werden kann.

Sollte die zu formatierende Diskette für den Rechner lesbare Daten enthalten, erfolgt zur Sicherheit eine Abfrage, ob wirklich formatiert werden soll, wobei Sie dann die Wahl zwischen

(A)abbruch, (W)iederholung, (F)ortfahren

haben. Bei Abbruch geht der Rechner zurück in die DOS-Befehlsebene. Es ist aber auch möglich eine neue Diskette in Laufwerk 1 einzulegen und die Funktion zu wiederholen. Der eigentliche Formatiervorgang beginnt nach der Eingabe von F, was ein Löschen der auf der Diskette befindlichen Daten zur Folge hat.

Nachdem eine Diskette formatiert ist, prüft der Computer, ob der Vorgang ordnungsgemäß abgelaufen ist und legt anschließend die Systemdaten, wie z.B. Name und Datum, darauf ab.

Die erfolgreiche Beendigung einer Formatanweisung wird durch Ausgabe von

ENDE

auf dem Bildschirm angezeigt.

Selbstverständlich können Sie auch mit nur einem Laufwerk eine Diskette formatieren. Die Laufwerknummer wird in diesem Fall mit 0 angegeben.

Sie müssen dann jedoch die Frage, ob System und Ziel identisch sind, mit "N" beantworten, da sonst Ihre System-Diskette durch Überformatierung zerstört wird und sich der Computer nach der Beendigung des Formatiervorgangs sozusagen "aufhängt".

Beispiel:

Formatiervorgang bei 2 Laufwerken, wenn die Zieldiskette alte Daten enthält, die gelöscht werden können.

```
Befehlseingabe:NDF 1 TESTDISK 06.05.83
Diskette wird formatiert
" NEW LINE ", wenn Zieldiskette in Laufwerk NR. 1
```

```
Diskette hat Daten
Zieldiskettenname, -datum: Genie I      11.02.83
<A>bbruch, <W>iederholung, <F>ortfahren
```

Die in Laufwerk 1 liegende Diskette trägt in diesem Beispiel den Namen Genie I und wurde am 11.02.1983 erstellt.

Wenn der Rechner auf der Diskette zwar etwas findet, aber Name und Datum nicht erkennen kann, erscheint die Meldung:

```
Diskette hat Daten
Zieldiskettenname, -daten: ?????????? ??????????
(A)bbruch, (W)iederholung, (F)ortfahren
```

In beiden Fällen können Sie nun die Formatierung abbrechen (A), eine neue Diskette in Laufwerk 1 einlegen und die Funktion wiederholen oder mit dem Formatiervorgang fortfahren (F), was ein Löschen der auf der Diskette befindlichen Daten zur Folge hat.

```
F
Formatiere
Prüfen
Schreibe nun Systemdaten
ENDE
Befehlseingabe:
```

Durch ENDE wird der erfolgreiche Formatiervorgang angezeigt.

Beispiel:

Formatieren einer Leerdiskette mit nur einem Laufwerk:

```
Befehlseingabe:NDF 0 TESTDISK 06.05.83
Diskette wird formatiert
Sind System und Ziel identisch? (Ja/Nein) N
" NEW LINE ", wenn Zieldiskette in Laufwerk Nr. 0
```

```
Formatiere
Prüfen
Schreibe nun Systemdaten
ENDE
" NEW LINE ", wenn ==> Systemdiskette in Laufwerk Nr. 0
```

```
Befehlseingabe:
```

Das Inhaltsverzeichnis einer Diskette

=====

Durch das Betriebssystem wird auf jeder Diskette ein Inhaltsverzeichnis angelegt und aufgezeichnet. Dieses Inhaltsverzeichnis kann durch DOS-Kommandos angezeigt werden.

Hierbei gibt es zwei Möglichkeiten. Es werden entweder nur die Namen, unter denen Anwender-Programme und Dateien abgespeichert sind oder aber alle gespeicherten Informationen, mit Angaben über den ausgenutzten Speicherraum, angezeigt.

Für den ersten Fall lauten die DOS-Kommandos, um das Inhaltsverzeichnis der im Laufwerk 0 liegenden Diskette zu erhalten

I,0 bzw. DIR,0

Für die Anzeige des Inhaltsverzeichnisses der Diskette im Laufwerk 1 sind folgende Kommandos zu verwenden, wobei die Kommata wieder durch Leerzeichen ersetzt werden können

I,1 bzw. DIR,1

Für die vollständige Anzeige aller Informationen des Inhaltsverzeichnisses einer Diskette gibt es die Kommandos

I,n,I,S,A bzw. DIR,n,I,S,A

In diesen Befehlen, die genau so mit Zwischenräumen bzw. Satzzeichen eingegeben werden müssen wie sie hier geschrieben sind, bedeutet n die Laufwerknummer.

Die Angaben I,S,B,A hinter der Laufwerknummer haben folgende Bedeutung.

I = Invisible (unsichtbare) Dateien werden angezeigt

S = Systemprogramme werden angezeigt

B = Dateien, auf die geschrieben wurde, werden angezeigt

Bei der Angabe A = Alles wird die Datei-Information erweitert ausgegeben. Folgende Informationen kommen dann zum Dateinamen hinzu.

Ende : Satznummer/Byte des Dateiendes

log : logische Satzlänge

Anz. : Anzahl der Sätze

Einh. : Platzbedarf der Datei in Einheiten

Erw. : Anzahl der Erweiterungen, mit je bis zu 32 Einh.

Der Zusatz SYS beim Programmnamen bedeutet, daß es sich dabei um ein Systemprogramm (Teil vom DOS) handelt.

CMD deutet an, daß dieses Maschinen-Programm direkt durch Angabe seines Namens (und evtl. Parameter) wie ein DOS-Befehl aufgerufen werden kann.

Der Anhang JOB weist den Benutzer darauf hin, daß die Datei lediglich Befehlsfolgen für den Rechner enthält.

Beim kurzen Verzeichnis werden die Namen in 4 Spalten ausgegeben. Beim vollständigen Verzeichnis wird dagegen für jede Datei eine komplette Zeile verwendet. Da auf dem Bildschirm nur eine begrenzte Zeilenanzahl gezeigt werden kann, ist NEW LINE zur Fortschaltung der Anzeige zu drücken.

Inhaltsverzeichnis einer G-DOS Systemdiskette nach Eingabe des Kommandos I 0 bzw. DIR 0:

Laufw. 0 G-DOS FC 14.03.84 40 Spuren 61 Fr.Pl. 70 Einh.

| | | | |
|-----------|----------|--------------|------------|
| BASIC/CMD | JOB/CMD | PROGMOD/BAS | SUPER/CMD |
| ASM/CMD | VL/CMD | DIRCHECK/CMD | LIST64/BAS |
| ITOH/CMD | DEMO/JOB | FORMLIST/BAS | |

Inhaltsverzeichnis einer G-DOS Systemdiskette nach Eingabe des Kommandos I 0 I S A bzw. DIR 0 I S A:

Laufw. 0 G-DOS FC 14.03.84 40 Spuren 61 Fr.Pl. 70 Einh.

| | Ende | log | Anz. | Einh. | Erw. | SIBEF....HBS |
|--------------|--------|-----|------|-------|------|--------------|
| GDOS/SYS | 5/000 | 256 | 5 | 1 | 1 | SI.....HB6 |
| SYS6/SYS | 35/000 | 256 | 35 | 7 | 1 | SIB.....HB7 |
| SYS14/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS22/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| BASIC/CMD | 18/000 | 256 | 18 | 4 | 1 | ..B.....HB0 |
| JOB/CMD | 33/000 | 256 | 33 | 7 | 1 | ..B.....0 |
| INHALT/SYS | 15/000 | 256 | 15 | 3 | 1 | SI.....HB5 |
| SYS7/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS15/SYS | 3/026 | 256 | 4 | 1 | 1 | SIB.....HB7 |
| SYS23/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS0/SYS | 15/000 | 256 | 15 | 3 | 1 | SIB.....HB7 |
| SYS8/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS16/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS24/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| PROGMOD/BAS | 2/050 | 256 | 3 | 1 | 1 | ..B.....0 |
| SUPER/CMD | 10/000 | 256 | 10 | 2 | 1 | ..B.....HB0 |
| SYS1/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS9/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS17/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS25/SYS | 10/000 | 256 | 10 | 2 | 1 | SIB.....HB7 |
| SYS2/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS10/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS18/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS26/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| ASM/CMD | 33/167 | 256 | 34 | 7 | 2 | ..B.....0 |
| SYS3/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS11/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS19/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS27/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| VL/CMD | 10/051 | 256 | 11 | 3 | 1 | ..B.....HB0 |
| SYS4/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS12/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS20/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS28/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| DIRCHECK/CMD | 15/000 | 256 | 15 | 3 | 1 | ..B.....HB0 |
| SYS5/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS13/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS21/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| SYS29/SYS | 5/000 | 256 | 5 | 1 | 1 | SIB.....HB7 |
| LIST64/BAS | 2/082 | 256 | 3 | 1 | 1 | ..B.....0 |
| ITOH/CMD | 1/210 | 256 | 2 | 1 | 1 | ..B.....0 |
| DEMO/JOB | 2/001 | 256 | 3 | 1 | 1 | ..B.....HB0 |
| FORMLIST/BAS | 3/086 | 256 | 4 | 1 | 1 | ..B.....0 |

Freier Speicherplatz einer Diskette

Von den ca. 100 K-Byte, die auf einer normalen einseitigen Diskette mit 40 Spuren und einfacher Schreibdichte als Speicher-
raum zur Verfügung stehen, werden auf der System-Diskette ca.
2/3 für das Betriebssystem usw. benötigt. Der restliche freie
Raum steht dem Anwender zur Verfügung. Auf den Disketten in den
Laufwerken 1 bis 3, die bei einer Formatierung nur kleine Teile
des Betriebssystems erhalten, können knapp 100 K-Byte gespei-
chert werden.

Es ist zu beachten, daß jeder Speichervorgang auf einer Disket-
te mindestens eine Einheit, das sind 5 Sektoren gleich
1,25 K-Byte benötigt.

Der Computer meldet den freien Speicherraum der Disketten in
den angeschlossenen Laufwerken nach Eingabe des DOS-Kommandos

FREE

Angezeigt wird jeweils die Kopfzeile, die auch beim Aufruf des
Inhaltsverzeichnisses erscheint:

| | | | | | | |
|--------|---|----------|----------|-----------|-----------|----------|
| Laufw. | 0 | G-DOS FC | 11.02.83 | 40 Spuren | 27 Fr.Pl. | 24 Einh. |
| Laufw. | 1 | TESTDISK | 11.11.11 | 40 Spuren | 62 Fr.Pl. | 77 Einh. |

Befehlsliste des G-DOS aufrufen

Auf der System-Diskette gibt es noch ein weiteres Verzeichnis,
das durch die DOS-Kommandos

LIB bzw. ?

ohne weitere Parameter aufgerufen wird.

Dieses Verzeichnis enthält alle System-Befehle und alle Be-
zeichnungen der System-Hilfsprogramme, die mit einem DOS-Kom-
mando aufgerufen werden können.

Befehlsliste des Betriebssystems G-DOS 2.1b:

Befehlseingabe: LIB

| | | | | | | | |
|------|-------|-----|--------|--------|--------|-------|-------|
| 0 | @ | AIK | APPEND | ATTRIB | AUTO | B2 | BL |
| BOOT | BREAK | CLS | CONT | COPY | CREATE | DATUM | DIR |
| DISK | DO | DR | DUMP | E | FORM | FREE | HIMEM |
| I | INFO | JKL | KILL | LC | LF | LIB | LIST |
| LOAD | | N | NDF | PAUSE | PD | PORT | PRINT |
| PROT | PURGE | R | S | STMT | UHR | V+ | V24 |
| Z | ZEIT | & | ! | ; | / | ? | > |

M> DDE
Befehlseingabe:

Löschen von Dateien und Programmen

=====

Das Löschen einzelner Dateien, zu denen auch Programme gehören, ist mit Hilfe des Befehls

KILL

möglich. Dieser Befehl kann sowohl in BASIC als auch als DOS-Befehl ausgegeben werden. Er lautet:

KILL "Dateiname:n" als BASIC-Kommando

KILL Dateiname:n als DOS-Kommando

Wird beim Namen der zu löschenden Datei nicht die Laufwerksnummer n definiert, sucht G-DOS beginnend bei Laufwerk 0 nach dieser Datei und löscht die erste, die es findet. Um zu verhindern, daß auf diese Art und Weise Dateien unbeabsichtigt gelöscht werden, sollte immer ein Laufwerk definiert werden, wie z.B.

KILL TESTPROG/CMD:1

Sollen direkt mehrere Programme gelöscht werden, so bietet sich das Kommando

PURGE n

des G-DOS an, wobei n wieder für die Laufwerksnummer steht. Um dem Anwender die Entscheidung über das Löschen oder Nichtlöschen einer Datei oder eines Programms zu überlassen, fragt G-DOS nach Eingabe des PURGE-Befehls jedes Programm und jede Datei der angegebenen Diskette ab und fordert vom Benutzer eine Entscheidung (Ja/Nein/Ende).

Beispiel:

Auf einer Diskette in Laufwerk 1 befinden sich 6 Testprogramme. Mit Hilfe des PURGE-Befehls sollen die Programme TEST1-TEST4 gelöscht werden.

```
Befehlseingabe: PURGE 1
TEST1            Datei löschen ? (J/N/E) J
TEST3            Datei löschen ? (J/N/E) J
TEST6            Datei löschen ? (J/N/E) N
TEST2            Datei löschen ? (J/N/E) J
TEST4            Datei löschen ? (J/N/E) J
TEST5            Datei löschen ? (J/N/E) E
```

Nach der Eingabe von E (wie Ende) werden alle durch J bestimmten Dateien gelöscht. Anschließend meldet sich der Rechner wieder mit

Befehlseingabe:

Uhrzeit und Datum =====

Genie-DOS gibt dem Anwender die Möglichkeit, während der Einschaltdauer des Genie Computers eine Uhr und ein Datum mitlaufen zu lassen, um beides bei Bedarf im Programm zu verwenden.

Datum und Uhrzeit werden in einer Zeichenkette mit 17 Zeichen unter dem Namen TIMES gespeichert. Dieser Name kann in BASIC-Befehlen gebraucht werden, um das eingestellte Datum und die augenblickliche Uhrzeit im Programm mitzuverwenden.

Datum und Uhrzeit werden in folgender Form gespeichert:

tt.mm.jj hh:mm:ss

Alle Zahlen sind zweistellig, gegebenenfalls mit einer Vornull einzugeben.

Die Stunden werden von 00 bis 23 angezeigt und müssen auch so eingegeben werden.

Zur Eingabe von Datum und Uhrzeit sind die folgenden DOS-Kommandos zu verwenden

DATUM 13.05.83 stellt das Datum auf den 13.5.83
ZEIT 12:50:00 stellt die Zeit auf 12.50

Über die beiden Kommandos DATUM und ZEIT lassen sich die Werte auch wieder abrufen.

Mit dem DOS-Kommando UHR kann eine laufende Zeitanzeige in die obere rechte Ecke des Bildschirms gesetzt werden. Diese Anzeige läßt sich durch das DOS-Kommando

UHR,N

und anschließendem Drücken der CLEAR-Taste löschen und mit

UHR,J

wieder aufrufen.

Wichtig:

Bei gleichzeitigem Floppy- und Kassettenbetrieb muß, wenn während des Diskettenbetriebes auf eine Kassette gespeichert oder von dieser gelesen werden soll, die Uhr ausgeschaltet sein.

Das Ein- und Ausschalten der Uhr ist nur vom BASIC aus möglich und wird wie folgt ausgeführt:

CMD" T" schaltet die Uhr, den blinkenden Cursor
und die Autorepeat-Funktion ab

CMD" R" schaltet die Uhr, den blinkenden Cursor
und die Autorepeat-Funktion wieder ein

Beispiel:

```
Befehlseingabe: DATUM 26.06.84
Befehlseingabe: ZEIT 12:50:00
Befehlseingabe: DATUM
26.06.84
Befehlseingabe: ZEIT
12:50:57
```

Aufruf von Uhrzeit und Datum in BASIC mit Hilfe von TIME\$:

```
10 DA$=LEFT$(TIME$,8)
20 ZT$=RIGHT$(TIME$,8)
30 PRINT"Datum: ";DA$
40 PRINT"Uhrzeit: ";ZT$
```

Programmlauf:

```
>RUN
Datum: 26.06.84
Uhrzeit: 12:56:25
READY
>
```

Datum und Uhrzeit können im DOS direkt und im BASIC über das CMD-Kommando gesetzt werden. Der Aufruf von TIME\$ ist nur aus dem BASIC heraus möglich. Der Ausdruck TIME\$ entspricht einer String-Variablen und kann somit auch jederzeit entsprechend behandelt werden.

Umbenennung einer Datei oder eines Programms

=====

Um Datei- oder Programmnamen zu ändern benutzen Sie den Befehl

```
N Altdateiname,Neudateiname
```

Wiederholung eines Befehls

=====

Soll ein Befehlssatz mehrfach hintereinander durchgeführt werden, (z.B. beim Erstellen mehrerer Duplikate einer Diskette), bewirkt die Eingabe

```
R
```

nach Ablauf der ersten Befehlsausführung eine sofortige Wiederholung. R muß mit anschließendem NEW LINE eingegeben werden. Wurde versehentlich mehr eingetippt und die zusätzlichen Zeichen durch Backspace gelöscht, kommt es zu einer Fehlermeldung.

Das Abspeichern von Programmen

Als Benutzer brauchen Sie sich in keiner Weise darum zu kümmern, an welcher Stelle die Speicherung auf der Diskette erfolgen soll. Sie müssen dem Programm oder der Datei lediglich einen Namen geben. Dieser wird automatisch ins Inhaltsverzeichnis der Diskette eingetragen und dient zum späteren Wiederauffinden der Programmdatei. Da das DOS auf Grund des freien Speicherplatzes auf der Diskette entscheidet, an welcher Stelle der Name ins Inhaltsverzeichnis eingetragen wird, kann es ohne weiteres vorkommen, daß der Name Ihres zuletzt abgespeicherten Programms am Anfang oder auch mitten im Inhaltsverzeichnis erscheint.

Zum Abspeichern von BASIC-Programmen auf einer Diskette dient der SAVE-Befehl. Die Daten werden dabei entweder im ASCII-Format (wie im Listing) oder in der komprimierten Form, in der sie auch im Speicher stehen, auf der Diskette abgelegt.

Für den Programmnamen gibt es einige Regeln, die genau einzuhalten sind. Er kann aus bis zu 3 Teilen bestehen:

- * Der eigentliche Name darf aus bis zu 8 beliebigen alphanumerischen Zeichen bestehen, wobei das erste jedoch ein Buchstabe sein muß.
- * Dem Namen kann - aber muß nicht - nach einem Schrägstrich eine Kennung aus 1 bis 3 alphanumerischen Zeichen folgen.
- * Wenn Sie mehrere Laufwerke besitzen, so ist immer die Nummer des Laufwerks, auf dem gespeichert werden soll, nach einem Doppelpunkt anzufügen. Wenn Ihnen dies jedoch gleich ist, oder aber nur ein Laufwerk angeschlossen ist, so kann diese Angabe entfallen. Es wird in diesem Fall automatisch die erste nicht schreibgeschützte Diskette mit freiem Speicherplatz benutzt.

Die Befehle zur Abspeicherung eines Programms lauten:

| | |
|----------------------|--|
| SAVE"Programmname" | (Abspeichern eines Programms in komprimierter Form) |
| oder | |
| SAVE"Programmname",A | Abspeichern eines Programms in ASCII-Format) |

Gültige Programmnamen sind zum Beispiel

für Anlagen mit
einem Laufwerk:

TEST
TEST:0
TESTPROG/BAS:0

für Anlagen mit
zwei Laufwerken:

TEST:1
TESTPROG/BAS:1

Sind mehrere Laufwerke angeschlossen, so sucht das System, wenn der Programmname ohne Laufwerknummer angegeben ist, zunächst auf allen Disketten, ob bereits Daten unter dem gewählten Namen verzeichnet sind. Ist dies der Fall, wird die alte Aufzeichnung beim Abspeichern überschrieben und ist damit verloren.

Beispiel:

Geben Sie folgendes Programm ein:

```
10 CLS
20 ?"Dies ist ein Testprogramm"
30 REM Die Abspeicherung erfolgt in zwei Formaten
40 SAVE"TESTPROG/BAS:0":REM speichert komprimiert
50 SAVE"TESTPROG/ASC:0",A:REM speichert in ASCII
60 REM nach SAVE im ASCII-Format wird das Programm beendet
```

Wenn Sie dieses Programm mit RUN starten, läuft das Laufwerk 0 an und das Programm speichert sich zweimal auf der Diskette ab. Dies erfolgt zuerst in komprimierter Form als TESTPROG/BAS und dann im ASCII-Format als TESTPROG/ASC. Der Programmlauf wird allerdings nach Ausführung der ASCII-Speicherung in Zeile 50 abgebrochen (wie durch END).

Das Laden von Programmen

=====

Um Programme von Diskette wieder in den Speicher des Rechners einladen zu können, bedienen Sie sich des Befehls

```
LOAD"Programmname"
```

wobei für den Programmnamen das gleiche gilt wie beim Abspeichern mit Hilfe des Befehls SAVE. Bei einem LOAD spielt es keine Rolle, in welchem Format das Programm auf der Diskette vorliegt.

Während bei Verwendung des Befehls LOAD ein auf Disk vorhandenes Programm lediglich in den Speicher eingelesen wird, erfolgt durch die Befehle

```
RUN"Programmname"
```

oder auch

```
LOAD"Programmname",R
```

nach dem Einladen ein sofortiger Start des gewünschten Programms.

Beispiel:

Legen Sie die Diskette mit den in Kapitel 14.1 erstellten Testprogrammen in Laufwerk 0. Nach der Eingabe von

```
LOAD"TESTPROG/BAS:0"
```

wird dieses Programm in den Speicher des Rechners eingeladen. Sie können es sich mit LIST wieder auf dem Bildschirm ansehen. Geben Sie aber

```
RUN"TESTPROG/BAS" oder LOAD"TESTPROG/BAS",R
```

ein, erfolgt die sofortige Ausführung des aufgerufenen Programms, d.h. in unserem Fall eine erneute zweifache Abspeicherung des Programms wie im letzten Kapitel beschrieben.

Es sei noch darauf hingewiesen, daß es auch einen DOS-Befehl LOAD gibt, der zum Laden von Programmen in Z80-Maschinensprache dient. Die Programme werden dabei in den Hauptspeicher des Computers entsprechend den im Programm enthaltenen Ladeadressen eingelesen.

Zur Erstellung derartiger Maschinenprogramme ist ein umfangreiches Wissen über Z80-Programmierung Voraussetzung. Ferner wird dazu ein Monitorprogramm oder ein Assembler benötigt, mit deren Hilfe die Eingabe der Z80-Befehle und das Abspeichern der Programme möglich ist.

Das Vereinigen von Programmen

=====

Der Befehl

```
MERGE"Programmname"
```

verbindet ein im Speicher des Computers vorhandenes Programm mit einem weiteren Programm, das sich auf der Diskette befindet.

Dabei sollten die Programme unterschiedliche Zeilennummern haben, da doppelte Nummern automatisch durch die Zeilen des Diskettenprogramms ersetzt werden. Die Anwendung des MERGE-Befehls liegt hauptsächlich darin, ein BASIC-Programm aus einzelnen Modulen aufzubauen und diese nach Fertigstellung zusammenzufügen.

Beispiel:

Schreiben Sie die Programmzeilen

```
20 PRINT"Diskettenmodul"  
30 END
```

und speichern Sie sie ab mit

```
SAVE"PROG1"
```

Löschen Sie nun mit NEW den Speicherinhalt des Rechners, und geben Sie die Zeilen

```
10 PRINT"Speichermodul"  
20 END
```

ein.

Mit Hilfe des nun folgenden Befehls

```
MERGE"PROG1"
```

verbinden Sie beide Module und es ergibt sich bei einem LIST das Gesamtprogramm

```
10 PRINT"Speichermodul"  
20 PRINT"Diskettenmodul"  
30 END
```

wobei die Zeile 10 bestehen bleibt, Zeile 20 des Speichermoduls gegen Zeile 20 des Diskettenmoduls ausgetauscht und Zeile 30 angehängt wird.

AUTO-Start
=====

Oft kann es zweckmäßig sein, daß bestimmte Programme selbständig nach dem Booten der Diskette anlaufen. Hierfür gibt es den DOS-Befehl

```
AUTO Programmname
```

durch den das angesprochene Programm sofort nach dem Booten in den Arbeitsspeicher geladen wird.

Es ist zu beachten, daß der Programmname nicht in Anführungszeichen stehen darf.

Da es sich hier um einen DOS-Befehl handelt, können mit ihm auch nur die in DOS direkt aufrufbaren Hilfsprogramme usw. angerufen werden.

Will man ein BASIC Programm laden, muß zunächst BASIC aufgerufen und dann das BASIC-Programm geladen werden. Der Befehl muß dann wie folgt geschrieben werden:

```
AUTO BASIC LOAD"Programmname"
```

oder wenn das betreffende Programm sofort starten soll

```
AUTO BASIC RUN"Programmname"
```

Besitzt man eine System-Diskette, auf der schon ein AUTO-Befehl steht, kann diese auch ohne Ausführung dieses Befehls gebootet werden, indem Sie beim Betätigen von RESET die NEW LINE-Taste gedrückt halten, bis "Befehlseingabe:" erscheint.

Die Speicherung eines AUTO-Befehls kann durch Eingabe von

AUTO

in der DOS-Befehlsebene ohne nachfolgendes Befehlswort aufgehoben werden.

Beispiel:

Nach dem Einschalten des Gerätes oder dem Drücken der RESET-Taste kommt es bei einem aktivierten AUTO-Start beim Speedmaster z.B. zu folgendem Funktionsablauf:

1

```
SPEEDMASTER (C)1983
UBM 2.0
```

```
Genie DOS 3.0 (c) TCS
26. Januar 1984
Speedmaster
00.00.00 00:00:33
```

2 BASIC RUN "TESTPROG"

3 Genie DOS 3.0
DISK-BASIC
10.August 1983

4 Nun erfolgt der Start Ihres Testprogrammes

In diesem Beispiel wird zunächst G-DOS 3.0 gebootet und angezeigt (1), anschließend kommt es zu Ausführung des AUTO-Start-Befehls (2), der BASIC lädt, anzeigt (3) und zuletzt das Programm TESTPROG aufruft und startet.

Aufruf von BASIC ohne Speicheränderung

=====
Damit ein in Arbeit befindliches BASIC-Programm nicht bei einem RESET durch das selbstätige Laden eines neuen Programms zerstört wird, muß in diesem Fall die Taste NEW LINE festgehalten werden.

Danach ist das Kommando

BASIC *

einzugeben, wodurch das zuletzt im Arbeitsspeicher abgelegte BASIC-Programm erhalten bleibt und auf dem Bildschirm gelistet wird.

Programmierhilfen unter Disk-BASIC =====

Durch das Disk-BASIC auf Ihrer Systemdiskette erhalten Sie einige Optionen, die Ihnen die Programmierarbeit in BASIC erheblich erleichtern können.

Durch einfache Tastenbetätigung lassen sich bestimmte Funktionen aufrufen, für die Sie ansonsten ganze Befehlswoorte eintippen müßten.

Außerdem sind auf den Systemdisketten noch einige Hilfsprogramme gespeichert, die das Entwickeln oder Austesten von Programmen erleichtern. Erklärungen zu den Programmen finden Sie im Anhang dieses Handbuchs.

Innerhalb dieser kurzen, einführenden Anleitung ist es nicht möglich, alle Möglichkeiten, die eine Systemdiskette bietet, ausführlich zu schildern.

Hier nun die wichtigsten Abkürzungen:

Es ist jeweils immer nur eines der folgenden Kommandos erlaubt. Ferner müssen die Abkürzungen immer am Anfang einer Zeile, unmittelbar hinter dem BASIC-Prompt ">" stehen. Eine Zeilennummer oder ein Backspace vorweg sind nicht erlaubt.

Die zuletzt bearbeitete Zeile eines Programms wird nach Eingabe eines

"." (Punkt)

angezeigt, oder steht bei Betätigung der Taste

"," (Komma)

im EDIT-Modus zur Verfügung.

Das Listen der jeweils nächsten Zeile ist durch Drücken der

Pfeil-abwärts-Taste

möglich.

Entsprechend erscheint bei der

Pfeil-aufwärts-Taste

der Inhalt der vorhergehenden Programmzeile.

Die gleichzeitige Betätigung der Tastenkombination

Shift-Pfeil-aufwärts

listet die erste Zeile des Programms.

Dementsprechend wird bei der Kombination

Shift-Pfeil-abwärts

die letzte Programmzeile angezeigt.

Eine komplette Bildschirmseite wird, von der zuletzt bearbeiteten oder angezeigten Zeile an, durch Drücken der Taste

"q"

nach einem automatischen CLS ausgegeben.

Das Kommando AUTO, zur selbständigen Zeilennumerierung des Rechners, kann auch durch die Abkürzung

A Zeilennummer (Automatic)

ersetzt werden.

Den Zeileneditor, den Sie im Normalfall mit dem Kommando EDIT erhalten, können Sie im Disk-BASIC durch den Großbuchstaben

E Zeilennummer (Edit)

aufrufen.

Dementsprechend gilt ein

D Zeilennummer (Delete)

für den DELETE-Befehl, um Zeilen zu löschen.

Anstelle des LIST geben Sie einfach

L Zeilennummer (List)

ein.

Außerdem haben sie die zwei wichtigen Befehle DU und DI durch den Erwerb Ihrer Systemdiskette erhalten.

DI aaaaa,nnnn (Delete and Insert)

löscht die existierende Zeile mit der Nummer aaaaa und schreibt ihren Inhalt als neue Zeile mit der Nummer nnnn ins Programm.

DU aaaaa,nnnn (Duplicate)

dupliziert eine existierende Zeile mit der Nummer aaaaa auf die Zeilennummer nnnn.

Verwendet man anstelle von aaaaa einen "." (Punkt), so wird die zuletzt editierte, bzw. zuletzt im Programm bearbeitete Zeile als alte Zeile behandelt.

Es gibt eine einfache Möglichkeit zu überprüfen, ob ein Programm Fehler enthält, die durch falsche Zeilennummern bedingt sind. Wird der BASIC-Befehl

```
RENUM U
```

als direktes Kommando eingegeben, überprüft der Computer das im Arbeitsspeicher befindliche Programm und meldet

```
Fehlerzeile  
zeilnr/U
```

wenn es die Zeilennummer nicht gibt. Der Text wird dabei nicht verändert.

Beispiel:

In einem Programm steht ein GOTO 10001. Diese Zeile ist jedoch nicht vorhanden.

Nach der Eingabe von RENUM U meldet der Computer, daß die nichtvorhandene Zeilennummer 10001 aufgerufen wird.

```
>RENUM U
```

```
Fehlerzeile  
10001/U Ende
```

```
>LIST  
210 GOSUB 10001  
READY  
>
```

Weiterhin lassen sich mit RENUM die Zeilennummern eines im Hauptspeicher stehenden Programms umändern. Der Befehl kann in vielen Versionen gebraucht werden. Die wichtigsten sind

```
RENUM ,          organisiert das Programm in 10er Schritten  
RENUM n:lnr,x   schreibt Programm ab n:lnr in x-er Schritten
```

Die allgemeine Form dieses Befehls lautet

```
RENUM neue Zeilen-Nr.,Schrittweite,von Zeile,bis Zeile,
```

Ausgabe von Referenzlisten für Konstanten und Variablen:

Mit dem BASIC-Befehl

```
REF*
```

wird eine Referenzliste eines BASIC-Programms mit Zeilennummern und Variablennamen auf dem Bildschirm ausgegeben.

```
REF$
```

aktiviert einen angeschlossenen Drucker und gibt dort die Referenzliste aus.

REFnn

veranlaßt die Ausgabe einer Referenzliste aller Variablen mit dem Namen nn. nn darf maximal aus zwei Buchstaben bestehen und nicht die Zeichen !, \$, # und % enthalten.

Die Ausgabe einer Referenzliste von allen Zahlen mit dem Wert zzzzz erfolgt nach dem Befehl

REFzzzzz

wobei die Zahl zzzzz maximal 5 Ziffern lang sein darf. Eine Anzeige hexadezimaler und oktaler Zahlen ist nicht möglich.

Beispiel:

```
>LIST
10 DIMA#(100)
20 PI=3.1415
30 A%=100
40 B#=12.234567823
50 C$="Testtext"
60 E$=C$
70 FORU=1TO100:A#(U)=B#:NEXTU:FORU=1TO100:NEXTU
80 GOTO 20
READY
>
```

Nach der Eingabe des Kommandos

```
REF*
```

erscheint die Referenzliste

```
1 70/2
20 80
100 10 30 70/2
A 10( # 30/% 70( #
B 40/# 70/#
C 50/$ 60/$
D 60/$
PI 20
U 70/5
READY
>
```

Links stehen die Konstanten, gefolgt von den Variablen, die im Programm vorkommen. Rechts daneben finden Sie die Zeilennummern und nach einem Schrägstrich, die Anzahl der in dieser Zeile jeweils vorkommenden Konstanten oder Variablen. Ferner wird der Typ der Variablen angegeben.

Fehler:

Das Microsoft Level II BASIC ist recht großzügig und toleriert kleine Eingabefehler, wie wenn z.B. Kleinschrift statt Großschrift verwendet oder zuwenig oder zuviel Zwischenraum gelassen wird.

Bei Eingaben im DOS müssen Sie sich dagegen genau an die Vorschrift halten. Alle Buchstaben werden zwar automatisch als Großbuchstaben verarbeitet, aber wenn Zeichen wie Kommata und Doppelpunkte oder auch Zwischenräume nicht exakt eingehalten werden, gibt es Fehlermeldungen.

Z.B.:

```
READY
>COPY 0 1 10.05.83
Syntaxfehler
READY
>
```

Eingabefehler:

Es wurde versucht im BASIC einen DOS-Befehl aufzurufen.

```
Befehlseingabe: COPY 0 1
Programm nicht gefunden
Befehlseingabe:
```

Eingabefehler:

Zwischen COPY und der Laufwerksnummer 0 muß mindestens ein Leerzeichen stehen.

```
Befehlseingabe: COPY 01 10.05.83
```

```
Syntax oder Trennzeichen oder Endzeichen unerlaubt
Befehlseingabe:
```

Eingabefehler:

Zwischen den Nummern der beiden Laufwerke muß mindestens ein Leerzeichen stehen.

```
READY
>CMD"DATUM 2.6.84
schlechte Parameter
READY
>
```

Eingabefehler:

Alle Elemente des Datum müssen zweistellig und jeweils durch einen Punkt getrennt sein.

Disk-BASIC Ergänzungen =====

Zwei der Ergänzungen, die das Disk-BASIC gegenüber dem Level II BASIC von Microsoft bietet, sind der Befehl

DEF FN Funktionsname (Variablenliste) = Ausdruck

und die Funktion

FN Funktionsname (Variablenliste)

Mit Hilfe dieser Befehle wird es Ihnen ermöglicht, eigene Funktionen zu definieren und unter dem zugewiesenen Funktionsnamen den Ausdruck wieder aufzurufen. Die in der Klammer des Funktionsanrufs stehenden, durch Kommata getrennten Variablen werden dann so verarbeitet, wie es in der Definitions-Anweisung festgelegt wurde.

Für den Funktionsnamen gelten die gleichen Bedingungen wie für Variablen. (siehe BASIC Manual). Die Variablenliste beinhaltet alle Variablen, die in der Funktion benötigt werden. Die Funktion darf die Länge einer Zeile nicht überschreiten. Sie sollten Funktionen immer am Anfang eines Programms definieren. Mit den Befehlen DEF FN und FN lassen sich auch Zeichenkettenvariablen verarbeiten.

Beispiel:

```
10 DEF FN PR(X,Y)=0.5*X*Y
20 INPUT A,B
30 E = FN PR(A,B)
40 PRINT E
50 GOTO 20
```

Die Funktion PR(X,Y) wird als Ergebnis der Rechnung $0.5 \cdot X \cdot Y$ definiert. Bei dem späteren Aufruf dieser Funktion in Zeile 30 werden die über INPUT eingegebene Variablen dem Klammerinhalt zugeordnet.

INSTR(n,Zeichenkette 1,Zeichenkette 2)

erlaubt es, innerhalb einer Zeichenkette nach einer anderen Kette zu suchen.

Er gibt als Wert die Anfangsposition der Zeichenkette 2 in der Zeichenkette 1 an. Gesucht wird von der Startposition n beginnend. n muß im Bereich 0 - 255 liegen. Wenn die gesuchte Kette nicht vorhanden oder n größer als die Länge der Zeichenkette 1 ist, wird als Antwort eine 0 ausgegeben.

Beispiel:

```
10 A$="Der Rolf ist ein kleiner Mann"
20 B$="Rolf"
30 C=INSTR(1,A$,B$)
40 PRINT C
```

Der Programmlauf ergibt die Position 5 als Antwort.

Um eine Anzahl von Zeichen innerhalb einer Zeichenkette durch eine Anzahl Zeichen einer anderen Kette zu ersetzen, können Sie sich des Disk-BASIC-Befehls

MID\$(Zeichenkettel,S,L) = Zeichenkette2

bedienen.

Durch diesen Befehl werden von der angegebenen Stelle S an L Zeichen der Zeichenkettel durch die gleiche Anzahl von Zeichen der Zeichenkette2 ersetzt. Um die Stelle S zu ermitteln, muß vom äußersten linken Zeichen der Zeichenkettel an gezählt werden.

Beispiel:

```
10 A$="Der Rolf ist ein kleiner Mann"
20 B$="Rolf": C$="Paul"
30 S=INSTR(1,A$,B$)
40 L=LEN(C$)
50 MID$(A$,S,L)=C$
60 PRINT A$
```

Der INSTR-Befehl sucht die Kette B\$ in A\$ und weist der Variablen S die Startposition von B\$ zu.

Beim Einfügen mit MID\$ ist zu berücksichtigen, daß die Zeichen nur geändert werden und keine Verschiebung des Rests der Kette stattfindet.

Somit erhalten Sie beim Programmlauf einen Austausch der Namens "Rolf" durch "Paul".

Der Befehl

LINEINPUT"Text";Zeichenkettenvariable

ist eine Erweiterung der INPUT-Funktion. Mit seiner Hilfe können alle Zeichen der Tastatur, also auch Kommata und Anführungsstriche eingegeben werden.

Im Gegensatz zur normalen INPUT-Funktion wird kein Fragezeichen angezeigt, es kann jedoch im vorhergehenden Text untergebracht werden.

Beim Drücken der NEW LINE-Taste wird der Zeichenkettenvariablen der gesamte, zuvor eingegebene, Text zugewiesen.

Beispiel:

```
10 PRINT"Geben Sie einen Text mit Kommata und"
20 LINEINPUT"Anführungszeichen ein!";T$
30 PRINT"Der Text lautete:"
40 PRINT T$
```

Weitere Informationen zum Befehl LINEINPUT sind noch im Abschnitt 19 über Dateien nachzulesen.

Die im Level II BASIC enthaltene USR-Funktion wurde im Disk-BASIC erweitert. Mittels des Befehls

DEFUSRn=Startadresse

ist es möglich bis zu 10 Maschinenroutinen zu definieren. Sie teilen der durch n (n= 0...9) ausgewählten USR-Funktion eine Startadresse zu, unter der sie aufgerufen werden kann.

Über den Befehl

Variable=USRn(Argument)

wird das definierte Maschinenprogramm aufgerufen (siehe auch BASIC Manual). Als Variable kann irgendeine unbenutzte eingesetzt werden (Dummy). Falls die Routine einen Wert ergibt, wird er der Variablen zugewiesen.

Beispiel:

```
10 DEFUSR1=&H01C9:REM Adresse der CLS-Routine
20 DEFUSR2=&H0049:REM Tastaturabfrage
30 DEFUSR3=&H022C:REM Stern blinken lassen
40 A=USR1(0):REM löscht Bildschirm
50 PRINT"Nächste Ausführung erst nach Tastendruck!"
60 A=USR2(0):REM Warteschleife bis zum Tastendruck
70 A=USR1(0):REM löscht Bildschirm
80 PRINT"Nach Tastendruck erscheint ein Stern ----->"
90 A=USR2(0):REM Warteschleife bis zum Tastendruck
100 PRINT"Stern kann per Tastendruck geschaltet werden."
110 PRINT"Dies erfolgt auch beim Drücken der BREAK-Taste!"
120 A=USR3(0):A=USR2(0):GOTO90
```

Dieses Programm definiert 3 Maschinenroutinen und löscht als erstes den Bildschirm mit dem Aufruf der Routine 1 in Zeile 40. Anschließend erwartet die Routine 2 in Zeile 60, daß irgendeine Taste gedrückt wird.

Nachdem ein Tastendruck erfolgt ist, bewirkt der erste Aufruf der Routine 3 in Zeile 120 das Einschalten eines Sternchens "*" in der oberen rechten Ecke des Bildschirms.

Nach Ausführung der Routine 2 in Zeile 120, kommt es zum zweiten Aufruf von Routine 3, womit das Sternchen wieder gelöscht wird.

Da auch die BREAK-Taste in dieser Abfrage mit berücksichtigt wird, läßt sich das Programm nur mit RESET abbrechen.

Dateien (Files)

=====

Unter einer Datei, englisch auch File genannt, versteht man eine nach bestimmten Gesichtspunkten geordnete Datenmenge, die so gespeichert ist, daß durch ein Programm auf sie zugegriffen werden kann.

Damit dies möglich ist, muß jede Datei einen Namen haben, durch den sie eindeutig und unverwechselbar gekennzeichnet ist. Die Regeln für die Namensgebung entsprechen den Regeln, die schon für die Benennung von Programmen beschrieben wurden, denn intern ist ein Programm für den Computer ebenfalls eine Datei.

Man muß wissen, daß die Be- und Verarbeitung der Daten einer Datei immer nur im Arbeitsspeicher des Computers erfolgen kann. Die Datei muß also jeweils in den Arbeitsspeicher geladen, dort bearbeitet und dann wieder auf die Diskette zurückgeschrieben werden.

Im wesentlichen unterscheidet man sequentielle Dateien und Dateien mit direktem Zugriff.

Die sequentielle Datenspeicherung auf einer Diskette arbeitet ähnlich wie die Speicherung auf einer Kassette. Die einzelnen Daten einer Datei, die jeweils einer Variablen zuzuordnen sind, werden in einer Folge hintereinander abgespeichert und müssen in der gleichen Folge auch wieder gelesen werden. Benötigt man die Daten eines bestimmten Blocks, müssen auch immer alle davor liegenden Datenblöcke gelesen und im Arbeitsspeicher geprüft werden. Die Blöcke können unterschiedlich lang sein. Nach jedem Block muß ein Trennzeichen gesetzt werden, an dem der Computer das Ende erkennt.

Demgegenüber werden bei Dateien mit direktem Zugriff die Daten in jeweils durch Indexziffern gekennzeichneten Sätzen gespeichert. Man kann auf jeden dieser Sätze direkt zugreifen und immer nur den benötigten Satz in einen Puffer des Arbeitsspeichers laden, um ihn dort weiter zu verarbeiten. Dieser Vorteil muß allerdings mit einigem Aufwand beim Programmieren erkauft werden.

Der Name einer Datei darf sich aus 1 bis 8 alphanumerischen Zeichen zusammensetzen, wobei das erste Zeichen immer ein Buchstabe sein muß.

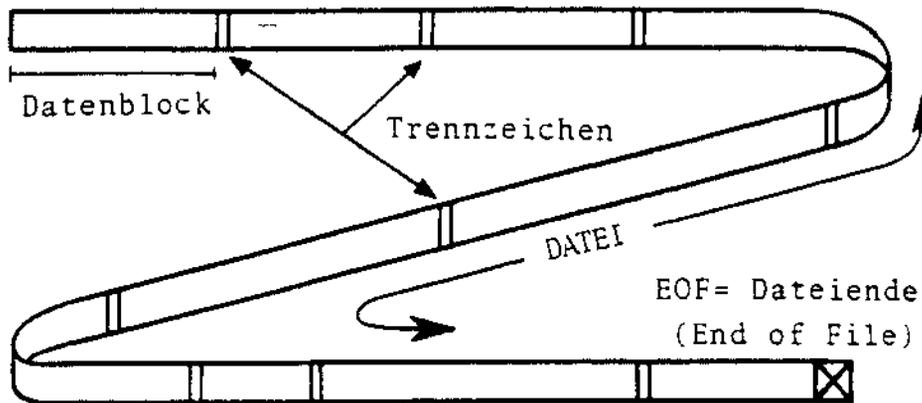
Dem Namen kann noch eine Ergänzung, aus 1 bis 3 alphanumerischen Zeichen bestehend und mit einem Schrägstrich beginnend angefügt werden. Wenn diese Ergänzung nicht angegeben wird, verwendet das System an ihrer Stelle Leerzeichen.

Gängige Dateitypbezeichnungen sind z.B.:

| | |
|------|---|
| /BAS | BASIC-Programm |
| /CMD | Maschinenprogramm |
| /DAT | Datei zu einem Programm gleichen Namens |
| /TXT | Textdatei |
| /JOB | Arbeitsdatei für Programmverkettung |
| /SYS | zum DOS gehörendes Systemprogramm |
| /OBJ | Datei in Maschinencode |

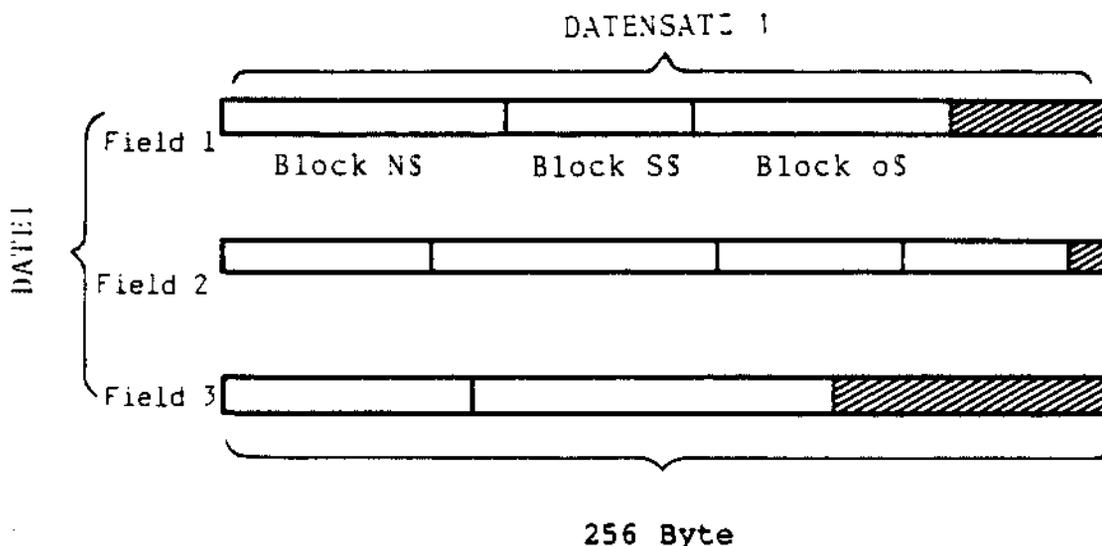
Schema einer sequentiellen Datei

Die einzelnen Datenblöcke, immer dem Inhalt einer Variablen entsprechend, werden logisch hintereinander gespeichert. Jedes Ende eines Datenblocks muß durch ein Trennzeichen gekennzeichnet sein. An das Ende der Datei setzt der Computer ein EOF. EOF heißt "End of File" bzw. Dateiende. Die Datei kann nur als Ganzes im Arbeitsspeicher des Rechners verarbeitet werden.



Schema einer Datei mit direktem Zugriff

Teile der Gesamtdatei können als Datensätze, auch Datenfelder (Fields) genannt, vor der Abspeicherung definiert werden. In ihnen wird den einzelnen Datenblöcken die Länge zugewiesen. Wenn das Feld nicht voll ausgenutzt wird, geht der entsprechende Speicherplatz verloren. Jedes Feld kann einzeln zur Bearbeitung in den Hauptspeicher geholt werden.



Öffnen und Schließen von Dateien

=====

Alle Anweisungen für die Dateiverwaltung auf der Diskette sind nur von BASIC aus möglich. Meist schreibt man sie in Programmzeilen.

Um den entsprechenden Speicherraum im Arbeitsspeicher für die Übernahme der Daten einer extern abgespeicherten Datei vorzubereiten, lautet zu Beginn jeder Dateibearbeitung der BASIC-Befehl

--
OPEN "Modus",Puffernummer,"Dateiname"

Mit diesem Befehl kann der Zugriff auf Disk-Dateien erfolgen. Er ist stets der erste in einer Kette von zusammengehörigen Befehlen, die zum Dateizugriff notwendig sind.

Als "Modus" muß dabei jeweils einer der folgenden Buchstaben in Anführungsstrichen eingesetzt werden.

- O (OUTPUT) für das Abspeichern von Daten auf eine sequentielle Datei
- I (INPUT) für das Lesen von Daten von einer sequentiellen Datei
- E (EXTEND) für das Anfügen weiterer Daten an eine sequentielle Datei
- R (RANDOM) für das Speichern auf oder das Lesen von einer Datei mit direktem Zugriff
- D Dateiinterner RANDOM-Zugriff
(Die Datei muß schon existieren und wird dabei nicht erweitert)

Als "Puffernummer" kann eine ganze Zahl zwischen 1 und 15 gewählt werden. Sie dient zur Kurzkennzeichnung einer Datei bei den eigentlichen Schreib- und Lesebefehlen.

Normalerweise können 3 Dateien gleichzeitig geöffnet sein. Reicht dies nicht aus, kann beim Aufruf von BASIC noch die Anzahl der maximal offenzuhaltenden Dateien angefügt werden.

BASIC 8

würde also bedeuten, daß Sie mit bis zu 8 offenen Dateien arbeiten können. Nur die auf diese Art bereitgestellte Anzahl der Puffer kann später bearbeitet werden.

Der "Dateiname" ist der Name der Datei, auf die zugegriffen werden soll. Es kann sich dabei auch um eine Variable handeln.

Wurde eine Datei durch OPEN geöffnet, muß sie am Ende durch

CLOSE Puffernummer

wieder geschlossen werden. Es lassen sich auch mehrere Dateien zugleich schließen, wenn deren Nummern durch Kommata getrennt in dem Befehl stehen. CLOSE ohne Parameter schließt alle offenen Dateien. Schließen Sie immer alle Dateien, bevor Sie weiterprogrammieren. Es kann Ihnen sonst passieren, daß bei einer Datenausgabe das aktuelle Datenende nicht im Inhaltsverzeichnis vermerkt wird, was zur Unlesbarkeit der benutzten Datei führt.

Sequentielle Dateien

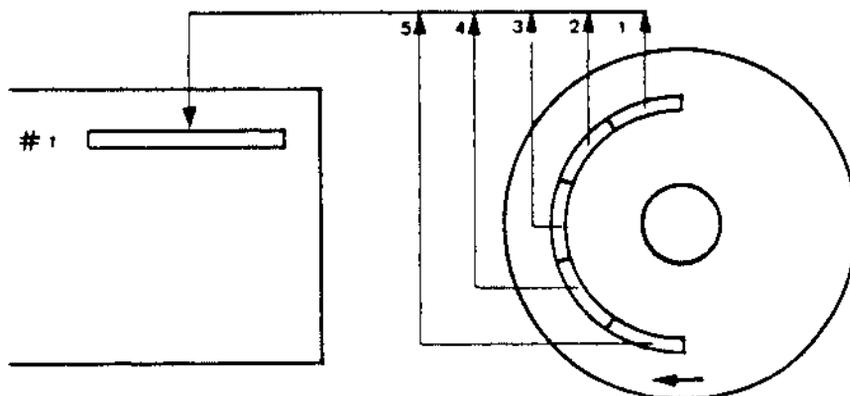
=====

Die sequentielle Ein- und Ausgabe ist die einfachste Art der Datenspeicherung. Die Anweisungen sind denen der Kassettenspeicherung sehr ähnlich.

Nehmen wir einmal an, daß sich auf der Diskette eine sequentielle Datei mit dem Namen ADRESSEN befindet. Durch den Befehl

```
OPEN"1",1,"ADRESSEN"
```

wird dieser Datei unter der Puffernummer 1 ein Bereich des Arbeitsspeichers zugewiesen. Dies bedeutet, daß die Datei ADRESSEN in den Befehlen jetzt nur noch mit #1 angesprochen werden muß. Nach dem Lesebefehl wird ein Sektor der Datei in den Arbeitsspeicher geladen und dort bearbeitet, indem die auf der Diskette abgespeicherten Informationen den entsprechenden Variablen zugewiesen werden. Bei einer sequentiellen Datei wird ein Sektor nach dem anderen gelesen und verarbeitet.



Die Befehle der sequentiellen Ein- und Ausgabe lauten

```
PRINT#           INPUT#  
PRINT# USING    LINE INPUT#  
                EOF(x)
```

Zur Datenspeicherung als sequentielle Datei lautet der Befehl

```
PRINT#Puffernummer,Variable;Variable;Variable
```

Schreibt man Daten auf die sequentielle Datei 'Puffernummer' (vergleichbar mit dem normalen PRINT-Befehl), muß sie zuvor durch ein OPEN geöffnet worden sein.

Statt einer Variablen kann auch ein Rechenausdruck oder eine in Anführungsstrichen stehende Zeichenkette eingesetzt werden. Die Abspeicherung erfolgt in die Datei, deren Namen im OPEN-Befehl zusammen mit der Puffernummer des PRINT#-Befehls gebraucht wurde.

Für die Trennung der Variablen im PRINT#-Befehl wird nach Zahlen-Variablen das Semikolon verwendet. Wenn in der gleichen Zeile nach einer Zeichenketten-Variablen noch weitere Variablen folgen, muß die Kombination ";","; also Semikolon-Anführungsstriche-Komma-Anführungsstriche-Semikolon zur Trennung verwendet werden.

Die gleiche Wirkung hat ein CHR\$(13), das am Ende eines PRINT#-Befehls in einer neuen Zeile erzeugt wird.

Ein Programm zur Abspeicherung der 3 Variablen A, B und Z\$ in die Datei mit dem Namen Versuch kann also in den folgenden Versionen geschrieben werden:

```
100 OPEN"O",1,"VERSUCH          100 OPEN"O",1,"VERSUCH
110 PRINT#1,A;Z$;"",";B        110 PRINT#1,A;Z$
120 CLOSE                       120 PRINT#1,B
                                130 CLOSE
```

In der ersten Version steht die besondere Trennzeichenkombination nach der Zeichenkette, im zweiten Fall steht nach dem String ein neuer PRINT#-Befehl, wodurch auch eine Trennung erfolgt.

Diese etwas umständliche Vorschrift wird verständlich, wenn man bedenkt, daß für das Wiedereinlesen einer sequentiellen Datenspeicherung ein INPUT#-Befehl verwendet wird, für den nur ein Komma oder NEW LINE als Trennzeichen gilt. Dies wiederum bedeutet, daß auch ein innerhalb einer Variablen stehendes Komma als Trennzeichen gewertet wird. Alles was nach dem Komma kommt wird beim Lesen dann schon der nächsten Variablen zugeordnet.

Alle Zeichen, also auch Zahlen, werden bei sequentiellen Dateien im ASCII-Format gespeichert. Nicht druckbare Zeichen müssen mit Hilfe der CHR\$(x) Funktion abgespeichert werden, wobei danach auch wieder das besondere Trennzeichen verwendet werden muß, wenn in der gleichen Zeile noch weitere Variablen folgen.

Mit Hilfe von PRINT USING lassen sich sequentielle Dateien auch in einem bestimmten Format schreiben. Diese Formate sind dabei die gleichen, wie in der BASIC-Anweisung PRINT USING.

Die Eingabe ist dabei wie folgt zusammengesetzt:

```
PRINT#Puffernummer,USING"Format";Variablen oder Ausdrücke
```

Das Programm

```
10 A$="Meier":B$="Peter":C=14.1478
20 OPEN"O",1,"TESTDAT"
30 PRINT#1,A$;"",";B$:REM Verwendung des bes. Trennzeichens
40 PRINT#1,USING"***#.***";C:REM formatierte Ausgabe
50 CLOSE1
60 END
```

schreibt die Stringvariablen A\$ und B\$, sowie die numerische Variable C im angegebenen Format ***#.*** als Zeichenkette **14.148 in die Datei mit dem Namen TESTDAT.

Zum Wiedereinlesen der durch PRINT# sequentiell gespeicherten Daten dient der Befehl

INPUT#Puffernummer,Variable,Variable,Variable

Durch diesen Befehl wird jeweils ein Datenblock, mit der Länge bis zum nächsten Trennzeichen, einer Variablen zugewiesen. Die Zuweisung erfolgt in der Reihenfolge, wie die Variablennamen im INPUT#-Befehl stehen, wobei natürlich auch immer der Variablentyp übereinstimmen muß.

Der INPUT#-Befehl richtet sich grundsätzlich immer nach den Trennzeichen, die bei der Speicherung aufgezeichnet wurden und erkennt dabei das Komma, einen neuen PRINT#-Befehl und nach Zahlen-Variablen auch ein Semikolon an. Waren bei der Aufzeichnung zwei Variablen nicht durch richtige Trennzeichen getrennt, werden die aufgezeichneten Daten nur der ersten Variablen zugewiesen.

Vor dem Einlesen der Daten über INPUT# muß der Puffer mit einem sequentiellen OPEN-Befehl geöffnet worden sein.

Im Gegensatz zur Kassettenspeicherung braucht der PRINT#-Befehl nicht genau mit dem INPUT#-Befehl übereinzustimmen. Die Daten, die mit einem PRINT#-Befehl gespeichert wurden, können z.B. auch mit mehreren INPUT#-Befehlen eingelesen werden.

Beispiele:

Das nun folgende Beispiel zeigt, daß sich ein völlig falsches Bild ergibt, wenn in den Zeichenketten Kommata verwendet werden.

```
40 CLEAR100
50 A$="1. Block;1. Teil;2. Teil"
60 B$="2. Block;1. Teil;2. Teil"
70 C$="3. Block"
80 D=12345
100 OPEN"O",1,"VERSUCH"
120 PRINT#1,A$;" ";D;B$
130 PRINT#1,C$
140 CLOSE:CLEAR
200 OPEN"I",1,"VERSUCH"
220 INPUT#1,A$,D,B$,C$:CLOSE
310 PRINT"A$=";A$
320 PRINT"B$=";B$
330 PRINT"C$=";C$
340 PRINT"D =" ;D
```

Wird dieses Programm mit RUN gestartet, erscheint nach kurzer Zeit auf dem Bildschirm die richtige Anzeige

```
A$=1. Block;1. Teil;2. Teil
B$=2. Block;1. Teil;2. Teil
C$=3. Block
D = 12345
```

Werden die Zeilen 50 und 60 des Programms gegen

```
50 A$="1. Block,1. Teil,2. Teil"  
60 B$="2. Block,1. Teil,2. Teil"
```

ausgetauscht, (Kommata innerhalb der Zeichenketten A\$ und B\$),
so erscheint nach Programmstart die falsche Ausgabe

```
A$=1. Block  
B$=1. Teil  
C$=2. Teil  
D = 1
```

Im folgenden Beispiel soll ein eindimensionales Feld in einer sequentiellen Datei gespeichert werden. Es muß dazu eine Schleife verwendet werden. Dies gilt auch für das Einlesen eines Feldes von einer sequentiellen Datei. Das Beispiel zeigt auch, daß man ohne das besondere Trennzeichen auskommt, wenn man nach jeder Zeichenketten-Variablen mit einem neuen PRINT#-Befehl beginnt.

```
10 CLEAR 500  
50 A$(1)="Maier": O$(1)="Hamburg"  
60 A$(2)="Schulz": O$(2)="Duisburg"  
70 A$(3)="Schmidt": O$(3)="Essen"  
80 P(1)=2000: P(2)=4100: P(3)=4300  
100 OPEN"O",1,"Versuch"  
110 FOR K=1 TO 3  
120 PRINT#1,A$(K);",",P(K);O$(K)  
140 NEXT:CLOSE  
200 OPEN"I",1,"VERSUCH"  
210 FOR K = 1 TO 3  
220 INPUT#1,A$(K),P(K),O$(K)  
230 NEXT:CLOSE  
300 FOR K = 1 TO 3  
310 PRINT P(K);O$(K),A$(K)  
320 NEXT
```

Der Programmlauf ergibt folgende Ausgabe auf dem Bildschirm:

```
2000 Hamburg   Maier  
4100 Duisburg  Schulz  
4300 Essen     Schmidt
```

Beginnt man mit einem neuen PRINT#-Befehl anstelle des besonderen Trennzeichens in Zeile 120, so erhält man nach Eingabe der Zeilen

```
120 PRINT#1,A$(K)  
130 PRINT#1,P(K);O$(K)
```

das gleiche Ergebnis.

Soll eine Erkennung der Kommata und Anführungszeichen als Trennzeichen vermieden werden , d.h. jede Zeichenkette von Diskette einlesbar sein, so muß der Befehl

```
LINE INPUT#Puffernummer,Zeichenketten-Variable
```

verwendet werden.

Alles was Sie bis zum Drücken der NEW LINE-Taste eingegeben, wird der im Befehl stehenden Zeichenketten-Variablen zugewiesen. Selbstverständlich darf in diesem Befehl nur ein Variablenname stehen.

Beispiel:

```
40 CLEAR100
50 A$="1. Block, 2. Block"
60 B$="3. Block, 4. Block"
70 C$="5. Block"
80 D=12345
100 OPEN"O",1,"VERSUCH"
120 PRINT#1,A$;" ";D;B$
130 PRINT#1,C$
140 CLOSE:CLEAR
200 OPEN"I",1,"VERSUCH"
220 LINE INPUT#1,A$
230 LINE INPUT#1,B$:CLOSE
310 PRINT"A$=";A$;PRINT"B$=";B$
320 PRINT"C$=";C$;PRINT"D =";D
```

Dieses Programm ergibt nach dem Start das falsche Ergebnis:

```
A$=1. Block, 2. Block, 12345 3. Block, 4. Block
B$=5. Block
C$=
D = 0
```

Tauschen Sie nun die Zeilen 120 - 150 und 220 - 240 aus.

```
120 PRINT#1,A$
130 PRINT#1,B$
140 PRINT#1,D;C$
150 CLOSE:CLEAR
220 LINE INPUT#1,A$
230 LINE INPUT#1,B$
240 INPUT#1,D,C$:CLOSE
```

Der Programmlauf ergibt nun die richtige Lösung mit

```
A$=1. Block, 2. Block
B$=3. Block, 4. Block
C$=5. Block
D = 12345
```

Für das Abspeichern und Einlesen von Daten in eine Datei verwendet man auch gern Feld-Variablen. Man kann die Datei zunächst mit einer FOR...NEXT Schleife aufbauen, dann mit einer weiteren Schleife speichern und auch wieder mit einer Schleife einlesen. Da derartige Dateien unterschiedliche Längen haben können, gibt es eine Funktion, die das Ende einer Datei meldet.

Diese Funktion (END OF FILE) lautet

EOF(Puffernummer)

Der Wert dieser Funktion kann einer Zahlenvariablen zugewiesen werden, läßt sich aber auch direkt in einem Entscheidungsbefehl innerhalb einer Schleife verwenden.

z.B. X = EOF(1) oder IF EOF(1) GOTO

Der Wert von X bzw. EOF bleibt 0, solange noch Daten gelesen werden können und wird -1, wenn das Dateiende erreicht ist.

Beispiel:

```
10 CLEAR100
20 OPEN"O",1,"TESTDAT"
30 FOR I=1TO7:A$(I)="TESTTEXT"+STR$(I)
40 PRINT#1,A$(I):NEXT:CLOSE
50 OPEN"I",1,"TESTDAT"
60 FOR I=1TO20:INPUT#1,A$(I)
70 PRINTA$(I)
80 IF EOF(1) THEN100
90 NEXT:CLOSE:END
100 PRINT I;"Datensätze eingelesen"
110 CLOSE
```

Dateien mit direktem Zugriff (RANDOM FILES)

=====

Dateien mit direktem Zugriff sind zwar etwas aufwendiger in der Programmierung, bieten jedoch einige Vorteile gegenüber sequentiellen Dateien. Der Hauptvorteil besteht darin, daß die Gesamtdatei noch in einzelne Felder eingeteilt werden kann, wobei jedes Feld allein aufrufbar ist und von der Diskette in den Arbeitsspeicher geladen und dort bearbeitet werden kann. Außerdem ist es möglich wahlweise zu schreiben oder zu lesen, ohne daß die Datei dazu erneut geöffnet werden muß.

Auch für Dateien mit direktem Zugriff muß zunächst durch den OPEN-Befehl Speicherraum im Arbeitsspeicher des Rechners reserviert werden. Die Befehlsfolge dazu lautet wie folgt:

```
OPEN"R",Puffernummer,"Dateiname"
```

Das R steht für RANDOM = direkter Zugriff. Ferner wird wie bei einer sequentiellen Datei eine Puffernummer und ein Dateiname gewählt, um eine bestimmte Datei anzusprechen.

Als nächstes ist die Aufteilung des Satzes notwendig. Ein Satz hat bei diesem Dateityp immer eine Länge von 256 Bytes. Die Länge der Datei ist lediglich durch das Speichermedium begrenzt.

Jeder Satz einer Datei kann in mehrere Pufferfelder aufgeteilt werden, die jeweils einer Text-Variablen zuzuordnen sind. Im Gegensatz zur sequentiellen Datei muß die Länge der Felder vorher festgelegt werden (FIELDING). Beim Abspeichern werden die Daten unabhängig von ihrer wirklichen Länge in den reservierten Speicherraum geschrieben. Wenn weniger Daten als reservierter Speicherraum vorhanden sind, bleibt der Rest frei. Versucht man mehr Daten zu speichern, wird der Überhang abgeschnitten.

Die Feldeinteilung erfolgt mit der Anweisung

```
FIELD Puffernummer, Blocklänge AS Variable, .....
```

Mit dieser Anweisung wird der Aufbau des Pufferspeichers im Arbeitsspeicher bestimmt. Als Variablennamen dürfen ausschließlich Namen für String-Variablen verwendet werden. Innerhalb eines Feldes können mehrere Variablen definiert werden. Die FIELD-Anweisung sollte direkt hinter dem OPEN-Befehl verwendet werden, um Fehler zu vermeiden.

Sollen mehrere Feldaufteilungen innerhalb eines Programms benutzt werden, empfiehlt es sich, die FIELD-Anweisung in Unterprogrammen abzulegen, welche dann bei Bedarf vom Hauptprogramm aufgerufen werden.

Zwischen 'Blocklänge', 'AS' und 'Variable' sollte jeweils ein Leerzeichen stehen. (Vermeiden Sie Variablennamen, die mit dem Buchstaben 'C' beginnen, da ein fehlendes Leerzeichen in diesem Fall die BASIC-Funktion 'ASC' aufrufen würde.

Es ist möglich die Blocklänge auch als numerischen Ausdruck anzugeben. Diese muß dann aber in jedem Fall in Klammern gesetzt werden, da sonst es sonst zu Fehlinterpretationen des Rechners kommen kann.

In dem Programmteil

```
310 FORI=0TO4
320 FIELD 1,(200-I*50) AS N$(I), (I*50) AS V$(I)
330 NEXTI
```

ist die Variable N\$(0) gleich der Variablen V\$(4). Es können sich also demnach auch mehrere Variablen innerhalb eines Pufferfeldes überlappen.

Da bei RANDOM-Dateien alle Variablen als Zeichenketten gespeichert und Zahlen als Binärzahlen dargestellt werden, benötigen sie nur wenig Speicherraum.

| | |
|------------------------------------|--------|
| eine Ganzzahl | 2 Byte |
| eine Gleitkommazahl | 4 Byte |
| eine Zahl mit erhöhter Genauigkeit | 8 Byte |

Sollen Zahlen in der Datei gespeichert werden, müssen die entsprechenden Blöcke in der FIELD-Anweisung nach dieser Regel in der Länge festgelegt werden.

Wenn der Puffer durch die FIELD-Anweisung vorbereitet ist, können die Daten in die Blöcke des Feldes geschrieben werden. Dabei gibt es verschiedene Befehle für Zeichenketten und Zahlen. Je nachdem, ob eine Zeichenkette links- oder rechtsbündig in dem für den Block reservierten Speicherraum untergebracht werden soll, gelten die Befehls Worte

| | |
|------|--|
| LSET | für linksbündig, also mit Leerstellen rechts |
| RSET | für rechtsbündig, also mit Leerstellen links |

Mit diesen beiden Befehlen weist man einer Puffer-Variablen den Wert einer im Programm stehenden Variablen zu. Es sind die einzigen Befehle, mit denen die Variablen in einem "FIELD" verändert werden können.

Wenn z.B. A\$ als "HAUS" definiert ist und für die Variable X\$ ein Block mit 10 Zeichen Länge reserviert wurde, ergibt eine Speicherung

| | | |
|----------------|-----------------|-----------|
| LSET X\$ = A\$ | den Blockinhalt | HAUS..... |
| RSET X\$ = A\$ | den Blockinhalt |HAUS |

Statt der hier geschriebenen Punkte werden Leerstellen gesetzt.

Ist ein Feld im Pufferspeicher aufgebaut, genügt ein Befehl für die Abspeicherung auf Diskette. Er lautet

PUT Puffernummer, Satznummer

Die Satznummer positioniert den Schreib/Lesekopf des Diskettenlaufwerks. Wenn Sie auf diese Nummer verzichten, werden die Daten beim ersten PUT-Befehl automatisch in den ersten Satz, beim zweiten Befehl in Satz 2 usw. geschrieben, da sich die Positionierung des Schreib/Lesekopfes schon an dieser Stelle befindet.

Der Befehl zum Einlesen eines ganzen Feldes (Datensatzes) in den Puffer ist dem Schreibbefehl sehr ähnlich und lautet

GET Puffernummer, Satznummer

Er bringt jeweils das ganze angesprochene Feld in den Puffer, von wo es weiterverarbeitet werden kann. Die Daten stehen im Puffer mit den Namen der Puffer-Variablen, mit denen direkt weitergearbeitet, oder deren Inhalt wieder anderen Variablen zugewiesen werden kann.

Beispiel:

Im folgenden Beispiel ist zu erkennen, daß die OPEN- und FIELD-Befehle für das Abspeichern und Einlesen gleich sind. Man kann sie also auch über ein Unterprogramm aufrufen. In den Zeilen 130 - 150 erfolgt die links- und rechtsbündige Zuweisung an die im FIELD-Befehl stehenden Puffer-Variablen. Der Programmlauf läßt erkennen, daß in der betreffenden Variablen auch immer alle Leerzeichen bis zur vollen Länge der im FIELD-Befehl definierten Kette enthalten sind. Der CLEAR-Befehl in Zeile 190 löscht die Variablenspeicher vor dem Einlesen.

```
50 A$="Heinz Jochen Wirths"
60 B$="Im Blumengarten 43"
70 C$="5300 Bonn 7"
100 OPEN"R",1,"VERSUCH"
110 FIELD#1,25 AS N$,25 AS S$,25 AS O$
130 LSET N$ = A$
140 RSET S$ = B$
150 LSET O$ = C$
180 PUT 1,1
190 CLOSE:CLEAR
200 OPEN"R",1,"VERSUCH"
210 FIELD#1,25 AS N$,25 AS S$,25 AS O$
220 GET 1,1
230 CLOSE
240 PRINT N$:PRINT S$:PRINT O$
```

Der Programmlauf ergibt die Bildschirmausgabe

```
Heinz Jochen Wirths
      Im Blumengarten 43
5300 Bonn 7
```

wobei Name und Ort linksbündig, die Straße jedoch rechtsbündig angezeigt werden. Die Blocklänge beträgt per Definition in Zeile 110 bzw. 210 jeweils 25.

Das folgende Beispiel zeigt, wie die einzelnen Felder mit Hilfe von normalen INPUT-Befehlen innerhalb der Speicherschleife gefüllt und dann abgespeichert werden. Nach Beantwortung der Frage nach der Datei wird nur der Datensatz des jeweilig gewählten Feldes gelesen und ausgedruckt.

```

10 CLEAR 1000
20 OPEN"R",1,"VERSUCH"
30 GOSUB 1000
40 FOR K=1 TO 3
50 INPUT A$,B$,C$
60 LSET N$=A$
70 LSET S$=B$
80 LSET O$=C$
90 PUT 1,K
100 NEXT K
110 CLOSE
120 PRINT"Welche Adresse soll wieder"
130 INPUT"aufgerufen werden (1/2/3)";P
140 OPEN"R",1,"VERSUCH"
150 GOSUB1000
160 GET 1,P
170 CLOSE
180 PRINT N$: PRINT S$: PRINT O$
190 END
1000 FIELD#1,20 AS N$,20 AS S$,25 AS O$:RETURN

```

Der Programmablauf bringt z.B. folgende Bildschirmausgabe

```

? Ulrich Holtz
?? Kirschallee 7
?? 1000 Berlin 63
? Hans-Dieter Lohr
?? Sigismundstraße 20
?? 5330 Königswinter 1
? Ellen Kunert
?? Auf dem Acker 3
?? 5300 Bonn 3
Welche Adresse soll wieder
aufgerufen werden (1/2/3)? 2
Hans-Dieter Lohr
Sigismundstraße 20
5330 Königswinter 1

```

Zum Abspeichern von Zahlen in Dateien mit direktem Zugriff gibt es 3 Funktionen, die sie in Zeichenketten umwandeln.

| | |
|--------------------------------|---|
| MKI\$ (Ganzzahl-Variable) | übergibt eine Ganzzahl an eine 2-stellige String-Variable |
| MKS\$ (Fließkomma-Variable) | übergibt eine Fließkommazahl an eine 4-stellige String-Variable |
| MKD\$ (Doppeltgenaue-Variable) | übergibt eine Zahl doppelter Genauigkeit an eine 8-stellige String-Variable |

Diese Funktionen werden in den LSET- oder RSET-Befehlen z.B. wie folgt verwendet:

```
LSET X$ = MKI$(A%)
RSET Y$ = MKS$(B)
RSET Z$ = MKD$(C#)
```

Wenn die Feldlänge vorher entsprechend dem abzuspeichernden Variablentyp richtig festgelegt wurde, spielt es keine Rolle, ob LSET oder RSET verwendet wird.

Zeichenketten werden vom Puffer übernommen, indem man die Puffer-Variable, unter der die Kette in der FIELD-Anweisung steht, verwendet.

Für die Übernahme von Zahlen gibt es wieder besondere Funktionen, mit denen die als Zeichenketten gespeicherten Werte in entsprechende Zahlen zurückverwandelt werden können.

```
CVI(2-Byte-Kette)   wandelt in Ganzzahl
CVS(4-Byte-Kette)   wandelt in Fließkommazahl
CVD(8-Byte-Kette)   wandelt in Zahl doppelter Genauigkeit
```

Die entsprechenden Befehle könnten zum Beispiel lauten:

```
A% = CVI(X$)   oder   B = CVS(Y$)   oder   C# = CVD(Z$)
```

Beispiel:

```
50 A = 1234.5
60 B# = 3.141592625
70 C% = 2345
100 OPEN"R",1,"VERSUCH"
110 GOSUB 1000
130 LSETX$=MKS$(A)
140 LSETY$=MKD$(B#)
150 LSETZ$=MKI$(C%)
180 PUT1,1
190 CLOSE: CLEAR
200 OPEN"R",1,"VERSUCH"
210 GOSUB 1000
220 GET1,1
230 PI# = CVD(Y$)
240 PRINT"PI =";PI#
250 IN = CVI(Z$)
260 N = CVS(X$)
270 PRINT"IN =";IN:PRINT" N =";N
280 END
1000 FIELD#1,4 AS X$,8 AS Y$,2 AS Z$
1010 RETURN
```

In den Zeilen 130 bis 150 dieses Programms werden die Zahlen in Zeichenketten umgewandelt und in Zeile 180 auf der Diskette abgelegt. Nach dem erneuten Einlesen der Variablen in 220 erfolgt wiederum die Umwandlung der Zeichenketten in die Werte der entsprechenden Zahlen (Zeilen 230, 250, 260).

Das FIELD-Kommando im Unterprogramm ab 1000 wird sowohl zum Ausgeben, als auch zum Einladen der Datei verwendet. Der Programmstart bringt das Ergebnis

```
PI = 3.141592625
IN = 2345
N = 1234.5
```

Besonders nützlich ist eine RANDOM-Datei, wenn eine Reihe von Daten jeweils in Verbindung mit einem kennzeichnenden Begriff gespeichert werden sollen. Dies ist zum Beispiel bei Namen-Dateien, die außer dem Namen noch die Adresse, die Telefonnummer usw. enthalten sollen, der Fall. Das Einlesen und Abspeichern erfolgt dann z. B. in einer FOR...NEXT-Schleife. Sie können davon später ein beliebiges Feld anrufen, bearbeiten und wieder abspeichern.

Es ist hilfreich, wenn bekannt ist, wieviel Datensätze der betreffenden Datei schon besetzt sind. Dazu gibt es eine besondere Funktion, die die letzte, schon verwendete Satznummer ausgibt (LENGTH OF FILE). Sie lautet

```
LOF(Puffernummer)
```

Diese Funktion benutzt man zum Beispiel nach einem PRINT-Befehl, aber auch in einer FOR-NEXT-Schleife als Grenze.

Beispiel:

Die Neueingabe eines Namens usw. erfolgt in dem folgenden Programmbeispiel ab Zeile 300 jeweils in ein Feld, dessen Nummer durch die LOF-Funktion + 1 bestimmt wird. Beim Suchen muß man berücksichtigen, daß die eingegebene Zeichenkette NA\$ nur den Namen enthält, während in N\$ der Name mit Leerzeichen gespeichert wurde. Damit der Vergleichsbefehl in 250 richtig arbeitet, muß vorher NA\$ auf insgesamt 25 Stellen mit Leerzeichen aufgefüllt werden, was in den Zeilen 225 und 227 erfolgt.

```
20 CLEAR 1000
100 INPUT "S für Suchen, N für Neueintragen";X$
110 IF X$ = "S" THEN 200
120 IF X$ = "N" THEN 300 ELSE 100
200 OPEN "R",2,"TELREG"
210 GOSUB 2000
220 INPUT "Bitte Namen eingeben";NA$
225 NA$=NA$+STRING$(25-(LEN(NA$)),32)
230 FOR K = 1 TO LOF(2)
240 GET2,K
250 IF NA$=N$ THEN CLOSE:GOTO500
260 NEXT
270 PRINT "Name nicht eingetragen"
280 CLOSE:GOTO100
300 OPEN "R",2,"TELREG"
310 INPUT "Name";NA$
320 INPUT "Strasse";ST$
330 INPUT "Ort";OTS
340 INPUT "Telefonnummer";TNS
```

```

350 GOSUB2000
360 LSET O$ = OT$
370 LSET S$ = ST$
380 LSET T$ = TN$
390 LSET N$ = NA$
400 LF = LOF(2)+1
410 PUT2,LF
420 CLOSE:GOTO100
500 PRINT:PRINTN$;T$:PRINT
510 PRINTS$;O$
520 END
2000 FIELD#2,25 AS N$,15 AS T$,25 AS O$,25 AS S$
2010 RETURN

```

Im Genie-DOS wurde die Funktion

LOC(Puffernummer)

definiert. Mit Hilfe dieser Funktion haben Sie die Möglichkeit die aktuelle Satznummer einer geöffneten Datei mit direktem Zugriff zu ermitteln.

Mit "Puffernummer" ist die Nummer des Puffers der zuvor geöffneten Datei gemeint. Die Abfrage ergibt die Satznummer des in diesem Puffer zuletzt mit GET oder PUT bearbeiteten Satzes. Bei dieser Zahl handelt es sich um einen Integer Wert zwischen 1 und 32767.

Das CMD-Kommando =====

Dieses Kommando erlaubt es dem Programmierer, aus laufenden BASIC-Programmen heraus oder aber auch als direkten Befehl ein DOS-Kommando auszuführen.

Man kann sich z.B. mit Hilfe des CMD-Kommandos das Inhaltsverzeichnis einer Diskette anschauen, ins DOS zurückspringen, die interne Uhr und das Datum setzen und vieles mehr.

Auf die verschiedenen Möglichkeiten soll hier nun noch etwas genauer eingegangen werden.

Der allg. Ausdruck zum CMD-Kommando lautet

```
CMD"Dosbefehl"
```

Er erlaubt es, allgemeine DOS-Befehle aufzurufen oder in ein BASIC-Programm einzubauen.

Nachdem der Computer den DOS-Befehl ausgeführt hat, kehrt er ins BASIC zurück und setzt den normalen Ablauf des BASIC-Programms fort, wenn es durch den DOS-Befehl nicht geändert oder zerstört wurde.

Beispiel:

```
10 CLS
20 PRINT"Möchten Sie das Inhaltsverzeichnis (I)"
30 PRINT"oder die DOS-Befehlsliste (B) sehen ?"
40 A$=INKEY$
50 IF A$=""THEN40
60 IF A$="I"THEN CMD"I 0"ELSE80
70 PRINT:PRINT"Inhaltsverzeichnis der Systemdiskette":END
80 IF A$="B"THEN CMD"? "ELSE40
90 PRINT:PRINT"DOS-Befehlsliste":END
```

Wenn der DOS-Befehl nur aus einem Zeichen besteht, so muß noch ein Leerzeichen angehängt werden, um zu verhindern, daß der Rechner versucht, den Befehl als CMD"x" in der erweiterten BASIC-Befehlsliste zu finden.

Bei dem Kommando

```
CMD"x"
```

kann das x durch verschiedene Zeichen mit unterschiedlichen Aufgaben ersetzt werden.

Mit Hilfe des Kommandos

```
CMD"C
```

werden alle Leerzeichen, außer innerhalb von Strings und alle Bemerkungen (REM und '), auch wo die Bemerkung in Zeilen als Folgestatement steht, aus dem Programmtext entfernt. Für den Fall, daß ein GOTO oder GOSUB eine Zeile anspringt, die ausschließlich aus einer Bemerkung besteht, muß das Programm entsprechend umgeschrieben werden.

Sollen die REM-Zeilen im Programm bestehen bleiben und nur die unnötigen Leerstellen entfernt werden, so steht das Kommando

```
CMD"C",S
```

zur Verfügung.

Mittels des Kommandos

```
CMD"C",R
```

lassen sich umgekehrt auch sämtliche REM-Zeilen herausnehmen, ohne die Leerstellen zu entfernen.

Die Ausgabe der aktuellen, sprich letzten G-DOS-Fehlermeldung, die an das BASIC übergeben wurde, erfolgt auf

```
CMD"E"
```

Das Kommando

```
CMD"O"
```

erlaubt das Sortieren von dimensionierten Variablen im Speicher, wobei es drei verschiedene Möglichkeiten gibt, mit diesem Befehl zu arbeiten.

```
CMD"O",Anzahl,vz Dimvar(Beginn),...  
CMD"O",Anzahl,vz Dimvar$(Beginn) (pos,länge),...  
CMD"O",Anzahl,*Indexdimvar,vz Dimvar(Beginn)...
```

Als "Anzahl" wird die Anzahl der zu sortierenden Elemente angegeben. Die Eingabe 0 bedeutet, daß alle Elemente bearbeitet werden.

Wenn Strings in umgekehrter alphabetischer Reihenfolge oder Beträge ihrer Größe nach sortiert werden sollen, ist mit "vz" die Art des Sortiervorganges anzugeben.

"Beginn" ist der Index, ab dem alle Variablen sortiert werden. Sollen alle Variablen betrachtet werden, so ist "Beginn" mit 0 anzugeben. Bei String-Variablen kann der Vergleich auch auf einen Teil der Variablen beschränkt werden. Es ist möglich den Sortiervorgang von bis zu 9 dimensionierten Variablen abhängig werden zu lassen. Gibt man beispielsweise

```
CMD"O",5,-A$(1)(1,3)
```

ein, so wird von den dimensionierten Variablen A\$(x) die Variable A\$(0) nicht sortiert. Bei A\$(5) endet die Sortierung, die in umgekehrter alphabetischer Reihenfolge erfolgt. Aus dem Klammersausdruck (1,3) geht hervor, daß nur die ersten 3 Zeichen der Variablen verglichen werden (ähnl. wie bei MID\$).

Wenn eine Variable, deren Position durch das Sortieren geändert wurde, auf die Diskette zurückgeschrieben werden soll, so müssen alle der entsprechenden Matrix zugeordneten Variablen zurückgeschrieben werden.

Hierbei empfiehlt sich die Indexsortierung. Die Dateistruktur kann dabei unverändert bleiben. Nur der die Reihenfolge des Zugriffes oder der die Ausgabe bestimmende Index sollte nach der Bearbeitung zurückgeschrieben werden.

Über den Befehl

CMD"J",Datum1,Datum2

ist eine Konvertierung des Datums möglich. Dabei wird der Ausdruck von "Datum1" konvertiert in die Stringvariable "Datum2" geschrieben. Zu beachten ist dabei, daß das "Datum1" im amerikanischen Format mm/tt/jj, bei dem zuerst der Monat angegeben ist, vorliegen muß.

Nach der Ausführung von CMD"J" enthält die Variable "Datum2" den entsprechenden Tag des Jahres in der Form ttt.

Beispiel:

```
10 CLS
20 D$=LEFT$(TIME$,8):DA$=D$
30 MID$(D$,3,1)=CHR$(47):MID$(D$,6,1)=CHR$(47)
40 D1$=MID$(D$,4,3)+LEFT$(D$,3)+RIGHT$(D$,2)
50 D2$=""
60 CMD"J",D1$,D2$
70 PRINT"Datum: ";DA$
80 PRINT"Das ist der Jahrestag Nr.: ";D2$
```

Umgekehrt läßt sich auch nach Angabe der Jahres- und Tageszahl ein dementsprechendes Datum im amerikanischen Format mm/tt/jj erzeugen. Dabei ist zu beachten, daß Jahres- und Tageszahl in der Form -jj/ttt anzugeben sind.

Beispiel:

```
10 CLS
20 INPUT"Geben Sie bitte die Jahreszahl ein ";JZ$
30 INPUT"Geben Sie bitte eine Tageszahl ein ";TZ$
40 D1$=CHR$(45)+RIGHT$(JZ$,2)+CHR$(47)+VZ$
50 D2$=""
60 CMD"J",D1$,D2$
70 MID$(D2$,3,1)=CHR$(46):MID$(D2$,6,1)=CHR$(46)
80 D$=MID$(D2$,4,3)+LEFT$(D2$,3)+RIGHT$(D2$,2)
90 PRINT"Die angegebene Tageszahl ";TZ$;" des Jahres ";JZ$
100 PRINT"entspricht dem Datum: ";D$
```

Mit Hilfe des Befehls

CMD"T"

schaltet man die Interruptkette und damit auch die Uhr und Autorepeatfunktion ab. Er kann entweder im Befehlsmodus oder innerhalb eines Programmes gegeben werden. Bei Arbeiten mit Kassetten als Datenspeichermedium, (Genie I/II) ist die Eingabe von CMD"T" zu Beginn unumgänglich.

Um ein CMD"T" wieder rückgängig zu machen muß

CMD"R"

eingegeben werden. Die systeminterne Interruptkette wird dadurch wieder aktiv, Zeitanzeige und Autorepeat sind wieder funktionsfähig.

CMD"S"

veranlaßt den Rechner das BASIC zu verlassen und G-DOS aufzurufen. Auf dem Bildschirm erscheint

Befehlseingabe:

Lautet das Kommando

CMD"S=Dosbefehl"

so führt das System den angegebenen DOS-Befehl aus, kehrt aber im Gegensatz zu

CMD"Dosbefehl"

nach seiner Ausführung nicht mehr ins BASIC zurück, sondern bleibt im DOS wie bei CMD"S".

Wenn während des Programmlaufs Zeilen des Programmtextes gelöscht werden sollen, so ist das Kommando

CMD"F",DELETE Anfangszeilennummer - Endzeilennummer

einzugeben. Mit Hilfe dieses Kommandos können Sie alle Textzeilen im Programm, die größer/gleich der "Anfangszeilennummer" und kleiner/gleich der "Endzeilennummer" sind, löschen. Innerhalb des zu löschenden Bereiches dürfen die keine aktive Definitionen (DEF FN), keine unabgeschlossen Unterprogramme (GOSUB ohne RETURN) und keine unabgeschlossen Schleifen (FOR ohne NEXT) benutzt werden.

Außerdem darf der Befehl CMD"F",DELETE nicht direkt, sondern nur innerhalb eines Programmes verwendet werden. Er muß dort die letzte Anweisung in einer Zeile sein und von der Zeile gefolgt werden, in der die Programmausführung fortgesetzt werden soll nachdem die betreffenden Zeilen gelöscht sind.

Das Kommando

CMD"F=Parameter"

führt je nach angegebenen Parametern folgende Funktionen aus.

Bei der Eingabe von

CMD"F=POPS"

(POPS = Pop Stack)

werden alle FOR-NEXT-Schleifen und Rückkehradressen aus GOSUB-Ebenen gelöscht. Dadurch erhält der Benutzer die Möglichkeit, alle noch nicht benutzten Rücksprünge und alle abgebrochenen FOR-NEXT-Schleifen zu löschen, da dies speziell bei rekursiver Programmierung zu Fehlern führen kann. CMD"F=POPS" arbeitet ähnlich wie CLEAR ohne jedoch die vorhandenen Variablen zu löschen.

CMD"F=POPR"

(POPR = Pop Return)

löscht nur die Rücksprungadressen der momentan bearbeiteten GOSUB-Ebenen und die in diesen GOSUB-Ebenen befindlichen FOR-NEXT-Schleifen.

Das Programm führt danach nicht das Statement, das auf das Return folgt, aus, sondern springt unmittelbar auf das dem CMD"F=POPR" folgende Statement.

Beim Kommando

CMD"F=POPN"Zähler

(POPR = Pop Return)

werden alle FOR-NEXT-Schleifen innerhalb der Schleife mit dem angegebenen Zähler gelöscht. Der folgende Programmablauf richtet sich nach der aktuellen Zeile. Fehlt die Angabe "Zähler", so wird nur die gerade bearbeitete FOR-NEXT-Schleife gelöscht.

Mit dem Kommando

CMD"F=SASZ",Zahl

(SASZ = String Area Size)

können Sie die Größe des Stringbereiches ändern, ohne daß die eventuell schon belegten Stringvariablen ihren Wert ändern. Die "Zahl" muß einen genügend großen Wert für die Zeichenkettenoperationen annehmen.

Der Befehl kann beispielsweise benutzt werden, wenn sich während eines Programms herausstellt, daß der Stringbereich zu klein oder zu groß ist.

Mit der folgenden Befehlsreihe lassen sich Variablen löschen:

CMD"F=ERASE",vn1,vn2,vn3,...

vn1,vn2, und vn3 sind hierbei die zu löschenden Variablennamen. Dieses Kommando kann auch bei dimensionierten Feldern verwendet werden. Der Speicherplatz steht nach der Ausführung des Befehls wieder zur Verfügung.

CMD"F=KEEP",vn1,vn2,vn3,...

bewirkt, daß alle Variablen, außer den angegebenen und den über DEFFN vereinbarten Variablentypen, gelöscht werden. Wenn eine angegebene Variable ein Feldname ist, so bleibt das gesamte Feld erhalten.

Alle Variablen, die nicht gelöscht werden sollen, müssen in einem Statement angegeben werden.

Mit dem Kommando

```
CMD"F=SWAP",vn1,vn2
```

läßt sich der Wert zweier Variablen vertauschen. Beide müssen in ihrem Typ gleich sein, d.h. es kann nur Integer gegen Integer, String gegen String, Fließkomma gegen Fließkomma usw. ausgetauscht werden.

Die beiden zu tauschenden Variablen müssen natürlich unterschiedliche Namen haben.

Mit G-DOS hat der Benutzer die Möglichkeit, seine Programme auch schrittweise ausführen zu lassen. Das Kommando dazu lautet

```
CMD"F=SS",Zeilennummer (SS = Single Step)
```

Wird keine Zeilennummer angegeben, führt der Rechner das Programm von Anfang an in Einzelschritten aus. Bei angegebener Zeilennummer erfolgt zunächst ein normaler Programmablauf. Sobald jedoch die angegebene Zeilennummer erreicht ist, schaltet der Computer auf Einzelschrittausführung um. Das Programm wird dabei Zeile für Zeile abgearbeitet, und der Benutzer hat vor jedem Befehl die NEW-LINE-Taste zu betätigen. In der oberen rechten Bildschirmecke wird jeweils die aktuelle Zeilennummer angezeigt.

Durch Drücken der BREAK-Taste läßt sich der Programmablauf wie üblich abbrechen. Wenn Sie dann am Programmtext nichts verändern, können Sie mit dem Befehl CONT den Programmablauf fortsetzen.

Die Einzelschrittausführung bleibt solange aktiv, bis Sie mit dem Kommando

```
CMD"F=SS",N
```

wieder in den normalen Ablauf-Modus übergehen.

DOS-Anpassungen

=====

Das Betriebssystem G-DOS erlaubt eine softwaremäßige Anpassung an verschiedene Diskettenformate und Laufwerkstypen durch Eingabe eines Anpaß-Befehls zu Beginn der Arbeit. Die Parameter dieses Befehls werden auf der System-Diskette gespeichert und bei jedem Booten mit berücksichtigt.

Nach der ersten Eingabe dieses Befehls braucht er also nur dann wieder eingegeben zu werden, wenn andere Laufwerke angeschlossen werden oder andere Disketten längerfristig zur Verwendung kommen sollen.

Der Anpaß-Befehl oder auch PD-Befehl wird im DOS als direktes Kommando gegeben und setzt sich aus dem Befehlswort

PD (PD = Parameter of Drives)

dem die Nummer des Laufwerks, in dem sich die zu ändernde System-Diskette befindet, folgen muß und verschiedenen wahl-freien Zusätzen zusammen.

Wegen der vielen Möglichkeiten, die der Pdrive-Befehl bietet, ist er recht kompliziert aufgebaut. Daher sollen hier nur die wichtigsten Parameter und ihre Wirkung erklärt werden.

PD, lwnr Befehlswort und Laufwerknummer, auf der die System-Diskette läuft

dann w a h l w e i s e

, lwnr Nummer des Laufwerks, für das die folgenden Parameter gelten sollen,

, TI= Interfacetyp
Nach dem Gleichheitszeichen sind die Kennbuchstaben für die Art des beim Anschluß des Laufwerks verwendeten Controllers (Interface) einzusetzen.

, TD= Typ des Diskettenlaufwerks
Hier ist ein Kennbuchstabe für die verwendete Diskettenart und deren Formatierung einzusetzen

, SP= Spuranzahl
Hier ist die Anzahl der Spuren, die auf der Diskette nutzbar sind, einzusetzen.

, SEK= Sektoren pro Spur
Hier ist die Anzahl der Sektoren pro Spur auf der Diskette einzusetzen. Bei Verwendung von doppelseitig arbeitenden Laufwerken ist hier eine Zahl, die der doppelten Anzahl der Sektoren einer Spur entspricht, anzugeben.

, SWZ= Spurwechselzeitfaktor
Hier wird die Zeitdauer der Impulse für die Kopfpositionierung eingesetzt.
SWZ=3 ist der sicherste Wert. Die meisten Laufwerke arbeiten jedoch auch mit 0 oder 1 einwandfrei.

- ,EIB= Einheiten im Block
Hier wird die Anzahl der Einheiten pro Block angegeben. Die Zahl darf zwischen 2 und 8 liegen. Beim G-DOS werden 5 Sektoren pro Einheit und 2 - 8 Einheiten pro Block (nicht Spur) erzeugt.
- ,SBIV= Startblock Inhaltsverzeichnis
Hier wird die Blocknummer angegeben, in der der erste Sektor des Inhaltsverzeichnisses steht.
- ,AEIV= Anzahl der Einheiten im Inhaltsverzeichnis
Hier wird die Anzahl der Einheiten angegeben, die für das Inhaltsverzeichnis beim NDF- oder COPY-Befehl reserviert werden.
- ,A Ein A am Ende eines PD-Befehls gibt an, daß er sofort geladen und aktiv werden soll, (allerdings nur bei fehlerfreier Eingabe).

Die folgenden Kennbuchstaben werden am häufigsten benötigt:

Für TI

- A bedeutet, daß das Standard-Interface für Diskettenoperation eingebaut ist.
- C bedeutet, daß ein Interface für doppelte Schreibdichte eingebaut ist.
- K bedeutet, daß die Spur 00 nicht mit der gleichen Schreibdichte der weiteren Spuren formatiert ist.
- L bedeutet, daß von einer Spur zur nächsten zwei Impulse gegeben werden. Es können mit 80 Spur-Laufwerken auch Disketten mit 35 oder 40 Spuren gelesen werden.
- E Beim Genie III wird dadurch auch das Arbeiten mit angeschlossenen 8 Zoll Laufwerken möglich.

Für TD

- A 5" einfache Schreibdichte, einseitig
- C 5" einfache Schreibdichte, doppelseitig
- E 5" doppelte Schreibdichte, einseitig
- G 5" doppelte Schreibdichte, doppelseitig

Für SP

Wird für TI die Option K gewählt, muß hier die Anzahl der Spuren evtl. um 1 kleiner eingesetzt werden als es der Spurenzahl entspricht, da eine zusätzliche Spur 0 in SD auf die Diskette geschrieben wird und kritische Laufwerke die letzte Spur nicht mehr fehlerfrei beschreiben oder lesen können. Mit guten Laufwerke lassen jedoch problemlos auch 41 oder 81 Spuren verarbeiten.

Für SEK

Bei doppelseitigen Laufwerken wird die Spur mit der gleichen Nummer auf Vorder- und Rückseite der Diskette als eine Spur behandelt. Die Sektorzahl pro Spur muß also bei diesen Laufwerken gegenüber einseitigen Laufwerken mit 2 multipliziert werden.

Eine 40 Spur G-DOS 2.2 Systemdiskette mit doppelter Schreibdichte kann z.B. nach Eingabe des Befehls PD 0 folgende Tabelle anzeigen:

Befehlseingabe:PD 0

```
0* TI=CK,TD=E,SP=40,SEK=18,SWZ=3,EIB=3,SBIV=24,AEIV=3
1* TI=CK,TD=E,SP=40,SEK=18,SWZ=3,EIB=3,SBIV=24,AEIV=3
2* TI=A,TD=A,SP=40,SEK=10,SWZ=3,EIB=2,SBIV=20,AEIV=2
3* TI=A,TD=A,SP=40,SEK=10,SWZ=3,EIB=2,SBIV=20,AEIV=2
4  TI=CKL,TD=E,SP=40,SEK=18,SWZ=3,EIB=3,SBIV=24,AEIV=3
5  TI=CK,TD=E,SP=40,SEK=18,SWZ=3,EIB=3,SBIV=24,AEIV=3
6  TI=A,TD=A,SP=80,SEK=10,SWZ=3,EIB=2,SBIV=40,AEIV=2
7  TI=A,TD=C,SP=80,SEK=20,SWZ=3,EIB=2,SBIV=40,AEIV=2
8  TI=CK,TD=E,SP=80,SEK=18,SWZ=3,EIB=3,SBIV=48,AEIV=3
9  TI=CK,TD=G,SP=80,SEK=36,SWZ=3,EIB=6,SBIV=48,AEIV=6
```

Befehlseingabe:

Die Bedeutung einiger Pdrive Parameter aus dieser Tabelle:

```
0* TI=CK,TD=E,SP=40,SEK=18,SWZ=3,EIB=3,SBIV=24,AEIV=3
```

ist die Standardformatierung für einseitige 5" Disketten mit 40 Spuren und doppelter Schreibdichte (Spur 0 einfache Schreibdichte).

```
2* TI=A,TD=A,SP=40,SEK=10,SWZ=3,EIB=2,SBIV=20,AEIV=2
```

ist die Standardformatierung für einseitige 5" Disketten mit 40 Spuren und einfacher Schreibdichte.

```
7  TI=A,TD=C,SP=80,SEK=20,SWZ=3,EIB=2,SBIV=40,AEIV=2
```

ist die Standardformatierung für doppelseitige 5" Disketten mit 80 Spuren und einfacher Schreibdichte.

```
9  TI=CK,TD=G,SP=80,SEK=36,SWZ=3,EIB=6,SBIV=48,AEIV=6
```

ist die Standardformatierung für doppelseitige 5" Disketten mit 80 Spuren und doppelter Schreibdichte. (Spur 0 einfache Schreibdichte)

Wenn Sie die Angaben für eine Diskettenstation ändern wollen, sind die Parameter folgendermaßen einzugeben

```
PD,0,1,TI=A,TD=A,SP=40,SEK=10,SWZ=3,EIB=2,SBIV=20,AEIV=2
```

Die Kommata können auch durch Leerzeichen ersetzt werden.

Für den Fall, daß in der Tabelle die von Ihnen gewünschte Einstellung bereits vorhanden, aber unter einer anderen Nummer abgelegt ist, können Sie z.B. die Einstellung von Nummer 9 auf Nummer 1 durch Eingabe des Befehls

```
PD,0,1=9,A
```

übertragen und sofort aktivieren.

Noch einige Ratschläge

Sie können nun, wenn Sie das bisher Beschriebene halbwegs verstanden und an einigen kleinen Beispielen ausprobiert haben, sowie die Grundkenntnisse von BASIC besitzen, Programme im Diskettenbetrieb entwickeln und abspeichern. Das Betriebssystem G-DOS bietet jedoch noch eine Fülle weiterer Möglichkeiten, die in den folgenden Abschnitten dieses Handbuchs beschrieben sind.

Es kann leider nie ausgeschlossen werden, daß der Inhalt einer Diskette durch eine versehentliche Eingabe falscher Befehle oder durch einen plötzlichen Stromausfall beschädigt wird. Meist werden dabei Teile der Aufzeichnungen unlesbar. Sie können dann nur noch austesten, welche Dateien bzw. Programme sich noch einwandfrei laden lassen und diese dann auf eine andere Disk kopieren. Die beschädigte Diskette ist nach einer Formatierung wieder als Leerdiskette verwendbar. Man sollte daher nach Möglichkeit von allen wichtigen Programmen sofort Sicherungs- und Arbeitskopien erstellen und nur mit diesen arbeiten. Sie können natürlich auch die komplette Diskette zur Sicherheit kopieren.

Bei Diskettenbetrieb gibt es eine Reihe von Steckverbindungen (auch im Innern der Laufwerke). Stellen Sie beim Betrieb unerklärliche Fehler fest, sollten Sie zunächst alle Stecker auf richtigen Anschluß und festen Sitz überprüfen.

Unerklärlichen Ärger kann es geben, wenn ein Laufwerk mit stark abweichender Drehzahl läuft. Um diese zu kontrollieren befindet sich meist auf der Drehachse eine Stroboskopscheibe mit weißen und schwarzen Feldern. Wird diese mit einer Leuchtstofflampe angestrahlt, so stehen die weißen und schwarzen Streifen in einem Feldring still, sofern das Laufwerk mit der richtigen Geschwindigkeit von 300 U/min läuft. Eine Nachregelung ist in der Regel mit Hilfe eines kleinen Trimpotis auf der Platine des jeweiligen Floppylaufwerks möglich. Weitere Fehler die auf das Laufwerk oder die Diskette zurückzuführen sind, werden vom Betriebssystem bei der Formatierung einer Disk mit NDF erkannt und ausgegeben.

Da Disketten doppelseitig beschichtet sind, können sie in einem einseitig arbeitenden Laufwerk auch umgedreht werden. Hierzu sind jedoch noch ein zweites Indexloch und eine zweite Schreibschutzkerbe in der Hülle anzubringen. Beides muß genau symmetrisch zur Mittelachse liegen. Seien Sie bei einer solchen Aktion bitte sehr vorsichtig, denn die Magnetschicht ist sehr empfindlich und kann leicht verkratzt werden.

Tai !!

DOS-Refahr

Was ist ein Betriebssystem ?

=====

Mit dieser Beschreibung erhalten Sie Ihre G-DOS-Systemdiskette, das sogenannte Betriebssystem.

Um zu verstehen, was ein Betriebssystem ist, sollen hier noch einmal die einzelnen Komponenten Ihrer Microcomputer-Anlage in Erinnerung gerufen werden:

Das Herzstück ist sicherlich der eigentliche Rechner, der zunächst den Prozessor (CPU=Central Processing Unit) enthält. Dieser Baustein ermöglicht durch eine hochintegrierte Schaltung die Verarbeitung von Daten, die ihm über Daten- und Adressleitungen zugeführt werden. Nach der Verarbeitung werden diese Daten wieder an periphere Verarbeitungseinheiten weitergegeben.

Eine weitere Einheit stellt der sogenannte RAM-Speicher dar (RAM = Random Access Memory), ein veränderbarer Schreib/Lese-Speicher, in dem Daten oder Programme für die Dauer ihrer Verarbeitung zwischengespeichert, abgelegt oder verarbeitet werden.

Programme, die manuell eingegeben oder von Band oder Diskette eingelesen werden, gelangen in den RAM-Speicher, wo sie bearbeitet (= editiert) werden können oder von wo aus sie nach einem Programmstart den Programmablauf kontrollieren.

Daten, die von einem Programm aus sortiert werden sollen, werden in den RAM-Speicher geladen und dort sortiert, bei Bedarf dann in sortierter Reihenfolge wieder auf ein externes Speichermedium zurückgeschrieben.

Im Gegensatz zum RAM-Speicher erlaubt der ROM-Speicher (ROM = Read Only Memory) nur Lese-Zugriffe, hier residiert z.B. bei Genie I und II der BASIC-Interpreter, auf den zwar für Leseoperationen zugegriffen werden kann, der Inhalt dieses Speichers ist aber nicht veränderbar.

Eingabetastatur und Bildschirm komplettieren das Microcomputersystem zu einer funktionstüchtigen Einheit.

Da alle Daten oder Programme, die sich im RAM-Speicher befinden, gelöscht werden, sobald die Anlage von der Spannungsversorgung getrennt wird, ist es notwendig, diese zur späteren Wiederverwendung auf ein externes Speichermedium zu übertragen, von wo sie bei Bedarf wieder in den Rechner eingelesen werden können.

Die preisgünstigste Lösung zur Datenspeicherung stellt sicherlich der Kassettenrekorder dar, wie er z.B. im Genie I serienmäßig eingebaut ist.

Auch bei Genie II und Speedmaster ist der Rekorderanschluß unproblematisch, da das notwendige Kassetteninterface bereits im Gerät enthalten ist und ein beliebiger Rekorder nur noch über ein Diodenkabel an den Rechner anzuschließen ist.

Bei Genie III wurde auf ein Kassetten-Interface verzichtet, da das Gerät in der Grundversion schon mit zwei Diskettenlaufwerken zur Datenspeicherung ausgestattet ist.

Die Nachteile der Kassettenspeicherung liegen in erster Linie in dem Zeitaufwand, wie ihn die Speicherung großer Programme oder Datenmengen mit sich bringt.

Die Anforderungen an eine effektive Nutzung der Computeranlage speziell in der kommerziellen Anwendung schließen deshalb die Verwendung eines Kassettenrekorders als externe Speichereinheit fast gänzlich aus.

Die Lösung dieser Problematik stellt für die meisten Anwendungen der Anschluß von Diskettenlaufwerken an die Computeranlage dar.

Bei dieser Art der Datenspeicherung beträgt der notwendige zeitliche Aufwand nur noch einen Bruchteil dessen, was z.B. bei einer Speicherung auf Kassetten notwendig wäre. Programme von mehreren Tausend Zeichen werden innerhalb von Sekunden zuverlässig auf die Disketten geschrieben, um genauso schnell wieder von dem Datenträger in den Arbeitsspeicher geladen werden zu können.

Da die Diskettenlaufwerke selbst über keine eigene "Intelligenz" verfügen, um Daten sinnvoll auf eine Diskette schreiben zu können, ist dazu ein Werkzeug notwendig, welches die komplizierten Vorgänge während einer Datenspeicherung steuert und überwacht.

Dieses Werkzeug stellt nun neben dem Floppy-Controller, der hardwaremäßig die Voraussetzungen für Diskettenoperationen schafft, von der Software her das "Betriebssystem" dar, welches praktisch die Verbindung zwischen dem Rechner und den (bis zu vier) angeschlossenen Diskettenlaufwerken herstellt.

Das Betriebssystem ist demzufolge weder ein Interpreter, der eine bestimmte Programmiersprache für den Anwender bereitstellt, in der dieser seine Anwendungen realisieren kann, noch stellt es ein Anwenderprogramm selbst dar, mit dem sich in der praktischen Anwendung individuelle Probleme lösen ließen.

Vielmehr gibt das Betriebssystem dem Anwender die Möglichkeit, durch ein Fülle neuer Befehle auf eine Reihe von Routinen zurückzugreifen, die in mittelbarem oder unmittelbarem Zusammenhang mit Disketten-Ein/Ausgabe-Operationen zusammenhängen.

Wichtig ist dabei grundsätzlich, daß sich die verschiedenen Ebenen, in denen sich einzelne Verarbeitungen abspielen, klar voneinander abgrenzen und nicht verwechselt werden dürfen.

Der eine Bereich ist die sogenannte DOS-Ebene (DOS = Disk Operating System), also das eigentliche Betriebssystem. Von hier aus kann neben vielen weiteren Funktionen das Inhaltsverzeichnis der Diskette aufgerufen werden, hier werden Maschinenprogramme gestartet oder in den Arbeitsspeicher geladen, oder Kopier- und Formatierungsfunktionen aufgerufen.

Die andere Ebene ist die der jeweiligen Programmiersprache, in der Anwenderprogramme erstellt und verarbeitet werden können. Auf der G-DOS-Systemdiskette finden Sie unter anderem das sog. Disk-BASIC unter der Bezeichnung BASIC/CMD.

Die unterschiedlichen Verarbeitungsebenen sind anhand ihrer unterschiedlichen Bereitschaftsmeldungen leicht zu unterscheiden. Das DOS meldet sich nach dem Start der Systemdiskette mit der Aufforderung "Befehlseingabe: ", während sich das Disk-BASIC nach dem Aufruf von BASIC/CMD mit "READY" meldet.

Beide Ebenen verfügen über unterschiedliche Befehlsvorräte, die nicht miteinander verwechselt werden dürfen.

Innerhalb der DOS-Ebene kann kein BASIC-Befehl verwendet werden, ohne daß es zu einer Fehlermeldung käme.

Aus dem BASIC gibt es allerdings die Möglichkeit, durch einen Befehlszusatz eine bestimmte DOS-Routine aufzurufen. Dadurch ist es z.B. möglich, aus einem Anwenderprogramm heraus eine Kopier- oder Formatfunktion aufzurufen.

Die Schreibweise für den DOS-Aufruf aus dem BASIC lautet in diesem Fall `CMD"Dosbefehl"`. Durch das dem DOS-Kommando vorangestellte CMD in Verbindung mit dem in Anführungszeichen gesetzten DOS-Befehl wird die Eingabe als DOS-Befehl interpretiert, der aus dem BASIC heraus ausgeführt werden soll. Mehr darüber finden Sie im dritten Teil dieses Handbuches.

Auf den folgenden Seiten werden die verschiedenen DOS-Kommandos in alphabetischer Reihenfolge aufgeführt und ihre Wirkungsweise durch einzelne Beispiele verdeutlicht.

SYNTAX: AIK Laufwerksnummer

=====

WIRKUNG: Die Adressmarken des Inhaltsverzeichnisses der Diskette werden umgeschrieben.
Unterschiedliche Floppy-Disk-Controller benutzen unterschiedliche Adressmarken. Ein Controller für doppelte Schreibdichte benutzt als Adressmarke innerhalb des Inhaltsverzeichnisses den Wert F8. Dieser Wert ist bei Controllern einfacher Schreibdichte FC.

Um die jeweiligen Disketten auch für einen anderen Typ lesbar zu machen, gestattet der Befehl AIK eine Korrektur der Adressmarken des Inhaltsverzeichnisses.

Dabei werden die auf der Diskette befindlichen Adressmarken von FC in F8 geändert.
Nach Umschreiben der Adressmarken kann eine entsprechende Single-Density-Diskette sowohl vom Controller 1771 (single-) als auch vom Controller 1791 (double density) gelesen werden, da auch der Controller für einfache Schreibdichte problemlos mit der Adressmarke F8 arbeiten kann.

ANWENDUNG: Anpassung von Disketten, die unter einem anderen Floppy-Disk-Controller angefertigt wurden, zur Nutzung auf einer Anlage, die den Typ F8 als Adressmarke verarbeitet.

Die Notwendigkeit der Anwendung dieser Funktion ergibt sich beispielsweise, wenn Disketten, die auf einer Genie I/II-Anlage mit Expansioninterface angefertigt wurden, für die Nutzung auf einer Anlage mit Diskettenlaufwerken der FC-Serie umgestellt werden sollen.

BEMERKUNG: Der Befehl AIK sollte normalerweise immer zur Änderung einer Diskette in einem der Laufwerke von 1 bis 3 genutzt werden, nicht auf Laufwerk Null.
Vor dem Befehlsaufruf muß unbedingt gewährleistet sein, daß die PD-Parameter für die Diskette in dem angesprochenen Laufwerk richtig eingestellt sind.
Falsche Parameter führen hier zur Zerstörung Ihrer Diskette!
Disketten, die auf die neue Adressmarke F8 konvertiert worden sind, lassen sich danach durch beide Controller-Typen problemlos verarbeiten.

BEISPIEL: AIK 1

Bewirkt, daß die Adressmarken des Inhaltsverzeichnisses der Diskette in Laufwerk 1 umgeschrieben werden.

SYNTAX: APPEND Dateil Datei2

=====

WIRKUNG: Dieses DOS-Kommando dient dazu, zwei Dateien oder Programme miteinander zu verbinden.

ANWENDUNG: Beide Files müssen sich vor der Ausführung des APPEND-Kommandos bereits auf der Diskette befinden. "Datei2" soll durch die "Dateil" erweitert werden. Die "Dateil" wird an das Ende der "Datei2" angefügt. "Dateil" bleibt dabei unverändert.

APPEND findet vor allem bei der Sicherung kleinerer Dateien (Texte) Anwendung.

BEMERKUNG: Wenn die auf der Diskette befindliche Datei ein typisches EOF-Zeichen besitzt, kann es bei der Ausführung des APPEND-Kommandos zu Schwierigkeiten kommen. BASIC-Files lassen sich beispielsweise auf diese Art nicht miteinander verbinden. Benutzen Sie in diesem Fall den Disk-BASIC-Befehl MERGE.

BEISPIEL: APPEND TEXT2/DAT:0 TEXT1/DAT:1

Der Textbaustein TEXT1/DAT befindet sich auf der Diskette im Laufwerk 1 und soll durch das Textmodul TEXT2/DAT, das sich auf der Disk in Laufwerk 0 befindet, erweitert werden.

Bei der Ausführung des Kommandos wird der Inhalt von TEXT2/DAT an das Ende von TEXT1/DAT in Laufwerk 1 gehängt.

SYNTAX: ATTRIB Dateiname Parameter

=====

WIRKUNG: Das Kommando ATTRIB versieht eine auf der Diskette befindliche Datei mit einem oder mehreren Attributen. Mögliche Parameter dazu sind im Einzelnen

INV
VIS
PROT=x
HKW=Hauptkennwort
BKW=Bearbeitungskennwort
ADE=x (J oder N)
ADF=x (J oder N)
BEA=x (J oder N)

Der Befehl ATTRIB gestattet es, Dateien im Inhaltsverzeichnis der Diskette unsichtbar (oder auch wieder sichtbar) zu machen (INV/VIS), die Zugriffsstufen für einzelne Dateien zu definieren (PROT=xxxx), Dateien oder Programme mit verschiedenen Kennworten zu schützen (HKW/BKW), die Zulässigkeit von Dateierweiterungen und Diskettenplatz-Freigaben zu definieren (ADE/ADF) und das Bearbeitungskennzeichen einer Datei zu setzen oder zu löschen (BEA).

Die notwendigen Eingaben dazu werden anhand der Beispiele zu diesem Befehl auf der nächsten Seite verdeutlicht.

BEMERKUNG: Die Auswirkung der Eigenschaften, die Sie einzelnen Programmen oder Dateien mit dem Befehl ATTRIB zugeordnet haben, sind zum Teil von anderen Bestandteilen des Betriebssystems abhängig. So ist die Vergabe von Kennworten zum Schutz von Dateien überhaupt nur sinnvoll, wenn der entsprechende Systemparameter (AA=J) so eingestellt ist, daß das Betriebssystem Kennwortkontrollen durchführt.

Genauso können auch Dateien, die mit dem Parameter INV "unsichtbar" gemacht wurden, durch eine entsprechende Erweiterung des DIR-Befehles trotzdem mit angezeigt werden.

BEISPIELE: ATTRIB TESTPROG/BAS INV

Der Eintrag der Programmes TESTPROG/BAS innerhalb des Inhaltsverzeichnisses wird unsichtbar gemacht. Obwohl das Programm nach dem Aufruf des Inhaltsverzeichnisses (mit DIR Laufwerknummer) nicht mehr angezeigt wird, läßt es sich weiterhin genauso laden oder verarbeiten, als wenn es sichtbar wäre. Erst die erweiterte Eingabe des DIR-Befehles (mit DIR I Laufwerknummer) bewirkt, daß auch diese unsichtbare Datei mit auf den Bildschirm angezeigt wird.

ATTRIB BASIC/CMD VIS

Die (vorher unsichtbare) Datei BASIC/CMD wird sichtbar gemacht und erscheint wieder in der Anzeige des Inhaltsverzeichnisses nach einem einfachen I-Befehl.

ATTRIB ADRESSEN/DAT PROT=Ausdruck

Als Ausdruck sind sieben verschiedene Angaben möglich, die die Zugriffsart eines Bearbeiters auf eine bestimmte Datei (in diesem Beispiel ADRESSEN/DAT) definieren, wenn die Kennwortkontrollen aktiviert sind. Die Ausdrücke haben folgende Bedeutung:

- KEIN Datei für Zugriffe durch den Bearbeiter vollkommen gesperrt, nur das Betriebssystem hat noch Zugang.
- START Es ist nur der Programmstart (im BASIC also mit RUN oder LOAD,R) zugelassen. Die Break-Taste wird blockiert.
- LESEN Erlaubt das Lesen den Programmes
- ÄNDERN Das Programm kann gestartet, gelesen und auch verändert werden.
- NAME Das Programm kann gestartet, gelesen und verändert werden, zusätzlich ist die Veränderung des Dateinamen zulässig.
- KILL Zusätzlich zum Parameter NAME ist hier auch das Löschen der Datei zulässig.
- NULL Alle Manipulationen sind zugelassen.
-

ATTRIB TRANSFER/BAS HKW=ITALIEN

Der Datei TRANSFER/BAS wird das Hauptkennwort ITALIEN zugewiesen. Bei Aktivierung der Kennworte und einer entsprechenden Zugriffsklassifizierung (siehe oben) ließe sich das Programm im BASIC nur durch die Eingabe von LOAD"TRANSFER/BAS.ITALIEN" laden. Wird das aktivierte Kennwort bei einem Ladeversuch nicht mit angegeben, so gibt das System die Fehlermeldung "Datei verwehrt Zugriff" aus.

ATTRIB TESTPROG/BAS BKW=ABC12XYZ

Abhängig vom Schutzgrad des Programmes TESTPROG/BAS kann der Bearbeiter nur auf das Programm zugreifen, wenn er das Kennwort ABC12XYZ mit dem Programmnamen eingibt, z.B. LOAD"TESTPROG/BAS.ABC12XYZ"

HINWEIS: Ein Kennwort darf aus maximal acht Zeichen bestehen, wobei das erste Zeichen ein Buchstabe sein muß!

ATTRIB KUNDEN/DAT ADE=x

Je nach dem Parameter, der für x eingesetzt wird, werden Dateierweiterungen über den für die Datei KUNDEN/DAT reservierten Speicherplatz hinaus zugelassen (ADE=J) oder verhindert (ADE=N).

ATTRIB LIEFER/DAT ADF=x

ADF=N verhindert die Verkürzung eines reservierten Diskettenspeichers. ADF=J läßt dagegen das Auseinanderreißen eines reservierten Diskettenplatzes zu, falls dies aufgrund einer geringen Datenmenge vom Betriebssystem her sinnvoll erscheint.

ATTRIB NUMMERN/DAT BEA=x

Wird eine Datei auf Diskette gespeichert oder durch einen Schreibzugriff in Teilen verändert, so setzt das DOS ein sogenanntes Bearbeitungskennzeichen. Diese Kennung ist besonders sinnvoll, da sie es gestattet, bei dem Aufruf des Inhaltsverzeichnisses der Diskette (mit DIR A) auf einen Blick sehen zu können, welche Dateien sich seit der letzten Löschung der Bearbeitungskennzeichen verändert haben. Auf diese Weise ist es zudem möglich, im Rahmen einer Datensicherung das Bearbeitungskennzeichen abzufragen und auf diese Weise nur die Dateien oder Programme auf die Sicherungskopie zu übertragen; die sich auch wirklich verändert haben. Durch den Parameter BEA=N setzen Sie das Bearbeitungskennzeichen für die einzelnen definierte Dateien wieder auf Null, mit BEA=J kann ein einzelnes Bearbeitungskennzeichen gesetzt werden. Je nach Parameter, der in diesem Beispiel für x eingesetzt wird, würde das Bearbeitungskennzeichen der Datei NUMMERN/DAT also gelöscht (BEA=N) oder gesetzt (BEA=J) werden.

HINWEIS: Sollen die Bearbeitungskennzeichen aller auf der Diskette befindlichen Dateien gelöscht werden, so bedient man sich des Kommandos PROT (siehe dort).

SYNTAX: AUTO

oder

AUTO Befehlsfolge

=====

WIRKUNG: Die nach dem AUTO-Kommando definierte Befehlsfolge wird nach jedem Start (booten) des Betriebssystems automatisch ausgeführt, ohne daß dazu vorher eine Tastatureingabe notwendig ist.

ANWENDUNG: Die Anwendung dieses Befehls ist überall dort sinnvoll, wo eine bestimmte Anwendung mit jedem Start der Diskette automatisch ausgeführt werden soll. Besonders in jeglicher Art von Anwendersoftware ist es wünschenswert, daß das eigentliche Programm, mit dem gearbeitet werden soll, automatisch lädt und startet, ohne daß der Benutzer sich erst mit einer eventuell komplizierten Befehlsfolge vertraut machen muß, um das Programm zu starten.

BEMERKUNG: Die Durchführung des AUTO-Kommandos kann durch das Festhalten der NEW LINE- oder ENTER-Taste unterdrückt werden, falls diese Möglichkeit nicht durch den entsprechenden Systemparameter (BD=N) ausgeschlossen wurde.
Ist das Betriebssystem über die Systemparameter (AB=J) in den sogenannten RUN-ONLY-MODE versetzt, muß vorher unbedingt ein AUTO-Kommando definiert worden sein, da der Ladevorgang sonst abbricht. Der Rechner ist damit vollkommen blockiert und kann nur mit einer anderen Systemdiskette wieder gestartet werden!

BEISPIELE: AUTO BASIC RUN"TESTPROG/BAS"

Die Diskette, die mit diesem AUTO-Kommando versehen wurde, lädt nach jedem Systemstart automatisch das BASIC, danach das Programm TESTPROG/BAS, welches auch direkt startet.

AUTO DIR:0

Nach jedem Systemstart wird automatisch das Inhaltsverzeichnis der Diskette in Laufwerk 0 auf dem Bildschirm angezeigt, bevor das Betriebssystem sich bereit meldet, den nächsten Befehl entgegenzunehmen.

AUTO

Ein eventuell auf der Diskette gespeichertes AUTO-Kommando wird aufgehoben.

SYNTAX: B2

=====

WIRKUNG: Dieser Befehl aktiviert das sogenannte LEVEL II-BASIC im Speicherbereich zwischen 0000H und 2FFFH.

Das Betriebssystem ist damit abgeschaltet, alle G-DOS-Befehle, sowie die Erweiterungen, die das Disk-BASIC dem Benutzer bietet, können nicht mehr angesprochen werden und führen zu Fehlermeldungen.

Das Microsoft Level II-BASIC meldet sich mit der Bildschirmausgabe

```
BASIC ?           (Genie III)
READY?           (Genie I/II)
Mem-Size?       (Speedmaster)
```

Geben Sie hier nur NEW LINE bzw. ENTER ein, so wird kein reservierter Speicherbereich definiert, BASIC kann den gesamten Speicherplatz nutzen. Soll für bestimmte Unterroutinen ein Speicherbereich reserviert werden, so geben Sie nach dem Fragezeichen die Adresse (dezimal) der höchsten Speicherstelle an, die durch BASIC genutzt werden darf, und beenden die Eingabe durch NEW LINE.

Der Bereich oberhalb dieser Adresse bis zur Speicherstelle 65535 ist damit für Ihre Unterprogramme geschützt.

Daß dieser Bereich nun für Unterprogramme reserviert ist, ist durch die Eingabe PRINT MEM sofort feststellbar, da hier nur die Anzahl der freien Bytes angezeigt wird, die das BASIC selbst nutzen kann. Je tiefer also die Speicheradresse ist, die Sie nach dem Fragezeichen eingeben, desto weniger freier Speicherplatz wird bei PRINT MEM angezeigt und steht dem BASIC damit zur Verfügung.

ANWENDUNG: Laden und bearbeiten von Programmen, die in einen Speicherbereich laden, der sonst durch das Betriebssystem belegt wird.

BEMERKUNG: Um aus der Ebene des Level II-BASIC wieder in das Betriebssystem zu gelangen genügt bei Genie III die Eingabe von SYSTEM gefolgt von NEW LINE, auf die Anzeige von Stern und Fragezeichen werden dann ein Schrägstrich und eine Null eingegeben. Das Diskettenlaufwerk 0 läuft an und lädt das Betriebssystem in den Arbeitsspeicher.

Bei Benutzung des Speedmasters oder eines Genie I/II muß der SYSTEM-Eingabe der Befehl POKE 14316,254 vorausgehen.

Die einfachste Art das DOS neu zu starten ist natürlich die Betätigung der RESET-Taste(n).

BEISPIELE: Eingabe: B2

Das Betriebssystem wird verlassen und Level II-BASIC gestartet, welches sich mit der Abfrage nach der HIMEM-Grenze meldet.

Diese Meldung ist je nach Rechner unterschiedlich,

entweder BASIC ? (Genie III)
oder Mem-Size? (Speedmaster)
oder READY? (Genie I/II) .

Rücksprung in das Disketten-Betriebssystem bei Genie III:

Eingabe: SYSTEM (NEW LINE)
Meldung: *?
Eingabe: /0 (NEW LINE)

Das Diskettenlaufwerk läuft an und das Betriebssystem meldet sich nach erfolgreichem Laden in den Arbeitsspeicher des Rechners mit der Aufforderung zur Befehlseingabe.

Ist auf der Diskette ein AUTO-Kommando definiert, so gelangt dieses zur Ausführung.

Rücksprung in das Disketten-Betriebssystem bei Speedmaster / Genie I / Genie II:

Eingabe: POKE 14316,254 (NEW LINE)
Meldung: READY
EINGABE: SYSTEM (NEW LINE)
Meldung: *?
Eingabe: /0 (NEW LINE)

Der Ladevorgang des Betriebssystems beginnt wie auch bei Genie III beschrieben.

SYNTAX: BL Parameter

WIRKUNG: Mit diesem Befehl wird definiert, ob der Cursor auf dem Bildschirm periodisch blinken soll oder nicht. Die Eingabe BL J bewirkt, daß der Cursor blinkt, mit BL N wird das Blinken abgeschaltet.

BEMERKUNG: Mit der Systemoption BH kann definiert werden, ob der Cursor direkt nach dem Systemstart blinken soll oder nicht.

Das Kommando BL J arbeitet bei Genie III nur im 64-Zeichen-Modus, bei 80-Zeichen-Darstellung wird der Cursor ungeachtet eines BL-Befehles immer als feststehende Linie dargestellt.

SYNTAX: BOOT

=====

WIRKUNG: Der Befehl BOOT ermöglicht eine programmierbare Neu-Initialisierung (Warmstart) des Betriebssystems. Der aktuelle Bildschirminhalt wird dabei gelöscht, und falls ein AUTO-Kommando auf der Diskette gespeichert ist, kommt dieses nach der Initialisierung des Betriebssystems zur Ausführung.

ANWENDUNG: Neue Initialisierung des Betriebssystems ohne Hardware-RESET, Start eines über ein Programm abgeänderten AUTO-Kommandos, Zurücksetzung aller Betriebssystem-Parameter auf die mit den Systemoptionen vordefinierten Werte.

BEISPIELE: BOOT

Der Bildschirminhalt wird gelöscht, das Diskettenlaufwerk springt an und lädt das Betriebssystem. Alle vordefinierten Systemparameter sowie die physikalischen Daten der angeschlossenen Laufwerke (PD-Tabelle) werden wieder abgefragt und auf die vordefinierten Werte gesetzt.

```
100 LINEINPUT "Befehl für Autostart : ";S$
110 SC$ = "AUTO "+S$
120 CMD SC$
130 CMD"BOOT"
```

Bei diesem BASIC-Programm muß zunächst eine Befehlsfolge S\$ eingegeben werden, die beim späteren booten automatisch ausgeführt werden soll. In Zeile 110 wird dieser String dem AUTO-Befehl hinzugefügt und in Zeile 120 auf Diskette geschrieben. Mit dem Kommando "BOOT" in 130 wird das Programm beendet, ins Betriebssystem gesprungen, ein Warmstart ausgeführt, das DOS neu geladen und der AUTO-Befehl zur Ausführung gebracht.

SYNTAX: BREAK Parameter

WIRKUNG: Dieser Befehl gestattet es, die Funktion der BREAK-Taste innerhalb der Tastatur Ihres Rechners zu sperren oder zuzulassen.

BREAK J schaltet die Funktion der BREAK-Taste ein und BREAK N sperrt die Taste.

Nachdem die BREAK-Taste abgeschaltet ist, bleibt der Betriebszustand bis zur nächsten Aufforderung "Befehlseingabe:" erhalten, danach ist die Funktion definiert, die mit der Systemoption AG voreingestellt wurde.

ANWENDUNG: Die häufigste Anwendung dieses Befehls findet sich als DOS-Aufruf innerhalb von BASIC-Programmen, um für eine bestimmte Operation entgegen der durch den Betriebssystem-Parameter AG eingestellten Tasten-Blockade die Funktion der BREAK-Taste zeitweilig zuzulassen.

Umgekehrt kann es auch sinnvoll sein, für die Dauer der Abarbeitung eines wichtigen Programmteiles die BREAK-Taste zu sperren, um ungewolltem Datenverlust durch eine eventuelle Betätigung der Taste entgegenzuwirken.

BEMERKUNG: Innerhalb der BASIC-Programmierung sollten Sie sich in jedem Fall vergewissern, daß die Funktion der BREAK-Taste nicht gesperrt ist, bevor Sie das Auto-Kommando zur automatischen Erzeugung von Zeilennummern verwenden, da die Eingabe der Programmzeilen in diesem Fall nur durch die BREAK-Taste beendet werden kann.

BEISPIELE: Funktionsaufrufe aus dem Basic heraus:

```
10 CLS
20 CMD"BREAK N"
30 FOR I=1 TO 255
40 POKE 15360 + 4*I,I
50 NEXT I
60 RUN
```

Das Programm kann durch den Aufruf in Zeile 20 nicht mehr unterbrochen, da die Funktion der BREAK-Taste abgeschaltet wurde

```
10 CMD"BREAK N"  
20 CLS  
30 FOR I=1 TO 255  
40 POKE 15360 + 4*I,I  
50 NEXT I  
60 LINEINPUT"Passwort eingeben : ";PW$  
70 IF PW$ = "BREAKEIN" THEN CMD"BREAK J" : GOTO 20  
80 GOTO 20
```

In diesem Fall wird das Programm einmal abgearbeitet, danach fragt Zeile 60 nach dem Passwort zur Wieder-Aktivierung der BREAK-Taste. Wird das richtige Schlüsselwort eingegeben, so springt das Programm nach Zeile 20, kann aber ab sofort wieder unterbrochen werden. Wird der falsche Begriff eingegeben, so startet das Programm wieder bei Zeile 20, die BREAK-Taste bleibt jedoch abgeschaltet.

SYNTAX: DDE Dateiname

=====

ANWENDUNG: DDE (= Disketten-Daten-Editor) ermöglicht den direkten Zugriff auf einzelne Diskettensektoren, um diese zu überprüfen oder ggf. auch zu modifizieren.

Der Name einer zu untersuchenden Datei kann direkt mit dem DDE-Aufruf kombiniert werden, ist dies nicht der Fall, so meldet sich DDE zunächst mit einer Bildschirmmeldung und fragt dann nach dem Namen der zu untersuchenden Datei. Im Namen der Datei kann auch eine Laufwerksnummer enthalten sein.

Sofern die angesprochene Datei auf der Diskette gefunden wird, wird der erste Sektor dieser Datei auf dem Bildschirm dargestellt, wobei die Ausrichtung dem 64-Zeichen-Modus entspricht. (Im 80-Zeichen Modus ist DDE nicht lauffähig!)

Dargestellt werden jeweils die 256 Byte, die sich in diesem Sektor befinden, linksbündig mit einer Spalte, welche die Sektornummer sowie die relative Byteadresse angibt, die eigentliche Sektorinformation dann in hexadezimaler Darstellung (Mitte) und in ASCII-Darstellung auf der rechten Seite.

Folgende Tastaturanweisungen sind möglich:

; ... einen Sektor vor
- ... einen Sektor zurück
+ ... Sprung in den letzten Sektor der Datei
= ... Sprung in den ersten Sektor der Datei
M ... Modifizierung der Sektordaten
Sxx ... Sprung in Sektor Nummer xx (hex)

Die Eingabe von M zur Modifizierung schaltet den Cursor ein der als rechteckiges Graphikzeichen dargestellt wird.

Die CLEAR-Taste erlaubt die Umschaltung der Änderungsmöglichkeit vom Hexadezimal- zum ASCII-Feld und umgekehrt.

Die Pfeiltasten steuern die Cursorbewegung zur gewünschten Änderungsposition.

Die dort angezeigten Daten können direkt überschrieben werden, wobei im hexadezimalen Feld die Kleinbuchstaben (a bis f) automatisch in Großbuchstaben umgewandelt werden.

Mit der Eingabe von NEW LINE wird der Änderungsmodus wieder verlassen und die geänderten Daten werden auf den ursprünglich von der Diskette eingelesenen Sektor zurückgeschrieben.

ANWENDUNG: Wird innerhalb des Änderungsmodus die BREAK-Taste gedrückt, so werden alle eventuell schon gemachten Änderungen wieder zurückgesetzt und der Sektor bleibt auf der Diskette unmodifiziert.

Im Anzeigemodus bewirkt der Druck auf die BREAK-Taste einen Rücksprung zur Abfrage nach einem Dateinamen und ein weiterer Druck auf die BREAK-Taste hat das Ende der DDE-Routine mit Rücksprung in die Ebene, von der aus DDE aufgerufen wurde, zur Folge.

Soll DDE nicht auf eine bestimmte Datei zugreifen, so kann—auf die Anfrage nach dem Dateinamen auch eine Laufwerksangabe erfolgen, um sektorweise die Daten einer ganzen Diskette untersuchen bzw. verändern zu können. Dazu ist auf die Namensabfrage ein Doppelkreuz (über der 3 auf der Tastatur Ihres Rechners) sowie die Nummer des anzusprechenden Laufwerkes anzugeben.

Angezeigt wird daraufhin der erste Sektor auf dieser Diskette, von dort aus kann über die vorgenannten Steuerfunktionen "geblättert" werden.

BEMERKUNG: Der Aufruf von DDE ist sowohl vom DOS als auch von Mini-DOS und BASIC aus möglich, wobei bei letztgenannter Ebene die Kennzeichnung "CMD" den DOS-Aufruf aus dem BASIC kennzeichnen muß.

WARNUNG : DDE erlaubt den direkten Zugriff auf die Diskettendaten. Deshalb sollten Sie unter allen Umständen in der Anfangszeit für die Dauer Ihrer DDE-Anwendungen die Diskette mit einem Schreibschutz versehen, um nicht aus Versehen wichtige Daten zu zerstören oder gar eine Diskette vollkommen unbrauchbar zu machen.

Der Umgang mit DDE erfordert in den meisten Fällen fundierte Kenntnisse in der Programmierung, um sinnvolle Modifikationen von Daten direkt auf der Diskette vornehmen zu können.

Achten Sie außerdem darauf, daß die BREAK-Taste aktiv ist, bevor Sie DDE starten. Nur so können Sie bei versehentlich falschen Eingaben in der Modifizierung den ursprünglichen Zustand des eingelesenen Sektors erhalten.

BEISPIELE: DDE BASIC/CMD

Der erste Sektor der Datei BASIC/CMD wird gelesen und die Sektordaten werden in hexadezimal- sowie in ASCII-Darstellung auf dem Bildschirm angezeigt.

DDE

Die Routine DDE meldet sich mit der Frage nach dem Namen einer Datei, die untersucht werden soll. Beantworten Sie diese Frage mit einem Dateinamen oder mit der Spezifikation eines Laufwerkes, dessen Diskettendaten gelesen werden sollen.

Aus dem BASIC:

```
10 CLS
20 CMD"DDE INFORM/DAT:0"
30 PRINT "Ende der Datei-Untersuchung mit DDE !"
40 END
```

In Zeile 20 wird DDE als DOS-Aufruf aus dem BASIC aktiviert und ermöglicht Zugriffe auf die Datei INFORM/DAT. Nach Beendigung der Routine durch BREAK wird das Programm mit Zeile 30 fortgesetzt.

Soll das BASIC nach dem Rücksprung aus DDE verlassen werden und ein Rücksprung in das Betriebssystem erfolgen, so wäre Zeile 20 zu ändern in `CMD"S=DDE INFORM/DAT:0"`.

SYNTAX: CLS

WIRKUNG: Der aktuelle Bildschirminhalt wird gelöscht und der Cursor auf Position Null in der linken oberen Ecke des Bildschirms gesetzt.

ANWENDUNG: Übersichtliche Bildschirmausgabe eines nachfolgenden Programmteiles, ohne daß Teile des vorangegangenen Bildschirminhaltes weiter dargestellt werden.

Anwendungen z.B. bei einem Wechsel von einem Menue in ein anderes innerhalb eines Anwenderprogrammes.

BEMERKUNG: Der Befehl CLS ist neben dem Betriebssystem mit gleicher Wirkung auch innerhalb des BASIC vorhanden.

SYNTAX: CONT Parameter

=====

WIRKUNG: Dieses Kommando wirkt auf eine JOB-Datei, die aktiviert wurde und sich in der Bearbeitung befindet. Als Parameter ist nur ein Zeichen zulässig.

CONT J Dieser Befehl bedeutet, daß alle nachfolgend erwarteten Tastatureingaben bei einer JOB-Unterbrechung aus der JOB-Datei gelesen werden. Gelesen wird dabei ab der Stelle, an der die Verarbeitung der JOB-Datei unterbrochen wurde. Gleichzeitig bewirkt CONT J einen Übergang in die Ebene aus der die JOB-Datei aufgerufen wurde.

CONT J muß eingegeben werden, um ein CONT N wieder rückgängig zu machen.

CONT N Hier handelt es sich um die Umkehrung von CONT J. Nach einer Unterbrechung innerhalb einer aktiven JOB-Datei werden alle Eingaben wieder von der Tastatur erwartet. CONT N ist solange aktiv bis ein CONT J ausgeführt wird.

CONT D unterscheidet sich von CONT J nur insofern, daß sich bei einem DOS-Aufruf aus dem BASIC der Betriebszustand (DOS CALL) nach einer Unterbrechung der JOB-Datei nicht verändert. Das bedeutet, daß Eingaben, die nachfolgend aus der JOB-Datei gelesen werden, automatisch weiter als DOS-Aufrufe interpretiert und abgearbeitet werden. Wird die Befehlsfolge der JOB-Datei nach einer Unterbrechung durch eine Reihe von G-DOS-Befehlen fortgesetzt, so werden diese auch ohne ein vorangestelltes "CMD" weiterhin als Befehle des Betriebssystems behandelt. Wenn die Fortsetzung der aktiven JOB-Datei mit CONT J definiert worden wäre, würde der DOS-Call-Modus abgeschaltet und die nachfolgenden G-DOS-Befehle produzierten eine Kette von (BASIC)-Syntaxfehlern.

BEMERKUNG: Das Kommando CONT mit dem dazugehörigen Parameter ist als GDOS-Befehl nur innerhalb von JOB-Dateien zulässig, eine direkte Eingabe innerhalb des Betriebssystems ruft eine Fehlermeldung hervor.

HINWEIS: Auch im BASIC gibt es einen Befehl CONT, der dazu dient, die Verarbeitung eines durch den STOP-Befehl oder die Betätigung der Break-Taste unterbrochenes Programm fortzusetzen.

BEISPIELE: Nutzung des Befehls CONT innerhalb von JOB-Dateien:

Unter der Voraussetzung, daß die drei folgenden JOB-Dateien aus dem BASIC heraus (zum Beispiel mit dem Befehl CMD"DO Name) aufgerufen würden, wären die folgenden beschriebenen Wirkungen zu beobachten:

JOB-Datei 1 , Name = TEST1/JOB

```
-----  
1      CLS  
2      DIR 0  
3      -- INFO  
4      PAUSE  
5      CONT J  
6      LOAD"TEST1/BAS"  
7      RUN  
8      TCS Computer GmbH  
9      CMD"LIB"
```

BASIC-Programm TEST1/BAS zur JOB-Datei TEST1/JOB:

```
10 CLS:INPUT"Stringvariable :";A$:I=0  
20 FOR A= 1 TO 12: PRINT I,A$:I=I+68:NEXTA
```

Die Datei TEST1/JOB bearbeitet die ersten fünf Befehle als DOS-Aufrufe, nach der Unterbrechung durch das Kommando PAUSE erfolgt der Rücksprung aus dem DOS-Aufruf in das BASIC durch CONT J. Die Befehle 6 und 7 der JOB-Datei werden deshalb auch wieder als BASIC-Befehle interpretiert und dementsprechend richtig ausgeführt.

Da mit CONT J alle Tastatureingaben bei einer Unterbrechung (z.B. während des Ablaufs eines BASIC-Programms) aus der JOB-Datei erfolgen, holt sich der Rechner die in Zeile 10 des BASIC-Programms verlangte String-Variable A\$ aus der JOB-Datei und führt dann das Programm fort. Nach Beendigung des Programms endet der Ablauf der JOB-Datei mit dem Aufruf der DOS-Befehlsliste. Das CMD" ist notwendig, da sich der Rechner immer noch in der BASIC-Ebene befindet.

Diese Job-Datei läuft nur dann fehlerfrei, wenn sie über CMD"DO TEST1/JOB aus dem BASIC aufgerufen wird. Ein Aufruf dieser Datei aus der DOS-Befehlsebene, hätte ihren Abbruch mit dem BASIC-Befehl LOAD"TEST1/BAS" zur Folge.

JOB-Datei 2 , Name = TEST2/JOB

```
1      CLS
2      DIR 0
3      INFO
4      PAUSE
5      CONT J
6      LOAD"TEST2/BAS"
7      RUN
8      TCS Computer GmbH
9      CMD"CONT D
10     LIB
```

Das BASIC-Programm TEST2/BAS ist identisch mit dem Programm TEST1/BAS.

Die Datei TEST2/JOB arbeitet bis einschließlich Punkt 8 wie die JOB-Datei TEST1/JOB. Das Kommando CMD"CONT D"bewirkt jedoch einen Rücksprung in die DOS-Ebene, sodaß der nachfolgende Befehle LIB wieder ohne vorangestelltes CMD" ausgeführt wird.

JOB-Datei 3 , Name = TEST3/JOB

```
1      CLS
2      DIR 0
3      INFO
4      PAUSE
5      CONT J
6      LOAD"TEST3/BAS",R
7      CMD"CONT N
8      CMD"PORT
9      CMD"LIB
```

BASIC-Programm TEST3/BAS zur JOB-Datei TEST3/JOB:

```
10 CLS:INPUT"Stringvariable: ";A$:I=0
20 FOR A= 1 TO 12: PRINT I,AI=I+68:NEXTA
30 CMD"CONT J
```

Wie die Datei TEST1/JOB bearbeitet auch TEST3/JOB die ersten fünf Befehle als DOS-Aufrufe. Nach dem Kommando PAUSE erfolgt der Rücksprung ins BASIC durch CONT J mit anschließendem Laden von TEST3/BAS. Der Befehl CMD"CONT N" bewirkt danach jedoch einen vorläufigen Abbruch der JOB-Ausführung, wobei die Abbruchstelle markiert und eigene Eingaben über die Tastatur ermöglicht werden.

Starten Sie nun das Programm TEST3/BAS mit RUN und geben Sie eine beliebige Stringvariable ein. Das Programm endet mit dem Kommando CMD"CONT J", wodurch wieder die Umschaltung auf die JOB-Datei ausgelöst wird deren weitere Ausführung hinter der Markierung beginnt (CMD"PORT" und CMD"LIB").

Der COPY-Befehl:

Lassen Sie uns der Beschreibung des COPY-Befehls zwei Nachrichten vorausschicken, eine gute und eine schlechte:

Zunächst die gute Nachricht:

Das COPY-Kommando des G-DOS bietet Ihnen durch eine Fülle von möglichen Optionen für die verschiedensten Anwendungen einen Komfort, wie er in kaum einem anderen Betriebssystem verwirklicht ist.

Und daraus resultiert auch gleich die schlechte Nachricht:

Die Fülle der möglichen COPY-Funktionen macht es notwendig, daß Sie sich intensiv mit diesem Teil des Handbuches vertraut machen.

Erfahrungsgemäß ist gerade der Umgang mit den Kopier Routinen des Betriebssystems für den Lernenden mit größeren Schwierigkeiten verbunden, da die Fülle der möglichen Parameter (von denen sich einige auch noch gegenseitig ausschließen) nicht mehr so leicht zu überblicken ist.

Zur Eingabe von Kopierkommandos werden folgende Voraussetzungen definiert:

Ein Dateiname besteht jeweils aus dem eigentlichen Namen, der bis zu acht Zeichen lang sein kann, wobei das erste Zeichen in jedem Fall ein Buchstabe sein muß.

Die Eingabe kann in Groß- oder Kleinbuchstaben erfolgen, alle Eingaben werden jedoch in der weiteren Verarbeitung vom Betriebssystem automatisch in Großbuchstaben konvertiert.

An einen Namen kann noch eine Namens Erweiterung (Extension) angefügt werden, welche durch einen Schrägstrich vom eigentlichen Namen abzutrennen ist. Durch diese Erweiterung können z.B. Dateien gekennzeichnet werden, die eine gemeinsame Eigenschaft aufweisen.

Das Betriebssystem erkennt zwei Extensions als Default-Werte, wenn ein Dateiname angegeben wird, nämlich die Erweiterungen /CMD und /JOB.

Wird innerhalb der DOS-Ebene ein Dateiname ohne Extension eingegeben, um ein Programm aufzurufen, so geht G-DOS automatisch davon aus, daß der Dateiname die Erweiterung /CMD aufweist.

Bei einem DO-Kommando wird automatisch die Namens Erweiterung /JOB vorausgesetzt, auch wenn diese nicht ausdrücklich mit angegeben wurde.

Soll eine Datei durch ein Passwort geschützt werden, so wird im Anschluß an den Dateinamen ein Punkt eingegeben, danach dann das Passwort mit bis zu acht alphanumerischen Zeichen, wobei das erste Zeichen wiederum ein Buchstabe sein muß.

Durch einen Doppelpunkt abgetrennt kann zudem die Nummer des Laufwerkes angegeben werden, in der sich die für eine Schreib- oder Leseoperation vorgesehene Diskette befindet.

Wird auf die Festlegung einer Laufwerknummer verzichtet, so werden die Inhaltsverzeichnisse der Disketten in den angeschlossenen Laufwerken der Reihe nach auf das Vorkommen des angegebenen Dateinamen hin durchsucht.

Wird der Name dabei nicht gefunden, wird bei Leseoperationen eine entsprechende Fehlermeldung ausgegeben, oder aber bei Schreiboperationen die Datei neu angelegt.

Alle Kopier Routinen werden sehr stark von den Systemoptionen beeinflusst, die mit dem S-Befehl zu verändern sind. Zum besseren Verständnis des COPY-Kapitels sollten Sie sich deshalb erst einmal mit der Bedeutung dieser Optionen vertraut machen.

Werden mit einem Kommando mehrere Parameter übergeben, so werden diese Parameter jeweils durch ein Komma oder ein Leerzeichen voneinander abgetrennt.

SYNTAX 1: COPY DATEIALT/TYP:QD DATEINEU/TYP:ZD Parameter

=====

WIRKUNG: Diese Eingabe des COPY-Kommandos erlaubt die Kopie einer einzelnen Datei (DATEIALT/TYP) auf Laufwerk QD auf die Diskette in Laufwerk ZD, wo die Datei den Namen DATEINEU/TYP erhält.

Die Zieldiskette in Laufwerk ZD muß vor der Ausführung dieses Befehles bereits formatiert sein, damit die zu kopierenden Daten auf diese Diskette geschrieben werden können.

Ist die Datei DATEIALT/TYP durch ein Kennwort geschützt, so ist dieses (durch einen Punkt abgetrennt) direkt nach dem Dateinamen anzufügen.

Soll die Datei auf der Zieldiskette den gleichen Namen tragen wie das Original auf der Quelldiskette, so genügt die einmalige Nennung des Dateinamens innerhalb des COPY-Kommandos.

Auch dabei ist die Laufwerknummer der Zieldiskette allerdings durch ein Leerzeichen (oder Komma) und einen Doppelpunkt getrennt einzugeben.

Die Eingabe COPY DATEIALT/TYP:QD :ZD würde in diesem Fall zur Übertragung des Files DATEIALT/TYP auf die Zieldiskette ausreichen.

Falls die innerhalb der PD-Tabelle festgelegten physikalischen Daten für die angeschlossenen Laufwerke von den Formatierungen der an dem Kopiervorgang beteiligten Disketten abweichen, so kann für diesen einen Kopiervorgang auch angegeben werden, auf welche PD-Daten das Betriebssystem zur Ausführung dieses Kommandos zurückgreifen soll.

Zu diesem Zweck können dem COPY-Kommando (wiederum durch ein Leerzeichen oder Komma abgetrennt) die Parameter QPDN=X und (oder) ZPDN=X angefügt werden. X kann dabei einen Wert zwischen 0 und 9 annehmen und definiert die Zeilennummer innerhalb der PD-Tabelle, aus der die physikalischen Daten von Quelldiskette (QPDN) und Zieldiskette (ZPDN) entnommen werden sollen.

Auf diese Art kann z.B. auf einem 80-Track Doppellaufwerk eine Dateikopie von einer doppelseitig beschriebenen 80-Track-Diskette auf eine einseitig beschriebene 40-Track-Diskette erfolgen, ohne daß die innerhalb der PD-Tabelle festgelegten Daten der angeschlossenen Laufwerke vorher geändert werden müßten.

Die Nennung von QPDN und ZPDN ist nur im Zusammenhang mit dem einen auszuführenden COPY-Kommando relevant, danach geht das Betriebssystem wieder von den innerhalb der PD-Tabelle definierten Daten für die Laufwerke QD und ZD aus.

BEISPIELE: COPY TEST/DAT:0 :1

Die Datei TEST/DAT, die sich auf der Diskette in Laufwerk Null befindet, wird mit gleichem Namen auf die Diskette in Laufwerk Eins kopiert.

COPY GENIETST/CMD:0 GUINNESS/CMD:2

Das Programm GENIETST/CMD in Laufwerk Null wird auf die Diskette in Laufwerk 2 kopiert, wo es unter dem Namen GUINNESS/CMD in das Inhaltsverzeichnis eingetragen wird.

COPY DEMO/BAS:1 DEMONEU/BAS:3 ZPDN=5

Das Programm DEMO/BAS wird von der Diskette in Laufwerk 1 auf die Diskette in Laufwerk 3 kopiert, wo es den Namen DEMONEU/BAS erhält. Die Formatierung der Zieldiskette entspricht den physikalischen Daten, die innerhalb der PD-Tabelle unter Zeile Nr. 5 definiert sind.

COPY TESTPROG/CMD:0 /OBJ:1

Die Datei TESTPROG/CMD wird von Laufwerk Null auf Laufwerk 1 kopiert, wo sie den Namen TESTPROG/OBJ erhält.

COPY MYTAPE/CMD:0 :2 QPDN=3 ZPDN=3

Die Datei MYTAPE/CMD wird von Laufwerk 0 auf Laufwerk 2 kopiert. Da für das Systemlaufwerk 0 und für die Zieldiskette in Laufwerk 1 abweichende PD-Daten definiert wurden, fordert das Betriebssystem nach Eingabe dieses Kommandos zum Diskettenwechsel auf. Die physikalischen Daten der beteiligten Disketten sollen den unter Zeile Nr. 3 der PD-Tabelle festgelegten Parametern entsprechen.

SYNTAX 2: COPY \$DATEIALT/TYP:QD DATEINEU/TYP:ZD Parameter

=====

WIRKUNG: In dieser Form ist die Auswirkung des Kopierkommandos fast identisch zur Arbeitsweise der SYNTAX 1.

Das dem Dateinamen vorangestellte Dollarzeichen bewirkt allerdings, daß das Betriebssystem davon ausgeht, daß ohne G-DOS kopiert werden soll und deshalb zum Diskettenwechsel auffordert.

ANWENDUNG: Kopie von Dateien, die von einer reinen Datendiskette auf eine andere Diskette übertragen werden sollen, ohne daß die Quelldiskette ein G-DOS-Betriebssystem beinhaltet.

SYNTAX 3: COPY LW DATEIALT/TYP DATEINEU/TYP

=====

WIRKUNG: In dieser Schreibweise wird dem Betriebssystem mitgeteilt, daß für die Kopie nur ein Diskettenlaufwerk (LW) als Quell- und Ziellaufwerk verwendet wird.

Zusätzlich können abweichend von den unter der angegebenen Laufwerknummer definierten PD-Daten durch die Parameter QPDN=x und ZPDN=y die physikalischen Daten von Quell- und Zieldiskette angegeben werden, wobei x und y die Zeile innerhalb der PD-Tabelle kennzeichnen, in der die Daten der beiden beteiligten Disketten stehen.

Soll für die Diskettenkopie nur das Laufwerk Null verwendet werden, so müssen sowohl Quell- als auch Zieldiskette ein einwandfreies G-DOS-Betriebssystem enthalten, da der Rechner sich sonst durch Zugriffsversuche auf nicht vorhandene Dateien aufhängen und über RESET neu gestartet werden müßte.

ANWENDUNG: Kopie von Dateien unter Verwendung nur eines Diskettenlaufwerkes, wobei in der Regel nicht das Laufwerk Null zur Kopie mit dieser Syntax herangezogen wird.

SYNTAX 4: COPY LW \$DATEIALT/TYP DATEINEU/TYP Parameter

=====

WIRKUNG: Diese Form des Kopierkommandos entspricht in der Wirkungsweise der Syntax 3, die zu kopierenden Dateien befinden sich hierbei jedoch auf Disketten, die ein anderes Format aufweisen oder überhaupt kein G-DOS-Betriebssystem enthalten.

Da das Betriebssystem bei dieser Kopieroutine mehrfach zum Diskettenwechsel auffordert, ist diese Form nur dann sinnvoll, wenn das Laufwerk Null für die Diskettenkopie verwendet werden soll (LW =0). Andernfalls wäre der Zeitaufwand für die Dateikopie unnötig groß.

Von den PD-Daten abweichende Diskettenformate der an der Kopie beteiligten Disketten können auch hier über die Parameter QPDN=x und ZPDN=y definiert werden.

ANWENDUNG: Kopie von Dateien, die sich auf Disketten ohne Betriebssystem befinden, auf Laufwerk Null.

SYNTAX 5: COPY QLW ZLW Parameter

WIRKUNG : Diese Form des Kopierkommandos wird für die Duplizierung kompletter Disketten verwendet.

QLW definiert dabei das Laufwerk, in dem sich die Quelldiskette befindet, deren Inhalt auf die in Laufwerk ZLW befindliche Zieldiskette kopiert werden soll.

Wird das Kommando ohne weitere Parameter eingegeben, so geht das Betriebssystem davon aus, daß die Formate von Quell- und Zieldiskette den in der PD-Tabelle festgelegten Daten entsprechen.

Der Benutzer wird gefragt, ob die Zieldiskette zunächst formatiert werden soll. Die entsprechende Frage ist mit J für Formatierung oder mit N für Nicht-Formatierung zu beantworten.

Im folgenden fragt G-DOS, ob die Systemdiskette, von der aus die Kopierroutine aufgerufen wurde, als Quell- oder Zieldiskette an der Kopieraktion beteiligt ist.

Auch diese Frage ist mit J oder N zu beantworten, das Betriebssystem fordert danach zum Einlegen der entsprechenden Quell- oder Zieldisketten auf, die in die angeforderten Laufwerke eingelegt und durch die Betätigung der NEW LINE-Taste bestätigt werden.

Beachten Sie genauestens die Anweisungen des Betriebssystems, damit sich auch wirklich die angeforderten Disketten in den richtigen Laufwerken befinden, weil es durch Fehlbedienungen zum Systemabsturz oder zum eventuellen Verlust wichtiger Daten kommen kann !

Neben dieser Grundform des COPY-Kommandos kann eine Fülle von Parametern definiert werden, um die der eigentlichen Diskettenkopie vorgeschalteten Abfragen zu unterdrücken oder der neuerstellten Kopie verschiedenen Attribute zuzuweisen.

Mögliche Parameter und ihre Bedeutung:

- TT.MM.JJ** Datumseingabe im Format Tag (zweistellig), Monat (zweistellig) und Jahr (zweistellig), jeweils durch einen Dezimalpunkt voneinander abgetrennt. Das angegebene Datum wird als Diskettendatum auf der Zieldiskette eingetragen. Wird auf die Datumsparameter verzichtet, so trägt das Betriebssystem automatisch das interne Datum des Rechners als Diskettendatum ein.
- J** - JA, in jedem Fall kopieren -
Die Diskettenkopie soll in jedem Fall durchgeführt werden, auch dann, wenn sich auf der Zieldiskette für das Betriebssystem lesbare Daten befinden. Der Parameter J darf nicht in Verbindung mit den Parametern N, AZN=, AZKW=, ZZND, BZN und BZD verwendet werden.
- N** - NEIN, nur wenn neue Diskette verwendet wird -
Die Diskettenkopie wird abgebrochen, falls sich auf der Zieldiskette für das Betriebssystem lesbare Daten befinden. Dieser Parameter kann nicht in Verbindung mit den Parametern J, AZN=, AZKW=, ZZND, BZN und BZD verwendet werden.
- FMT** - Zieldiskette formatieren -
Vor dem eigentlichen Kopiervorgang wird die Zieldiskette formatiert, die entsprechende Abfrage (Formatieren J/N) durch das Betriebssystem entfällt damit.
- NFMT** - Zieldiskette nicht formatieren -
Das Betriebssystem soll davon ausgehen, daß die Zieldiskette bereits formatiert ist und die Diskettenkopie direkt ausgeführt werden kann. Auch hier entfällt die entsprechende Abfrage.
- KDWA** - Keinen Diskettenwechsel Abwarten -
Die Kopieraktion soll sofort beginnen, ohne daß das Betriebssystem weitere Abfragen vorschaltet. Quell- und Zieldiskette müssen sich zum Zeitpunkt dieses Kommandos bereits in den entsprechenden Laufwerken befinden.
Wird nur der Parameter KDWA definiert (ohne FMT oder NFMT), so geht das Betriebssystem davon aus, daß die Zieldiskette formatiert werden soll.
- Tritt während des Kopiervorganges ein Schreib- oder Lesefehler auf, so wird bei gesetztem Parameter KDWA die Aktion mit einer entsprechenden Fehlermeldung abgebrochen. Die Unterbrechung eines Kopiervorganges durch Betätigung der Hochpfeil-Taste ist nicht möglich.

- QKW=xxx - Quelldisketten-Kennwort=xxx -
Das Zugriffskennwort der zu kopierenden Diskette wird mit diesem Parameter angegeben, wenn die Systemparameter AA=J und AR=N gesetzt sind. Wird das Zugriffskennwort fehlerhaft eingegeben, so ist eine Kopie der entsprechenden Diskette nicht möglich.
- NZKW=yyy - Neues Zieldisketten-Kennwort=yyy -
Mit der Diskettenkopie wird der Zieldiskette direkt ein Zugriffskennwort zugeordnet. Diese Angabe und die Verwendung des Parameters IVU schließen sich gegenseitig aus.
- ZZND - Zeige Zieldiskettennamen und -datum -
Das Betriebssystem soll den Diskettennamen sowie das auf einer formatierten Zieldiskette enthaltene Datum auf dem Bildschirm anzeigen. Dieses ermöglicht dem Anwender, sich noch einmal davon zu überzeugen, ob die in das Ziellaufwerk eingelegte Diskette auch die richtige ist.
Nach der Anzeige dieser Daten kann der Anwender über Fortfahren oder Abbruch entscheiden.
- AZN=xxx - Alter Zieldisketten-Name -
Wenn der auf der Zieldiskette eingetragene Name nicht mit dem als xxx angegebenen Altnamen übereinstimmt, fragt das Betriebssystem den Anwender, ob die Kopieraktion fortgesetzt oder abgebrochen werden soll.
- BZN - Behalte Ziel-Namen -
Die Zieldiskette behält den alten Namen, der in ihrem Inhaltsverzeichnis eingetragen war. Die Verwendung von BZN schließt die gleichzeitige Benutzung der Parameter J, N, IVU und NZN aus.
- QN=yyy - Quelldisketten-Name=yyy -
Der Name der Quelldiskette wird mit dem als yyy angegebenen Namen verglichen. Wird bei diesem Vergleich keine Übereinstimmung festgestellt, so fragt das Betriebssystem den Anwender, ob die Diskettenkopie fortgesetzt oder abgebrochen werden soll.
- NZN=zzz - Neuer Zieldisketten-Name=zzz -
Der als zzz definierte Name wird als Diskettenname auf der Zieldiskette eingetragen. Die gleichzeitige Verwendung der Parameter BZN oder IVU ist ausgeschlossen.
- BZD - Behalte Zieldisketten-Datum -
Das im Inhaltsverzeichnis der Zieldiskette bereits eingetragene Datum wird nicht verändert bzw. durch ein neues Systemdatum überschrieben. Die gleichzeitige Verwendung der Parameter J, N, SQD und IVU ist ausgeschlossen.

- SQD - Schreibe Quelldisketten-Datum -
Das Datum, welches auf der Quelldiskette eingetragen ist, wird mit der Kopie auch auf die Zieldiskette übertragen. Die gleichzeitige Verwendung der Parameter BZD oder IVU ist ausgeschlossen.
- IVU - Inhaltsverzeichnis unverändert -
Das Inhaltsverzeichnis (INHALT/SYS) der Zieldiskette soll nicht - wie bei der Diskettenkopie normalerweise üblich - durch das Betriebssystem neu angelegt werden, sondern ist unverändert zu lassen. Die Datei GDOS/SYS wird unverändert direkt von der Quelldiskette übernommen, ohne daß Daten der PD-Spezifikation geändert würden.
- Die Verwendung dieses Parameters ist z.B. dann sinnvoll, wenn Daten im Inhaltsverzeichnis der Zieldiskette unlesbar geworden sind oder aber das Inhaltsverzeichnis geKILLte, aber noch nicht gelöschte Dateien enthält, die noch wiederhergestellt werden sollen.
Die gleichzeitige Verwendung der Parameter BZN, NZN, NZKW und SQD ist ausgeschlossen.
- QPDN=xx - Quelldisketten-PD-Nummer=xx -
Die physikalischen Daten der Quelldiskette entsprechen den unter Zeile xx der PD-Tabelle definierten Werten.
- ZPDN=yy - Zieldisketten-PD-Nummer=yy -
Die physikalischen Daten der Zieldiskette entsprechen den unter Zeile yy der PD-Tabelle definierten Werten.

SYNTAX 6: COPY QLW# ZLW# Parameter

WIRKUNG: Bis auf IVU sind die Parameter der Syntax 5 auch hier verwendbar, zusätzlich ist es aber möglich, sich durch Einzeldateikopien mit der Abfrage der zu kopierenden Dateien durch das Betriebssystem das Kopieren größerer Mengen von Dateien zu erleichtern. Weitere Parameter erlauben, automatisch alle Dateien zu kopieren, die bestimmten vom Anwender definierten Spezifikationen genügen.

Die Parameter lauten im einzelnen:

EDK

- Einzel-Datei-Kopie -
Dem Betriebssystem wird mitgeteilt, daß die Diskettenkopie dateiweise erfolgen soll, es wird jeweils eine Datei in den Arbeitsspeicher eingelesen und danach auf die Zieldiskette geschrieben. Dateien, die im Laufe der Zeit erweitert worden sind und sich physikalisch auf verschiedene Positionen auf der Diskette verteilt haben, werden so wieder aneinandergesetzt, wenn der freie Speicherplatz auf der Zieldiskette dies erlaubt. Dieses ist für Schreib-/Leseoperationen von Vorteil, da der Kopf des Diskettenlaufwerkes die zu verarbeitenden Daten nicht an verschiedenen Stellen "sammeln" muß.
Der Parameter EDK ist für Kopien in Syntax 6 notwendig und wird nur dort benutzt.

FRAG

- Anwender fragen, ob Datei kopiert werden soll -
Die im Inhaltsverzeichnis der Quelldiskette aufgeführten Namenseinträge werden der Reihe nach auf dem Bildschirm angezeigt. Das Betriebssystem fragt zu jeder Datei, ob sie kopiert werden soll oder nicht (Datei kopieren (J/N/A/W) ?)
Dabei stehen als mögliche Antworten neben J (Ja) und N (Nein) noch A und W zur Verfügung, wobei A den Abbruch der Dateiabfrage bewirkt und die Kopie der Dateien startet, für die die Kopierabfrage bis dahin mit J beantwortet worden ist.
Hat der Anwender sich in der Beantwortung einer Frage geirrt, so kann die komplette Routine durch Eingabe des W für Wiederholung neu gestartet werden. Soll die Kopieroutine gänzlich abgebrochen werden, so ist auch bei dieser Syntax der Hochpfeil zu drücken und das Betriebssystem kehrt nach entsprechender Meldung wieder in die Kommandoebene zurück.

FRD

- Freie Dateien -
Der Kopierbefehl bezieht sich nur auf Anwenderdateien, unsichtbare Files sowie Systemdateien werden nicht kopiert.

- /TYP** - nur Dateien dieses TYPs -
Kopiert werden nur die Dateien, die als Namens-erweiterungen die unter /TYP angegebene Zeichenfolge aufweisen.
- BEA** - Bearbeitete Dateien -
Alle Dateien, deren Bearbeitungskennzeichen gesetzt ist, werden kopiert. Dieses Kennzeichen wird mit jedem Schreibzugriff auf einen Sektor der entsprechenden Datei gesetzt und kann nur über die Befehle PROT oder ATTRIB zurückgesetzt werden.
In Verbindung mit dem Parameter BEA ist es möglich, automatisch alle Dateien kopieren zu lassen, die seit dem letzten Löschen aller Bearbeitungskennzeichen aktualisiert worden sind.
- NVD** - Nur vorhandenen Dateien -
Es werden nur die Dateien kopiert, die im Inhaltsverzeichnis der Zieldiskette bereits eingetragen sind. Ist eine Datei dort nicht aufgeführt, so wird sie auch bei der Kopie nicht mit übertragen.
Eine derartige Überprüfung wird in der Regel bei der Arbeit mit Datensicherungs-Disketten herangezogen.
- FMT** - Zieldiskette formatieren -
Bevor der Kopiervorgang beginnt, wird die Zieldiskette zunächst formatiert. Soll die Zieldiskette ein anderes Format erhalten, muß in der Syntax 6 der Parameter FMT in jedem Fall angegeben sein, da die G-DOS-Systemdateien auf diese Weise an der richtigen Stelle im Inhaltsverzeichnis der Zieldiskette eingetragen werden
- IDL=Dateiliste/IDL** - inclusive Dateien in Liste -
Das Betriebssystem gestattet es, die Dateien, die bei einem Kopiervorgang berücksichtigt werden sollen, in einer Datei zusammenzufassen. Diese Datei erhält die Namens-erweiterung /IDL und wird auf der Diskette gespeichert.
- Die einzelnen Dateinamen werden dabei in sequentieller Form in die Datei geschrieben und enden jeweils mit einem Zeichen für Zeilenende (0D Hex).
Zusätzlich dürfen Kommentarzeilen in diese Dateien eingebunden werden, die mit einem Semikolon (;) beginnen müssen.
Zur Erstellung einer /IDL-Datei wird diese zunächst über CREATE erzeugt, danach können die einzelnen Einträge mit JOB/CMD oder einem kurzen BASIC-Programm in diese Datei eingetragen werden.
Nach dem letzten Eintrag innerhalb der Datei dürfen nur noch Nullen folgen. Wird der Kopiervorgang mit dem /IDL-Parameter gestartet, liest das Betriebssystem die Namen der zu kopierenden Dateien der Reihe nach aus dem IDL-File und überträgt nur die dort aufgeführten Dateien auf die Zieldiskette.

XDL=Dateiliste/XDL

- exclusive Dateien in Liste -

Eine Datei mit der Namensweiterung /XDL wird erzeugt (wie bei IDL), in der diejenigen Dateien aufgeführt werden, die vom Kopiervorgang ausgenommen werden sollen. Bei der Diskettenkopie werden nur Dateien übertragen, die innerhalb der /XDL-Datei nicht aufgeführt sind.

AZKW=Kennwort

Ist der Parameter NFMT gesetzt, während die Kennwortkontrollen aktiviert sind (siehe Systemoptionen), so wird das Kennwort der Zieldiskette mit diesem Parameter angegeben. Stimmt diese Angabe nicht mit dem Kennwort der Zieldiskette überein, so wird der Kopiervorgang abgebrochen.

SBIV=xx

Bei der Formatierung der Zieldiskette wird das Inhaltsverzeichnis beginnend mit dem ersten Sektor des durch xx festgelegten Blocks eingetragen. Der Startblock des Inhaltsverzeichnisses kann auf diese Weise abweichend von dem innerhalb der PD-Tabelle definierten SBIV-Parameter positioniert werden. Die gleichzeitige Verwendung des Parameters NFMT ist unzulässig.

AEIV=yy

Für das Inhaltsverzeichnis der neuen Diskette wird bei der Formatierung die Anzahl von yy Einheiten bereitgestellt.

Damit kann auch dieser Parameter abweichend von dem vordefinierten PD-Tabellenwert AEIV angegeben werden. Dem Inhaltsverzeichnis müssen mindestens zwei Einheiten zugeordnet werden, der Maximalwert für yy ist sechs.

Auch bei der Verwendung dieses Parameters ist die gleichzeitige Angabe des Parameters NFMT ausgeschlossen.

SYNTAX: CREATE Dateiname Parameter

WIRKUNG: Dieser Befehl gestattet dem Anwender, eine Datei neu anzulegen, wobei von vornherein festgelegt werden kann, wieviel Platz für die Datei auf der Diskette reserviert werden soll.

Ist auf der Diskette bereits eine Datei unter dem angegebenen Namen vorhanden, so wird diese mit der Ausführung des Create-Befehls unwiederbringlich gelöscht. Der Anwender ist deshalb aufgefordert, seine bestehenden Dateien durch Kennworte zu schützen!

Als Parameter können folgende Begriffe verwendet werden:

LOG=xxx Dieser Parameter legt die Länge eines einzelnen logischen Datensatzes innerhalb der anzulegenden Datei fest. Dieser Wert muß zwischen 0 (=256) und 255 liegen. Wird mit dem Create-Befehl auf die Festlegung des LOG-Parameters verzichtet, so wird als Länge des einzelnen Datensatzes automatisch 256 angenommen.

ANZ=xxx Festlegung der Anzahl von logischen Datensätzen, die mit dem Create-Befehl anfangs angelegt werden sollen.

ADE=z Dieser Parameter legt fest, ob später eine automatische Dateierweiterung möglich sein soll oder nicht. Als Eingabe kann ADE=J für mögliche Dateierweiterungen oder andernfalls ADE=N angegeben werden.

ADF=z Dieser Parameter erlaubt dem Betriebssystem die automatische Freigabe des nichtbelegten Dateiplatzes durchzuführen (ADF=J) oder er unterbindet diese Möglichkeit (ADF=N).

BEISPIELE: CREATE TEST/DAT:1 LOG=100 ANZ=50

Auf der Diskette in Laufwerk 1 wird die Datei TEST/DAT angelegt, und zwar mit 50 logischen Datensätzen zu je 100 Zeichen Länge. Falls auf der Zieldiskette bereits eine Datei unter dem Namen TEST/DAT vorhanden war, so wird deren Inhalt unwiederbringlich gelöscht.

CREATE INFO/DAT:0 ANZ=20 ADF=N ADE=N

Auf der Diskette in Laufwerk Null wird die Datei INFO/DAT mit einer Anzahl von 20 logischen Datensätzen angelegt, wobei jeder einzelne Datensatz eine Länge von einem Sektor (= 256 Zeichen) auf der Diskette belegt. Automatische Dateiplatzfreigabe sowie Dateierweiterung werden ausgeschlossen.

SYNTAX: DATUM

oder

DATUM tt.mm.jj

WIRKUNG: Wird nur das Befehlswort DATUM eingegeben, so erfolgt die Anzeige des aktuellen Systemdatums auf dem Bildschirm.

Werden als Parameter tt.mm.jj (Tag.Monat.Jahr) mit angegeben, so wird das Systemdatum auf diese eingegebenen Werte gesetzt und (bei GENIE III) als laufendes Datum von der eingebauten Echtzeituhr weitergeführt.

BEISPIELE: DATUM

Das aktuelle Systemdatum wird auf dem Bildschirm angezeigt.

DATUM 10.07.84

Das Systemdatum wird auf den 10. Juli 1984 gesetzt.

SYNTAX: DIR Parameter

WIRKUNG: Das Kommando DIR entspricht dem Kommando I und bewirkt die Anzeige des Inhaltsverzeichnisses einer Diskette.

Je nach Art der mit diesem Befehl kombinierten Parameter läßt sich der Umfang der angezeigten Informationen festlegen.

Als Parameter kann zunächst einmal die Laufwerknummer angegeben werden, für die das Inhaltsverzeichnis angezeigt werden soll. Falls diese Angabe entfällt, wird das Inhaltsverzeichnis derjenigen Diskettenstation angezeigt, die innerhalb der Systemdaten des Betriebssystems unter dem Parameter AN als Standardwert angegeben ist.

Zusätzlich dürfen als Parameter die Angaben S,I,A,B,P und /TYP angegeben werden.

Dabei bedeutet S, daß auch die Systemdateien des Betriebssystems mit der Namensweiterung /SYS angezeigt werden.

Der Parameter I erlaubt auch die Anzeige der unsichtbaren Dateien innerhalb des Inhaltsverzeichnisses.

Wird B als Parameter ergänzt, so werden nur die Dateien angezeigt, deren Bearbeitungskennzeichen gesetzt ist.

P bewirkt, daß die Ausgabe des Inhaltsverzeichnisses über einen angeschlossenen Drucker ausgegeben wird.

Durch die Kennzeichnung des /TYP mit dem DIR-Kommando ist es möglich, nur die Dateien anzeigen zu lassen, deren Namensweiterung aus dem als /TYP angegebenen Begriff besteht.

A erlaubt es, neben den Namenseinträgen noch eine Reihe von Zusatzinformationen über die auf der angesprochenen Diskette befindlichen Dateien anzeigen zu lassen.

Dieses sind

| | |
|--------|---|
| Ende: | Satznummer/Byte des Dateiendes |
| Log: | Länge des logischen Satzes |
| Anz.: | Anzahl der Sätze in der Datei |
| Einh.: | Anzahl der durch die Datei belegten Einheiten |
| Erw.: | Anzahl der Dateierweiterungen |

Zusätzlich werden bei der A-Option zum DIR-Befehl über einzelne Kennbuchstaben noch folgende Informationen angezeigt:

1. S - Bei der Datei handelt es sich um ein Systemprogramm
2. I - Die Datei ist innerhalb des Inhaltsverzeichnisses unsichtbar gemacht worden
3. B - Das Bearbeitungskennzeichen der Datei ist gesetzt
4. E - Eine Erweiterung der Datei wurde abgeschlossen
5. F - Eine Freigabe des Dateiplatzes wurde abgeschlossen
6. H - Ein Hauptkennwort ist definiert
7. B - Ein Bearbeitungskennwort existiert
8. S - Stufe des zu dieser Datei definierten Zugriffsschutzes (siehe ATTRIB)

Wird der Nennung von Laufwerk Null als Ziellaufwerk für den DIR-Befehl ein Dollarzeichen vorangestellt, so geht das Betriebssystem davon aus, daß die Diskette, deren Inhaltsverzeichnis angezeigt werden soll, selbst kein Betriebssystem enthält.

Vor der Anzeige der Daten des Inhaltsverzeichnisses wird das Betriebssystem deshalb zu einem Diskettenwechsel auffordern.

BEMERKUNG: Sind so viele Dateien auf der angesprochenen Diskette, daß sie nicht gleichzeitig auf einer Bildschirmseite dargestellt werden können, so unterbricht das Betriebssystem die Anzeige nach jeweils 15 Zeilen. Im 64x16 Zeichenmodus ist das eine Bildschirmseite.

Im linken unteren Teil erscheint ein Fragezeichen, welches auf die Unterbrechung der Anzeige hinweist. Sobald die NEW LINE- oder ENTER-Taste betätigt wird, fährt das Betriebssystem mit der Anzeige der weiteren auf der Diskette befindlichen Dateien fort.

BEISPIELE: DIR

Die Dateien auf der Diskette im Laufwerk, das über die Systemoption AN angegeben ist, werden angezeigt.

DIR 0 S

Das Inhaltsverzeichnis der Diskette in Laufwerk 0 wird auf dem Bildschirm ausgegeben, wobei auch die Dateien des Betriebssystems angezeigt werden.

DIR 1 A P

Das Inhaltsverzeichnis der Diskette in Laufwerk 1 wird über den angeschlossenen Drucker ausgegeben, wobei auch alle Zusatzinformationen zu den einzelnen Dateien ausgedruckt werden.

DIR 0 /BAS

Alle Dateien der Diskette in Laufwerk 0, die die Namens Erweiterung /BAS tragen, werden auf dem Bildschirm angezeigt.

DIR 2 B

Nur die Dateien, deren Bearbeitungskennzeichen gesetzt ist, werden von der Diskette in Laufwerk 2 auf dem Bildschirm angezeigt.

DIR \$0

Das Betriebssystem fordert zum Diskettenwechsel in Laufwerk 0 auf, um danach das Inhaltsverzeichnis einer Diskette auf dem Bildschirm anzuzeigen, die kein eigenes Betriebssystem enthält.

DIR \$0 I S A P

Das Betriebssystem fordert zum Diskettenwechsel in Laufwerk 0 auf, um danach das erweiterte Inhaltsverzeichnis einer Diskette (mit Zusatzinformationen) incl. der unsichtbaren (invisible) und systeminternen Dateien auf einem Drucker auszugeben.

SYNTAX: DISK Laufwerknummer=Typ

WIRKUNG: Der Befehl DISK ermöglicht es, dem Betriebssystem für nachfolgende Verarbeitungen ein durch TYP spezifiziertes Diskettenformat festzulegen, wobei dieses Format durch einen Buchstaben von A bis P gekennzeichnet wird.

Nach Ausführung des DISK-Befehls kann eine Diskette in dem angegebenen Laufwerk damit auch bearbeitet werden, wenn die physikalischen Daten der Diskette von den durch die PD-Tabelle vordefinierten Werten abweichen.

Wird als Laufwerknummer das Systemlaufwerk (Laufwerk Null) angegeben, fordert das Betriebssystem zum Diskettenwechsel auf.

Die Diskettenformate von A bis P sind dabei folgendermaßen festgelegt:

| TYP | SPUREN | LAUFWERK | DICHTE | SEITEN | EIB | AEIV | SBIV | TI | TD |
|-----|--------|----------|--------|--------|-----|------|------|-----|----|
| A | 40 | 40 | SD | SS | 2 | 2 | 20 | A | A |
| B | 40 | 40 | SD | DS | 4 | 4 | 20 | A | C |
| C | 40 | 40 | DD | SS | 3 | 3 | 24 | CK | E |
| D | 40 | 40 | DD | DS | 6 | 6 | 24 | CK | G |
| E | 40 | 80 | SD | SS | 2 | 2 | 20 | AL | A |
| F | 40 | 80 | SD | DS | 4 | 4 | 20 | AL | C |
| G | 40 | 80 | DD | SS | 3 | 3 | 24 | CKL | E |
| H | 40 | 80 | DD | DS | 6 | 6 | 24 | CKL | G |
| I | 80 | 80 | SD | SS | 2 | 2 | 40 | A | A |
| J | 80 | 80 | SD | DS | 4 | 4 | 40 | A | C |
| K | 80 | 80 | DD | SS | 3 | 3 | 48 | CK | E |
| L | 80 | 80 | DD | DS | 6 | 6 | 48 | CHK | G |
| M | 40 | 80 | DD | SS | 2 | 2 | 17 | CL | E |
| N | 80 | 80 | DD | SS | 2 | 2 | 17 | CK | E |
| O | 40 | 80 | SD | SS | 2 | 2 | 17 | AL | A |
| P | 40 | 80 | DD | SS | 2 | 2 | 17 | CKL | E |

Ist die Laufwerkeinstellung mit dem DISK-Befehl verändert worden, sollten Sie zunächst mit DIR überprüfen, ob das Inhaltsverzeichnis der Zieldiskette auch lesbar ist.

Sollte dieses Inhaltsverzeichnis in einem von der Voreinstellung abweichenden Block (SBIV) beginnen, kann es dennoch durch das Betriebssystem gefunden werden und der entsprechende Block wird systemintern vermerkt.

Zur Sicherheit sollte die Lesbarkeit der auf der Diskette befindlichen Dateien überprüft werden, indem man sich eine Datei mittels LIST auf dem Bildschirm anzeigen läßt.

BEISPIELE: DISK 1=E

In dem Laufwerk 1 (80-Spur-Laufwerk) kann eine einseitige 40-Spur-Diskette einfacher Schreibdichte gelesen werden, die mit dem Format von zwei Einheiten im Block angelegt wurde.

DISK 0=P

Das Betriebssystem fordert zum Diskettenwechsel auf, da die zu bearbeitende Diskette in Laufwerk Null gelesen werden soll.

Nach der vorgegebenen TYP-Definition ist diese Diskette eine 40 Spur-Diskette, die in doppelter Schreibdichte einseitig formatiert wurde und zwei Einheiten im Block aufweist.

SYNTAX: DO Dateiname

=====

WIRKUNG: Der Befehl DO dient dazu, Job-Dateien aufzurufen und damit eine Folge von Befehlen zu aktivieren, die innerhalb dieser Job-Datei definiert wurden.

Dateien dieses Typs sind durch das Hilfsprogramm JOB/CMD auf Ihrer Betriebssystemdiskette zu erstellen und im Inhaltsverzeichnis der Diskette an der Namenserweiterung /JOB zu erkennen.

Sinn dieser Dateien ist es, eine Folge von Befehlen nacheinander automatisch auszuführen, ohne daß dazu zwischendurch Tastatureingaben erforderlich sind. Alle Kommandos werden der Reihe nach aus der Job-Datei gelesen und ausgeführt.

Der Name der Datei, die über den DO-Befehl aktiviert wird, kann entweder mit oder ohne Extension eingegeben werden.

Wird der Anhang /JOB nicht extra eingegeben, so geht das Betriebssystem aufgrund des DO-Befehls von sich aus davon aus, daß der Dateiname die Erweiterung /JOB aufweist.

Ist die Datei nicht oder mit einer anderen Namenserweiterung im Inhaltsverzeichnis eingetragen, so führt ein Aufruf über DO zu einer Fehlermeldung (Datei nicht im Inhaltsverzeichnis).

ANWENDUNG: Der Aufruf von Job-Dateien über den DO-Befehl findet überall seine Anwendung, wo zur Ausführung eines Programmes eine ganze Reihe von Befehlen nacheinander ausgeführt werden müssen, ohne daß der Anwender selbst sich mit den teilweise recht komplizierten Befehlsfolgen auseinandersetzen müßte.

Die Eingabe komplexer Kommandofolgen über eine Job-Datei hilft dem Benutzer eines Anwenderprogrammes außerdem, Eingabefehler zu vermeiden.

BEMERKUNG: Die Beschreibung des Programmes JOB/CMD zur Erzeugung von Job-Dateien finden Sie im Anhang dieses Handbuchs bei den Beschreibungen der mit Ihrem Betriebssystem gelieferten Hilfsprogramme.

Beispiele für Job-Dateien sind auch im Zusammenhang mit dem DOS-Befehl CONT aufgeführt.

BEISPIELE: DO TEST/JOB

Innerhalb der DOS-Ebene wird durch diesen Befehl die Datei TEST/JOB aktiviert, welche ihrerseits die Eingabekontrolle übernimmt.

Die Datei dieses Namens könnte im DOS auch durch das Kommando DO TEST aufgerufen werden.

Aus dem BASIC:

```
CMD"DO SORTER/JOB"
```

Über einen DOS-Aufruf wird die Job-Datei SORTER/JOB aktiviert, die in diesem Fall sowohl aus DOS- als auch aus BASIC-Befehlen bestehen kann.

DOS-Kommandos innerhalb einer solchen Datei müssen unter bestimmten Voraussetzungen vor dem eigentlichen Befehl ein vorangestelltes CMD" aufweisen, da die Eingaben aus der Job-Datei sonst als zum BASIC gehörig interpretiert werden. Nähere Informationen darüber finden Sie unter dem Befehl CONT.

Innerhalb der Job-Datei ist es allerdings auch möglich, über den Befehl CMD"S" das BASIC vollkommen zu verlassen und die Kontrolle wieder dem Betriebssystem zu übergeben.

Dieser Schritt kann vor allem dann notwendig sein, wenn der Arbeitsspeicher des Rechners für einen Sortiervorgang gebraucht wird und nicht mehr durch ein umfangreiches BASIC-Programm belegt sein darf. Oft ist dazu auch die zeitweilige Änderung vorhandener HIMEM-Grenzen notwendig.

Nach einem solchen Kommando müssen innerhalb der Job-Datei natürlich G-DOS-Befehle stehen, da das BASIC als Interpreter nicht mehr zur Verfügung steht.

Soll aus der DOS-Ebene wieder automatisch in das BASIC zurückgekehrt werden, so muß der Interpreter aus der Job-Datei durch das Kommando BASIC/CMD wieder neu geladen werden.

SYNTAX: DR Parameter Text

=====

WIRKUNG: Mit DR ist es möglich, aus dem Betriebssystem Texte direkt an den angeschlossenen Drucker auszugeben. Die eingegebenen Daten werden auf die parallele Drucker-schnittstelle geleitet. Ein Text kann dabei maximal 76 Zeichen beeinhalteln, die unabhängig von der Eingabeschreibweise immer als Großbuchstaben ausgedruckt werden.
Ist der Drucker nicht in Druckbereitschaft, so wird eine entsprechende Fehlermeldung auf dem Bildschirm ausgegeben.

Als Parameter kommen die Zeichen \$ und # in Frage. Es handelt sich dabei um Druckersteuerzeichen. Ein \$ ersetzt dabei den ESC-Code (ASCII 27), der bei fast allen Druckersteuerung benutzt wird. Das # wandelt einen folgenden Buchstaben in einen entsprechenden Controlcode um. Es wird dabei ein Controlcode ausgegeben, der dem ASCII-Wert des Buchstaben minus 64 entspricht.

ANWENDUNG: Ausgabe von Texten bzw. Kommentaren direkt aus der Betriebssystemebene unter Ausnutzung der Druckersteuerzeichen.

BEISPIELE: (Steuerzeichen für einen ITOH 8510A)

DR \$P Das ist Proportionaldruck
DR \$E Das ist Eliteschrift
DR #O Das ist Engschrift

SYNTAX: DUMP Dateiname vvvv bbbb eeee llll

WIRKUNG: Der Befehl DUMP ermöglicht die direkte Speicherung eines durch Startadresse vvvv und Endadresse bbbb festgelegten Speicherbereiches auf eine Diskette.

Wird durch den DUMP-Befehl ein komplettes ausführbares Maschinenprogramm auf die Diskette geschrieben, so gibt die Einsprungadresse eeee den Entry-point dieses Programmes an, wenn es im direkten Befehlsmodus aufgerufen wird.

Wird eine Verschiebung des Speicherinhaltes gewünscht, um z.B. Platz für Programme, die in denselben Speicherbereich laden, zu schaffen, so kann mit dem Parameter llll eine Ladeadresse für die Datei angegeben werden.

Falls mit dem DUMP-Befehl ein bestimmtes Laufwerk angesprochen werden soll, so ist die Laufwerknummer durch einen Doppelpunkt abgetrennt direkt an den gewünschten Dateinamen anzufügen.

Alle Adressen werden innerhalb des DUMP-Befehles durch ein Leerzeichen (oder Komma) voneinander getrennt und können sowohl in dezimaler als auch in hexadezimaler Schreibweise eingegeben werden. Bei hexadezimaler Eingabe ist der entsprechenden Zahl ein H anzufügen, welches diese Zahl für das Betriebssystem als Hexadezimalzahl kenntlich macht.

Start- und Endadresse (vvvv,bbbb) des zu speichernden Bereiches müssen in Verbindung mit dem DUMP-Kommando auf jeden Fall angegeben werden, die anderen Adressen sind optional.

Wird keine Einsprungadresse (eeee) angegeben, so erhält die auf diese Weise auf Diskette geschriebene Datei automatisch den Einsprungpunkt 402DH.

ANWENDUNG: Sichern von Maschinenprogrammen aus dem Arbeitsspeicher des Rechners die Diskette, sowie Schreiben eines definierten Speicherbereiches auf Diskette zur Untersuchung mittels geeigneter Disketten-Editoren (z.B. DDE).

BEMERKUNG: Der Anwender ist für die Richtigkeit der mit dem DUMP-Befehl verbundenen Adressenangaben selbst verantwortlich, da das Betriebssystem die Logik der eingegebene Definitionen nicht überprüft. Eine Einsprungadresse wird durch die entsprechende Eingabe nicht erzeugt, sondern nur für die Disk-Ein/Ausgabe definiert. Sollten also die im DUMP-Befehl angegebene Einsprungadresse und die echte Einsprungadresse des abzuspeichernden Programms nicht übereinstimmen, wird es zu einem Absturz des Systems kommen.

BEISPIELE: DUMP TESTDUMP/OBJ:1 7000H 7FFFH

Der Speicherinhalt des Rechners ab der Adresse 7000H wird bis zur Adresse 7FFFH einschließlich unter dem Namen TESTDUMP/OBJ auf die Diskette in Laufwerk 1 geschrieben.

DUMP GUINNESS/CMD:0 EF00H FFOBH F00BH

Der Speicherinhalt der Rechners wird unter dem Namen GUINNESS/CMD auf die Diskette in Laufwerk 0 geschrieben, und zwar ab der Adresse EF00H bis zur Adresse FFOBH, die Einsprungadresse des Maschinenprogramms ist F00BH.

DUMP MEMORY/OBJ:0 0000H FFFFH

Der gesamte Speicherinhalt des Computers wird unter dem Namen MEMORY/CMD auf die Diskette in Laufwerk 0 geschrieben.

Aus dem BASIC:

```
10 FORI=-8192TO-8181:READA:POKEI,A:NEXTI
20 DEFUSR1=-8192:POKEI5360,191
30 A=USR1(0)
40 CMD"DUMP BILD:0 15360 16383
50 POKEI5360,72:A=USR1(0)
60 CMD"LOAD BILD:0
70 GOTO 70
80 DATA 33,0,60,17,1,60,1,255,3,237,176,201
```

Das Programm schreibt zunächst den Bildschirm mit CHR\$(191) weiß und schreibt diesen Speicherbereich auf der Diskette im Laufwerk 0 unter dem Namen BILD mit Hilfe des DUMP-Befehls unter Voranstellung eines CMD" ab.

Nun wird der Bildschirm mit CHR\$(72) gefüllt und daran anschließend der zuvor auf Disk gedUMpte Videospeicherbereich erneut geladen, was wiederum ein weißes Bild hervorruft. Diese Anwendung kann insbesondere bei der Erstellung von Bildschirmmasken interessant sein.

HINWEIS: Falls im Inhaltsverzeichnis der Diskette eine Datei existiert, die den innerhalb des DUMP-Befehls definierten Namen trägt, so wird diese mit der Ausführung des Kommandos überschrieben.

SYNTAX: E Fehlercode

WIRKUNG: E ist die Abkürzung von Error und ermöglicht die Ausgabe der Fehlermeldung, die dem angegebenen Fehlercode entspricht. Die Fehlercodes mit der entsprechenden Meldung finden Sie im Anhang dieses Handbuchs.

ANWENDUNG: Derart simulierte Fehlermeldungen können innerhalb von Programmen benutzt werden, um dem Anwender evtl. mögliche Probleme anzuzeigen.

BEISPIEL: E 31

Der Fehler mit dem Code 31, "Programm nicht gefunden", wird auf dem Bildschirm angezeigt.

SYNTAX: FREE Parameter

WIRKUNG: Wird innerhalb der DOS-Ebene der Befehl FREE eingegeben, so werden für alle angeschlossenen Diskettenlaufwerke die Kopfzeilen des jeweiligen Inhaltsverzeichnisses angezeigt. Aus dieser Anzeige gehen die Diskettenamen, -Daten sowie die Anzahl der freien Plätze im Inhaltsverzeichnis der Disketten und die Anzahl der auf den jeweiligen Disketten freien Einheiten hervor.

Das Kommando gestattet damit, sich auf einfache Weise einen Überblick darüber zu verschaffen, wieviel Diskettenspeicherplatz auf den unterschiedlichen Laufwerken noch als Massenspeicher zur Verfügung steht.

Wird dem Befehl als Parameter ein P mitgegeben so erfolgt die Ausgabe der Daten auf den angeschlossenen Drucker.

ANWENDUNG: Abfrage des noch verbliebenen Speicherplatzes auf den Disketten in den angeschlossenen Laufwerken. Kontrolle der in den Laufwerken befindlichen Disketten über die Diskettenamen.

BEMERKUNG: Zur Ausführung des FREE-Kommandos muß die Formatierung der Disketten in den angeschlossenen Laufwerken den Daten entsprechen, die innerhalb der PD-Tabelle für die einzelnen Diskettenstationen definiert wurden.

Liegt diese Übereinstimmung nicht vor, so erfolgt innerhalb der Ausführung des FREE-Befehles ein Abbruch der Funktion mit der Fehlermeldung "Lesefehler Inhaltsverzeichnis".

SYNTAX: FORM Zeichenkette

=====

WIRKUNG: FORM dient zur Ausgabe von besonders definierten Zeichenketten an den Drucker, z.B. Zeichen für die Umschaltung auf Breitschrift oder Unterstreichung. Diese Zeichenketten werden mit Hilfe von DDE in die Datei SYS28/SYS eingetragen (Systemoption AA muß auf N stehen, damit keine Kennwortkontrolle erfolgt).

Die Zeichenkette wird Zeichen für Zeichen nach der Tabelle in SYS28/SYS in die Druckersteuerbefehle übersetzt.

Erlaubt sind innerhalb der Zeichenkette folgende Zeichen:

A,B,C,....,P,X,Y sowie 0,1,2,....,9 und sämtliche Sonderzeichen der Tastatur.

Vordefiniert sind Punkt (".") und Leerzeichen (" "). Punkt gibt einen Zeilenvorschub und Leerzeichen ein Leerzeichen aus.

Die Zeichenfolgen, die über die Zeichen A-P ausgegeben werden, stehen im zweiten Sektor von SYS28/SYS. Jede Zeile (jeweils 16 Byte) gehört zu einem Zeichen.

Die auszugebende Zeichenfolge muß mit einem 03H-Endezeichen abschließen.

Die Zeichenfolgen, die über die Ziffern 0-9, sowie über die Sonderzeichen ausgegeben werden, stehen im vierten Sektor von SYS28/SYS. Für jedes erlaubte Zeichen stehen vier Byte zur Verfügung. Auch hier muß jede Zeichenkette mit 03H abschließen, es stehen also 3 Byte für jedes Zeichen zur Verfügung.

Die Zeichenkette für Y belegt die zweite Hälfte des vierten Sektors in SYS28/SYS. Es stehen somit 128 Zeichen zur Verfügung. Hier lassen sich längere Steuersequenzen oder Seitenüberschriften eintragen.

Die Zeichenkette für X belegt den letzten Sektor im GDOS/SYS. Sie haben damit 256 Byte zur Verfügung, z.B. für einen ganzen Briefkopf.

Innerhalb eines FORM-Befehls können beliebige Zeichen aneinandergereiht werden.

WARNUNG: Erstellen Sie sich eine Sicherungskopie von SYS28/SYS durch den Befehl

```
COPY SYS28/SYS SYS28/ALT
```

bevor Sie mit DDE Änderungen am Original vornehmen.

HINWEIS: Durch den FORM-Befehl haben Sie die Möglichkeit für verschiedene Drucker, z.B. Itoh 8510 oder Star, mit unterschiedlichen SYS28/SYS-Dateien eine identische Druckersteuerung zu programmieren. Sie müssen diese Dateien natürlich unter verschiedenen Namen, z.B. SYS28/ITH oder SYS28/STA auf der Diskette abgelegt haben und diese bei Bedarf nach SYS28/SYS kopieren. In einem BASIC-Programm erfolgt dann der Aufruf durch CMD"FORM Zeichenkette" .

Zur Ausführung der LF-Funktion muß der Drucker eingeschaltet und auf Druckbereitschaft selektiert sein, andernfalls wird die Fehlermeldung "Drucker nicht bereit" ausgegeben.

BEISPIELE: FORM .

Dieser Befehl gibt einen Zeilenvorschub an den Drucker aus. (Dieses Zeichen ist vordefiniert!)

FORM A B

Dieser Befehl gibt die unter A und B definierten Zeichenketten, durch ein Leerzeichen voneinander getrennt, an den Drucker aus. Der Drucker wird die Zeichenketten jedoch erst nach einem Zeilenvorschub ausgeben, wie im vorherigen Beispiel.

Mit dem mitgelieferten SYS28/SYS sollte Ihr Drucker folgendes ausgeben:

AAAA,AAAA,AAAA, BBBB,BBBB,BBBB,

Es bietet sich an, die Zeichenkette für B so zu ändern, daß Ihr Drucker auf Breitschrift schaltet.

=====

WIRKUNG: Die DOS-Routine zur Programmierung der Funktionstasten wird aufgerufen.

Zum Zeitpunkt des Aufrufes dieses Kommandos können im Arbeitsspeicher des Rechners befindliche Programme zerstört werden, da zur Ausführung ein Teil dieses Speichers benutzt wird.

Acht Funktionstasten (F1 bis F8) können mit Kommandofolgen belegt werden, wobei insgesamt 256 Zeichen zur Verfügung stehen.

Ein NEW LINE-Zeichen darf dabei mit eingegeben werden (Darstellung auf dem Bildschirm als kleines Rechteck), was die Belegung einer einzelnen Taste mit einer ganzen Befehlsfolge ermöglicht.

Mit einer solchen Belegung ist in der praktischen Anwendung ein automatischer Funktionsablauf zu starten, wie er sonst nur über JOB-Dateien zu realisieren ist.

Die Arbeitsweise der F#-Routine ist auf dem Bildschirm selbsterklärend aufgebaut, mögliche Eingaben werden in jeder Ebene angezeigt.

ANWENDUNG: In Programmen, die zum Nutzen des Anwenders möglichst bedienungsfreundlich gestaltet werden sollen, z.B. Belegung der Funktionstasten mit komplexen Befehlsfolgen aus der COPY-Reihe.

BEMERKUNG: Im Normalfall werden die Funktionstastenbelegungen nach dem Abschluß der Routine auf der Diskette gespeichert, damit sie mit dem nächsten Start des Systems wieder automatisch zur Verfügung stehen. Hat der Anwender diese Speicherung durch einen Schreibschutz auf der Diskette verhindert, so ist die geänderte Belegung nur bis zum nächsten RESET aktiv.

SYNTAX: HIMEM Adresse

WIRKUNG: Das Kommando HIMEM gestattet es, in Verbindung mit der nachfolgend genannten Adresse die höchste Speicherstelle zu definieren, die das BASIC innerhalb des Arbeitsspeichers nutzen kann.

Speicherstellen oberhalb dieser Adresse (bis FFFFH) bilden einen geschützten Speicherbereich, in den Maschinen-Unterprogramme geladen werden können, ohne daß diese durch das Laden eines BASIC-Programmes zerstört würden.

Die HIMEM-Adresse kann sowohl in dezimaler als auch in hexadezimaler Schreibweise angegeben werden, wobei letztere Eingabeform zur Kennzeichnung als Hexadezimaladresse ein nachgestelltes H benötigt.

Wird innerhalb des Betriebssystems nur HIMEM ohne eine nachfolgende Adressangabe eingegeben, so erscheint die gerade aktuelle HIMEM-Grenze auf dem Bildschirm.

Die definierte Grenze des geschützten Speicherbereiches bleibt bestehen, bis eine neue HIMEM-Adresse genannt wird oder der G-DOS-Befehl 0 zur Ausführung kommt, der den Arbeitsspeicher bis zur höchsten Adresse mit Nullen überschreibt (=löscht). Nach dem 0-Kommando ist demnach der Wert von HIMEM immer FFFFH.

ANWENDUNG: Schutz eines bestimmten Bereiches im oberen Teil des Arbeitsspeichers gegen die Nutzung durch BASIC, Assembler usw.

BEMERKUNG: Innerhalb der Systemparameter kann unter dem Parameter AP die Adresse eingegeben werden, die sofort nach dem Laden des Betriebssystems die HIMEM-Adresse darstellen soll.

Mit jedem Ladevorgang (Boot) des Betriebssystems wird die dort festgelegte Adresse abgefragt und HIMEM automatisch auf diesen Wert festgesetzt, ohne daß dazu eine Tastatureingabe notwendig wäre.

BEISPIELE: HIMEM EFOOH

Der Speicherbereich des Rechners kann nur noch bis zur Adresse EFOOH durch BASIC-Programme genutzt werden, der oberhalb dieser Speicherzelle liegende Bereich bis FFFFH ist geschützt und kann Maschinenroutinen aufnehmen, die in diesen Bereich laden.

HIMEM

Die gerade aktuelle HIMEM-Grenze wird auf dem Bildschirm angezeigt. Ist durch kein vorangegangenes Kommando eine abweichende HIMEM-Grenze definiert worden, wird hier die Adresse angezeigt, die unter dem System-Parameter AP festgelegt wurde.

HINWEIS: Daß ein durch HIMEM geschützter Adressbereich für die Nutzung durch BASIC nicht mehr zur Verfügung steht, läßt sich leicht durch die Abfrage PRINT MEM innerhalb des BASIC feststellen, da die Anzahl der hier angezeigten freien Zeichen um die Anzahl der Bytes des geschützten Speicherbereiches kleiner geworden ist. BASIC hat eine eigene Möglichkeit HIMEM zu setzen. Dieses HIMEM des BASICs ist unabhängig vom HIMEM des DOS.

SYNTAX: I Parameter

WIRKUNG: Das Kommando I entspricht dem Kommando DIR und bewirkt die Anzeige des Inhaltsverzeichnisses einer Diskette.

Ausführliche Informationen finden Sie dazu unter dem Kommando DIR.

SYNTAX: INFO Parameter

WIRKUNG: Die Eingabe von INFO als Befehlseingabe innerhalb des Betriebssystems bewirkt die Anzeige der aktuellen Treiberadressen für Bildschirm, Drucker und Tastatur, dazu die Eingangsdaten aller aktiven Ports und die Darstellung des verfügbaren Zeichensatzes auf dem Bildschirm.

Außerdem werden die aktuelle HIMEM-Grenze für einen reservierten Speicherbereich sowie die Seitenlänge für den angeschlossenen Drucker angezeigt.

Da einzelne dieser Daten innerhalb der Verarbeitung von Programmen verändert werden können, kann man durch die Erweiterung des INFO-Kommandos mit verschiedenen Parametern diese Daten wieder auf ihren Initialisierungsstatus zurücksetzen.

Als Parameter sind zu diesem Zweck folgende Buchstaben anzuhängen:

M für den Bildschirm (Monitor)
T für die Tastatur
D für den angeschlossenen Drucker
Z für die Drucker-Zeilenzahl pro Seite

ANWENDUNG: Information über die aktuellen Zustände der aktiven Ports, Kontrolle des zur Verfügung stehenden Zeichensatzes sowie diverser Zusatzdaten.
In Verbindung mit den möglichen Parametern lassen sich Treiberdaten auf ihren Initialisierungsstatus zurücksetzen.

BEMERKUNG: Der INFO-Befehl ist unter anderem nützlich, um die Definition eines Z-Kommandos (siehe Z) zu überprüfen, da der aktuell nutzbare Zeichenvorrat auf dem Bildschirm angezeigt wird.
So kann auf einen Blick festgestellt werden, ob im Bereich der darstellbaren Sonderzeichen gerade die Blockgraphikzeichen oder die inverse Schriftdarstellung aktiviert sind (nur bei Genie III).

BEISPIELE: INFO

Auf dem Bildschirm erscheint die Anzeige des verfügbaren Zeichensatzes, die aktuellen Treiberadressen für Bildschirm, Tastatur und Drucker, die Eingangsdaten der aktiven Ports sowie HIMEM-Grenze und die Seitenzahl eines angeschlossenen Druckers.

INFO T M

Rücksetzung der Initialisierungsdaten für Tastatur und Bildschirm auf ihren ursprünglichen Status.

SYNTAX: JKL

=====

WIRKUNG: Der Befehl JKL dient dazu, den aktuellen Bildschirminhalt auf einen angeschlossenen Drucker auszugeben.

Der Ausdruck des Bildschirminhaltes erfolgt dem Format entsprechend, welches im CRTC-Kontrollblock (37F0H bis 37FFH) betriebssystem-intern festgelegt wurde. Dies trifft jedoch nur für das Genie III zu. (Siehe dazu auch Bildschirmformate unter "##".) Beim den Rechner Genie I/II ist das Format durch die Hardware fest definiert.

Steuerzeichen innerhalb der auszudruckenden Daten werden als Leerzeichen dargestellt. Um Graphikzeichen in den Ausdruck einzubeziehen, muß eine Reihe von Voraussetzungen erfüllt sein:

An den Rechner muß ein grafikfähiger Matrixdrucker angeschlossen sein, der über Einzelnadelansteuerung die Erzeugung von Graphikzeichen zuläßt.

Die Systemoptionen AK und AX müssen so eingestellt sein, daß eine Druckausgabe von Graphikzeichen zugelassen ist.

Falls diese Voraussetzungen nicht erfüllt sind, werden auf dem Bildschirm dargestellte Graphikzeichen als Leerzeichen an den Drucker gegeben.

Da der Zeichensatz eines Druckers sich im Normalfall besonders innerhalb der darstellbaren Graphikzeichen vom verfügbaren Zeichensatz Ihres Computers unterscheidet, ist zudem für einen Bildschirmausdruck der Genie-Graphikzeichen eine vorher zu ladende Software notwendig, welche die auf dem Bildschirm dargestellten Symbole in die entsprechenden Druckerodes umwandelt.

Solche Routinen finden Sie innerhalb der Zusatzprogramme, die auf Ihrer Betriebssystem-Diskette enthalten sind, unter dem Namen ITOH/CMD für die Matrixdrucker Itoh EG3085 und EG3100 und STAR/CMD für die STAR-Matrixdrucker.

ANWENDUNG: Die Hauptanwendung des Befehles JKL ist sicherlich als DOS-Aufruf aus einem BASIC-Programm, wenn eine dort dargestellte Bildschirmseite auf einen Drucker ausgegeben werden soll.

BEMERKUNG: Neben der Eingabe von JKL als Befehlseingabe kann auch durch `g l e i c h z e i t i g e s` Drücken der Tasten J, K und L ein Bildschirmausdruck gestartet werden, falls diese Möglichkeit durch den Systemparameter AD nicht ausgeschlossen wurde.

BEISPIELE: Unter der Voraussetzung, daß der Systemparameter AD=J definiert ist, drücken Sie gleichzeitig die Tasten "J", "K" und "L".

Der aktuelle Bildschirminhalt wird auf einen angeschlossenen Drucker ausgegeben.

Aus dem BASIC:

```
10 CLS : CLEAR 1000
20 FOR I=1 to 15
30 L$=STRING$(I,32)
40 TX$="Das ist ein Test für die JKL-Funktion !"
50 PRINT L$; : PRINT TX$
60 NEXT I
70 CMD"JKL"
80 END
```

Nachdem der Bildschirm gelöscht und der in den Zeilen 30 und 40 definierte Text in 15 Bildschirmzeilen dargestellt wurde, startet das Programm durch Zeile 70 einen Bildschirmausdruck auf dem angeschlossenen Drucker.

SYNTAX: KILL Dateiname:Laufwerknummer

WIRKUNG: Dieser Befehl dient dazu, einzelne auf der Diskette befindliche Dateien oder Programme zu löschen und ist deshalb mit größter Vorsicht zu verwenden.

Neben dem Namen der zu löschenden Datei sollte in jedem Fall eine Laufwerknummer angegeben werden, damit ein versehentliches Löschen einer Datei ausgeschlossen wird.

Wird ein Laufwerk mit dem KILL-Befehl nicht ausdrücklich definiert, sucht das Betriebssystem beginnend mit Laufwerk Null die Disketten nach dem angegebenen Dateinamen ab, um die erste Datei, die unter diesem Namen gefunden wird, zu löschen.

Mit der Löschung einer Datei aus dem Inhaltsverzeichnis der Diskette ist diese Datei zwar nicht unwiederbringlich zerstört, die Rekonstruktion einer einmal gelöschten Datei erfordert allerdings fundierte Kenntnisse der Programmierung und der Struktur des Betriebssystems bzw. Inhaltsverzeichnisses. Ohne diese Voraussetzungen ist der Anwender für eine solche Prozedur erfahrungsgemäß überfordert.

ANWENDUNG: Löschen einer einzelnen Datei aus dem Inhaltsverzeichnis einer Diskette.

BEMERKUNG: Sollen mehrere auf einer Diskette befindliche Dateien gleichzeitig gelöscht werden, sollte dazu der G-DOS-Befehl PURGE benutzt werden.

BEISPIELE: KILL GUINNESS/CMD:1

Die Datei GUINNESS/CMD auf Laufwerk 1 wird aus dem Inhaltsverzeichnis der Diskette gelöscht.

KILL TESTPROG

Die Inhaltsverzeichnisse der Disketten in den angeschlossenen Laufwerken werden nach dem Dateinamen TESTPROG durchsucht, beim ersten Auffinden dieser Datei wird diese gelöscht.

KILL ADRESS/DAT:2

Die Datei ADRESS/DAT auf Laufwerk 2 wird aus dem Inhaltsverzeichnis der Diskette gelöscht.

HINWEIS: Wird mit dem Namen der zu löschenden Datei ein Laufwerk spezifiziert, so führt der KILL-Befehl zu einer Fehlermeldung, falls sich die angesprochene Datei nicht im Inhaltsverzeichnis der entsprechenden Diskette befindet.

Diese Meldung erfolgt ungeachtet der eventuellen Existenz des angegebenen Dateinamen auf einer Diskette in einem anderen Laufwerk.

SYNTAX: LC Parameter

=====

WIRKUNG: LC erlaubt die Einschaltung der Kleinschrift für die Darstellung der Tastatureingaben auf dem Bildschirm. Dabei sind folgende Parameter möglich:

- J Darstellung von Kleinbuchstaben ist zugelassen. Die SHIFT-Taste hat eine normale Wirkung (Beim Genie III abhängig von der LOCK-Taste).
- N Kleinbuchstaben-Darstellung wird unterbunden. Die SHIFT-Taste hat eine umgekehrte Wirkung (wie unter normalem Level II BASIC).

Folgende Parameter gelten nicht für das Genie III:

- J,J Wie bei einer Schreibmaschine werden die von der Tastatur eingegebenen Zeichen auf dem Bildschirm in Kleinbuchstaben dargestellt. Mit der SHIFT-Taste können Großbuchstaben erzeugt werden.
- J,N Die Eingabe von Kleinbuchstaben über die Tastatur wird unterbunden, während Kleinbuchstaben, die über ein Programm an den Bildschirm ausgegeben werden, dargestellt werden.

BEMERKUNG: Ist der Modus LC J definiert, kann auch über die Tastenkombination SHIFT-NULL die Tastatureingabe umgestellt werden.

Nach gleichzeitigem Druck auf die Tasten SHIFT und Null (0) werden entweder nur Großbuchstaben zugelassen, oder Kleinbuchstaben, die über die SHIFT-Taste in Großbuchstaben umgewandelt werden können. Die Funktion der SHIFT-Null-Kombination hängt von dem vorher aktiven Status der Tastatureingabe ab, Kleinschrift kann also über SHIFT-Null sowohl ein, als auch ausgeschaltet werden. Beim Genie III erzeugt SHIFT-Null statt der Umschaltung ein Leerzeichen.

Die Systemparameter BG und BF definieren, ob direkt mit dem Start des Betriebssystems die Kleinbuchstabendarstellung aktiviert ist oder nicht.

BEISPIEL: LC J,N

Die Darstellung von Kleinbuchstaben innerhalb der Bildschirmdarstellung ist zugelassen, Tastatureingaben erfolgen in Kleinschrift, sofern nicht die SHIFT-Taste gleichzeitig mit der entsprechenden Buchstabeneingabe gedrückt wurde. Die Kombination SHIFT-Null kann die Tastatureingabe auf ausschließliche Großbuchstabenerzeugung umschalten.

SYNTAX: LF Parameter

=====

WIRKUNG: LF dient der Definition der Seitenlänge für den angeschlossenen Drucker bzw. zu dessen Positionierung.

LF,6 definiert die Seitenlänge mit sechs Zeilen je Zoll, auf einer Druckseite sind damit 72 Zeilen darstellbar.

LF,8 definiert eine Zeilenzahl von 96 pro Druckseite, also acht Zeilen je Zoll.

LF ohne Parameter aktiviert eine Routine zur Positionierung des Druckers.

Nach dieser Aktivierung sind folgende Tastenfunktionen wirksam:

| | |
|------------------|--|
| HOCHPFEIL | bewirkt einen Zeilenvorschub des Druckers bis zum Seitenende |
| PFEIL NACH UNTEN | ergibt einen Zeilenvorschub |
| LINKSPFEIL | Umschaltung der Transportrichtung auf negativen Zeilenvorschub (nur für Itoh 8510) |
| RECHTSPFEIL | Umschaltung der Transportrichtung auf positiven Zeilenvorschub (nur für Itoh 8510) |
| CLEAR | stellt den internen Zeilenzähler auf Null (siehe auch INFO 2) |
| NEW LINE | beendet die LF-Routine |

Da die SteuerCodes der Drucker sich je nach Type und Fabrikat voneinander unterscheiden, sind die Parameter zur Steuerung der LF-Funktion innerhalb des Betriebssystems je nach verwendetem Druckertyp abzuändern.

Die Voreinstellung entspricht den Daten für den Itoh Matrixdrucker 8510.

HINWEIS: Die folgende Änderung in SYS28/SYS macht den LF-Befehl für die Star-Drucker verfügbar. Ändern Sie mit DDE den letzten Sektor von SYS28/SYS.

Sie finden ab Byte 6AH folgende Zeichen:

1B41 0300 1B42 0300 1B72 0300 1B66 03FE

Ändern Sie diese Zeichen in:

1B32 0300 1B30 0300 1B10 0300 1003 00FE

Dadurch ändert sich auch die Bedeutung von LINKSPFEIL und RECHTSPFEIL (nur bei einem Star Radix-Drucker wirksam). LINKSPFEIL transportiert das Papier um eine Zeile zurück. RECHTSPFEIL transportiert das Papier um eine Zeile vor. Der aktuelle Zeilenzähler stimmt nach einem Rücktransport nicht mehr.

BEMERKUNG: Zur Ausführung der LF-Funktion muß der Drucker eingeschaltet und auf Druckbereitschaft selektiert sein, andernfalls wird die Fehlermeldung "Drucker nicht bereit" ausgegeben.

BEISPIELE: LF 6

Die Seitenlänge pro Druckseite wird für den angeschlossenen Drucker auf sechs Zeilen pro Zoll festgesetzt, insgesamt pro Seite 72 Zeilen.

LF 8

Die Seitenlänge pro Druckseite wird auf 96 Zeilen festgesetzt, was einer Zeilenzahl von acht pro Zoll entspricht.

LF

Die Routine zur Druckerpositionierung wird aktiviert, über die Pfeiltasten läßt sich der Papiertransport steuern. NEW LINE beendet diese Routine und bewirkt Rücksprung in den G-DOS-Befehlsmodus.

SYNTAX: LIB

WIRKUNG: Das Kommando LIB benötigt keine weiteren Parameter. Die möglichen G-DOS-Befehlswoorte werden auf dem Bildschirm angezeigt.

BEMERKUNG: Die Funktion des LIB-Befehles kann auch durch die Eingabe eines Fragezeichens innerhalb der DOS-Ebene aufgerufen werden.

BEISPIEL: LIB

Die Befehlsliste des Betriebssystems wird auf dem Bildschirm dargestellt.

SYNTAX: LIST Dateiname Startzeile Zeilenzahl

WIRKUNG: Mit dem LIST-Befehl kann der Inhalt einer auf der Diskette befindlichen Datei auf dem Bildschirm angezeigt werden.

ANWENDUNG: Eine sinnvolle Anwendung findet dieser Befehl eigentlich nur in Verbindung mit Textdateien, da nicht darstellbare Steuerzeichen innerhalb einer Datei immer nur als Punkt auf dem Bildschirm angezeigt werden.

Wenn Sie innerhalb des Kommandos eine Startzeile angeben, beginnt das Listing auf dem Bildschirm erst bei dieser Zeile. Weiterhin läßt sich durch Angabe einer Zeilenzahl die Anzahl der zu listenden Zeilen begrenzen.

Ein auf dem Schirm ablaufendes Listing kann durch Drücken der Pfeil-rechts-Taste unterbrochen werden. Die Betätigung der NEW LINE-Taste hat den Fortlauf des Listings zur Folge. Über die Pfeil-oben-Taste läßt sich ein Listing vollkommen abbrechen.

BEISPIELE: Im folgenden Beispiel ist vorausgesetzt, daß das Programm TESTPROG/BAS im ASCII-Format auf der Diskette abgespeichert wurde.

```
LIST TESTPROG/BAS 100 10
```

Vom BASIC-Programm TESTPROG/BAS werden auf Grund dieses Befehls ab Zeile 100 weitere 10 Zeilen gelistet.

LIST DEMO/JOB

Der File DEMO/JOB wird komplett auf dem Bildschirm angezeigt. Das Listing kann über Pfeil-rechts gehalten und mit NEW LINE fortgeführt werden. Pfeil-oben bedeutet Abbruch des Listings.

SYNTAX: LOAD Filename

=====

WIRKUNG: Mit Hilfe dieses Befehls können Sie Files, die in Z80-Assembler geschrieben sind, in den Speicher des Rechners einladen. Programme werden dabei nicht gestartet, sondern nur den Ladeadressen entsprechend im RAM abgelegt.

ANWENDUNG: Der LOAD-Befehl wird immer dann benutzt, wenn Programme oder Daten zwar schon im Speicher vorhanden sein müssen, jedoch erst später abgerufen werden (z.B. im BASIC als `USR`-Aufruf). Weiterhin ist eine sinnvolle Anwendung des LOAD-Befehls beim Einlesen von Bildschirmmasken zu finden.

BEMERKUNG: Die mit LOAD einzuladenden Files müssen im korrekten Format auf der Diskette vorliegen, d.h. es dürfen sich keine Überschneidungen mit dem DOS ergeben.

BEISPIEL: Einladen einer Bildschirmmaske:

```
10 FOR I=15360 TO 16383
20 POKE I,191:NEXT I
30 CMD"DUMP VIDEO/BLD:0 15360 16383
50 CLS:PRINT"Jetzt wird das Bild wieder geladen!"
60 CMD"LOAD VIDEO/BLD:0
70 GOTO 70
```

Das Programm schreibt zunächst den Bildschirm mit `CHR$(191)` weiß und speichert diesen Speicherbereich auf der Diskette im Laufwerk 0 unter dem Filenamen `VIDEO/BLD` mit Hilfe des `DUMP`-Befehls ab. Nach einem `CLS` wird das zuvor auf Diskette gespeicherte Bild erneut mit `LOAD` in dem Video-RAM geladen und auf dem Bildschirm wiedergegeben.

SYNTAX: N DATEIALT/TYP DATEINEU/TYP
=====

WIRKUNG: Durch den Befehl N ist es möglich, einer auf der Diskette befindlichen Datei einen neuen Namen zuzuweisen.

Nach Ausführung des Befehls trägt die Datei DATEIALT/TYP die neue Bezeichnung DATEINEU/TYP.

Eine Laufwerksangabe darf logischerweise nur nach der Nennung des alten Dateinamen erfolgen, da der Befehl auch nur eine einzige Datei bearbeitet.

ANWENDUNG: Umbenennung eines Programms innerhalb des Inhaltsverzeichnisses einer Diskette.

BEMERKUNG: Alle Parameter, die einer Datei unter ihrer alten Bezeichnung zugewiesen waren (Sichtbarkeit, Erweiterungsmöglichkeit etc.) werden durch diesen Befehl nicht beeinflusst.

BEISPIELE: N TESTADRS/BAS:0 ADRESSEN/BAS

Das Programm TESTADRS/BAS auf der Diskette in Laufwerk 0 erhält den neuen Namen ADRESSEN/BAS.

N COMMAND/OBJ:1 KOMMANDO/CMD

Die Datei COMMAND/OBJ, die sich auf der Diskette in Laufwerk 1 befindet, erhält den neuen Namen KOMMANDO/CMD.

N TROMME/DAT TCS/DAT

Die Disketten in den angeschlossenen Diskettenlaufwerken werden nach dem Dateinamen TROMME/DAT durchsucht, sobald diese Datei in einem der Inhaltsverzeichnisse gefunden wird, erhält sie den neuen Namen TCS/DAT.

SYNTAX: NDF Laufwerk-Nr. Diskname Datum Kennwort Parameter

WIRKUNG: Der Befehl NDF dient zur Formatierung einer neuen Diskette in dem angegebenen Laufwerk.

Bei der Formatierung wird die Diskette in Spuren und Sektoren aufgeteilt, die dem Betriebssystem die Orientierung auf der Diskette bei Schreib- oder Leseoperationen ermöglichen.

Zudem werden mit der Formatierung die auf jeder Diskette notwendigen Dateien GDOS/SYS und INHALT/SYS automatisch angelegt.

Wenn außer dem anzusprechenden Diskettenlaufwerk keine weiteren Parameter mitgeteilt werden, so wird die Diskette gemäß den diesem Laufwerk zugeordneten Daten innerhalb der PD-Tabelle formatiert.

Einzelne Parameter gestatten aber auch, davon abweichend verschiedene physikalische Daten anders zu definieren.

Die möglichen Parameter zum Befehl NDF lauten:

N
J
KDWA
ZZND
AZN=ZIELNAME
BZN
SBIV=XX
AEIV=YY
ZPDN=ZZ
SPUR=xx
STOP=yy
MAG

Da der größte Teil der zulässigen Parameter auch bei der Verwendung des COPY-Kommandos erklärt ist, sind hier nur die dort nicht aufgeführten Parameter beschrieben:

SPUR=xx - xx gibt die Spur an, bei der die Formatierung der Diskette beginnen soll

STOP=yy - yy legt die Anzahl der zu formatierenden Spuren fest

MAG entspricht einer Formatierung, die dem Löschen der Diskette durch einen Magneten gleichkommt. Die Dateien GDOS/SYS und INHALT/SYS werden **n i c h t** eingetragen!

ANWENDUNG: Durch die Parameter SPUR=xx und STOP=yy ist auch eine teilweise Formatierung einer Diskette möglich. Das kann z.B. sinnvoll sein, wenn eine nur teildefekte Diskette innerhalb eines festgelegten Bereiches neu formatiert werden soll, weil ein Sektor oder eine Spur nicht mehr lesbar sind.

BEMERKUNG: Befinden sich auf der zu formatierenden Diskette Daten oder Programme, so werden diese mit der Ausführung des NDF-Kommandos unwiederbringlich (!) gelöscht !

Alle nicht durch GDOS/SYS und INHALT/SYS belegten Sektoren werden während der Formatierung mit einem sogenannten Formatierungsmuster beschrieben, welches bei G-DOS aus den Daten 6D und B6 besteht.

Dadurch kann unter Umständen schon bei der Formatierung der Diskette festgestellt werden, ob diese einen physikalischen Defekt oder einen Mangel in der Beschichtung der Diskettenoberfläche aufweist.

Ein Sektor, der nach der Formatierung noch nicht zur Datenspeicherung genutzt wurde, ist z.B. bei Untersuchung der Diskette mittels DDE leicht anhand des Formatierungsmusters 6DB6 zu erkennen.

Werden einzelne Parameter, die normalerweise zur Syntax des NDF-Befehls gehören, bei der Eingabe ausgelassen, so sind nachfolgende Parameter unter Umständen durch die Verwendung von Kommata als Platzhalter vom Befehlswort abzutrennen.

Würde z.B. zur Formatierung der Befehl NDF 1 MAG lauten, so entspräche das erzeugte Format den Daten, die in Zeile 1 der PD-Tabelle für Laufwerk 1 definiert sind, GDOS/SYS und INHALT/SYS würden eingetragen und die Diskette erhielte den Namen MAG.

Soll die Diskette aber ohne Dateieinträge vollkommen gelöscht werden (entsprechend der Option MAG), so muß das entsprechende Kommando NDF 1,,,MAG lauten.

BEISPIELE: NDF 1

Die Diskette in Laufwerk 1 wird gemäß den in der PD-Tabelle festgelegten physikalischen Daten formatiert, nachdem das Betriebssystem zum Einlegen der Zieldiskette in das angesprochene Laufwerk aufgefordert hat.

NDF 2,,,KDWA

Ohne weitere Warteabfragen startet die Formatierung der Diskette in Laufwerk 2.
Die Kommata stehen für Diskname und Datum.

NDF 1,,,,SPUR=25,STOP=3

Beginnend mit Spur 25 werden drei Spuren der Diskette in Laufwerk 1 neu formatiert.

NDF 1 DATADISK,,,ZPDN=3

Die Diskette in Laufwerk 1 wird formatiert, erhält den Namen DATADISK, die Formatierungsdaten entsprechen den in der Zeile 3 der PD-Tabelle festgelegten Parametern.

NDF 3,,,,MAG

Die Diskette in Laufwerk 3 wird vollkommen gelöscht, ohne daß die Dateien GDOS/SYS und INHALT/SYS eingetragen werden.

NDF 1 TESTDISK 28.08.84 TCS KDWA

Die Diskette in Laufwerk 1 wird ohne weitere Abfragen formatiert. Sie erhält dabei den Namen TESTDISK, das Datum 28.08.84 und kann nur noch unter Angabe des Kennwortes TCS bearbeitet werden.

SYNTAX: PAUSE Nachricht

WIRKUNG: Der Befehl PAUSE veranlaßt das Betriebssystem, in eine Warteschleife zu springen, die erst durch die Betätigung der NEW LINE- oder ENTER-Taste wieder verlassen wird.

Mit der Ausführung des Kommandos wird die Meldung "Taste 'ENTER' wenn fortfahrbereit !" auf dem Bildschirm ausgegeben.

Zusätzlich zu dieser Meldung kann der Anwender selbst eine Nachricht angeben, die während einer Programmunterbrechung angezeigt werden soll. Durch ein Leerzeichen (oder Komma) abgetrennt wird diese Nachricht bei der Befehlseingabe hinter dem PAUSE-Kommando angefügt. Als Nachricht sind Eingaben mit einer Länge von bis zu 73 Zeichen zulässig.

ANWENDUNG: Innerhalb des Betriebssystems ist der Befehl PAUSE fast nur innerhalb von Job-Dateien gebräuchlich, deren Verarbeitung aus einem bestimmten Grund zeitweilig unterbrochen werden soll.

Häufiger sind Anwendungen als DOS-Aufruf aus dem BASIC, wo der PAUSE-Befehl einen Stop der Programmverarbeitung bewirkt, bis der Anwender über die Tastatur die weitere Verarbeitung freigibt.

BEMERKUNG: Wie immer bei DOS-Aufrufen aus dem BASIC ist dem Befehl PAUSE bei Verwendung innerhalb von BASIC-Programmen die Kennzeichnung CMD" voranzustellen.

BEISPIELE: Aus dem BASIC:

```
10 CLS
20 LINEINPUT Q$
30 CMD"PAUSE,Bitte den Drucker bereit machen !"
40 LPRINT Q$
50 END
```

Nachdem der Anwender den zu druckenden Text eingegeben hat, gibt das Programm die innerhalb des DOS-Aufrufes in Zeile 30 definierte Nachricht aus. Sobald die NEW LINE-Taste gedrückt wird, fährt das Programm in der Verarbeitung mit Zeile 40 fort.

Beispiele für die Verwendung des PAUSE-Befehls innerhalb von Job-Dateien finden Sie in diesem Handbuch unter den Beispielen zum G-DOS-Befehl CONT.

SYNTAX: PD Laufwerknummer Parameter

=====

WIRKUNG: Der PD-Befehl dient zur Festlegung der physikalischen Daten für die angeschlossenen Diskettenlaufwerke.

Die dazu gespeicherten Daten sind in einer Tabelle aufgeführt, die bei der Eingabe

PD Laufwerknummer <NEW LINE>

auf dem Bildschirm angezeigt wird. Diese Tabelle enthält zehn Zeilen (numeriert von Null bis Neun), von denen jede Zeile einen bestimmten Formatierungstyp kennzeichnet.

ANWENDUNG: Die Parameter innerhalb einer Zeile kennzeichnen im einzelnen folgende Spezifikationen:

TI = Typ des verwendeten Diskettencontrollers
TD = Typ des verwendeten Diskettenlaufwerks
SP = Anzahl der Spuren auf der Diskette
SEK = Anzahl der Sektoren pro Spur
SWZ = Zeitfaktor für den Spurwechsel des Lesekopfes
EIB = Einheiten im Block
AEIV = Anzahl der Einheiten für Inhaltsverzeichnis
SBIV = Startblock für das Inhaltsverzeichnis

Jede Zeile steht für ein Diskettenlaufwerk, wobei die tatsächlich gemäß der Systemoption AL angeschlossenen Laufwerke durch einen Stern vor der entsprechenden Zeile gekennzeichnet sind.

Alle weiteren Zeilen haben demnach für die Spezifikation der aktuellen physikalischen Daten der angeschlossenen Laufwerke keine Bedeutung, sie sind aber hilfreich, wenn die Parameter für einen bestimmten Diskettentyp komplett ausgetauscht werden sollen. In diesem Fall könnte z.B. das Kommando "PD 0 1=6" eingegeben werden, und alle in Zeile sechs definierten Werte werden in die Zeile eins übertragen, welche die Daten für das Laufwerk eins enthält.

Mit dem PD-Kommando werden Daten auf der Diskette geändert, die mit jedem Systemstart neu initialisiert werden. Deshalb darf sich auf der Diskette, auf der die PD-Daten geändert werden sollen, auf keinen Fall ein Schreibschutz befinden !

Die erste Zahl nach PD kennzeichnet das Laufwerk, in der sich die Diskette befindet, deren Spezifikation geändert werden soll.

Die zweite Zahl gibt an, welche Zeile innerhalb der PD-Tabelle geändert werden soll.

Danach werden dann entweder einzelne Parameter angegeben, die zu verändern sind, oder aber alle Parameter einer Zeile durch die Werte einer anderen Zeile aus der Tabelle ersetzt.

PARAMETER: TI kennzeichnet den Typ des verwendeten Disketten-controllerinterface:

- TI=A Standard-Disk-Controller-Typ (FDC 1771)
Adressmarke Inhaltsverzeichnis = FC
TD = A oder C
- TI=B Zulässig sind TD = A, B, C oder D
- TI=C Disk-Controller-Typ FDC 1771 oder FDC 1791
Adressmarke Inhaltsverzeichnis = F8 bei
FDC 1791.
TD = A, C, E oder G

Von diesen TI-Parametern darf entsprechend dem verwendeten Controller jeweils nur einer angegeben werden. Nachfolgend können die im folgenden aufgeführten Buchstaben jedoch als Zusatzinformation angehängt werden:

- H Bevor die Schreib/Leseköpfe der Laufwerke aktiviert werden, soll eine Zeitverzögerung stattfinden.
- I Die Numerierung der Sektoren beginnt mit Sektor 1 (statt Sektor 0)
- J Die Numerierung der Spuren beginnt mit Spur 1 (statt Spur 0)
- K Die erste Spur auf der Diskette ist in einfacher Schreibdichte zu schreiben, alle weiteren Spuren in doppelter Schreibdichte. Solange während des Systemstarts angenommen wird, die Diskette sei in einfacher Schreibdichte formatiert, muß K als Parameter mit angegeben sein, damit dieser Diskettentyp gelesen werden kann.
- L Durch diese Option wird der Step-Impuls für den Schreib/Lesekopf des Diskettenlaufwerkes verdoppelt.
Das ist notwendig, wenn 40-Spur-Disketten auf 80-Spur-Laufwerken verarbeitet werden sollen. In diesem Fall wird nur jede zweite Spur gelesen bzw. beschrieben.
Da nicht alle Laufwerke in der Lage sind, Disketten in diesem Modus einwandfrei zu lesen, ist bei dieser Verarbeitungsart besondere Vorsicht geboten.
In keinem Fall ist es sinnvoll, mit einem 80-Spur-Laufwerk auf einer 40-Spur-Diskette Schreiboperationen durchzuführen. Die Toleranzen in der Verarbeitung der auf einer Diskette gespeicherten Daten sind so eng, daß in diesem Fall mit großer Wahrscheinlichkeit Probleme zu erwarten sind, wenn eine auf diese Weise bearbeitete Diskette später auf einem 40-Spur-Laufwerk gelesen werden soll.

M Verarbeitung von Disketten, die im TRS-80 Model III-Format erstellt sind.
Diese Option gestattet Leseoperationen von einer entsprechenden Diskette oder auch eine Disketten-Kopie mit dem COPY-Parameter IVU.
Alle weiteren Verarbeitungen bleiben jedoch ausgeschlossen

TD TD legt den Typ des verwendeten Diskettenlaufwerks fest. Es bedeuten dabei:

TD=A einfache Schreibdichte, einseitig, 5.25 Zoll
TD=B einfache Schreibdichte, einseitig, 8 Zoll
TD=C einfache Schreibdichte, doppelseitig, 5.25 Zoll
TD=D einfache Schreibdichte, doppelseitig, 8 Zoll
TD=E doppelte Schreibdichte, einseitig, 5.25 Zoll
TD=F doppelte Schreibdichte, einseitig, 8 Zoll
TD=G doppelte Schreibdichte, doppelseitig, 5.25 Zoll
TD=H doppelte Schreibdichte, doppelseitig, 8 Zoll

SP SP legt fest, wieviele Spuren auf dem betreffenden Laufwerk benutzt werden sollen.
Wenn TI=K gesetzt wurde (s.o.), wird hier die Anzahl der Spuren ohne Spur Null angegeben.

SEK SEK gibt an, wieviele Sektoren pro Spur vorhanden sein sollen. Bei allen doppelseitigen Laufwerken ist zu beachten, daß die Anzahl der Sektoren durch zwei teilbar sein muß.
In Abhängigkeit von der TD-Festlegung gelten folgende Maximalwerte für die Anzahl der Sektoren:

| | | |
|----------|-------------|----------|
| Bei TD=A | 10 Sektoren | (SEK=10) |
| TD=B | 17 Sektoren | (SEK=17) |
| TD=C | 20 Sektoren | (SEK=20) |
| TD=D | 34 Sektoren | (SEK=34) |
| TD=E | 18 Sektoren | (SEK=18) |
| TD=F | 26 Sektoren | (SEK=26) |
| TD=G | 36 Sektoren | (SEK=36) |
| TD=H | 52 Sektoren | (SEK=52) |

SWZ SWZ kann einen Wert von Null bis Drei aufweisen und gibt den Faktor für die Impulszeit an, die der Controller für die Kopfpositionierung des Laufwerks zur Verfügung stellt.

Die Verzögerungen betragen bei

| | | | |
|-------|------------------|-------|------------------|
| SWZ=0 | 5 Millisekunden | SWZ=1 | 10 Millisekunden |
| SWZ=2 | 20 Millisekunden | SWZ=3 | 40 Millisekunden |

Der korrekte Wert ist den technischen Daten zu dem verwendeten Laufwerk zu entnehmen, im Zweifelsfall ist SWZ=3 zu wählen.

EIB EIB gibt die Anzahl der Einheiten an, die innerhalb der internen Betriebssystem-Organisation zu einem Block zusammengefasst werden.
Eine Einheit besteht dabei aus fünf Sektoren mit jeweils 256 Byte Länge.
Die Einteilung in sogenannte Blöcke wird vorgenommen, um einen möglichst schnellen Zugriff auf die Diskettendaten zu gewährleisten.
EIB darf jeden ganzzahligen Wert zwischen zwei und acht annehmen.

SBIV SBIV kennzeichnet den Startblock für das Inhaltsverzeichnis der Diskette. Um einen schnellen Zugriff auf die Diskettendaten zu gewährleisten, sollte dieser Block nach Möglichkeit auf der mittleren Spur der Diskette angelegt werden. Bei 80-Spur-Doppelkopflaufwerken ist dies z.B. Block 48, wenn EIB=6 definiert wurde.

AEIV AEIV legt die Anzahl der Einheiten fest, die für das Inhaltsverzeichnis der Diskette bereitgestellt werden sollen. Dadurch wird also die Größe und damit auch das Fassungsvermögen des Inhaltsverzeichnisses beeinflusst.
Es müssen mindestens zwei Einheiten bereitgestellt werden, der Maximalwert für AEIV beträgt sechs.

HINWEIS: Die mit dem PD-Kommando übergebenen Daten werden im Systemprogramm GDOS/SYS abgespeichert und erst mit einem neuen Systemstart initialisiert.

Soll eine Änderung innerhalb der PD-Daten sofort (ohne neuerlichen Systemstart) wirksam werden, so ist nach Eingabe aller Parameter zusätzlich (durch ein Komma oder Leerzeichen abgetrennt) ein A einzugeben.

In diesem Fall arbeitet das Betriebssystem ab der Ausführung des PD-Kommandos sofort mit den neuen Tabellendaten.

WARNUNG: Wird mit der Eingabe des PD-Befehls ein falscher Parameter übergeben, so erscheint unter der fehlerhaften Zeile die Meldung: >>!ACHTUNG!<<

Dieser Fehler sollte sofort korrigiert werden, da jeder Fehler innerhalb der PD-Tabelle einen neuen Systemstart verhindert.

Erfolgt ein RESET, während unzulässige Daten innerhalb der PD-Tabelle gespeichert sind, gibt das Betriebssystem die Meldung

UNZULÄSSIGE INITIALISIERUNGSDATEN AUF DER DISKETTE !

aus und der Rechner "hängt sich auf", ohne in die DOS-Ebene zur Befehlseingabe zu gelangen.

BEISPIELE: PD 0

Diese Eingabe bewirkt, daß die aktuelle PD-Tabelle der Diskette in Laufwerk Null auf dem Bildschirm angezeigt wird.

PD 0 1=2 A

Auf der in Laufwerk Null befindlichen Diskette wird die PD-Tabelle dahingehend geändert, daß alle in Zeile 1 abgelegten PD-Daten durch die Werte der Zeile 2 ersetzt werden.

Das nachgestellte A bewirkt, daß diese Änderung sofort wirksam wird, ohne daß dazu ein neuer Systemstart notwendig wäre.

PD 0 1 TI=CK TD=E SEK=18 EIB=3 A

Innerhalb der PD-Tabelle der in Laufwerk Null befindlichen Diskette werden die Parameter TI, TD, SEK und EIB so geändert, daß eine einseitig beschriebene Diskette doppelter Schreibdichte mit 18 Sektoren pro Spur und drei Einheiten im Block verarbeitet werden kann.

Alle anderen in Zeile 1 der PD-Tabelle aufgeführten Parameter (SP, SWZ, SBIV, AEIV) bleiben unverändert. Die Änderung der PD-Daten wird durch das nachgestellte A sofort wirksam.

PD 0 2=5 SP=79 SWZ=3 A

In der PD-Tabelle der Diskette in Laufwerk Null werden die in Zeile 2 abgelegten Daten komplett durch die in Zeile 5 festgelegten Parameter ersetzt, davon abweichend werden zusätzlich die Parameter SP und SWZ so verändert, daß die Spurenzahl für Laufwerk zwei mit 79 und die Spurwechselzeit mit 40 Millisekunden definiert ist.

Auch diese Änderung wird durch das nachgestellte A sofort wirksam.

PD 0 A

Die PD-Tabelle der Diskette in Laufwerk Null wird auf dem Bildschirm angezeigt und die innerhalb der Tabelle festgelegten Daten neu initialisiert. Diese Funktion ist z.B. nach der Ausführung des DISK-Befehls sinnvoll, damit das Betriebssystem wieder zu den ursprünglichen Initialisierungsdaten zurückkehrt.

SYNTAX: PORT

=====

WIRKUNG : Der Befehl PORT listet alle Ports, deren INP-Signale ungleich FF hex/255 dez sind mit den jeweiligen Eingabewerten, die diese Ports momentan aufweisen, auf dem Bildschirm.

SYNTAX: PRINT Dateiname Startzeile Zeilenzahl

=====

WIRKUNG: Dieser Befehl hat die gleiche Funktion wie der Befehl LIST. Der Unterschied besteht lediglich in der Ausgabe. Während ein LIST auf den Bildschirm gelenkt wird, kommt es bei einem PRINT zu einer Druckerausgabe.

ANWENDUNG: Die Steuerungen und Anwendungsmöglichkeiten des PRINT-Befehls entnehmen Sie bitte der Erklärung zum LIST-Befehl.

SYNTAX: PROT Laufwerknummer Parameter

=====

WIRKUNG: Dieser Befehl erlaubt es, verschiedene auf der Diskette gespeicherte Kontrolldaten zu verändern. Als Parameter sind dazu folgende Eingaben möglich:

BKL
ZU, AUF
KW=Kennwort
NAME=Diskname
DATUM=Diskdatum

BKL Der Parameter BKL (=Bearbeitungskennzeichen löschen) bewirkt in Verbindung mit dem PROT-Kommando, daß die Bearbeitungskennzeichen aller im Inhaltsverzeichnis der Diskette aufgeführten Dateien zurückgesetzt werden.
Wird dem Parameter BKL ein Dateiname vorangestellt, so wird nur das Bearbeitungskennzeichen dieser einen Datei zurückgesetzt.

KW Mit dem Parameter KW=xxxxxxx wird der Diskette das neue Kennwort xxxxxxxx zugewiesen.

ZU Den auf der Diskette gespeicherten Dateien wird das Haupt-Kennwort der Diskette als deren Kennwort zugewiesen. Unsichtbare sowie Systemdateien bleiben unverändert.

AUF Die Kennworte der freien Dateien werden gelöscht.

DATUM=xxx Die Diskette in dem angegebenen Laufwerk wird mit dem neuen Datum xxx und/oder dem neuen Diskettenname=yyy versehen.
NAME=yyy

BEISPIELE: PROT 1 BKL

Die Bearbeitungskennzeichen aller Dateien in Laufwerk 1 werden gelöscht.

PROT TEST/BAS:0 BKL

Das Bearbeitungskennzeichen der Datei TEST/BAS in Laufwerk Null wird gelöscht.

PROT 1 NAME=NEUDISK DATUM=09.07.84

Auf der Diskette in Laufwerk 1 werden der Begriff NEUDISK als Diskettenname sowie als Datum der 09.07.84 eingetragen.

SYNTAX: PURGE Laufwerknummer Parameter

WIRKUNG: Der Befehl PURGE gestattet es, mehrere auf einer Diskette befindliche Dateien in einem Arbeitsgang zu löschen.

Dabei werden die Dateinamen der Reihe nach auf dem Bildschirm angezeigt und der Anwender wird gefragt, ob die entsprechende Datei gelöscht werden soll oder nicht. —

Die entsprechende Frage ist mit J für Ja, N für Nein oder E für Abbruch der Funktion zu beantworten.

Als Parameter können dem Befehl Dateiattribute mitgegeben werden, um die Auswahl der Dateinamen auf diejenigen zu beschränken, die eine oder mehrere Voraussetzungen erfüllen.

So kann z.B. eine Namensweiterung (/TYP) als Kennzeichen verwendet werden, um die Auswahl der angebotenen Dateien zu beschränken und nicht alle Systemdateien in die Abfrage mit einzubeziehen.

BEISPIELE: PURGE 0 FRD

Die freien Anwenderdateien (ohne System- und sichtbare Dateien) werden der Reihe nach auf dem Bildschirm gelistet, der Anwender wird über jede Datei gefragt, ob diese gelöscht werden soll oder nicht.

PURGE 0 /BAS

Alle Programme mit der Namensweiterung /BAS werden der Reihe nach auf dem Bildschirm angezeigt und dem Anwender die Entscheidung überlassen, ob die entsprechende Datei gelöscht werden soll oder nicht.

SYNTAX: R

WIRKUNG: Die Eingabe von R im Befehlsmodus des G-DOS bewirkt die Wiederholung des letzten DOS-Kommandos, das vor dem R-Befehl ausgeführt wurde. Deshalb kann diese Eingabe vollkommen unterschiedliche Auswirkungen haben, nur davon abhängig, welches die letzte Operation war, die G-DOS vor diesem Kommando ausgeführt hat.

ANWENDUNG: Wiederholung verschiedenener G-DOS-Kommandos durch eine einzige Tastatureingabe, ohne daß eventuell komplexe Befehlsfolgen noch einmal komplett neu eingegeben werden müßten.

Häufige Anwendung im Zusammenhang mit Kopierbefehlen, die in umfassender Form nur einmal eingegeben werden müssen, um dann durch das R-Kommando wiederholt werden zu können.

BEMERKUNG: Ist der Eingabe von R kein ausführbarer G-DOS-Befehl vorausgegangen, führt der Aufruf dieser Funktion zu einer Fehlermeldung.

BEISPIELE: Voraussetzung: Das letzte Kommando, welches G-DOS ausgeführt hat, war der Aufruf des Inhaltsverzeichnisses mit dem Befehl DIR 1 I S A.

Sollen die Inhaltsverzeichnisse mehrerer verschiedener Disketten angezeigt werden, so reicht es aus, die jeweilige Diskette in Laufwerk 1 einzulegen und danach R einzugeben.

Nach jedem R-Aufruf wird wiederum das Inhaltsverzeichnis der Diskette in Laufwerk 1 mit allen dazu definierten Parametern angezeigt.

Voraussetzung: Das letzte Kommando, das G-DOS ausgeführt hat, war eine Diskettenkopie mit der Kommando-
folge COPY 0 1,,FMT EDK ZPDN=2.

Nach Abschluß der ersten Diskettenkopie genügt zur Aktivierung eines weiteren Kopiervorganges die Eingabe von R im Befehlsmodus des Betriebssystems.

Das Kopier-Kommando wird mit allen vorher definierten Parametern neu gestartet.

SYNTAX: S Laufwerknummer Parameter

=====

WIRKUNG: Durch den Befehl S wird eine Fülle von Parametern festgelegt, welche die Initialisierungsdaten für das Betriebssystem nach einem Systemstart definieren.

Die Eingabe von S in Verbindung mit einer Laufwerknummer zeigt die Systemdaten, die auf der Diskette in dem angesprochenen Laufwerk gespeichert sind, auf dem Bildschirm an.

Die Parameter zur Beeinflussung der Initialisierungsdaten werden im folgenden in alphabetischer Reihenfolge erklärt:

- AA=x** Dieser Parameter definiert die Kennwortaktivierung. Ist AA=J, so sind die Kennworte aktiv, bei AA=N werden Kennworte nicht berücksichtigt.
- AB=x** Das System kann in den sogenannten RUN-ONLY-MODE versetzt werden. AB=J bedeutet, daß der Anwender keine Möglichkeit hat, einen Programmablauf zu unterbrechen oder in das Betriebssystem zu gelangen.
J als Zuordnung zum Systemparameter AB bedeutet, daß die BREAK-Taste abgeschaltet wird und die Systemdaten AD, AE und AF auf N gesetzt werden.
Bevor das Betriebssystem nach einer Umstellung auf RUN-ONLY gestartet wird, muß unbedingt ein AUTO-Kommando definiert worden sein, da das Betriebssystem jeden Zugang zur Befehlseingabe blockiert und mit der Diskette nicht mehr gearbeitet werden könnte. Gleichzeitig verhindert die Definition AB=J den Abbruch eines AUTO-Start durch Festhalten der NEW LINE-Taste.
- AC=x** Dieser Parameter legt die Funktion der Tastatur Entprellroutine fest:

J bedeutet, daß die Entprellung aktiv ist, bei N ist die Entprellroutine abgeschaltet. (Genie III: AC=N)
- AD=x** Ist AD=J definiert, so ist die Tastaturfunktion JKL zum Bildschirmausdruck aktiviert, AD=N unterdrückt die JKL-Funktion.

- AE=x** Die Tastatur-Funktion 123 zum Einsprung in den DEBUG-Monitor ist zugelassen, wenn AE=J ist, bei Definition von AE=N wird diese Tastatur-Funktion ignoriert.
- AF=x** Der Sprung in die Mini-Befehlseingabe (Mini-DOS) ist zugelassen, wenn AF=J definiert ist, bei AF=N zeigt die Tastaturfunktion DFG keine Auswirkung.
- AG=x** Die BREAK-Taste ist aktiv, wenn AG=J ist, die Festlegung AG=N sperrt die Funktion dieser Taste.
- AI=x** Dieser Parameter legt fest, ob Bit 4 der Adresse 436CH gesetzt ist oder nicht. Eine Reihe von Maschinenprogrammen (z.B. DEBUG) fragt diese Adresse ab, um die Zulässigkeit von Kleinbuchstabendarstellung abzuprüfen. Ist Bit 4 gesetzt (AI=J), so ist die Kleinbuchstabendarstellung zugelassen, ist dieses Bit nicht gesetzt (AI=N), ist diese Darstellung nicht erlaubt.
- AJ=x** Bestimmt, ob die G-DOS-Tastaturroutine oder die Level-II-BASIC-Tastaturabfrage aktiv ist. Ist AJ=J definiert, ist die Tastaturentprellung aktiv, die Tastenfunktionen 123, DFG und JKL werden abgefragt. Ist dieser Parameter auf N gesetzt, so entfallen diese Erweiterungen der normalen Tastaturabfrage. Unabhängig von der Voreinstellung dieses Systemparameters kann die Funktion AJ=N erzwungen werden, wenn während des Systemstarts die Hochpfeil-Taste festgehalten wird.
- AL=xx** Der als xx eingetragene Wert (zwischen 1 und 4) gibt an, wieviele Diskettenlaufwerke an der benutzten Anlage betrieben werden. Dadurch kann bei Befehlen ohne Laufwerks-Spezifikation die Suche auf diese Anzahl von Laufwerken begrenzt werden. Ist xx z.B. = 2, so existieren an der Anlage die Laufwerke 0 und 1.
- AM=xx** xx steht für die Anzahl der erlaubten Zugriffsversuche für Schreib- oder Leseoperationen auf der Diskette. Nach dieser Anzahl von Versuchen wird der Vorgang im Fehlerfall mit einer entsprechenden Meldung abgebrochen.
- AN=xx** Der Parameter AN definiert, auf welchem Laufwerk das Inhaltsverzeichnis der Diskette gelesen werden soll, wenn mit den Eingaben DIR oder I keine Laufwerknummer angegeben wurde.

- AO=xx** Dieser Parameter definiert das Laufwerk für die Eröffnung neuer Dateien. Soll eine Datei eröffnet werden, ohne daß eine Laufwerknummer ausdrücklich angegeben wurde, so werden die Disketten in den angeschlossenen Drives nach dem Vorhandensein einer Datei unter dem angegebenen Namen abgesucht. Ist die Datei noch nicht vorhanden, so wird sie auf der Diskette in dem durch den Parameter AO definierten Laufwerk angelegt.
- AP=xxxx** Durch Angabe dieses Parameters wird HIMEM gesetzt und damit der Speicherbereich ab Adresse xxxx reserviert. xxxx kann in hexadezimaler Schreibweise eingegeben werden, wobei der Zahl ein H anzuhängen ist, andernfalls geht das Betriebssystem von einer dezimalen Adressendefinition aus. Wie auch alle anderen Systemparameter wird AP nur beim Booten des Systems abgefragt.
- AQ=xx** Der Parameter AQ legt fest, ob die Funktion der CLEAR-Taste ein- oder abgeschaltet ist. Ist xx=J, so kann der Bildschirm durch Betätigung der CLEAR-Taste gelöscht werden, ist xx=N, so ist diese Funktion nicht aktiviert. Dabei kann der Bildschirm nach wie vor durch die Eingabe von CLS gelöscht werden.
- AR=xx** Dieser Parameter legt fest, ob eine Diskettenkopie (COPY-Syntax 5 und 6) ohne die Eingabe von Kennworten erlaubt ist oder nicht. Ist AR=J, so kann eine Kopie ohne Kennworte erfolgen, ist AR=N, so werden Kennworte verlangt. Der Parameter hat bei AA=N keine Wirkung.
- AS=xx** Dieser Parameter beeinflusst die Großbuchstabenumwandlung im BASIC. Ist AS=N, so werden nur die Befehle in Großbuchstaben umgewandelt, ist AS=J, so erfolgt eine generelle Umwandlung in Großbuchstaben durch den BASIC-Editor.
- AT=xx** Dieser Parameter legt fest, ob innerhalb von JOB-Dateien auch Tastatureingaben definiert werden können, die nicht durch NEW LINE abgeschlossen werden müssen. Ist AT=J, so ist diese Möglichkeit gegeben, bei AT=N dürfen dagegen in der JOB-Datei nur vollständige Befehle enthalten sein, während Einzelbuchstaben von der Tastatur eingegeben werden müssen.

AU=xx Wenn AU=J definiert wurde, so ist die automatische Wiederholung der gedrückten Eingabetaste (AUTO-REPEAT-FUNKTION) aktiviert, ist AU=N, so ist diese Funktion abgeschaltet.

AV=xx Der Parameter AV legt den Verzögerungsfaktor für die automatische Eingabewiederholung fest. Der Wert von xx bestimmt dabei, wieviele Zyklen von jeweils 25 Millisekunden vergehen sollen, bevor die AUTO-REPEAT-FUNKTION einsetzt.

AW=xx Der Parameter AW legt die Anzahl der Schreibversuche fest, die das Betriebssystem beim Auftreten von Prüfzahlfehlern vornimmt, bevor die Entscheidung über weitere Versuche oder den Abbruch der Routine dem Anwender überlassen wird.

AX=xx Der maximale ASCII-Code der auf einen Drucker auszugebenden Zeichen. xx muß dabei kleiner als 256 sein. Alle oberhalb des durch xx definierten Grenzwertes liegenden Zeichen werden bei der Druckausgabe als Leerzeichen ausgegeben.

AY=xx Ist der Parameter AY=J gesetzt, so wird mit dem Systemstart die Eingabe von Zeit und Datum vom Anwender verlangt. Ist der Parameter AY=N, so bleiben Datum und Zeit auf den intern gegebenen Werten. DIESER PARAMETER IST BEI GENIE III OHNE WIRKUNG !

AZ=xx Ein Parameter AZ=J bewirkt (wie bei AY=J) die Frage nach der Eingabe von Zeit und Datum. Ist AZ=N, so werden Zeit und Datum nicht auf Null gesetzt, sondern verbleiben auf dem Wert, den sie vor dem RESET hatten. DIESER PARAMETER IST BEI GENIE III OHNE WIRKUNG !

BC=xx Ist BC=J gesetzt, so kann der Anwender den Ablauf einer JOB-Routine durch eine Tastatureingabe abbrechen, ist BC=N, so wird diese Möglichkeit unterdrückt.

BD=xx Dieser Parameter legt fest, ob die Ausführung eines AUTO-Kommandos durch das Festhalten der NEW LINE- bzw. ENTER-Taste unterbunden werden kann oder nicht. Diese Möglichkeit besteht, wenn BD=J ist. Ist BD=N gesetzt, so kann der automatische Programmaufruf nicht unterbunden werden.

- BE=xx** Bei BE=J ist innerhalb der DOS-Ebene der R-Befehl aktiviert, der die Wiederholung der letzten Befehlseingabe bewirkt. Mit BE=N ist der R-Befehl inaktiv.
- BF=xx** In Abhängigkeit von diesem Parameter ist der Kleinschrifttreiber mit dem Systemstart aktiviert (BF=J) oder nicht (BF=N). Bei Aktivierung entspricht die Tastatur damit der einer Schreibmaschine (Großbuchstaben mit SHIFT).
(Die Funktion dieses Parameters ist bei GENIE III durch die LOCK-TASTE ersetzt.)
- BG=xx** Der Parameter legt fest, ob Groß-/Kleinschreibung zugelassen sind oder nicht. Ist BG=J definiert, wird die DOS-Funktion LC,J bei Systemstart automatisch aktiv, bei BG=N sind nur Großbuchstaben möglich.
- BH=xx** Bei BH=J ist mit dem Start des Betriebssystems automatisch das Blinken des Cursors aktiviert, ist BH=N definiert, so ist das Cursorblinken abgeschaltet (gilt nicht für den 80x25 Zeichenmodus).
- BI=xxx** Dieser Parameter gestattet es, das Cursorzeichen festzulegen. Sie können den ASCII-Code xxx des gewünschten Zeichens, sowohl in dezimaler als auch in hexadezimaler Schreibweise, (durch ein nachgestelltes H gekennzeichnet), eingeben.
- BJ=xx** Dieser Parameter ist in Relation zu der Taktfrequenz zu setzen, mit der der Rechner arbeitet. Die Bezugsgröße ist dabei die Taktfrequenz, mit der GENIE I und II arbeiten, nämlich 1.77 MHz. Der Wert von xx gibt nun den Faktor an, um den die Rechnerfrequenz schneller ist als diese Bezugsfrequenz.
- BK=xx** Dieser Parameter legt fest, ob die G-DOS-Funktion AIK zugelassen ist oder nicht. BK=J erlaubt die Funktion AIK, bei BK=N ist sie unterdrückt.
- BN=xx** BN=J bedeutet, daß die Adressmarke des Inhaltsverzeichnisses einem durch einen Double-Density-Controller geschriebenen Format entspricht.
Mit BN=N wird die Adressmarke im üblichen Format auf Single-Density-Disketten geschrieben.
Da G-DOS beide Adressmarken akzeptiert, ist dieser Parameter in erster Linie aus Kompatibilitätsgründen wichtig, um Disketten anderer Betriebssysteme verarbeiten zu können.

=====

WIRKUNG: Der Befehl SIO initialisiert die beiden seriellen Schnittstellen des Genie IIIs und zeigt die gültige Einstellung am Bildschirm an. Alle Parameter werden auf der Diskette gespeichert. Der Befehl ermöglicht den Datenaustausch durch einfache BASIC-Befehle. Dazu werden die Kassettenbefehle des BASIC so geändert, daß die Ein-/Ausgabe nicht auf einen Kassettenrekorder sondern an eine der beiden Schnittstellen geht. Es ist möglich, einen seriellen Drucker anzuschließen und bei Bedarf alle Druckausgaben auf diesen umzulenken.

ANWENDUNG: Initialisierung der seriellen Schnittstelle, Anzeige der gültigen Einstellung, Umleitung der Kassettenbefehle, Umleitung der Druckbefehle an einen seriell angeschlossenen Drucker.

Die möglichen Parameter sind:

SIO : Laufwerk Kanal Baudrate Parität Wortlänge
Stopbits Protokoll WAIT/NOWAIT PR/NOPR

Die Parameter bedeuten im einzelnen:

| | |
|------------|---|
| : Laufwerk | Laufwerk, auf dem die Parameter gespeichert werden sollen. Fehlt diese Angabe, wird automatisch Laufwerk :0 benutzt. Der Doppelpunkt muß angegeben werden. Diese Angabe muß die erste Angabe der Parameterliste sein! |
| Kanal | A oder B bezeichnet die Schnittstelle, deren Einstellung geändert werden soll. Fehlt diese Angabe, wird automatisch Kanal A angenommen. |
| Baudrate | Geschwindigkeit der Datenübertragung in Baud (Bits pro Sekunde). Möglich sind 19200, 9600, 4800, 2400, 1800, 1200, 600, 300, 200, 150, 134.5, 110, 75 und 50 Baud. |
| Parität | EVEN, ODD oder NO. Stellt die Parität der übertragenen Zeichen ein. |
| Wortlänge | 5, 6, 7 oder 8 Bits werden pro Zeichen gesendet. Üblich sind 7 oder 8 Bits. |
| Stopbits | 1, 2 oder 1.5 Stopbits sind möglich. Üblich sind 1 oder 2 Stopbits pro gesendetem Zeichen. |

Protokoll DTR, RTS oder XON sind möglich. Das Protokoll steuert den Datenaustausch mit dem angeschlossenen Gerät (z.B. einem Drucker), um Zeichenverluste zu verhindern.

DTR benutzt den DTR- bzw. DSR-Anschluß der Schnittstelle für die Steuerung. Dies ist beim Druckeranschluß üblich.

RTS benutzt den RTS- bzw. CTS-Anschluß für die Steuerung. Dies ist beim Datenaustausch mit anderen Rechnern üblich.

XON benutzt zur Steuerung die ASCII-Zeichen XON (Code 19) und XOFF (Code 17). Dies wird üblicherweise bei der Datenübertragung über Telefon benutzt, da keine zusätzlichen Leitungen zur Verfügung stehen. Der SIO-Befehl unterstützt dieses Protokoll nur beim Senden.

WAIT Die Angabe von Wait bewirkt beim Senden, daß das nächste Zeichen erst bei leerem Datenpuffer des SIO-Bausteins gesendet wird. Die Übertragung kann dadurch etwas langsamer werden. Benutzen Sie WAIT nur, wenn unerklärliche Übertragungsfehler auftreten.

NOWAIT hebt WAIT auf.

PR (PRINT) lenkt alle Druckausgaben auf die gewählte Schnittstelle um.

NOPR hebt die Wirkung von PR auf. Die Druckausgabe erfolgt durch die Standard-Druckerschnittstelle.

BEMERKUNG: Alle Parameter können in beliebiger Reihenfolge angegeben werden (bis auf :Laufwerk). Da alle Werte auf der Diskette gespeichert sind, werden nicht angegebene Parameter nicht verändert. Mit einem SIO-Befehl lassen sich beide Schnittstellen gleichzeitig initialisieren.

Da die Kassettenbefehle nach einem SIO-Befehl auf die serielle Schnittstelle wirken, ist es von BASIC aus möglich, durch PRINT#-1, bzw. PRINT#-2, Daten an den Kanal A oder den Kanal B auszugeben. Allerdings ist die Verwendung von TAB() oder ", " in solchen PRINT# Befehlen nicht möglich. Genauso lassen sich durch INPUT#-1, oder INPUT#-2, Daten von den Schnittstellen einlesen. Das gilt auch für den LINE INPUT# Befehl. Bei der Eingabe wartet der Rechner auf ein Zeichen von der Schnittstelle. Wenn nichts kommt, dann kann der Vorgang durch die BREAK-Taste jederzeit abgebrochen werden. Das gleiche gilt auch beim Senden von Zeichen an ein nicht bereites oder nicht vorhandenes Gerät.

BEISPIELE: SIO

Beide Schnittstellen werden initialisiert, alle Parameter bleiben erhalten.

SIO :1

Wie im ersten Beispiel, aber die Parameter werden von der Diskette im Laufwerk :1 übernommen.

SIO A 300 7 2 EVEN NOWAIT XON

Kanal A wird auf 300 Baud, 7 Bit Wortlänge, 2 Stopbits und EVEN-Parity (gerade Parität) gesetzt. WAIT wird abgeschaltet, es ist sowieso meist nur bei hohen Baudraten sinnvoll. Das Protokoll wird als XON/XOFF-Steuerung festgelegt, wie es für Modembetrieb notwendig ist. Diese Einstellung ist typisch für den Anschluß an einen entfernt aufgestellten Großrechner. Das Programm TERM, das bei TCS erhältlich ist, läßt ihr Genie IIIs als Terminal arbeiten. Die Einstellung von Kanal B wird nicht geändert.

SIO A 1200 B 9600 DTR WAIT 8 1 NO PR

Die Baudrate des Kanals A wird auf 1200 Baud eingestellt. Kanal B wird für die Ansteuerung eines seriellen Druckers vorbereitet (9600 Baud, DTR-Protokoll, 8 Bit Wortlänge, 1 Stopbit, NO Parity, Warten bis Puffer leer ist). PR aktiviert Kanal B als Drucker-schnittstelle.

SIO B NOPR

Die Druckerausgabe erfolgt wieder auf der Standard-Druckerschnittstelle.

SYNTAX: STMT Textzeile

=====

WIRKUNG: Die angegebene Textzeile wird in Großbuchstaben auf dem Bildschirm ausgegeben und der nächste Befehl ohne Unterbrechung ausgeführt.

ANWENDUNG: Dieses Kommando findet nur dann eine sinnvolle Anwendung, wenn es über einen DOS-Call angesprochen wird (z.B. JOB-Datei) der das Kommando nicht anzeigt.
Sie haben damit die Möglichkeit, Bildschirmmeldungen auszugeben, ohne die Ausführungen des DOS dabei zu unterbrechen. Es können über STMT auch mehrere Zeilen angezeigt werden. In diesem Fall muß es sich jedoch beim letzten Zeileninhalt um ein PAUSE-Kommando mit oder ohne Textzeile handeln.

BEMERKUNG: Die Eingabe des STMT-Befehls als direktes Kommando ist nicht sinnvoll, da der Text in diesem Fall sowieso auf dem Bildschirm erscheint.

BEISPIEL: Ausgehend von der Tatsache, daß ein Programm über eine JOB-Datei abgearbeitet wird und der Benutzer darüber informiert werden soll, welcher Programmteil sich gerade in der Bearbeitung befindet, lassen sich über das Kommando STMT derartige Meldungen einblenden, ohne daß das DOS in seinen Ausführungen unterbrochen wird.

STMT Datensicherung ist beendet!
PAUSE Bitte die Originaldisk in Laufwerk 1 legen!

Wenn sich die Meldungen über mehrere Zeilen hinziehen, muß am Ende ein PAUSE-Kommando stehen. In diesem Beispiel wartet der Rechner auf den Druck auf die NEW LINE-Taste, bevor er seine Arbeit fortsetzt.

SYNTAX: UHR Parameter

=====

WIRKUNG: Mit der Eingabe UHR J wird in der rechten oberen Ecke des Bildschirms die aktuelle Zeitanzeige eingeschaltet.

Obwohl die Uhr geräteintern kontinuierlich weiterläuft, kann es in der Anzeige zu Sprüngen in der Sekundenanzeige kommen, da systeminterne Zugriffe die laufende Zeitanzeige unterbrechen.

Durch die Eingabe UHR N wird die Anzeige der Uhrzeit auf dem Bildschirm wieder abgeschaltet.

ANWENDUNG: Darstellung der Uhrzeit auf dem Bildschirm während anderer Abläufe innerhalb des Betriebssystems und auch während der Verarbeitung von BASIC-Programmen.

BEMERKUNG: Wird im G-DOS der Befehl 0 ausgeführt, so erfolgt damit automatisch auch eine Abschaltung der durch UHR J aktivierten Zeitanzeige.

BEISPIELE: UHR J

Die Zeitanzeige auf dem Bildschirm wird aktiviert, in der rechten oberen Bildschirmecke erscheint die Uhrzeit im Format HH:MM:SS

UHR N

Die Uhrzeitanzeige auf dem Bildschirm wird wieder abgeschaltet.

Aufruf aus dem BASIC:

```
10 CLS
20 E$=INKEY$ : IF E$="" THEN 20
30 CMD"UHR J"
40 A$=INKEY$ : IF A$="" THEN 40
50 CMD"UHR N"
60 GOTO 10
```

Dieses BASIC-Programm gestattet das Ein- und Ausschalten der Uhrenanzeige auf dem Bildschirm durch jeweiligen Druck auf eine beliebige Taste. Das Programm ist durch Druck auf die BREAK-Taste zu beenden.

SYNTAX: V+ Parameter

WIRKUNG: Dieser Befehl bewirkt, daß mit jedem Schreibvorgang eines Sektors eine Verifizierung der gespeicherten Daten erzwungen wird.

Als Parameter können J oder N übergeben werden, um die Verifizierung ein- oder auszuschalten.
V+ ohne Parameter wird automatisch als V+ J interpretiert.

=====

WIRKUNG: Mit Hilfe dieses Befehls können Sie die im Genie III eingebaute serielle Schnittstelle (V24) initialisieren und einstellen.

Weiterhin ist dadurch die Möglichkeit gegeben, zwei über V24 miteinander verbundene Rechner auf ordnungsgemäße Datenübertragung zu testen.

ANWENDUNG: Nach der Eingabe des V24-Kommandos meldet sich das Genie III mit einem Menue. Gleichzeitig wird dabei eine Standardeinstellung der seriellen Schnittstelle vorgenommen, die unter dem Menue zu sehen sind. Wenn Sie eine von dieser Standardeinstellung abweichende Übertragungsart festlegen wollen, erlaubt das Menue folgende Eingaben:

F Übertragungs-Format
B Baud Rate
S Start Communication
E Ende

Die Wortlänge wird im Übertragungs-Format definiert. Sie haben die Wahl zwischen 5, 6, 7 und 8 Bit Worten. Weiterhin muß im Format die Anzahl der Stop-Bits festgelegt werden. Geben Sie hier den entsprechenden Buchstaben ein (A=1, B=1.5, C=2). Zuletzt kommt es noch zur Einstellung der "Parity". Es genügt, wenn jeweils der erste Buchstabe von Odd, Even oder No angegeben wird.

Sollte die gewünschte Baud-Rate von der Standard-Einstellung abweichen, so haben Sie nach der Eingabe von B über eine Tabelle eine Baud-Rate zwischen 50 und 38400 einzustellen.

Um nach Anpassung der beiden miteinander verbundenen Geräte zutesten, ob die Datenübertragung einwandfrei vollzogen wird und die Schnittstelle zu aktivieren, muß über S die Verbindung der Geräte hergestellt werden. Die jeweiligen Tastatureingaben werden dann beim Partnergerät auf dem Bildschirm ausgegeben. Die Datenübertragung bricht ab, sobald die BREAK-Taste betätigt wird.

Ein Druck auf E beendet die V24-Routine. Die geänderten Daten zur Ansteuerung des seriellen Ports bleiben jedoch erhalten bis der Rechner abgeschaltet oder eine neue Einstellung vorgenommen wird.

SYNTAX: Z Parameter (gilt nur für Genie III und IIIs)

=====

WIRKUNG: Mit Hilfe dieses Kommandos können Sie beim Genie III den Zeichensatz verändern.

Folgende zwei Parameter sind möglich:

- Z X Alle Zeichen mit einem ASCII-Wert von 128 bis 191 werden in Blockgrafik-Darstellung umgeschaltet.
- Z Y Die Blockgrafik der Zeichen mit einem ASCII-Wert von 128 bis 191 wird unterbunden. Der Rechner bringt unter den ASCII-Werten 128 bis 255 nochmals die Zeichen von 0 bis 127 jedoch in invertierter Darstellung auf den Bildschirm. Beim Genie IIIs ist mit Z Y auch Fett- oder Kursivschrift möglich (siehe Befehl ZL).
- Z (nur Genie IIIs) An den Bildschirm wird das Steuerzeichen 22 (Umschaltung auf inverse Schrift) ausgegeben. Die Umschaltung durch PRINT CHR\$(22); in BASIC ist weiterhin möglich.

ANWENDUNG: Unter BASIC wird beim Genie III durch den Befehl CMD"Z Y" die Möglichkeit gegeben, inverse Schrift auf dem Bildschirm darzustellen. Nach der Eingabe des CMD"Z Y" arbeitet der Rechner jedoch noch in normaler Schrift. Erst durch die Ausgabe des ASCII-Wertes 22 wird die Inverssschrift aktiv. Eine Abschaltung der Inverssschrift erfolgt ebenfalls über dieses Zeichen.

BEISPIEL: Schreiben Sie das BASIC-Programm

```
10 CLS
20 CMD"Z Y"
30 PRINT"Inverssschrift nur über PRINT CHR$(22)"
40 PRINT"Normalschrift ",CHR$(22),"Inverssschrift"
50 PRINT CHR$(22),"Wieder Normal"
```

und starten Sie es mit RUN. Der Rechner wird den Text in der entsprechenden Darstellungsform auf dem Bildschirm ausgeben.

SYNTAX: ZEIT

oder

ZEIT hh:mm:ss

=====

WIRKUNG: Wird nur das Befehlswort ZEIT eingegeben, so erfolgt die Anzeige der aktuellen Systemzeit auf dem Bildschirm.

Wenn als Parameter hh:mm:ss (Stunde:Minute:Sekunde) angegeben werden, so wird die Systemzeit auf dieses Werte gesetzt und (bei Genie III) als laufende Zeit von der eingebauten Echtzeituhr weitergeführt.

BEISPIELE: ZEIT

Die aktuelle Systemzeit wird auf dem Bildschirm angezeigt.

ZEIT 12:50:00

Die Systemzeit wird auf 12:50 Uhr gesetzt und beim Genie III auch von der Echtzeituhr übernommen.

=====

WIRKUNG: Mit diesem Kommando können Sie beim Genie IIIs einen Zeichensatz von der Diskette laden.

Folgende Parameter sind möglich:

- | | |
|-------------|---|
| ZL Datei | Die Datei wird in den Zeichengenerator kopiert. Die Umschaltung ist sofort wirksam. |
| ZL Datei X | Die Datei wird in den Zeichengenerator geladen. Danach werden unabhängig vom Inhalt der Datei auf den Zeichen 128 bis 191 Blockgrafikzeichen dargestellt. Die Wirkung ist die gleiche wie bei ZL Datei und anschliessendem Z X. |
| ZL Datei Y | Die Datei wird in den Zeichengenerator geladen. Dabei werden in den Zeichen 128 bis 255 inverse Zeichen erzeugt. |
| ZL Datei YA | Statt der inversen Zeichen werden ALTERNATE-Zeichen (Schattenschrift) erzeugt. |
| ZL Datei YB | Erzeugt BOLD-Zeichen (Fettschrift) |
| ZL Datei YC | Erzeugt ITALIC-Zeichen (Kursivschrift) |

HINWEIS: Nachfolgende Z X - und Z Y - Befehle schalten zwischen Grafik und der beim letzten ZL gewählten Schriftart um (Y, YA, YB oder YC). Diese Umschaltung betrifft nur die Zeichen 0-9 und die entsprechenden Sonderzeichen. Inverse oder Schräggestellte Zeichen werden dabei immer aus dem Standardzeichensatz entnommen. Benutzen Sie, wenn ihnen das nicht gefällt den Befehl ZL Datei mit Zusatz Y, YA, YB oder YC.

Zeichensätze werden mit dem Programm CHARDEF/BAS, das bei TCS erhältlich ist, erzeugt. Auf ihrer Systemdiskette befinden sich nur die Zeichensätze STD (Standard) und ASCII (Amerikanisch). Als Datei ist also nur STD oder ASCII zulässig.

BEISPIELE: ZL STD

Dieser Befehl lädt den Standardzeichensatz in den Zeichengenerator. Er ändert nur die ersten 128 Zeichen, da der Zeichensatz STD nur 128 Zeichen lang ist.

ZL ASCII Y

Dieser Befehl lädt den amerikanischen Zeichensatz. Statt der Umlaute werden eckige Klammern erzeugt. Zusätzlich werden alle Zeichen invertiert abgespeichert. Überzeugen Sie sich durch den Befehl INFO von der Wirkung dieses Befehls.

ZL STD YB

Dieser Befehl lädt den Standardzeichensatz und erzeugt zusätzlich Fettschrift. Geben Sie jetzt den Befehl Z ein, dann erscheinen alle Ausgaben in Fettschrift.

ZL ASCII X

Dieser Befehl lädt den amerikanischen Zeichensatz und erzeugt Grafikzeichen.

ZL STD Y
Z X
INFO

Diese Befehlsfolge stellt den Zustand nach dem Einschalten des Rechners her und zeigt den aktiven Zeichensatz an.

SYNTAX: 0,VON=xxxx,BIS=yyyy,HIMEM=zzzzz

=====

WIRKUNG: Alle Speicherzellen ab der Speicheradresse xxxx bis einschließlich der Adresse yyyy werden mit Nullen ausgefüllt, alle vorherigen Speicherinhalte in diesem Bereich gelöscht.

Als neue Grenze des für BASIC-Programme nutzbaren Speicherraumes wird die unter dem Parameter HIMEM angegebene Adresse zzzz definiert.

Alle vorher aktivierten Umleitungen (siehe Befehl #) werden zurückgesetzt.

Eventuelle Hintergrund-Maschinenprogramme (z.B. Spooler, Packer etc.) werden abgebrochen und gelöscht.

Alle zugeschalteten Routinen werden wieder aus der Interrupt-Kette entfernt, z.B. eine eingeschaltete Zeitanzeige auf dem Bildschirm aufgehoben.

ANWENDUNG: Das Ausnullen eines kompletten Speicherbereiches ist immer dann sinnvoll, wenn zu befürchten ist, daß Speicherinhalte innerhalb des zu löschenden Bereiches den weiteren Programmablauf behindern.

Dieser Effekt tritt zuweilen bei der Nutzung von Maschinen-Unterprogrammen auf, die vorher in einen geschützten Speicherbereich geladen worden sind.

In diesem Fall ist es zweckmäßig, vor dem Einlesen eines neuen Programmes den Arbeitsspeicher komplett "zu reinigen", indem die einzelnen Zellen auf den Wert Null gesetzt werden.

Sinnvoll ist die Verwendung dieses Befehls zudem, wenn eine bestimmte Routine in den Arbeitsspeicher geladen, dort modifiziert und danach wieder auf Diskette gedUMPT werden soll.

Alle nicht genutzten Speicherplätze innerhalb des gedUMPTen Bereiches sind danach bei vorangegangenen Ausnullen des Speichers leichter zu identifizieren.

BEMERKUNG: Werden mit dem Befehl keine weiteren Parameter übergeben, so erhält VON automatisch den Wert 5200H, BIS den Wert FFFFH und HIMEM den Wert FFFFH. In diesem Falle wird also der komplette freie Arbeitsspeicher gelöscht.

(Siehe dazu auch HIMEM und DUMP).

BEISPIELE: 0,VON=7000H,BIS=8FFFFH,HIMEM=F000H

Diese Eingabe bewirkt, daß alle Speicherzellen ab der Adresse 7000H bis zur Adresse 8FFFFH einschließlich den Inhalt Null erhalten.

Der Speicherbereich zwischen 5200H und 6FFFFH bleibt unverändert, ebenso alle Speicherzellen oberhalb der Adresse 8FFFFH.

Der Wert für HIMEM wird auf F000H festgesetzt.

0,HIMEM=EF00H

Der gesamte Speicherbereich zwischen 5200H und FFFFFH wird gelöscht, alle innerhalb dieses Bereiches liegenden Speicherzellen erhalten den Inhalt Null.

Die HIMEM-Grenze wird auf die Adresse EF00H festgelegt.

0

Der gesamte Speicherbereich zwischen 5200H und FFFFFH wird gelöscht, alle innerhalb dieses Bereiches liegenden Speicherzellen erhalten den Inhalt Null.

Die HIMEM-Grenze wird automatisch auf FFFFFH festgelegt, also kein Speicherbereich für die Nutzung durch ein geschütztes Programm reserviert.

WIRKUNG: Die Bildschirmdarstellung wird auf das Format 64 Zeichen mal 16 Zeilen umgestellt.

ANWENDUNG: Wiederherstellung der Initialisierungsdaten für die Bildschirmausgabe, Vorbereitung auf die Verarbeitung von 64-Zeichen-Software (z. B. von Genie I/II oder Speedmaster).

ERGÄNZUNG: Die Bildschirm-Kontrollcodes bei Genie III

| DEZ | HEX | FUNKTION |
|-----|-----|--|
| 03 | 03 | Ende der Textausgabe durch das Betriebssystem |
| 07 | 07 | Akustisches Signal ausgeben |
| 08 | 08 | Cursor eine Pos. zurück, altes Zeichen löschen |
| 09 | 09 | Cursor eine Tabulatormarke vor |
| 10 | 0A | Cursor an den Anfang der Folgezeile, diese löschen |
| 11 | 0B | wie 0A |
| 12 | 0C | wie 0A |
| 13 | 0D | wie 0A |
| 14 | 0E | Cursor anschalten |
| 15 | 0F | Cursor ausschalten |
| 22 | 16 | Invertierung (Bit 7 bei alphanum. Zeichen Ein/Aus) |
| 23 | 17 | Zeichenabstand verdoppeln |
| 24 | 18 | Cursor eine Position zurück |
| 25 | 19 | Cursor eine Position vor |
| 26 | 1A | Cursor abwärts |
| 27 | 1B | Cursor aufwärts |
| 28 | 1C | Cursor auf Position Null zurück |
| 29 | 1D | Cursor zurück zum Zeilenanfang |
| 30 | 1E | Ab Cursorposition bis zu Ende der Zeile löschen |
| 31 | 1F | Ab Cursorposition Bildschirm bis zur Endposition löschen |

=====

WIRKUNG: Die Bildschirmdarstellung wird auf das Format 80 Zeichen mal 24 Zeilen umgestellt.

Bestehende BASIC-Programme, die auf eine Bildschirmdarstellung von 64 Zeichen mal 16 Zeilen ausgerichtet sind, müssen an das neue Format angepaßt werden.

Da Teile der Tastatur- und Bildschirmroutinen verändert werden, kann es beim Laden von unmodifizierten Maschinenprogrammen zu Systemabstürzen mit allen damit verbundenen negativen Begleiterscheinungen kommen.

ANWENDUNG: Bessere Ausnutzung des Bildschirmplatzes, interessant vor allem für Programme, die größere Mengen an Information gleichzeitig auf dem Monitor darstellen sollen (z.B. Textverarbeitung etc.)

BEMERKUNG: Der Cursor, der sich normalerweise mit der Systemoption BI definieren läßt, ist im 80-Zeichen-Modus nicht mehr variabel, hier wird er immer als schmaler Strich unter der nächsten Eingabeposition angezeigt.

Aus dem BASIC sind die Bildschirmpositionen in der unteren Bildschirmhälfte **n i c h t** mit dem POKE-Befehl erreichbar. POKE und PEEK zur Modifizierung bzw. Abfragen von Bildschirmpositionen sind nur für den Adressbereich von 15360 bis 16383 zulässig.

Der untere Bildschirmteil kann dagegen über die Funktion PRINT@ erreicht werden.

Während im 64-Zeichen-Modus durch diese Funktion die Positionen von Null bis 1023 erreichbar sind, können im 80-Zeichen-Betrieb insgesamt 1920 Bildschirmpositionen direkt angesprochen werden. Der Bereich für PRINT@ liegt dabei zwischen Null und 1919.

SYNTAX: @

oder

@,KEINE

oder

@,statt,an,...,an

=====

WIRKUNG: Dieser Befehl können Drucker- und Monitorausgaben, sowie Tastatureingaben umgeleitet werden.

Die Eingabe von @ ohne weitere Angaben bewirkt die Anzeige der momentan aktivierten Umleitungen.

Die Eingabe von @,KEINE löscht alle Umleitungen, die momentan aktiv sind.

Die Eingabe von "@,statt,an" ermöglicht es, Ein- oder Ausgabekanäle des Rechners den jeweiligen Erfordernissen entsprechend umzuleiten.

Diese Kanäle sind im einzelnen

| | | |
|-------|---|---------------------------------|
| MO | = | Bildschirmausgabe |
| DR | = | Druckerausgabe |
| TA | = | Tastatureingabe |
| NL | = | Niemandsland |
| ST=xx | = | Speicherstelle xx (hexadezimal) |

ANWENDUNG: Der Befehl findet seine Anwendung, wenn Ein- oder Ausgaben verschiedener Kanäle Ihres Systems auf einen anderen Kanal umgeleitet werden sollen.

So kann z.B. eine Druckerausgabe, die innerhalb eines Programmablaufes erforderlich ist, auch auf den Bildschirm umgeleitet werden, falls ein Drucker momentan nicht zur Verfügung steht oder innerhalb der Programmentwicklung die für eine Druckausgabe bestimmten Daten überprüft werden sollen.

Der Parameter NL gestattet es, einen bestimmten Kanal vollkommen für Ausgaben zu sperren.

ST als Parameter kennzeichnet den Start einer oberhalb von 51FFH beginnenden Maschinen-Routine, die als CALL behandelt wird und dementsprechend durch ein RET abgeschlossen werden muß. Die ersten 16 Byte sind dabei für eine interne Nutzung durch das Betriebssystem zu reservieren, die Registerinhalte müssen gesichert werden. G-DOS übergibt bei Ausgaben das entsprechende Byte in Register C und erwartet den Eingabewert in Register A, wobei der Wert Null als keine Eingabe gewertet wird.

Bei Abschluß der Routine über RET müssen die Register wieder ihre ursprünglichen Werte zurückerhalten haben.

BEMERKUNG: Die Anwendung des Parameters ST setzt fundierte Kenntnisse der Maschinensprachen-Programmierung voraus und sollte dementsprechend nur seine Anwendung finden, wenn diese Voraussetzung erfüllt ist.

BEISPIELE: @ dr mo

Diese Eingabe bewirkt, daß alle Ausgaben, die programmgemäß über einen Drucker erfolgen sollen, auf das Bildschirm-Display umgeleitet werden.

Diese Art der Umleitung wirkt auch auf das DOS-Kommando JKL, welches einen Ausdruck des Bildschirm-inhaltes bewirkt.

Die DOS-Funktion DR läßt sich nicht auf den Bildschirm umleiten, falls dabei ist kein Drucker angeschlossen ist, erscheint auf dem Display eine Fehlermeldung (Drucker nicht bereit).

@ TA TA ST=EF00H

Eine Eingabe wird sowohl von der Tastatur als auch von der Maschinen-Routine, deren Beginn bei Speicherplatz EF00H liegt, akzeptiert.

@ MO NL

Alle Bildschirmausgaben werden unterdrückt, bis die Umleitung durch ein gegenteiliges Kommando wieder aufgehoben wird.

@

Umleitungen, die momentan aktiv sind, werden auf dem Bildschirm angezeigt.

@ KEINE

Alle aktivierten Umleitungen werden aufgehoben (wie auch bei Eingabe von 0 für Speicherlöschung)

SYNTAX: > Parameter

WIRKUNG: Das Zeichen ">" ist die Abkürzung für den Befehl COPY. Diesem Befehl sind im vorliegenden Handbuch mehrere Seiten gewidmet. Schlagen Sie bitte für weitere Erläuterungen in diesem Abschnitt nach.

BEISPIEL: > 0 1

Diese Kombination entspricht dem Befehl

COPY 0 1

(Weitere Beispiele siehe unter COPY).

SYNTAX: & oder
 &,N

WIRKUNG: Die Eingabe des Zeichens "&" ohne Parameter bewirkt, daß vor der Ausführung eines Programmes DEBUG automatisch aufgerufen wird, um z.B. die aktuellen Register- und Speicherinhalte zu überprüfen.

Diese vorgeschaltete Routine kann durch die Eingabe von "&,N" wieder aufgehoben werden.

ANWENDUNG: Die Anwendung dieses Befehls liegt in erster Linie innerhalb der Programmerstellung und -entwicklung, wo die Veränderung von Registerdaten durch den Programmablauf überprüft werden soll.

SYNTAX: ?

WIRKUNG: Dieser Befehl ist gleichbedeutend mit LIB und zeigt die möglichen G-DOS-Befehle auf dem Bildschirm an.

SYNTAX: ## Parameter (gilt nur für Genie III und IIIs)

=====

WIRKUNG: Das GENIE III/IIIs verfügt über einen 2k Video-Speicher. Der Befehl ## dient dazu, die zweite Bildschirmseite nutzbar zu machen, da diese über eine direkte Adressierung nicht erreichbar ist.

Wird ## ohne Parameter eingegeben, so wird die aktuelle Bildschirmseite auf die zweite, nur indirekt adressierbare, Videoseite kopiert.

Durch ein Komma oder Leerzeichen abgetrennt können außerdem folgende Parameter angegeben werden:

- N (neu) erlaubt die Wiederherstellung der Bildschirmparameter, wenn diese z.B. durch ein Anwenderprogramm in Basic direkt geändert worden sind, ohne die entsprechenden G-DOS-Routinen dafür zu benutzen.
- V (voll) schaltet auf eine Bildschirmdarstellung im Format 64x32. Dieses Format bedarf jedoch einer Feineinstellung des Monitors am Rechner. Die JKL-Routine ist in der Lage, eine Hardcopy der gesamten Bildschirmseite zu erstellen.
- H (halb) schaltet auf normales Bildschirmformat 64x16 (wie bei GENIE I/II)
- S (spezial) Bildschirmdarstellung im Format 64x24, wobei 16 Zeilen in der Mitte des Bildschirms dargestellt werden, oberhalb (4300H-43FFH) und unterhalb (4000H-40FFH) dieses Bereiches können jeweils vier Zeilen als Kommentarzeilen angesprochen werden (auch hier Adressierung nur indirekt über ##, ##,T oder ##,\$ möglich!).
- X Bildschirmdarstellung im Format 80x24
- T (Tausch) tauscht den Inhalt der beiden Bildschirmseiten untereinander aus
- \$ kopiert den Inhalt des Arbeitsspeichers ab FC00H in den zweiten Bildschirmbereich.

HINWEIS: Bei der Ausführung des Befehls ## \$ aus dem BASIC heraus ist hinter dem \$-Zeichen noch ein Komma zu setzen. Ansonsten gibt der Rechner eine Fehlermeldung aus.

Ziffer (von 0 bis 7) legt den Beginn des dargestellten Bildschirmbereiches fest. Dadurch ist eine vertikale Verschiebung des Beginns der Darstellung um jeweils vier Bildschirmzeilen (256 Byte)

Programmunterbrechungen:
=====

Es sind drei verschiedene Tastenkombinationen möglich, mit denen bei entsprechender Voreinstellung der Systemparameter ein Programmablauf unterbrochen werden kann. Alle Kombinationen arbeiten, indem drei Tasten gleichzeitig (!) gedrückt werden und können demnach nur innerhalb von Programmteilen funktionieren, in denen die Tastaturabfrage aktiv ist. Möglich sind im einzelnen die Kombinationen

- J K L Der Bildschirminhalt wird auf den angeschlossenen Drucker ausgegeben
- l 2 3 Das Programm DEBUG wird gestartet
- D F G Sprung in die Mini-DOS-Ebene (Mini-Befehlseingabe)

Die Beschreibungen von JKL, DEBUG und Mini-DOS sind an anderer Stelle in diesem Handbuch aufgeführt.

Das Mini-DOS:

=====

Dieser Betriebssystem-Modus ist installiert worden, um auch innerhalb eines BASIC-Programmes DOS-Befehle verarbeiten zu können, ohne daß ein DOS-Aufruf über ein vorangestelltes "CMD" notwendig wäre.

Der Befehlssatz innerhalb der Mini-DOS-Ebene ist allerdings gegenüber dem gesamten Befehlsvorrat des G-DOS eingeschränkt: Die Befehle COPY, NDF, APPEND, DO, CONT und F# sind in diesem Modus nicht ausführbar, alle anderen G-DOS-Befehle können angewendet werden.

Syntax: M> Dateiname:QLW# Dateiname:ZLW#

=====

ANWENDUNG: Dieses sogenannte Mini-COPY-Kommando findet seine Anwendung in erster Linie in der Mini-DOS-Ebene, wo die normalen COPY-Funktionen ausgeschlossen sind, kann aber gleichermaßen in der normalen DOS-Ebene verwendet werden.

Mit Mini-COPY ist es möglich, eine einzelne Datei zu kopieren, wobei die komplette Eingabe der Dateinamen sowie der an der Kopieroutine beteiligten Laufwerke (QLW# als Quell-, ZLW# als Ziellaufwerk) notwendig ist.

Ein normales COPY-Kommando arbeitet, indem der Arbeitsspeicher des Rechners als Zwischenablage für die von der Quelldiskette eingelesenen Daten fungiert. Von dort aus werden die Daten dann erst auf die Zieldiskette übertragen.

Da das Mini-DOS in der Regel aufgerufen wird, während sich ein Programm im Arbeitsspeicher befindet und bearbeitet wird, fehlt hier natürlich der Raum für die Zwischenablage der Daten beim COPY-Kommando. Ein COPY-Kommando könnte in diesem Fall bei großen zu kopierenden Dateien mit der Meldung ZU WENIG SPEICHER abgebrochen werden.

In diesem Fall hilft die Mini-COPY-Funktion, da hier nur kleine Datenmengen in den Arbeitsspeicher übertragen und von dort aus auf die Zieldiskette geschrieben werden.

Bei der Kopie einer einzelnen Datei mit dem Mini-COPY-Kommando "M>" ist unbedingt der volle Name der Zieldatei sowie die dazugehörige Laufwerksnummer anzugeben.

Beispiel: M> DATEIL/DAT:1 DATEIL/DAT:0

Mit Hilfe diese Kommandos innerhalb der Mini-DOS-Ebene wird ein File mit dem Namen DATEIL/DAT von einer Diskette in Laufwerk 1 auf die Diskette in Laufwerk 0 mit dem gleichen Namen kopiert.

Syntax: ! DEMO/JOB

=====

Durch die Eingabe eines Aufrufezeichens und nachfolgendem Dateinamen kann innerhalb der Mini-Befehlseingabe eine JOB-Datei aktiviert werden (die Verwendung von DO ist innerhalb der Mini-DOS-Ebene unzulässig!).

Syntax: ;

=====

Das Semikolon (;) bewirkt, daß die Bearbeitung wieder zu dem Punkt zurückkehrt, von dem aus das Mini-DOS aufgerufen wurde.

Syntax: /

=====

Ein Schrägstrich (/) bewirkt dagegen, daß das Betriebssystem zur normalen G-DOS-Befehlseingabe springt. Ein Programmablauf, aus dem das Mini-DOS aufgerufen wurde, wird in diesem Fall also vollkommen abgebrochen.

Das Programm DEBUG:

=====

Mit dem System-Programm DEBUG besitzen Sie auf Ihrer G-DOS-Diskette eine wertvolle Hilfe zur Programmierung in Maschinensprache.

Sie haben die Möglichkeit, laufende Programme zu unterbrechen und den Speicherinhalt des Rechners sowie die Daten auf der Diskette zu kontrollieren oder zu modifizieren. Ein über DEBUG unterbrochenes Programm kann in Einzelschritten weitergeführt werden.

DEBUG wird durch das gleichzeitige Drücken der Tasten 1,2, und 3 aufgerufen (vorausgesetzt, daß der Systemparameter AE=J gesetzt ist).

Außerdem wird DEBUG gestartet, wenn im Programm ein Breakpoint vorhanden ist (Befehl RST 30H) oder ein Sprung auf die Adresse 440DH erfolgt.

Wenn DEBUG über das DOS-Kommando "&" aktiviert wurde, wird es vor der Ausführung eines Programmes automatisch gestartet (siehe auch &-Befehl).

Nach dem Start von DEBUG werden alle Register im Benutzer-Stack gerettet und alle Breakpoints gelöscht. Der Stack des Rechners wird von DEBUG benutzt.

Die Position des Cursors auf dem Bildschirm an der untereren linken Ecke zeigt Ihnen an, daß DEBUG auf eine Eingabe wartet. Eingaben müssen immer durch ein NEW LINE bzw. ENTER abgeschlossen werden. Sie haben somit noch die Möglichkeit, Eingabefehler durch Betätigung der Backspace-Taste (LINKSPFEIL) zu korrigieren. SHIFT-Backspace löscht die gesamte Eingabe.

Nachdem DEBUG gestartet ist, erfolgt die Anzeige des Speichers in hexadezimalen Werten (links) und in ASCII-Zeichen (rechts).

DEBUG kann über den Befehl G4400 oder Q verlassen werden. Wenn Sie nach einer Programmunterbrechung durch DEBUG das Kommando G ohne Startadresse eingeben, wird der normale Programmablauf an der Unterbrechungsstelle wieder aufgenommen.

Sollte Sie ein Kommando des DEBUG falsch eingeben oder das von Ihnen eingegebene Kommando nicht in der Befehlsliste existieren, meldet sich DEBUG mit der Frage "wie?" und wartet bis NEW LINE bzw. ENTER gedrückt wird. Erst dann ist DEBUG bereit, weitere Kommandos entgegenzunehmen.

Die Befehle des DEBUG:

- ; Im S-Modus (siehe dort) läßt sich durch diese Eingabe der nächste 256 Byte lange Block auf dem Bildschirm darstellen.
- Im X-Modus (siehe dort) wird der nächste 64 Byte lange Block im Bereich 1 dargestellt.
- n; Über n können die im X-Modus dargestellten Bereiche 2 und 3 angegeben werden, um dort den nächsten 64 Byte langen Block darzustellen.
- Im S-Modus läßt sich durch diese Eingabe der vorherige 256 Byte lange Block auf dem Bildschirm darstellen.
- Im X-Modus wird der vorherige 64 Byte lange Block im Bereich 1 dargestellt.
- n- Über n können die im X-Modus dargestellten Bereiche 2 und 3 angegeben werden, um dort den vorherigen 64 Byte langen Block darzustellen.
- C Das C (wie Call) hat eine ähnliche Wirkung wie der Befehl I, jedoch mit dem Unterschied, daß in dem durch DEBUG unterbrochenen Programm ein Unterprogramm vollständig (ohne Unterbrechung) abgearbeitet wird.
- Dxxxx Über diesen Befehl, bei dem xxxx eine hexadezimal anzugebende Speicheradresse ist, haben Sie die Möglichkeit, im S-Modus eine beliebige Speicherstelle anzuspringen, von der ab ein 256 Byte langer Speicherbereich angezeigt wird.
- Im X-Modus hat dieses Kommando die gleiche Wirkung auf den 64 Byte langen Block im Bereich 1.
- nDxxxx Sollen im X-Modus in den Bereichen 2 und 3 bestimmte Blöcke angezeigt werden, so müssen diese durch die Angabe n definiert werden.
- F Einem F-Kommando ohne weitere Parameter muß ein F-Befehl mit näheren Angaben vorausgegangen sein. Der Rechner setzt damit die Suche nach den zuvor angegebene Zeichen hinter der Adresse des zuletzt gefundenen Zeichens fort.

Fxxxx,xx1,xx2,xx3,xx4

Nach Eingabe dieses Kommandos wird ab der hexadezimal anzugebenden Speicherstelle xxxx nach der maximal 4 Byte langen Zeichenfolge xx1,xx2,xx3,xx4 gesucht, wobei xx1 bis xx4 aus jeweils 2 Buchstaben oder Ziffern besteht.

Bei der Angabe von einem oder mehreren Bytes muß auch die Speicherstelle xxxx definiert sein bei der die Suche beginnen soll.

Wenn DEBUG das angegebene Byte oder die Bytefolge im Speicher findet, wird im Bereich I des X-Anzeigemodus der entsprechende Speicherausschnitt ausgegeben, wobei die gesuchten Zeichen ab der ersten Position in der dritten Zeile zu finden sind. Sollte innerhalb des Speichers die angegebene Bytefolge nicht vorhanden sein, wird im Bereich I ein 64 Byte langer Block mit der Anfangsadresse FFE0H angezeigt.

BEISPIEL: F4000,54,43,53

Dieser F-Befehl mit der angegebenen Parameterliste wird ab Speicherzelle 4000H nach der Zeichenfolge 54 43 53 suchen. Dies entspricht einer Suche nach dem Wort TCS.

G

Nach dieser Eingabe werden alle Registerwerte zurückgespeichert und das Programm an der Stelle weitergeführt, an der unterbrochen wurde. Diese Stelle stimmt mit der Angabe im PC-Register überein.

HINWEIS: Sollte ein unterbrochenes Programm mit dem Übergang in DEBUG eine Eingabe von Ihnen verlangen, so ist diese nach Ausführung des G-Befehls durchzuführen. In diesem Fall ist kein Cursor im Anzeigefeld des DEBUG sichtbar.

Gxxxx,yyyy,zzzz

Evtl. durchgeführte Änderungen an den Registerwerten werden übernommen und das aufrufende Programm setzt seinen Ablauf an der Unterbrechungsstelle (abgelegt im PC-Register) fort.

Soll das Programm an einer bestimmten Stelle fortgeführt werden, muß die hexadezimale Adresse xxxx angegeben werden.

Die Adressen yyyy und zzzz geben weitere Unterbrechungsadressen an. DEBUG setzt an diese Stellen den RST 30H-Befehl und sorgt für eine Zwischenspeicherung des Original-Inhalts. Nach erneutem Aufruf der definierten Speicherstellen, werden die RST 30H-Befehle wieder gegen die Originalwerte ausgetauscht.

- I Mit Hilfe dieses Befehls ist es möglich, ein durch DEBUG unterbrochenes Programm in Einzelschritten auszuführen, und dabei den Speicher- und Registerzustand zu kontrollieren.
- HINWEIS: Es sind keine Einzelschritte unterhalb der Speicheradresse 5200H oder Sprünge in diesen Speicherbereich gestattet.
Bei der Ausführung eines Programms im "Single-Step"-Modus wird der RST 30H-Befehl verwendet. Daraus geht hervor, daß ein mit DEBUG zu bearbeitendes Programm keine Rücksprünge auf sich selbst beinhalten und das auf einen Einzelschritt folgende Byte kein Datenbyte sein darf.
- Ln,drs Um einen Sektor von der Diskette in den Speicherbereich 4200H bis 42FFH einzulesen, müssen Sie den L-Befehl benutzen. "n" steht für die Laufwerksnummer und "drs" für die relative Sektornummer. Ein eingelesener Sektor kann sowohl im X- als auch im S-Modus angezeigt werden. Sie können dann z.B. den Speicherinhalt mit dem Befehl F nach Zeichenketten durchforschen, mit M modifizieren und anschließend über W wieder auf die Diskette zurückschreiben.
- HINWEIS: Sollte ein Programm mit einem Kennwort versehen sein, folgt auf die Eingabe des L-Kommandos die Ausgabe einer Fehlermeldung.
- Mxxxx Das steht für Modifikation und bedeutet, daß DEBUG auf die hexadezimal anzugebende Adresse xxxx springt, automatisch in den S-Modus übergeht und zur Eingabe von Änderungen (ebenfalls hexadezimal) bereit ist.
Mit Hilfe der vier Pfeiltasten können Sie den Cursor im Anzeigefeld verschieben. Das M-Kommando kann nur durch Druck auf NEW LINE bzw. ENTER abgeschlossen werden.
- Q Mit dem Kommando Q (Quit) wird DEBUG verlassen und in die DOS-Befehlsebene übergewechselt. Die gleiche Auswirkung hat das Kommando G4400.
- Rrr,xxxx Nach der Eingabe dieses Befehls wird der Wert des Registerpaares rr auf den angegebenen hexadezimalen Wert xxxx geändert.
- S Dieses Kommando bringt die Anzeige des DEBUG in den S-Modus. D.h. es wird ein 256 Byte langer Speicherblock auf dem Bildschirm ausgegeben, dessen Anfangsadresse mit der ersten Adresse des ersten Bereiches im X-Modus übereinstimmt.

WRn,drs

Der W-Befehls ist das Gegenteil vom L-Befehl, indem Sie mit seiner Hilfe den 256 Byte lange Speicherbereich von 4300H bis 42FFH auf die im Laufwerk n befindliche Diskette auf den relativen Sektor drs schreiben können. Sollte der Sektor geschützt sein, bleibt dieser Zustand auch nach der Änderung erhalten.

Im übrigen gilt das gleiche wie unter der L-Kommando bereits beschrieben.

HINWEIS: Sollte ein Programm mit einem Kennwort versehen sein, folgt auf die Eingabe des L-Kommandos die Ausgabe einer Fehlermeldung.

X

Dieses Kommando bringt die Anzeige des DEBUG in den X-Modus.

Es werden dabei drei Speicherbereiche mit einer Länge von jeweils 64 Byte auf dem Bildschirm in Gruppen von vier Zeilen ausgegeben. Die drei Bereiche können einen unterschiedlichen Inhalt haben. Die 5., 10. und 15. Zeile beinhalten die aktuellen Zustände der Z80-Register.

Zeile 5: Register AF, BC, DE, HL
Zeile 10: Register AF', BC', DE', HL'
Zeile 15: Register PC, SP, IX, IY

Von den Registern AF und AF' wird auch noch die Bitmaske für das Statusbyte F hinter dem Registerwert ausgegeben. Wenn innerhalb der Bitmaske ein Bit gesetzt ist, so ist dies durch einen Buchstaben kenntlich gemacht. Nicht gesetzte Bits werden durch ein Minus (-) angezeigt.

Die Buchstaben innerhalb der Bitmaske haben folgende Bedeutung:

| | | |
|--------|----------------|-------------------------------|
| Bit 0 | C (Carry) | steht für Übertrag |
| Bit 1 | N | steht für Subtraktion |
| Bit 2 | P (Parity) | steht für Parität o. Überlauf |
| Bit 3 | unbenutzt | |
| Bit 4 | H (Half carry) | steht für halber Übertrag |
| Bit 5 | unbenutzt | |
| Bit 6 | Z (Zero) | steht für Null |
| Bit 7: | S (Sign) | steht für minus |

Teil III

Disk-BASIC

Allgemeine Informationen

=====

Das auf Ihrer System-Diskette unter dem Filenamen BASIC/CMD abgelegte und auch aus der DOS-Befehlsebene so aufzurufende Disk-BASIC, erweitert den beim Genie I/II im ROM stehenden Microsoft Level II BASIC-Interpreter um eine Vielzahl nützlicher Befehle, wobei diese nicht unbedingt mit der diskettenorientierten Programmierung zusammenhängen müssen. Bei der G-DOS Version 2.1a zum Genie III und 3.0 zum Speedmaster beinhaltet das Modul BASIC/CMD natürlich auch noch das gesamte Level II BASIC.

Da das Disk-BASIC auf das Microsoft Level II BASIC aufbaut, sind Grundkenntnisse in der BASIC-Programmierung zum Verständnis der folgenden Seiten Voraussetzung.

Der größte Vorzug des Disk-BASIC liegt natürlich in der Möglichkeit, Programme von einer Diskette zu laden oder abzuspeichern, sowie Dateien durch die sequentielle Methode oder auch durch wahlfreien Zugriff (RANDOM) zu bearbeiten.

Dabei ist die sequentielle Methode der Datenspeicherung noch leicht verständlich, da sie einer Speicherung auf Kassette sehr ähnlich ist.

Für Programmierung von Dateien nach der Methode des wahlfreien Zugriffs (RANDOM) ist dagegen schon etwas mehr Aufwand und ein genaues Studium des entsprechenden Kapitels in diesem Handbuch unumgänglich. Nur so können Sie die Vielseitigkeit und Einsatzmöglichkeiten dieser Methode der Dateibearbeitung voll ausschöpfen.

Fehlermeldungen

=====

Die aus zwei Buchstaben bestehende Fehlermeldungen, die unter Level II BASIC bei einem Programmierfehler innerhalb eines BASIC-Programms ausgegeben werden, erscheinen bei Disk-BASIC als vollständige deutsche Texte.

So wird z.B. aus der einfachen Meldung

?SN-ERROR in Zeilennummer

unter Disk-BASIC die ausführliche Angabe

Syntaxfehler in Zeilennummer

Hinzu kommen noch die Meldungen, die auf Fehler bei der Arbeit mit den Diskettenlaufwerken hinweisen. Eine entsprechende Tabelle finden Sie hierzu im Anhang 13 dieses Handbuchs.

SYNTAX: BASIC Parameter

=====

WIRKUNG: Mit Hilfe dieses Befehls wird von der Systemdiskette der File BASIC/CMD geladen und gestartet. Es sind dabei die Angaben verschiedener Parameter möglich:

```
BASIC
BASIC *
BASIC nn
BASIC sssss
BASIC bef
BASIC nn,sssss,bef
BASIC sssss,nn,bef
BASIC nn,sssss
BASIC nn,bef
BASIC sssss,bef
```

Diese Parameter haben folgende Bedeutung:

- * beim Start des Disk-BASIC erfolgt keine Neuinitialisierung. Ein zuvor eingegebenes BASIC-Programm wird nicht gelöscht und alte Parameter werden übernommen. Ein noch im Speicher vorhandenes BASIC-Programm wird sofort gelistet. Befindet sich kein BASIC-Programm im Speicher, erfolgt wieder ein Rücksprung in die DOS-Befehlsebene. Wenn zuvor nn kleiner als 2 definiert wurde, wird BASIC * nicht ausgeführt.
- nn hiermit wird die Zahl der maximal zur Verfügung stehenden Disk-Ein/Ausgabe-Bereiche angegeben. D.h. es handelt sich hier um die Angabe der maximalen File-Anzahl, die Sie im Programm gleichzeitig eröffnen können. Der Wert n darf zwischen 0 und 15 liegen und ist standardmäßig auf 3 eingestellt. In der Regel wird den Daten bei der Ausgabe auf Diskette eine Blocklänge von 256 Byte zugewiesen. Wenn Sie jedoch auch mit Field-Item-Dateien arbeiten wollen, deren Satzlänge kleiner als 256 Byte ist, muß hinter der Angabe nn noch ein V eingegeben werden.
- sssss mit Hilfe von sssss können Sie dem Rechner die zur Verfügung stehende Speichergröße mitteilen. Die Angabe entspricht der Frage nach READY? im Level II BASIC und wird benötigt, um beispielsweise Maschinenroutinen oberhalb der angegebenen Speichergrenze vor BASIC zu schützen. sssss kann wahlweise in dezimaler oder hexadezimaler Schreibweise eingegeben werden. Ohne Angabe von sssss wird der Wert angenommen, der durch den HIMEM-Befehl vereinbart wurde.

bef an dieser Stelle können Sie BASIC-Befehle oder Kommandos angeben, die als Zeichenkette dem BASIC nach der Initialisierung vorliegen und automatisch ausgeführt werden.

Da in der DOS-Befehlsebene nur ein Tastatureingabepuffer von 80 Zeichen zur Verfügung steht, darf der BASIC-Aufruf mit allen Parametern diese Länge nicht überschreiten.

Wird BASIC über das AUTO-Kommando aufgerufen, ist nur eine Eingabe von 32 Zeichen möglich. Wenn die Zeile zur Initialisierung des BASIC einen Fehler enthält, erfolgt automatisch ein Rücksprung in die DOS-Befehlsebene.

BEISPIELE: BASIC

Dieser Aufruf bewirkt Initialisierung des Disk-BASIC mit Bereitstellung von drei Ein/Ausgabe-Bereichen. Die Speicherobergrenze wird entsprechend HIMEM gesetzt.

BASIC,LOAD"TEST/BAS"

Die Initialisierung des Disk-BASIC erfolgt wie zuvor beschrieben, jedoch wird unmittelbar danach noch das Programm TEST/BAS geladen.

BASIC,5,40000,RUN"TEST/BAS"

Bei der BASIC-Initialisierung werden fünf Ein/Ausgabe-Bereiche vorbereitet, die Speicherobergrenze auf 40000 gesetzt und das Programm TEST/BAS geladen und gestartet.

BASIC,5,40000,CLEAR800:CLS:RUN"TEST/BAS"

Außer der bereits zuvor beschriebenen Ausführungen werden nun noch 800 Byte für den Stringbereich des Rechners reserviert, der Bildschirm gelöscht und das Programm TEST/BAS gestartet.

BASIC,5V

Auch hier werden wieder fünf Ein/Ausgabe-Bereiche reserviert, wobei jedoch jedem Bereich 256 Byte Speicher zusätzlich zur Verfügung stehen, um die Bearbeitung von Field-Item-Files mit einer Länge von weniger als 256 Byte zu ermöglichen.

SYNTAX: DEF FN Funktionsname (Variablenliste) = Funktion

=====

WIRKUNG: Einem Funktionsnamen wird eine Funktion zugeordnet, deren Wert von den angegebenen Variablen abhängt.

ANWENDUNG: Mit Hilfe dieses Befehls können Sie Funktionen definieren, die der Rechner später beim Aufruf durch FN Funktionsname bearbeitet.

BEMERKUNG: Der Funktionsname kann aus einer beliebigen Anzahl von Zeichen bestehen, wobei das erste Zeichen immer ein Buchstabe sein muß. Bei der Bearbeitung des Funktionsnamens werden jedoch immer nur die ersten beiden Zeichen betrachtet.
In der Variablenliste müssen alle Variablen, die in der definierten Funktion benötigt werden, enthalten sein. Die Variablen werden in der Liste durch Komma getrennt aufgeführt.
Die Funktion selbst, z.B. mathematische Berechnungen oder Bearbeitungen von Zeichenketten und ähnliches, darf keine Doppelpunkte enthalten und die Länge einer Zeile nicht überschreiten.
Als ausgesprochen sinnvoll erweist sich die Anwendung von DEF FN besonders dann, wenn spezielle Funktionen mehrmals im Programm bearbeitet werden müssen. Sie sparen Zeit (durch Vermeidung von Unterprogrammansprüngen) und auch Speicherplatz.

BEISPIEL: Bei dem folgenden Beispiel handelt es sich um eine Umwandlung von Zeitangaben der Form hh:mm:ss in ihre dezimalen Werte.
Die dazu notwendige Umrechnung wird als Funktion definiert und später über FN aufgerufen.
Wissenswert ist dabei, daß die in der definierten Funktion zur Berechnung gewählte Variable Z\$ bei der später folgenden Eingabe der Zeiten nicht verwendet werden muß.
Im Beispiel wird die Funktion DZ(Z\$) mit 5 verschiedenen Variablen T\$(I) aufgerufen.

```
10 CLS: CLEAR500
20 DEF FN DZ#(Z$)=VAL(LEFT$(Z$, INSTR(Z$, ":")-1))+VAL
(MID$(Z$, INSTR(Z$, ":")+1, 2))*60+VAL(RIGHT$(Z$, 2)) /3
600
30 PRINT"Geben Sie 5 Zeiten im Format hh:mm:ss ein:"
40 PRINT:FORI=1TO5
50 PRINTCHR$(8)". " ;:LINEINPUT" Zeiteingabe: ";T$(I)
60 NEXTI:CLS
70 FORI=1TO5
80 DZ#(I)=FN DZ#(T$(I))
90 PRINT"Der Dezimalwert der Zeit ";T$(I);" lautet";
DZ#(I)
100 NEXTI
```

SYNTAX: DEFUSRn=Startadresse

=====

WIRKUNG: Die Einsprungadresse eines Maschinenunterprogramms wird definiert, wobei n einen Wert zwischen 0 und 9 annehmen kann.

ANWENDUNG: In Verbindung mit der USRn-Funktion können Sie bis zu 10 verschiedene Maschinenunterprogramme zur Ausführung kommen lassen.

BEMERKUNG: Wenn Sie den Wert n nicht angeben wird n=0 angenommen.
Die Routine kann in diesem Fall mit USR oder USR0 aufgerufen werden.

BEISPIEL: Im folgenden Programm wird ab Speicherstelle -8192 ein 12-Byte langes Maschinenprogramm abgelegt. In Zeile 20 erfolgt die Definition der Einsprungadresse und mit Zeile 40 kommt es zur Ausführung, wobei die Variable D eine Dummy-Variable darstellt.

```
10 CLS:FORI=-8192TO-8181:READA:POKEI,A:NEXTI
20 DEFUSR2=-8192
30 FORC=33TO191:POKE15360,C
40 D=USR2(0)
50 FORZ=1TO25:NEXTZ,C
60 GOTO30
100 DATA 33,0,60,17,1,60,1,255,3,237,176,201
```

SYNTAX: ERROR Codennummer

WIRKUNG: Das ERROR-Kommando zeigt die Fehlermeldung zur angegebenen Codennummer auf dem Bildschirm an.

BEMERKUNG: Unter Level II - BASIC werden Programmierfehler üblicherweise in einer abgekürzten Form am Bildschirm angezeigt.

Das Disk-BASIC enthält nun ein Modul (SYS13/SYS), welches bewirkt, daß anstelle der Abkürzungen ausführliche Fehlermeldungen auf dem Schirm ausgegeben werden.

22 der möglichen BASIC-Fehlermeldungen lassen sich mit Hilfe des ERROR-Befehls abrufen.

Die DOS-Fehler, mit Werten größer oder gleich 23, lassen sich über den ERROR-Befehl nicht erzeugen.

BEISPIEL: Folgende Tabelle zeigt Ihnen den Zusammenhang zwischen Codennummer, Abkürzung und ausführlicher Erklärung.

| Code | Abkürzung | Fehler-Erklärung |
|------|-----------|------------------------------|
| 1 | NF | NEXT ohne FOR |
| 2 | SN | Syntaxfehler |
| 3 | RG | RETURN ohne GOSUB |
| 4 | OD | Daten fehlen |
| 5 | FC | Unerlaubter Funktionsaufruf |
| 6 | OV | Ueberlauf |
| 7 | OM | Zuwenig Speicher |
| 8 | UL | Zeile fehlt |
| 9 | BS | Bereichsueberschreitung |
| 10 | DD | Neue Felddimension |
| 11 | /0 | Null als Teiler |
| 12 | ID | Eingabefehler |
| 13 | TM | Typabweichung |
| 14 | OS | Zuwenig Textspeicher |
| 15 | LS | Text\$ zu lang |
| 16 | ST | Text\$-Formel zu kompliziert |
| 17 | CN | Unerlaubtes CONT! |
| 18 | NR | kein RESUME |
| 19 | RW | RESUME ohne Fehler |
| 20 | UE | Fehler unbekannt |
| 21 | MO | Operand fehlt |
| 22 | FD | Schlechte Dateidaten |

SYNTAX: INSTR(n, Zeichenkette 1, Zeichenkette 2)

WIRKUNG: Innerhalb einer Zeichenkette 1 wird nach einer Zeichenkette 2 gesucht und deren Anfangsposition, von der linken Seite der Kette 1 ab Position n aus gezählt, festgestellt.

ANWENDUNG: Die INSTR-Funktion wird oft in Verbindung mit den Funktionen LEFT\$, RIGHT\$ und MID\$ genutzt, um die Position einer bestimmten Zeichenkette innerhalb einer weiteren Kette festzustellen.

BEMERKUNG: Die Position n muß nicht mit angegeben werden, wenn die Zeichenkette 1 komplett, von der linken Seite ausgehend, auf Zeichenkette 2 hin untersucht werden soll.
Wenn n größer ist als die Länge der Kette 1, Kette 1 die Länge 0 hat oder die Kette 2 nicht in Kette 1 gefunden wird, ergibt INSTR den Wert 0. Wenn Kette 2 Null ist ergibt INSTR bei angegebenem n den Wert n, andernfalls 1.

BEISPIEL: Im folgenden Programm wird das Alphabet auf ein + untersucht. Das + steht im String A\$ genau in der Mitte, also an 14. Stelle.

```
10 A$="ABCDEFGHILKLM+NOPQRSTUVWXYZ"  
20 B$="+"  
30 PRINT"Das + liegt an";INSTR(1,A$,B$);". Stelle."
```

SYNTAX: KILL"Dateiname"

=====

WIRKUNG: Löscht die angegebene Datei oder das Programm auf der Diskette.

ANWENDUNG: Durch das "Killen" von alten Dateien und Programmen können Sie sich auf einer Diskette wieder freien Speicherplatz schaffen.

BEMERKUNG: Der Eintrag einer gelöschten Datei bleibt im Inhaltsverzeichnis der Diskette für den Benutzer unsichtbar stehen, sodaß es evtl. einmal vorkommt, daß der Rechner beim Ablegen eines Programms die Meldung 'Inhaltsverzeichnis voll' ausgibt, obwohl auf der Disk noch freier Speicherplatz vorhanden ist. In diesem Fall schafft der Befehl "C" des Programms DIRCHECK Abhilfe, indem das Inhaltsverzeichnis gesäubert wird.

SYNTAX: LINE INPUT "Text"; Zeichenkettenvariable

oder

LINE INPUT Zeichenkettenvariable

=====

WIRKUNG: Die Ausführung von LINE INPUT ist ähnlich wie bei INPUT. Es werden jedoch auch Kommata und Anführungszeichen als Bestandteil der Stringvariablen akzeptiert.

ANWENDUNG: LINE INPUT wird immer dann eingesetzt, wenn bei der Eingabe der Zeichenkettenvariablen auch Kommata und Anführungszeichen vorkommen können. Außerdem wird das bei INPUT oft als lästig empfundene "?" unterdrückt.

BEMERKUNG: Wie bei einem normalen INPUT haben Sie die Möglichkeit, der Eingabe einen Textstring vorzuschicken. Mit NEW LINE erfolgt die Übergabe der eingegebenen Zeichenkette an die angegebene Variable.

BEISPIELE: Eine Anwendung des LINE INPUT-Befehls finden Sie im Beispiel des Abschnitts über DEF FN.

SYNTAX: LOAD"Programmname"
oder
LOAD"Programmname",R

WIRKUNG: Durch den Befehl LOAD wird ein im Speicher stehendes Programm gelöscht und durch ein neues, von der Diskette geladenes Programm ersetzt. Außerdem werden alle Variablen gelöscht und offene Dateien geschlossen (wie bei NEW).

BEMERKUNG: Wenn Sie die Option R wählen, wird das Programm nicht nur geladen, sondern auch gestartet (wie bei RUN). Geöffnete Dateien werden dabei jedoch nicht automatisch geschlossen.

SYNTAX: MID\$(Zeichenkette 1,m,n) = Zeichenkette 2

=====

WIRKUNG: Die MID\$-Funktion ermöglicht es, Teile einer Zeichenkette durch eine andere Zeichenkette zu ersetzen.

ANWENDUNG: Veränderung einer Zeichenkette 1 durch eine Kette 2. Dazu muß die Startposition m innerhalb der Kette 1 definiert werden, ab der der Austausch von n Zeichen durch die Zeichenkette 2 ausgeführt werden soll. Um m zu ermitteln muß vom äußerst linken Zeichen der Kette 1 an gezählt werden.

BEMERKUNG: Für den Fall, daß die Zeichenkette 2 sich wegen ihrer Länge nicht komplett in Kette 1 einfügen läßt, werden die überschüssigen Zeichen von der rechten Seite der Zeichenkette 2 abgeschnitten. Wenn die Angabe n fehlt, wird der Teil von Zeichenkette 1 ab Position m durch Kette 2 ersetzt, der der Länge von Kette 2 entspricht oder falls diese zu lang ist, bis zum Ende von Kette 1.

BEISPIEL:

```
10 CLS:A$="Dies ist ein Test zur MID$-Funktion."  
20 B$="TEST"  
30 C$=A$  
40 S=14:L=4  
50 MID$(A$,S,L)=B$  
60 PRINT"A$ lautete: ";C$  
70 PRINT"A$ ist nun: ";A$
```


SYNTAX: SAVE"Programmname"

oder

SAVE"Programmname",A

=====

WIRKUNG: Dieses Kommando wird dazu benutzt, um ein BASIC-Programm auf Diskette abzuspeichern.
Die Option A bedeutet, daß die Ausgabe des Programms im ASCII-Format erfolgen soll. Ohne A wird das Programm komprimiert auf der Diskette abgelegt.

BEMERKUNG: Beim Abspeichern wird ein evtl. auf der Disk befindliches Programm gleichen Namens gelöscht und durch das neue ersetzt.
Die ASCII-Ausgabe ist notwendig, um einen Programmtext beispielsweise durch ein weiteres Programm bearbeiten zu können.
Eine ASCII-Abspeicherung innerhalb eines laufenden Programms hat, (wie bei END), die Beendigung des Programms zur Folge.

BEISPIEL: Den Unterschied zwischen einem komprimiert und einem ASCII-mäßig abgespeicherten Programm zeigt folgendes Beispiel.

```
10 CLS
20 GOSUB 1000
30 PRINT "Das Programm wird nun in komprimierter"
40 PRINT "Form abgespeichert."
60 SAVE "TEST1/BAS:0"
70 GOSUB 1000
80 PRINT "Das Programm wird nun im ASCII-Format"
90 PRINT "abgespeichert."
100 PRINT "Hier endet der Programmablauf."
110 PRINT "Erneuter Programmstart mit RUN 200."
120 SAVE"TEST2/BAS:0",A
200 CLS:GOSUB 1000:CLS
210 PRINT "So sieht das Programm komprimiert aus:"
220 CMD"LIST TEST1/BAS:0"
230 GOSUB 1000
240 CLS:PRINT"Das gleiche Programm im ASCII-Format:"
250 CMD"LIST TEST2/BAS:0"
260 END
1000 PRINTCHR(10)"Drücken Sie die NEW LINE-Taste!"
1010 A$=INKEY$
1020 IF A$ <> CHR$(13) THEN GOTO 1010
1030 RETURN
```

SYNTAX: TIMES\$

WIRKUNG: Es handelt dabei um keinen Befehl, sondern um eine unter Disk-BASIC automatisch belegte Variable, die abgerufen werden kann.

ANWENDUNG: Unter der Zeichenkette TIMES\$ sind Datum und Uhrzeit der im Rechner enthaltenen Uhr abgelegt und können somit in BASIC-Programmen aufgerufen werden.

BEMERKUNG: Die Zeichenkette TIMES\$ besteht aus 17 Zeichen. Das darin enthaltene Datum und die Uhrzeit werden über die Datum-Zeit-Routine des G-DOS ermittelt. Datum und Uhrzeit sind in TIMES\$ im Format tt.mm.jj hh:mm:ss abgelegt.

BEISPIEL: Folgendes Programm verwendet das im Rechner stehende Datum und die Uhrzeit.

```
10 CLS: CLEAR 500
20 LINEINPUT "Geben Sie die genaue Zeit ein "; Z$
30 A$ = "ZEIT " + Z$
40 CMD A$
50 LINEINPUT "Geben Sie das heutige Datum ein "; D$
60 B$ = "DATUM " + D$
70 CMD B$
80 PRINT "Heute, am "; LEFT$(TIMES$, 8); ", ";
90 PRINT "wurde um "; RIGHT$(TIMES$, 8); " Uhr ";
100 PRINT "das Programm beendet."
```

SYNTAX: USRn(Argument)

=====

WIRKUNG: Ein über DEFUSR definiertes Maschinenunterprogramm wird aufgerufen.

ANWENDUNG: Mit Hilfe von USRn ist es möglich, bis zu 10 verschiedene Maschinenunterprogramme (n=0,1,...,9) aufzurufen. Zuvor müssen den USR-Routinen über DEFUSR die Startadressen zugeteilt werden.

BEMERKUNG: Wird n nicht mit angegeben, so ruft der Rechner automatisch die USR0-Routine auf. Bei dem Argument handelt es sich um einen Integer-Ausdruck mit einem Wert zwischen -32768 und 32768, (häufig wird 0 eingesetzt).

BEISPIEL: siehe DEFUSR-Funktion

Abkürzungen und Programmierhilfen

=====

Das Disk-BASIC des Genie-DOS erlaubt es, durch einfache Tastenbetätigung, bestimmte Funktionen aufzurufen, für die Sie normalerweise ganze Befehlswoorte eingeben müssten.

Diese Abkürzungen erweitern die Möglichkeiten des Programmisten um ein vielfaches.

Es ist dabei jedoch immer nur ein Kurzkommando pro Eingabezeile erlaubt. Außerdem muß das Zeichen jeweils an der ersten Position der Zeile eingegeben werden. Vorlaufende Zeilennummern führen zu einer Fehlermeldung.

Funktioniert ein Eintasten-Befehl nicht (z.B. "."), dann drücken Sie die BREAK-Taste und versuchen Sie es erneut.

SYNTAX: . (Punkt)

=====

WIRKUNG: Die zuletzt bearbeitete Zeile eines BASIC-Programms wird gelistet.

SYNTAX: ↑ (Pfeil aufwärts)

=====

WIRKUNG: Der vorhergehende Zeile eines BASIC-Programms wird gelistet. Bei Programmstart erscheint immer wieder die erste Zeile.

SYNTAX: ↓ (Pfeil abwärts)

=====

WIRKUNG: Die nächste Zeile eines BASIC-Programms wird gelistet. Beim Erreichen des Programmendes erscheint immer wieder die letzte Zeile.

SYNTAX: SHIFT ↑ (SHIFT Pfeil aufwärts)

oder

; (Semikolon)

=====

WIRKUNG: Die erste Zeile eines BASIC-Programms wird gelistet.

SYNTAX: SHIFT ↓ (SHIFT Pfeil abwärts)

oder

/ (Schrägstrich)

=====

WIRKUNG: Die letzte Zeile eines BASIC-Programms wird gelistet.

SYNTAX: @ (Klammeraffe)

=====

WIRKUNG: Von der zuletzt bearbeiteten oder angezeigten Zeile an wird eine komplette Bildschirmseite ausgegeben.

SYNTAX: : (Doppelpunkt)

=====

WIRKUNG: Es kommt zur Ausgabe einer kompletten Bildschirmseite mit den Zeilen des BASIC-Programms, die vor der zuletzt bearbeiteten oder angezeigten Zeile liegen.

SYNTAX: , (Komma)

=====

WIRKUNG: Die Eingabe eines Kommas bewirkt den Sprung in den EDIT-Modus für die zuletzt angezeigte oder bearbeitete Zeile.

SYNTAX: A Zeilennummer, Zeilenabstand

=====

WIRKUNG: Dieser Befehl schaltet die automatische Zeilennummerierung wie der AUTO-Befehl im Level II BASIC ein. Das A muß immer an der ersten Stelle einer Zeile angegeben werden. Dem A hat eine Zeilennummer oder ein Punkt zu folgen.

SYNTAX: D Anfangszeilennummer - Endzeilennummer
=====

WIRKUNG: Löscht wie beim DELETE-Befehl im Level II BASIC eine Einzelzeile oder ein Programmteil, das durch die Anfangs- und Endzeilennummer angegeben werden muß. Das D-Kommando muß an der ersten Stelle einer Zeile eingegeben werden.

SYNTAX: E Zeilennummer
=====

WIRKUNG: Das E-Kommando hat die gleichen Wirkungen wie der EDIT-Befehl und Level II BASIC. Das E muß immer am Zeilenanfang stehen.

SYNTAX: L Anfangszeilennummer - Endzeilennummer
=====

WIRKUNG: Diese Abkürzung arbeitet wie der LIST-Befehl unter Level II BASIC und ermöglicht damit die Anzeige einer oder mehrerer Zeilen, die durch ihre Anfangs- oder Endzeilennummer angegeben werden müssen. Das L muß immer am Zeilenanfang stehen.

SYNTAX: DU alte Zeilennummer, neue Zeilennummer
=====

WIRKUNG: Durch dieses Kommando wurde die Möglichkeit geschaffen, eine Zeile zu duplizieren.

SYNTAX: DI alte Zeilennummer, neue Zeilennummer
=====

WIRKUNG: Dieses Kommando bedeutet "Delete an Insert", d.h. eine Zeile wird an ihrer alten Stelle gelöscht und mit einer neuen Zeilennummer wieder ins Programm eingefügt.

SYNTAX: MERGE"Programmname"

WIRKUNG: Ein Programm auf der Diskette wird mit einem im Speicher des Rechners residenten Programm verbunden. (Wie LOAD ohne zu löschen!)

ANWENDUNG: Der MERGE-Befehl ermöglicht es Ihnen, aus laufenden BASIC-Programmen heraus, aber auch als direktes Kommando, weitere Programmteile nachzuladen, ohne das bereits im Speicher vorhandene Programm und dessen Variablen zu zerstören.

BEMERKUNG: MERGE hat eine unterschiedliche Wirkung, je nachdem ob es als direktes Kommando oder innerhalb eines Programms benutzt wird.

Direktes Kommando:

Das von der Diskette kommende Programm überlagert das im Speicher stehende, sodaß jede Zeile, die die gleiche Zeilennummer trägt, ersetzt wird. Vergewissern Sie sich vor Ausführung des Befehls MERGE, daß genügend Speicherplatz für das entstehende Gesamtprogramm vorhanden ist.

Aufruf im Programm:

Verwenden Sie ein MERGE innerhalb eines Programms, so wird dieses nach Ausführung des MERGE-Befehls in der nächsten Zeile fortgesetzt. MERGE muß außerdem immer der letzte Befehl einer Zeile sein, da nachfolgende Statements nicht mehr erkannt werden. Die Zeilennummern des nachzuladenden Programmteils, z.B. eines Unterprogramms, müssen größer sein als die größte Zeilennummer des aufrufenden Programms. D.h. das Modul wird immer hinten an das Programm angehängt.

Schaffen Sie vor dem MERGE mit Hilfe von CMD"F",DELETE genügend Platz, damit sich die Zeilennummern nicht überschneiden.

BEISPIEL: 40 REM *****
50 REM Dieser Teil des Programms befand
60 REM sich unter dem Namen "TEILL/BAS"
70 REM auf der Diskette und hat Zeile 40
80 REM des im Speicher stehenden Programms,
90 REM gegen die Zeile mit Sternen ausgetauscht.

Geben Sie diesen Programmteil ein und speichern Sie ihn unter TEILL/BAS:0 auf der Diskette ab. Löschen Sie anschließend das im Speicher stehende Programm mit NEW.

Geben Sie nun das zweite Programm ein:

```
10 REM Dies ist der Teil des Programms,  
20 REM welcher sich vor der Ausführung des  
30 REM Befehls MERGE im Speicher des Rechnes befand.  
40 REM =====
```

Vereinigen Sie jetzt beide Programmteile durch die
Eingabe von

```
MERGE"TEIL1/BAS:0"
```

und listen Sie das dabei entstandene Programm.

SYNTAX: REF Parameter

WIRKUNG: Der Befehl dient zur Erstellung der Referenzliste eines BASIC-Programms mit Zeilennummern und Variablennamen.

Dabei sind als Parameter folgende Angaben zulässig.

| | |
|---------------|---|
| REF | Hier muß zuvor ein REF nnnnn oder REF Variable ausgeführt worden sein. Die nächste Zeile mit einer entsprechenden Referenz wird dann angezeigt. Sind keine weiteren Zeilen mehr vorhanden, wird "Textende" auf dem Bildschirm ausgegeben. Nach erneutem REF erscheint wieder die erste Zeile. |
| REF * | Eine komplette Referenzliste wird auf dem Bildschirm ausgegeben. |
| REF \$ | Eine komplette Referenzliste wird auf dem Drucker ausgegeben. |
| REF nnnnn | Es wird eine Referenzliste von allen Zahlen mit dem Wert nnnnn erstellt, wobei nnnnn maximal 5-stellig sein darf und zwischen 1 und 99999 liegen muß. |
| REF Variable | Erstellt eine Referenzliste von allen Variablen mit dem entsprechenden Namen. Der Variablenname darf maximal zwei Zeichen lang sein und keine Typ-Bezeichnung (\$,!,#,%) enthalten. |
| REF"Text | Erzeugt eine Referenzliste des angegebenen Textes, mit der es beispielsweise möglich ist, bestimmte Worte in Strings oder Kommentaren aufzufinden. |
| REF=Befehl | Ähnlich wie bei REF"Text haben Sie hier die Möglichkeit nach BASIC-Befehlen zu suchen. Der angegebene Text wird dabei, wenn möglich, in das entsprechende Token umgewandelt. Wenn kein Token zum angegebenen Text existiert, wird der Befehl wie REF"Text behandelt. |
| REF * nnnnn | Kombination zwischen REF * und REF nnnnn |
| REF \$ nnnnn | Kombination zwischen REF \$ und REF nnnnn |
| REF*Variable | Kombination zwischen REF * und REF Variable |
| REF\$Variable | Kombination zwischen REF * und REF Variable |

BEISPIEL: Bei dem folgenden Kurzen Listing handelt es sich um ein Programm zur Umwandlung von Dezimalzahlen in Hexadezimalzahlen. Der Aufbau und die Funktion des Programms sollen hier nicht näher erläutert werden. Es geht vielmehr darum, eine durch den Rechner erstellte Referenzliste zu diesem Programm zu zeigen.

```

10 CLS
20 CLEAR1000
30 PRINT STRING$(64,42);
40 PRINT "****";
50 PRINT TAB(17) "Dezimal-Hexadezimal Umwandlung";
60 PRINT TAB(61) "****";
90 PRINT STRING$(64,42);
110 DEF FN H2$(ZW%)=MID$("0123456789ABCDEF",INT(ZW%/16)+1,1)+MID$("0123456789ABCDEF",ZW%-INT(ZW%/16)*16+1,1)
120 DEF FN H4$(A%)=FNH2$(ASC(MID$(MKI$(A%),2)))+FNH2$(ASC(MKI$(A%)))
130 PRINT@320,"Geben Sie eine Zahl zwischen -32768 und 65535 ein!"
150 PRINT@520,"Dezimalzahl      : ";:LINEINPUTW$:A!=VAL(W$)
160 IF A!>65535ORA!<-32768THENPRINT@538,"FALSCH EINGABE!":GOTO 190
170 IFA!>32767THENA%=A!-65536ELSEA%=A!
180 PRINT@584,"Hexadezimalzahl: ";:PRINT" ";FNH4$(A%)
190 PRINT@712,"Weiter ? (J/N)"
200 A$=INKEY$
210 IFA$=" "THEN200ELSEIF A$="J"ORA$="j"THENPRINT@538,STRING$(20,32):PRINT@712,STRING$(20,32):GOTO150ELSEIFA$="N"ORA$="n"THENENDELSEGOTO200

```

Nach Eingabe des Befehls REF * erfolgt die Ausgabe folgender Liste auf dem Bildschirm:

```

      1  110/4
      2  120
     16  110/3
     17  60
     20  210/2
     32  210/2
     42  30 90
     61  80
     64  30 90
    150  210
    190  160
    200  210/2
    320  130
    520  150
    538  160 210
    584  180
    712  190 210
   1000  20
  32767  170
  32768  160
  65535  160
  65536  170

```

```

A   120/%3 150/! 160/!2 170/%2 170/!3 180/% 200/$
    210/$5
H2  110($ 120($2
H4  120($ 180($
W   150/$2
ZW  110/%4

```

Auf der linken Seite dieser Tabelle stehen die Konstanten, gefolgt von den Variablen. Rechts daneben befindet sich jeweils die dazugehörigen Zeilennummer und nach einem Schrägstrich der Typ und die Anzahl der in der Zeile vorkommenden Konstanten oder Variablen.

Die erste Zeile der Tabelle sagt z.B. aus, daß die Konstante 1 in der Zeile 110 viermal vorkommt. Das A kommt im Programm in verschiedenen Zusammenhängen vor, z.B. als String-Variable, Integer-Variable und Variable einfacher Genauigkeit.

Die rechte Seite der Referenzliste hinter der Variablen A zeigt, daß die Integer-Zahl A% dreimal in Zeile 120, zweimal in Zeile 170 und einmal in Zeile 180 zu finden ist. Weiterhin steht die Variable A! mit einfacher Genauigkeit einmal in Zeile 150, zweimal in Zeile 160 und dreimal in der Zeile 170. Als String-Variable tritt A\$ einmal in Zeile 200 und fünfmal in Zeile 210 auf.

Anhand dieses Beispiels sehen Sie, daß mit Hilfe des REF-Befehls auf bequeme Art Listen aller in einem Programm verwendeten Konstanten und Variablen erstellt werden können.

SYNTAX: RENEW

WIRKUNG: Ein durch den Befehl NEW gelöschttes BASIC-Programm wird durch diesen Befehl wieder restauriert.

BEMERKUNG: Wenn sich kein gelöschttes Programm im Speicher des Rechners befindet, sollte dieser Befehl vermieden werden, da er in diesem Fall ein unsinniges Programm erzeugen kann.

SYNTAX: RENUM Parameter

WIRKUNG: Mit diesem Befehl lassen sich Programmteile verschieben und umnumerieren.
Als Parameter sind folgende Angaben möglich:

```
RENUM ,                               (das Komma ist wichtig!)
RENUM nnnnn
RENUMM , zzzzz
RENUM , , aaaaa
RENUM , , , eeeee
RENUM nnnnn, zzzzz
RENUM nnnnn, , aaaaa
RENUM , zzzzz, aaaaa
RENUM nnnnn, , , eeeee
RENUM , zzzzz, , eeeee
RENUM , , aaaaa, eeeee
RENUM nnnnn, zzzzz, aaaaa
RENUM nnnnn, zzzzz, , eeeee
RENUM nnnnn, , aaaaa, eeeee
RENUM , zzzzz, aaaaa, eeeee
RENUM nnnnn, zzzzz, aaaaa, eeeee
```

Die Parameter haben dabei folgende Bedeutung:

nnnnn ist die neue Anfangszeilennummer
zzzzz ist der Zeilenabstand im neu nummerierten Programmteil
aaaaa ist die Zeilennummer, bei der die Umnummerierung beginnen soll
eeeeee ist die Zeilennummer, bei der die Umnummerierung enden soll

Die Werte nnnnn, zzzzz, aaaaa und eeeee müssen im Bereich zwischen 1 und 65629 liegen. Werden nnnnn und zzzzz nicht angegeben (durch Angabe zweier aufeinander folgender Kommata), so erhalten sie den Standardwert 10. Bei ausbleibender Angabe der Werte aaaaa und eeeee erfolgt eine Einstellung auf 0 bzw. 65529.

Wenn bei der Umnummerierung vom Rechner ein Fehler im Programm bemerkt wird, geht das RENUM automatisch in den Modus RENUM U über (siehe dort) und die Zeilen bleiben unverändert.

BEMERKUNG: RENUM verschiebt Teile des Programms wenn die neuen Zeilennummern dies erfordern.

BEISPIELE: RENUM ,

Das gesamte BASIC-Programm wird mit neuen Zeilennummern versehen, wobei die erste Zeile die Nummer 10 erhält und alle weiteren in einem Zeilenabstand von 10 geändert werden.

RENUM 100,50

Das gesamte BASIC-Programm wird neu nummeriert. Die erste Zeile trägt die Nummer 100 und alle weiteren Zeilennummern haben einen Abstand von 50.

RENUM ,,,10000

Alle Zeilen des Programms, bis einschließlich Zeilennummer 10000, werden von 10 aus beginnend, im Zeilenabstand 10 neu durchnummeriert.

RENUM 40000,1,3500,4000

Die Zeilen im Bereich von 3500 bis 4000 werden nach 40000 verschoben und im Zeilenabstand 1 umnummeriert.

SYNTAX: RENUM U

=====

WIRKUNG: Ein BASIC-Programm wird damit auf Fehler durch falsche Zeilennummerierung untersucht. Es werden aber auch sonstige Fehler, die bei einem RENUM auftreten könnten, zur Anzeige gebracht. Ein mit RENUM U bearbeitetes Programm bleibt in seiner Form unverändert.

Die Anzeige einer Fehlermeldung erscheint auf dem Bildschirm in der Form:

Fehlerzeile
nnnnn/Kommentar Ende

nnnnn steht dabei für die Zeilennummer und als Kommentar kann ein U bei fehlender Zeile, ein X bei einer Zeile mit Syntaxfehler und ein S bei einer Zeile mit falscher Zeilennummer erscheinen.

BEISPIEL: 10 GOSUB: ' In dieser Zeile fehlt die Zeilennummer
20 GOSUB 1000: ' Zeile 1000 existiert nicht
30 GOSUB 100000: ' Zeilennummer 100000 ist falsch
40 GOSUB 100
50 END
100 REM Unterprogramm
110 RETURN

Der Aufruf des Befehls RENUM U ergibt die Bildschirmausgabe:

Fehlerzeile
10/X 1000/U 30/S Ende

SYNTAX: &Hnnnn

=====

WIRKUNG: nnnn steht für eine 4-stellige hexadezimale Zahl, die mit dieser Eingabe direkt verarbeitet oder aber in eine entsprechende Dezimalzahl umgewandelt werden kann. (n=0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

BEMERKUNG: Wenn Sie für nnnn Zahlen eingeben, die größer oder gleich 8000H sind, so erhalten Sie die Ausgabe eines negativen Dezimalwertes.

BEISPIEL: Die folgende Tabelle zeigt Ihnen den Zusammenhang zwischen den hexadezimalen Eingaben und den entsprechenden dezimalen Ausgaben:

| hexadezimale Eingabe | dezimale Ausgabe |
|----------------------|------------------|
| &H1 | 1 |
| &H2 | 2 |
| : | : |
| : | : |
| &H7FFF | 32767 |
| &H8000 | -32768 |
| &H8001 | -32767 |
| : | : |
| : | : |
| &HFFFE | -2 |
| &HFFFF | -1 |

SYNTAX: &Onnnnnn
oder
&nnnnnn

=====

WIRKUNG: nnnnnn steht für eine 6-stellige oktale Zahl, die mit diesem Befehl direkt verarbeitet oder aber auch in ihren entsprechenden dezimalen Wert umgewandelt werden kann. (n=0,1,2,3,4,5,6,7)

BEMERKUNG: Als größtmögliche oktale Zahl können Sie &O177777 eingeben. Oktalzahlen, deren Wert größer als &O77777 ist, ergeben bei der dezimalen Ausgabe einen negativen Wert.

BEISPIEL: Die folgende Tabelle zeigt Ihnen den Zusammenhang zwischen der oktalen Eingabe und der dezimalen Ausgabe von Zahlenwerten:

| oktale Eingabe | dezimale Ausgabe |
|----------------|------------------|
| &O1 | 1 |
| &O2 | 2 |
| : | : |
| : | : |
| &O77777 | 32767 |
| &O100000 | -32768 |
| &O100001 | -32767 |
| : | : |
| : | : |
| &O177776 | -2 |
| &O177777 | -1 |

Die CMD-Kommandos

=====

Die CMD-Kommandos erlauben es Ihnen, weitere Disk-BASIC-Anweisungen sowie die meisten DOS-Befehle als direkte Kommandos oder auch von Programmen ausgehend auszuführen.

Der allgemeine Ausdruck des CMD-Kommandos lautet:

CMD"Befehl"

Wenn der 'Befehl' aus mehreren Zeichen besteht und nicht mit 'S=' oder 'F=' beginnt, interpretiert der Rechner die Eingabe als DOS-Befehl. (siehe CMD"Dosbefehl")

SYNTAX: CMD"C",Parameter

=====

WIRKUNG: Wenn kein Parameter angegeben ist, werden alle Leerzeichen (ausgenommen in Strings) und REMs (') aus dem Programm entfernt.

Der Parameter S bewirkt, daß nur die Leerzeichen (ausgenommen in Zeichenketten und REMs) entfernt werden.

Bei der Parametereingabe R bleiben dagegen alle Leerzeichen erhalten und es werden nur sämtliche REMs entfernt.

ANWENDUNG: Das CMD"C"-Kommando bietet sich als wertvolle Hilfe an, wenn Sie sich aus einem mit REMs und Leerzeichen übersichtlich gestalteten, dafür aber auch umfangreichen Programm eine speicherplatzsparende Arbeitsversion erstellen wollen.

BEMERKUNG: Der Befehl darf nicht aus einem laufenden BASIC-Programm heraus benutzt werden, da alle Variablen bei seiner Ausführung gelöscht und das Programm verändert werden.

Da auch Zeilen gelöscht werden, die ausschließlich aus einem REM bestehen, ist vor der Ausführung des Befehls darauf zu achten, daß diese während des Programmablaufes nicht angesprungen werden.

Unter gewissen Bedingungen kann die Ausführung von CMD"C" problematisch werden. Aus diesem Grunde werden zunächst einmal alle Statements innerhalb einer Zeile komprimiert. Die neu entstandene Zeile wird anschließend wieder in den Ausgangszustand zurückgebracht und wiederum mit dem Zeileninhalt der Originalzeile verglichen. Wenn es bei diesem automatisch ausgeführten Vergleich zu keiner Übereinstimmung kommt, bleibt die Zeile unverändert und die Zeilennummer wird auf dem Bildschirm ausgegeben.

SYNTAX: CMD"E"

WIRKUNG: Dieser Befehl zeigt auf dem Bildschirm den letzten DOS-Fehler an, sofern ein solcher innerhalb eines BASIC-Programmablaufes aufgetreten ist. Andernfalls gibt der Rechner die Meldung "kein Fehler" aus.

SYNTAX: CMD"F",DELETE Anfangszeile-Endzeile

=====

WIRKUNG: Alle Textzeilen im Programm größer/gleich der Anfangszeile und kleiner/gleich der Endzeile werden gelöscht.

ANWENDUNG: Dieser Befehl kann in Verbindung mit dem Befehl MERGE zum Einsatz kommen, indem Sie damit Teile eines laufenden Programmes entfernen und an deren Stelle Programmabschnitte mit MERGE wieder einsetzen, ohne dabei Variableninhalte zu verlieren.

BEMERKUNG: Der zu löschende Programmteil darf nicht Bestandteil einer FOR-NEXT-Schleife, eines Unterprogrammes oder eines DEF FN-Statements sein. Variablen, die durch ein DEF FN im gelöschten Programmteil definiert wurden, bleiben nicht erhalten. Die Größe des Stringbereiches wird durch den Befehl nicht verändert. CMD"F",DELETE darf nicht als direkter Befehl, sondern nur innerhalb eines Programmes aufgerufen werden, und muß dabei immer die letzte Anweisung innerhalb einer Zeile sein, da nachfolgende Statements nicht mehr erkannt werden. Unmittelbar nach CMD"F",DELETE muß die Zeile folgen, mit deren Bearbeitung das Programm fortgesetzt werden soll (z.B. die Zeile, die mit MERGE einen neuen Programmteil von der Diskette einlädt).

BEISPIEL: 10 REM Dieser Teil befand sich auf der Diskette und
20 REM wurde mit Hilfe des MERGE-Befehls geladen

Geben Sie diese beiden Zeilen ein und speichern Sie diese unter dem Namen TEST1/BAS ab.
Tippen Sie nun das folgende Programm ein und starten es mit dem Befehl RUN.

```
10 PRINT"Die Zeilen 10 und 20 werden gelöscht,"
20 PRINT"TEST1/BAS hinzugeladen und alles gelistet."
200 CMD"F",DELETE 10-20
300 MERGE"TEST1/BAS"
400 CLS:PRINT"LISTING: ":PRINT:LIST
```

SYNTAX: CMD"F=KEEP",Variable1,Variable2,Variable3,...

WIRKUNG: Mit Ausnahme der angegebenen und über den Befehl DEF FN definierten Variablen werden alle gelöscht.

BEMERKUNG: Alle nicht zu löschenden Variablen müssen angegeben werden. Handelt es sich bei einer der angegebenen Variablen um den Namen eines Feldes, so bleibt das gesamte Feld erhalten. Alle zu erhaltenden Variablen müssen innerhalb eines Statements stehen. Da es aber mitunter vorkommen kann, daß die Menge der anzugebenden Variablen die maximale Zeilenlänge überschreiten würde, ist es möglich, das Kommando auch über mehrere Zeilen laufen zu lassen. Die Ankündigung einer Folgezeile wird dem Rechner durch ein abschließendes Komma in der fortzusetzenden Zeile mitgeteilt.

BEISPIEL: Der Programmabschnitt

```
:  
:  
1000 CMD"F=KEEP",A$,B#,FV$(0),FSS$(0),  
1010 X,Z!  
:  
:
```

bewirkt, daß bis auf A\$, B#, X, und Z! sowie die Felder FV\$(0) und FSS\$(0) alle Variablen gelöscht werden. Die Variablen in der Zeile 1010 werden mit ins CMD"F=KEEP"-Kommando übernommen, da die Zeile 1000 mit einem Komma abschließt.

SYNTAX: CMD"F=SWAP",Variable1,Variable2

=====

WIRKUNG: Die Werte der zwei angegebenen Variablen werden vertauscht.

BEMERKUNG: Der Typ der beiden zu vertauschenden Variablen muß gleich sein. Es können also z.B. keine Integer-Variablen mit Variablen doppelter Genauigkeit vertauscht werden. Die Namen der Variablen müssen selbstverständlich unterschiedlich sein.

BEISPIELE: CMD"F=SWAP",A\$,B\$

tauscht den Inhalt der Zeichenkette A\$ mit dem Inhalt von B\$.

SYNTAX: CMD"F=POPN",Variable

=====

WIRKUNG: Der Befehl bezieht sich ausschließlich auf die FOR-NEXT-SCHLEIFE mit der angegebenen Zählvariablen. Die Schleife wird gelöscht und der weitere Programmablauf erfolgt mit dem Statement hinter dem CMD"POPN"-Befehl. CMD"POPN" hat also den gleichen Effekt wie das letzte NEXT der Schleife. Wenn keine Variable angegeben wird, löscht der Rechner die zur Zeit bearbeitete FOR-NEXT-Schleife.

SYNTAX: CMD" F=POPR "

=====

WIRKUNG: Die augenblicklich ausgeführte Unterroutine und die in ihr enthaltenen FOR-NEXT-Schleifen werden gelöscht.
Dadurch ist der Befehl einem RETURN ähnlich, jedoch mit dem Unterschied, daß der Programmablauf nicht mit dem hinter dem GOSUB stehenden Statement, sondern mit dem auf das CMD"POPR" folgenden Statement fortgeführt wird.

SYNTAX: CMD"F=POPS"

WIRKUNG: Das Kommando löscht alle offenen Unterprogramme und FOR-NEXT-Schleifen indem die Rücksprungadressen und Kontrollblöcke auf die Anfangsposition gesetzt werden.

ANWENDUNG: Der Befehl kann in unübersichtlichen Programmen eingesetzt werden, um sicher zu sein, daß der Stack wieder zurückgesetzt ist. Speziell bei rekursiver Programmierung kann es sonst zu Fehlern im Programmablauf kommen.

SYNTAX: CMD"F=SASZ",Zahl

=====

WIRKUNG: Dieses Statement ändert den Stringbereich, ohne daß die schon vorher belegten Variablen ihren Wert verändern oder gelöscht werden.
(SASZ bedeutet "String Area Size".)

ANWENDUNG: Wenn sich bei einem Programmlauf herausstellt, daß der festgelegte Stringbereich zu groß oder zu klein ist, kann dieses Kommando nutzbringend eingesetzt werden.

BEMERKUNG: Die 'Zahl' muß einen genügend großen Wert für die Zeichenkettenoperationen beinhalten. Wenn die 'Zahl' zu groß oder zu klein ist, erfolgt eine Fehlermeldung des Rechners.

SYNTAX: CMD"F=ERASE",Variable1,Variable2,Variable3,...

=====

WIRKUNG: Mit Hilfe dieses Kommandos ist es möglich innerhalb eines Programms die angegebenen Variablen zu löschen.

BEMERKUNG: Wenn die Variablen einem Feld angehören und dieses nicht mehr benötigt wird oder mit einer anderen Größe neu definiert werden soll, kann ebenfalls ein CMD"F=ERASE" ausgeführt werden, wobei der dabei im Rechner entstehende freie Speicherplatz wieder zur Verfügung steht.

BEISPIEL:

```
10 CLS: CLEAR500: DIM A$(80)
20 PRINT "Feld auf 80 dimensionieren --> MEM ="; MEM
30 FOR I=1 TO 80: A$(I) = "TEST": NEXT I
40 FOR I=1 TO 80: PRINT A$(I);: NEXT: PRINT
50 CMD "F=ERASE", A$(0): DIM A$(5)
60 PRINT "Feld löschen und auf 5 dimensionieren --> MEM ="; MEM: PRINT
70 FOR I=1 TO 5: A$(I) = STRING$(64, 64+I): NEXT I
80 FOR I=1 TO 5: PRINT A$(I);: NEXT
90 GOTO 90
```

Zunächst wird durch Zeile 10 das Feld A\$ auf 80 Elemente dimensioniert. Zeile 30 ordnet dann allen Elementen den String "TEST" zu. Das CMD"F=ERASE" in Zeile 50 löscht das Feld wieder und es kommt zu einer erneuten Feld-Dimensionierung. Achten Sie auf den unterschiedlichen freien Speicherplatz, der durch PRINT MEM angezeigt wird.

SYNTAX: CMD"F=SS",Parameter

=====

WIRKUNG: Ein Programm wird nach der Eingabe dieses Befehls in Einzelschritten ausgeführt und die jeweils bearbeitete Zeilennummer hinter dem Zeichen @ in der oberen rechten Ecke des Bildschirms angezeigt. Durch Drücken der NEW LINE-Taste wird das Programm dann zeilenweise abgearbeitet.

ANWENDUNG: Oftmals ist es bei der Programmierung sinnvoll, ein Programm Schritt für Schritt, mit gleichzeitiger Anzeige der gerade bearbeiteten Zeilennummer, abzuarbeiten.

Auf diese Art lassen sich beispielsweise Programmierfehler leicht aufspüren.

Folgende Möglichkeiten sind durch Angabe von Parametern gegeben:

CMD"F=SS"

Schaltet den Rechner auf Einzelschrittverarbeitung (SS=Single Step)

CMD"F=SS",Zeilennummer

Schaltet den Rechner bei Erreichen der angegebenen Zeilennummer in den Einzelschritt-Modus.

CMD"F=SS",N

Schaltet die Einzelschrittverarbeitung wieder ab

BEMERKUNG: Wenn die Einzelschrittverarbeitung beim Programmende nicht wieder abgeschaltet wird, bleibt der Rechner auch bei einer nachfolgenden Verarbeitung eines neu in den Arbeitsspeicher geladenen Programms in diesem Modus.

In diesem Fall kann der Befehl zur Aufhebung auch als direktes Kommando innerhalb der BASIC-Ebene eingegeben werden, bevor das neue Programm gestartet wird.

SYNTAX: CMD"J",Datum1,Datum2

=====

WIRKUNG: 'Datum1' wird konvertiert in die Stringvariable 'Datum2' geschrieben. Dabei muß 'Datum1' im amerikanischen Format (mm/tt/jj) vorliegen. 'Datum2' enthält dann nach Ausführung des Befehls CMD"J" den entsprechenden Tag des Jahres in der Form ttt. Der umgekehrte Fall ist ebenso durchführbar.

BEMERKUNG: Wenn ein Datum der Form mm/tt/jj erzeugt werden soll, so ist zu beachten, daß 'Datum1' in der Form -jj/ttt angegeben werden muß.

BEISPIELE: Umwandlung eines Datums in die entsprechende Tageszahl des angegebenen Jahres :

```
10 CLS: CLEAR500
20 DEF FN D1$(A$)=MID$(LEFT$(A$,8),4,3)+LEFT$(LEFT$(A$,8),3)+RIGHT$(LEFT$(A$,8),2)
30 LINEINPUT"Geben Sie das heutige Datum in deutscher Form ein ";D$:DD$=D$
40 D$="DATUM "+D$:CMD D$
50 D1$=FN D1$(D$)
60 MID$(D1$,3,1)="/" : MID$(D1$,6,1)="/"
70 CMD"J",D1$,D2$
80 PRINT"Das Datum ";DD$;" gibt den ";D2$;" Tag des Jahres 19"+RIGHT$(D1$,2);" an."
```

Umwandlung einer angegebenen Tages- und Jahreszahl in das entsprechende Datum:

```
10 CLS: CLEAR500
20 DEF FN D$(D2$)=MID$(D2$,4,3)+LEFT$(D2$,3)+RIGHT$(D2$,2)
30 LINEINPUT"Bitte Tages- und Jahreszahl in der Form -jj/ttt eingeben: ";D1$
40 CMD"J",D1$,D2$
50 MID$(D2$,3,1)="." : MID$(D2$,6,1)="."
60 PRINT"Tages- und Jahreszahlen entsprechen dem ";FN D$(D2$)
```

SYNTAX: CMD"O",Parameterliste

=====

WIRKUNG: Dieses Kommando wird dazu benutzt, dimensionierte Variablen im Speicher des Rechners zu sortieren. Vom Inhalt der Parameterliste ist es abhängig, ob eine direkte oder eine Index-Sortierung ausgeführt wird.

ANWENDUNG: Folgende Parameter sind möglich:

CMD"O",n,Sortart,Variable(Feld1),Variable(Feld2),...
CMD"O",n,Sortart,Stringvariable(Feld1)(x,y),...
CMD"O",n,*Indexvar(Feld),Sortart,Variable(Feld1),...

Die ersten beiden Parameterlisten haben eine direkte Sortierung zur Folge.

n gibt die Anzahl der von der Sortierung zu behandelnden Elemente eines Feldes an. Die dadurch evtl. nicht mit angegebenen Variablen bleiben von der Sortierung unberücksichtigt. Wenn n=0 ist, richtet sich die Sortierung nach der Anzahl der im ersten Feld stehenden Variablen. Wird n größer angegeben, als Variablen im Feld vorhanden sind, kommt es zu einem FC-Error.

Mit 'Sortart' kann die Art der Sortierung definiert werden. Wenn z.B. Beträge nach ihrer Größe (der größte zuerst) oder Zeichenketten in umgekehrter alphabetischer Reihenfolge sortiert werden sollen, zeigt dies ein '-' (Minuszeichen) als 'Sortart' dem Rechner an. Wird die 'Sortart' nicht festgelegt, erfolgt die Sortierung in aufsteigender Reihenfolge. Der 'Sortart' folgen die zu sortierenden Variablenfelder. Mindestens ein Feld muß angegeben werden. Die Sortierung kann auf maximal neun Felder ausgedehnt werden. Bei der Sortierung hat das erste Feld die höchste und das letzte Feld die niedrigste Priorität.

Stringvariablen werden nach dem numerischen Wert ihres ASCII-Characters sortiert. Hat ein String die Länge Null, so wird ein negativer Wert angenommen. In der Regel dienen dem Sortiervorgang alle in einer Stringvariablen enthaltenen Zeichen. Sie können jedoch ähnlich wie bei der MID\$-Funktion eine Begrenzung in der Form (x,y) angeben. In diesem Fall beginnt die Sortierung bei der Startposition x und bezieht sich von dort ausgehend auf einen y Zeichen langen Teil der Variablen.

Bei der direkten Sortierung ist die dabei entstehende Folge der Elemente in allen Feldern gleich. Die Sortierung der Felder wird nicht nacheinander ausgeführt, sondern immer gleichzeitig. Wenn also z.B. im ersten Feld das 10. Element an die 20. Stelle gerückt wird, vollzieht sich dieser Vorgang auch in allen anderen Feldern.

Da bei der direkten Sortierung die Variablenzeiger im Speicher des Rechners verändert werden, hat das zur Folge, daß beim Abspeichern einer in ihrer Position verschobenen Variablen auf die Diskette auch alle weiteren, diesem Feld zugehörigen Variablen zurückgeschrieben werden müssen.

Bei einer Index-Sortierung ist dies nicht der Fall!

Ein indirekter Sortiervorgang (Index-Sortierung) wird dem Rechner, wie in der dritten aufgeführten Parameterliste dargestellt, durch Eingabe eines '*' (Stern) vor dem Indexvariablenfeld mitgeteilt. Die folgenden Angaben der Parameterliste entsprechen der Liste der direkten Sortierung. Bei einer Indexsortierung muß ein Indexvariablenfeld als Integerfeld sowie das erste Variablenfeld angegeben sein. Weitere Variablenfelder (2 - 9) können noch hinzugefügt werden.

Beim Sortiervorgang wird lediglich das Indexvariablenfeld sortiert und verändert. In ihm sind die Zeiger enthalten, die einen sortierten Zugriff auf die weiteren Variablenfelder ermöglichen.

Da diese von der Sortierung unberührt bleiben, muß nach der Ausführung des CMD"O"-Befehls nur noch das Index-Feld auf die Diskette zurückgeschrieben werden.

BEISPIELE: Das folgende Programm erzeugt ein Variablenfeld mit dem Namen RN!, in dem zunächst 30 Variablen mit randommäßig angelegten Zufallszahlen zwischen Null und 100 unsortiert abgelegt sind. In der Zeile 70 werden dann alle Variablen dieses Feldes in absteigender Reihenfolge (Minuszeichen!) direkt sortiert.

```
10 CLS: DIM RN!(40)
20 PRINT "Unsortiert:"
30 FOR I=0 TO 29
40 RN!(I)=RND(100)
50 PRINT RN!(I);
60 NEXT I: PRINT
70 CMD"O", 30, -RN!(0)
80 PRINT: PRINT "Sortiert:"
90 FOR I=0 TO 29: PRINTRN!(I);: NEXT I
```

Das nächste Programm zeigt die Anwendung einer Index-Sortierung von Stringvariablen. Wichtig ist, daß die FOR-NEXT-Schleifen immer bei 0 beginnen, da sonst nicht alle Namen bei der Sortierung berücksichtigt werden.

Zunächst werden dem Variablenfeld N\$ zehn Elemente in unsortierter Reihenfolge zugewiesen. Die indirekte Sortierung (Stern!) dieser zehn Variablen erfolgt durch Zeile 70 in aufsteigender alphabetischer Reihenfolge.

```

10 CLS: CLEAR500: DIM IN$(10), N$(30)
20 PRINT "Geben Sie bitte 10 verschiedene Namen ein:"
30 PRINT
40 FOR I=0 TO 9
50 PRINT USING "##. "; I+1; : LINEINPUT N$(I)
60 NEXT I
70 CLS: PRINT "Unsortiert:": PRINT
80 FOR I=0 TO 9
90 PRINT USING "##. "; I+1; : PRINT N$(I)
100 NEXT I
110 CMD"O", 10, *IN$(0), N$(0)
120 PRINT@30, "Alphabetisch sortiert:": PRINT
130 U=158
140 FOR I=0 TO 9
150 PRINT@U, ; N$(IN$(I));
160 U=U+64
170 NEXT I

```

```

CMD"O", 100, N$(0)

```

Die ersten 100 Variablen des angegebenen Feldes werden in aufsteigender Reihenfolge direkt sortiert. Sollte das Feld mehr als 100 Elemente beinhalten, bleiben die restlichen unbeachtet und werden nicht sortiert.

```

CMD"O", 150, -N$(50)

```

Vom Variablenfeld mit dem Namen N\$ werden die Elemente N\$(50) bis N\$(149) in absteigender alphabetischer Folge direkt sortiert, die restlichen Variablen bleiben unberücksichtigt.

```

CMD"O", 0, A(1), N$(1)(4,3)

```

Die Felder A und N\$ werden in aufsteigender Reihenfolge, bezogen auf Feld A, direkt sortiert. Sollte Feld A mehrere Variablen mit gleichem Wert besitzen, wird nach Feld N\$ sortiert, wobei in diesem Fall nur das 4., 5. und 6. Zeichen der Variablen herangezogen wird. Alle Elemente ab A(1) und N\$(1) werden zur Sortierung herangezogen, da die Anzahl n mit Null angegeben ist.

CMD"O",100,*IN%(0),A(1),N\$(1)

Bei dieser indirekten Sortierung werden zunächst die ersten 100 Elemente des Indexvariablenfeldes IN% automatisch durch den Rechner belegt. In diesem Feld befinden sich dann die Variablenzeiger auf die Felder A und N\$. Dieses Indexfeld wird entsprechend dem Inhalt der Variablenfelder sortiert, sodaß diese, in ihrer Reihenfolge weiterhin unverändert, trotzdem in aufsteigender alphabetischer Folge mit den Integerwerten aus IN% als Index abgerufen werden können. IN%(0) steht dann für das kleinste Element aus A bzw. N\$ und IN%(99) für das größte. A(0) und N\$(0) werden nicht zur Sortierung herangezogen.

SYNTAX: CMD"R"

=====

WIRKUNG: Aktiviert die systeminterne Interruptkette. Zeit-
anzeige, Autorepeatfunktion etc. werden wieder
aktiv. (Ein CMD"R" entspricht dem Z80-Assembler-
befehl EI.)

ANWENDUNG: Der Befehl wird immer dann benutzt, wenn die Inter-
ruptkette ausgeschaltet war.

SYNTAX: CMD"S"

=====

WIRKUNG: Verlassen des BASIC mit Übergang in die DOS-Befehlsebene.

ANWENDUNG: Verschiedene DOS-Funktionen lassen sich nicht als CMD"Dosbefehl" aus dem BASIC aufrufen. Dazu wird das BASIC über CMD"S" verlassen.

SYNTAX: CMD"T"

=====

WIRKUNG: Schaltet die systeminterne Interruptkette ab. Auto-repeatfunktion, Zeitanzeige etc. sind nicht mehr möglich. (CMD"T" entspricht dem Z80-Assemblerbefehl DI.)

ANWENDUNG: Wenn der Kassettenrecorder noch als Datenspeicher neben der Diskettenstation benutzt werden soll, (z.B. bei Genie I/II und Speedmaster), ist vor einer Programm- oder Datenübertragung vom oder zum Recorder der Befehl CMD"T" einzugeben, da sonst durch die Interrupts des Rechners Daten verloren gehen könnten.

SYNTAX: CMD"Dosbefehl"

=====

WIRKUNG: Über dieses Kommando ist der Aufruf eines DOS-Befehls aus der BASIC-Ebene möglich. Die Befehle können in Programme eingesetzt werden. Der Rechner führt dann den DOS-Befehl aus, kehrt anschließend ins BASIC zurück und setzt, wenn das aufrufende Programm nicht durch das CMD-Kommando zerstört wurde, den Programmablauf fort.

BEMERKUNG: Der hinter dem CMD folgende DOS-Kommandostring wird vom BASIC in den DOS-Eingabepuffer geschrieben und dort bearbeitet. Aus diesem Grunde darf der DOS-Befehl auch nur maximal 80 Zeichen enthalten. Wenn der DOS-Befehl fehlerhaft eingegeben wird, zeigt der Rechner dies mit einer entsprechenden Fehlermeldung an, bricht mit der abschließenden Meldung "DOSfehler" die Ausführung des Kommandos ab und kehrt wieder ins BASIC zurück.

BEISPIELE: CMD"I 0"

Das Inhaltsverzeichnis der Diskette in Laufwerk 0 wird angezeigt. Es genügt nicht, nur CMD"I" anzugeben, da in diesem Falle der Rechner versucht, den Befehl in der erweiterten BASIC-Befehlsliste zu finden. Besteht ein DOS-Befehl wie in diesem Fall nur aus einem Zeichen, muß immer noch ein Leerzeichen angehängt werden (CMD"I ").

CMD"DIRCHECK/CMD"

Das Programm DIRCHECK/CMD wird aus dem BASIC heraus aufgerufen und gestartet.

CMD"COPY SUPER/CMD:0 :1"

Das Programm SUPER/CMD wird von der Diskette in Laufwerk 0 auf die Diskette in Laufwerk 1 kopiert.

CMD"PD 0 1=0,A"

Die PDrive-Einstellung für Laufwerk 1 wird geändert, Laufwerk 0 entsprechend eingestellt und sofort aktiviert.

Dateien (Files)

=====

Das nun folgende Kapitel dürfte wohl die Beschreibung der wichtigsten Funktion Ihres Rechners enthalten - Verarbeitung und Handhabung von Dateien.

Unter einer Datei, (auch File genannt), versteht man allgemein eine nach festgelegten Kriterien geordnete Datenmenge. Dies kann natürlich auch ein Programm, der Inhalt des Bildschirmspeichers oder ähnliches sein. Die jeweilige Datenmenge ist auf der Diskette derart abgelegt, daß von Programmen aus auf sie zugegriffen werden kann. Aus diesem Grunde muß jede Datei einen für sie spezifischen Namen erhalten.

Zum Aufbau dieses Dateinamens gibt es einige Bestimmungen auf die an dieser Stelle näher eingegangen werden soll.

Die Bezeichnung einer Datei hat folgenden allgemeinen Aufbau:

Name/Extension.Kennwort:Laufwerksnummer

Der eigentliche Name darf aus bis zu acht alphanumerischen Zeichen bestehen, wobei darauf zu achten ist, daß das erste Zeichen stets ein Buchstabe sein muß. Es steht ihnen frei, ob Sie noch eine sogenannte Extension hinter dem Namen anbringen, die es Ihnen später erleichtert, die Art einer Datei, z.B. nach Aufruf des Inhaltsverzeichnisses, zu erkennen. Sie besteht aus einem Schrägstrich, gefolgt von bis zu 3 alphanumerischen Zeichen. Wenn die Ergänzung nicht angegeben wird, verwendet der Rechner an ihrer Stelle Leerzeichen.

Folgende Ergänzungen sind üblich:

| | |
|------|---|
| /SYS | Systemprogramm zum DOS |
| /CMD | Maschinenprogramm, (Start erfolgt durch Aufruf des Namens in der DOS-Ebene) |
| /BAS | BASIC-Programm, (nach Laden des Disk-BASIC erfolgt der Start mit Hilfe des Befehls RUN) |
| /DAT | Datei zu einem Programm gleichen Namens |
| /TXT | Textdatei |
| /JOB | Arbeitsdatei für Programmverkettungen, (Aufruf über den DOS-Befehl DO) |
| /OBJ | Datei in Maschinencode |

Das Kennwort dient zum Schutz der Datei vor unerlaubtem Zugriff. Es besteht wie der Dateiname aus acht alphanumerischen Zeichen, wobei auch hier das erste Zeichen ein Buchstabe sein muß. Die Kennwortkontrolle muß über den Systemparameter AA=J eingeschaltet sein. Das Kennwort für eine Datei wird mit dem DOS-Befehl ATTRIB vereinbart, ebenso die Art des erlaubten Zugriffs.

Enthält Ihre Anlage mehrere Laufwerke, ist es ratsam, hinter der Extension, mit einem Doppelpunkt vorweg, auch noch die Nummer des Laufwerks anzugeben, auf dem die Datei gespeichert oder geladen werden soll. Wenn Sie auf diese Nummer verzichten, obwohl mehrere Laufwerke angeschlossen sind, so wird die Datei auf allen angeschlossenen Laufwerken gesucht. Wird die Datei nicht gefunden, benutzt G-DOS zum Abspeichern einer neuen Datei das Laufwerk, das mit dem Systemparameter AO vereinbart wurde. Sollte die nicht gefundene Datei nur gelesen werden, erfolgt die Fehlermeldung "Datei nicht gefunden". Sollte ein Laufwerk angesprochen werden, dessen Tür geöffnet ist oder das keine Diskette enthält, erfolgt die Fehlermeldung "Gerät nicht verfügbar". Wird eine ungültige Laufwerksnummer angegeben, z.B. ":9", quittiert G-DOS diesen Versuch mit "unzulässiges oder fehlendes Laufwerk".

Sequentielle Dateien (PRINT/INPUT-Files)

=====

Die einfachste Art der Dateiverwaltung erfolgt mit Hilfe von sequentiellen Dateien.

Dabei sind einzelne Datenblöcke logisch hintereinander auf der Diskette gespeichert (im Prinzip wie bei einem Kassettenrecorder). Das Ende eines Datenblocks wird durch den Rechner automatisch gekennzeichnet, indem er ein "EOF" (End Of File) an den Schluß setzt.

Der Nachteil der sequentiellen Dateiverwaltung liegt darin, daß bei einer Veränderung der Dateidaten immer der gesamte Inhalt einer Datei von der Diskette ein- und ausgelesen werden muß.

Voraussetzung für jegliche Dateibearbeitung ist das Öffnen mittels des OPEN-Befehls.

SYNTAX: OPEN "Modus", #Puffernummer, "Dateiname"

=====

WIRKUNG: Um eine neue Datei anzulegen oder eine bereits vorhandene zu bearbeiten muß sie mittels des OPEN-Befehls geöffnet werden. Über den "Modus" teilen Sie dem Rechner mit, ob eine Aus- oder Eingabe auf den durch "Dateiname" definierten File erfolgen soll.

"I" (INPUT) als "Modus" zeigt dem Rechner an, daß von der angegebenen sequentiellen Datei gelesen werden soll.

"O" (OUTPUT) als "Modus" zeigt dem Rechner an, daß auf die angegebene sequentielle Datei eine Ausgabe erfolgen soll.

"E" (EXTEND) als "Modus" zeigt dem Rechner an, daß weitere Daten an eine bereits existierende sequentielle Datei angehängt werden sollen.

Als "#Puffernummer" kann eine beliebige Zahl zwischen 1 und 15 angegeben werden. Sie dient bei den Ein- und Ausgaben von Daten als Kurzzeichen der geöffneten Datei. Das Zeichen "#" kann auch entfallen. Zwei zu öffnende Dateien dürfen nicht mit der gleichen Puffernummer versehen werden. Der Rechner gibt dann die Fehlermeldung "Datei war offen" aus.

Die Puffernummern dürfen nicht größer sein als die Anzahl der Dateien, die beim Aufruf vom BASIC eingegeben wurden (maximal 15). Fehlt eine solche Angabe, können 3 Dateien gleichzeitig geöffnet werden (Puffernummer 1-3).

Wenn einer Datei ein derartiges Schutzwort zugeteilt wird, muß dieses bei jedem folgenden OPEN-Befehl immer mit angegeben werden, da sonst der Dateizugriff verwehrt wird. Der Dateiname und das Kennwort werden durch einen Punkt voneinander getrennt. Dieser "Dateischutz" funktioniert jedoch nur, wenn im DOS die Systemparameter entsprechend eingestellt sind (AA=J).

Es ist nicht möglich eine Datei unter gleichem Namen mit und ohne Kennwort auf der Diskette abzulegen. In diesem Fall gibt der Rechner die Fehlermeldung "Dateizugriff verwehrt" aus. Eine Datei kann zu einem Zeitpunkt immer nur in einem Modus geöffnet werden. Eine weiterer Öffnungsversuch hätte die Fehlermeldung "Datei war offen" zur Folge.

Um eine sequentielle Datei beispielsweise vom INPUT- auf den OUTPUT-Modus zu schalten, muß sie zwischenzeitlich geschlossen werden.

BEISPIELE: OPEN"O",7,"TEST/DAT.SECRET:1"

Voraussetzung zur Ausführung dieses Befehls ist der Aufruf des Disk-BASIC unter Bereitstellung von mindestens 7 Dateipuffern (BASIC,7). Die sequentielle Datei "TEST/DAT" mit dem Kennwort "SECRET" auf der Diskette in Laufwerk 1 wird dann zur Ausgabe von Daten geöffnet und die 7 als Puffernummer zugewiesen.

OPEN"O",1,"VERSUCH/BAS:0"

Das BASIC-Programm VERSUCH/BAS auf der Diskette in Laufwerk 0 wird als sequentielle Datei unter der Puffernummer 1 zur Ausgabe von Daten geöffnet. Ein Kennwort ist nicht angegeben.

OPEN"I",#1,"TEST/TXT"

Die Textdatei "TEST/TXT" wird unter Puffernummer 1 zur Eingabe vorbereitet. Da keine Laufwerknummer angegeben ist, sucht der Rechner, beginnend bei Laufwerk 0, nach dem Dateinamen und öffnet den ersten File, den er unter dem angegebenen Namen findet.

OPEN MO\$,X,NA\$

Sie können auch Variablen an der Stelle von "Modus", "Puffernummer" und "Dateiname" eingeben, sofern ihr Inhalt nicht mit den Bestimmungen der OPEN-Parameter in Konflikt gerät.

MO\$ ist in unserem Beispiel ein String, der den gewünschten Modus enthält, X ist die Puffernummer als Zahlenvariable und in NA\$ steht der Dateiname.

SYNTAX: CLOSE Puffernummer,Puffernummer,Puffernummer,...

WIRKUNG: Soll die Ein- oder Ausgabe auf eine Datei beendet werden, so ist sie wieder zu schließen.

Dieser Befehl beendet den Zugriff auf die durch die Puffernummern angegebenen offenen Dateien. Die im Puffer definierten Variablen stehen nicht mehr zur Verfügung und eine Änderung der Dateigrösse wird im Inhaltsverzeichnis der Diskette eingetragen.

BEMERKUNG: Eine CLOSE ohne angegebene Puffernummern schließt alle offenen Dateien.

Bei einer sequentiellen Datei werden zuerst die Puffer gelöscht und anschließend die Datei geschlossen.

Wenn Sie einen NEW-Befehl eingegeben schließt der Rechner ebenfalls automatisch alle offenen Dateien und löscht dann das Programm und die Variablen, die sich in seinem Speicher befinden.

Es ist sehr wichtig, daß vor dem Wechsel einer Diskette immer alle Dateien geschlossen werden. Es können sonst Daten verloren gehen, da das Ende der Datei nicht ins Inhaltsverzeichnis eingetragen wurde. Im Extremfall wird dadurch die gesamte Diskette unlesbar.

Eine Datei muß auch geschlossen sein, wenn sie gelöscht werden soll. Ansonsten zeigt der Rechner beim nächsten CLOSE die Fehlermeldung "schlechte FCB Daten" an.

BEISPIELE: CLOSE

Schließt automatisch alle offenen Dateien.

CLOSE 1,4,7,12

Schließt die Dateien, denen beim OPEN-Befehl die Puffernummern 1, 4, 7 und 12 zugewiesen wurden.

Bei einer sequentiellen Datei werden alle Zeichen im ASCII-Format ausgegeben. Aus diesem Grunde müssen Sie nicht druckbare Zeichen mittels CHR\$(X) ausgeben.

Für die Ein- und Ausgabe von Daten auf sequentielle Dateien werden folgende Anweisung verwendet:

| | |
|--------------|------------|
| PRINT# | LINEINPUT# |
| PRINT# USING | EOF(x) |
| INPUT# | |

SYNTAX: PRINT#Puffernummer,Variablenliste

WIRKUNG: Dieser Befehl schreibt die in der Variablenliste angegebenen Daten sequentiell in eine zuvor durch den Befehl OPEN spezifizierte Datei auf die Diskette.

Die Angabe der zur geöffneten Datei gehörenden Puffernummer (1-15) ist unbedingt notwendig.

BEMERKUNG: Innerhalb der Variablenliste können auch Zeichenketten stehen.

Wenn in der Variablenliste mehrere Variablen aufgeführt werden, so sind sie durch ein Semikolon voneinander zu trennen.

Zeichenkettenvariablen muß als Trennzeichen immer die Kombination Semikolon-Anführungszeichen-Komma-Anführungszeichen-Semikolon folgen.

Diese Prozedur ist auch bei der Ausgabe von Zeichen über den Befehl CHR\$(X) notwendig.

Die gleiche Wirkung der Trennzeichen hat auch ein neuer PRINT#-Befehl in einer weiteren Zeile.

Auf den ersten Blick erscheint Ihnen diese Sache vielleicht etwas umständlich. Sie wird jedoch verständlich, wenn man weiß, daß beim sequentiellen Einlesen von Daten über INPUT# nur Kommata oder ein weiteres INPUT# in einer neuen Zeile als Trennzeichen angesehen werden. Dies bedeutet natürlich auch, daß alle Kommata innerhalb von Zeichenkettenvariablen als Trennzeichen betrachtet werden, also der einem Komma folgende Text bereits der nächsten Variablen zugeordnet wird.

BEISPIEL: PRINT#3,A;B;A\$;"","";"B\$";","";"TESTTEXT"

Diese PRINT-Anweisung gibt zunächst die Zahlenvariablen A und B auf eine geöffnete sequentielle Datei aus. Diese Variablen, sowie die erste darauf folgende Zeichenkette, sind durch ein Semikolon getrennt. Im weiteren Verlauf der Ausgabe befinden sich nur die Zeichenketten A\$, B\$ und "TESTTEXT", die durch die Zeichenkombination ";" voneinander getrennt werden müssen.

SYNTAX: PRINT#Puffernummer,USING"Format";Variablenliste

=====

WIRKUNG: Wie bei der normalen PRINTUSING-Funktion aus dem Level II BASIC ermöglicht Ihnen diese Funktion, Daten in einem bestimmten Format in eine sequentielle Datei auszugeben.

BEMERKUNG: Als Daten können numerische Werte oder Zeichenketten verarbeitet werden. Das Format läßt sich sowohl direkt als auch über Stringvariablen festlegen. Folgende Zeichen können benutzt werden:

Dieses Zeichen gibt die Stellung jeder Ziffer eines Zahlenwerts an. Die Anzahl der #-Zeichen bildet das gewünschte Format. Wenn der Zahlenwert kleiner ist als die Anzahl der formatierten Stellen im Feld, werden die nicht benutzten Stellen links von der Zahl mit Leerstellen und rechts mit Nullen aufgefüllt.

Ein Dezimalpunkt kann irgendwo in das formatierte Zahlenfeld eingesetzt werden. Es erfolgt eine automatische Rundung der Zahl, wenn durch Formatierung Nachkommastellen unterdrückt werden. Wird ein Komma in eine Position zwischen der ersten Ziffer und dem Dezimalpunkt gesetzt, so erscheint in der Ausgabe nach je drei Vorkommaziffern ein Komma.

% Das %-Zeichen dient bei der Eingabe zur Definition eines Zeichenfeldes, welches mehr als ein Zeichen enthält. Die Länge eines derart formatierten Feldes ist gleich der Anzahl der Leerzeichen zwischen den Prozentzeichen plus 2.

In der Ausgabe finden wir das %-Zeichen, wenn das angegebene Feld nicht groß genug ist, um alle Ziffern des Zahlenwerts aufzunehmen.

Tritt dieser Fall ein, so wird hinter dem %-Zeichen die gesamte Zahl unformatiert angezeigt.

** Zwei Sterne am Anfang eines Feldes bewirken, daß die unbenutzten Positionen links vom Dezimalpunkt mit dem *-Zeichen aufgefüllt werden. Beachten Sie dabei, daß auch die beiden Sterne weitere Stellen im Feld bilden.

\$\$ Zwei Dollarzeichen am Anfang eines Formatfeldes erzeugen ein gleitendes \$-Zeichen vor der höchsten Stelle der ausgegebenen Zahl.

**\$ Es handelt sich hier um die Kombination zwischen ** und \$\$. Es werden also leere Positionen links von der Zahl mit Sternen aufgefüllt und vor der höchsten Stelle ein \$-Zeichen ausgegeben.

- + Ein Plus am Anfang oder Ende des Formatfeldes veranlaßt den Rechner bei positiven Zahlen ein '+' und bei negativen Zahlen ein '-' an Anfang bzw. Ende der Ausgabe zu setzen.
- Ein Minus am Ende der Formatanweisung hat die Ausgabe eines '-' hinter negativen Zahlen und eines Leerzeichens hinter positiven Zahlen zur Folge.
- ! Das Ausrufezeichen bewirkt, daß nur der erste Buchstabe der angegebenen Zeichenkette ausgegeben wird. Weiterhin ist es bei der Angabe mehrerer Zeichenketten innerhalb der Variablenliste möglich, jeweils die ersten Buchstaben der einzelnen Strings miteinander zu verketteten. Werden im Format mehrere '!' verwendet, folgen bei der Ausgabe dem ersten Buchstaben jeder angegebenen Zeichenkette soviele Leerzeichen, wie zwischen den Ausrufezeichen im Format stehen.

BEISPIELE: PRINT#1,USING"*\$####.##;1234.5678

Diese Anweisung hätte die Ausgabe von **\$1234.57 auf die sequentielle Datei mit der Puffernummer 1 zur Folge.

```

10 A$="TROMMESCHLÄGER"
20 B$="COMPUTER"
30 C$="SYSTEME"
40 OPEN"O",3,"TESTDAT"
50 PRINT#3,USING"!";A$,B$,C$
60 CLOSE

```

Dieses Programm schreibt in die Datei TESTDAT, die die Puffernummer 3 trägt, jeweils den ersten Buchstaben der 3 angegebenen Strings A\$, B\$ und C\$. D.h. auf der Diskette steht als Eintrag ein TCS.

SYNTAX: INPUT#Puffernummer,Variablenliste

WIRKUNG: Auf der Diskette stehende Datenblöcke mit einer Länge bis zum jeweiligen nächsten Trennzeichen werden den in der Variablenliste stehenden Variablen zugewiesen. Die Zuweisung ist von der Reihenfolge der Variablen in der Liste abhängig. Der Variablentyp muß dabei natürlich übereinstimmen.

ANWENDUNG: Der INPUT#-Befehl ermöglicht es, die durch PRINT# in einer sequentiellen Datei abgelegten Daten wieder in den Rechner zu laden und den Variablen zuzuordnen.

BEMERKUNG: Vor einem INPUT# muß die gewünschte sequentielle Datei durch ein OPEN"I" unter Zuweisung einer Puffernummer geöffnet werden. Der Aufbau der Variablenliste und die Anordnung der Trennzeichen innerhalb dieser Liste müssen nicht mit der Variablenliste des PRINT#-Befehls übereinstimmen. Als Trennzeichen zwischen allen Variablentypen genügen Kommata. Natürlich können alle Daten auch einzeln mit Hilfe mehrerer INPUT#-Zeilen gelesen werden.

BEISPIEL:

```
10 A$="Trommeschläger":B$="Computer":C$="Systeme"  
20 D$="GmbH":E$="Sankt Augustin 2":A=5205  
30 OPEN"O",2,"TESTDAT"  
40 PRINT#2,B$;" ";D$;" ";E$;" ";A  
50 PRINT#2,USING"!!! !";A$,B$,C$  
60 CLOSE  
70 OPEN"I",1,"TESTDAT"  
80 INPUT#1,X$,Y$,Z$,P,F$  
90 CLOSE  
100 PRINT F$,X$,Y$,P,Z$
```

Bei diesem Programm kommen gleich mehrere Statements zur Anwendung.

In Zeile 40 erfolgt die Ausgabe der Variablen B\$, D\$, E\$ und A auf die in Zeile 30 geöffnete Datei TESTDAT mit der Puffernummer 2. Anschließend werden die Variablen A\$, B\$ und C\$ mit Hilfe von PRINT USING formatiert ausgegeben, sodaß jeweils nur der erste Buchstabe dieser Zeichenketten abgelegt wird.

Dem ersten Zeichen des Strings C\$ wird dabei noch ein Leerzeichen hinzugefügt.

Nach dem Schließen und erneuten Öffnen der Datei zum Übergang in den INPUT-Modus, liest der Rechner die abgespeicherten Daten wieder ein und ordnet sie den Variablen F\$, X\$, Y\$, Z\$ und P zu, die durch Zeile 100 auf dem Bildschirm ausgegeben werden.

SYNTAX: LINEINPUT#Puffernummer,Zeichenkettenvariable

=====

WIRKUNG: Jede Zeichenkette, einschließlich Kommata und Anführungszeichen, kann mittels dieses Statements von der Diskette eingelesen und der angegebenen Stringvariablen zugewiesen werden.

ANWENDUNG: Da über den normalen INPUT#-Befehl keine Kommata und Anführungszeichen gelesen werden können, sondern diese als Trennzeichen angesehen werden, ist der Einsatz von LINEINPUT# in diesen Fällen unumgänglich.

BEMERKUNG: Wie beim INPUT# muß auch beim LINEINPUT# die gewünschte Datei zuvor zum sequentiellen Einlesen von Daten unter Zuweisung einer Puffernummer geöffnet werden.
Innerhalb eines LINEINPUT#-Statements werden einer Stringvariablen immer soviele Zeichen zugeordnet bis ein Carriage Return (CHR\$(13)) gelesen wird oder 255 Zeichen erreicht sind.

BEISPIEL: Es ist über LINEINPUT# möglich, ein im ASCII-Format abgespeichertes BASIC-Programm Zeile für Zeile verschiedenen Zeichenkettenvariablen zuzuordnen. Das folgende Programm ist zunächst als ASCII-File TEST/BAS (SAVE"TEST/BAS",A) auf der Diskette abzuspeichern und anschließend mit RUN zu starten. Das Programm wird als sequentielle Datei geöffnet und zeilenweise dem Variablenfeld A\$ zugewiesen. Die Ausgabe dieses Variablenfeldes zeigt danach, daß jede Variable jeweils den Inhalt einer Zeile des BASIC-Programms enthält.

```
10 CLS
20 CLEAR 500
30 OPEN"I",1,"TEST/BAS"
40 FORI=1TO9
50 LINEINPUT#1,A$(I)
60 NEXTI
70 FORI=1TO9
80 PRINTA$(I)
90 NEXTI
```

SYNTAX: EOF(Puffernummer)

=====

WIRKUNG: Es handelt sich hier um eine logische Funktion, die den letzten Satz einer sequentiellen Datei erkennt. Es können nur die Werte -1 und 0 bei der Abfrage von EOF vorkommen. Dabei ist EOF gleich -1, wenn das Dateiende erreicht ist und 0, wenn noch weitere Daten folgen.

ANWENDUNG: EOF (End Of File) wird benutzt, um das Ende einer sequentiellen Datei zu finden, wenn Daten von einer Diskette über INPUT# oder LINEINPUT# eingelesen werden und die Länge der Datei nicht bekannt ist.

BEISPIELE: Im Programm kann EOF auf zwei verschiedene Arten eingesetzt werden:
Es ist möglich den Wert von EOF einer Variablen zuzuweisen, z.B. A=EOF(1).
Die Funktion lässt sich aber auch direkt, innerhalb einer IF-Abfrage, verwenden. IF EOF(1) THEN ... bewirkt z.B. die Ausführung des Statements hinter THEN, wenn das Ende der bearbeiteten Datei erreicht ist, ansonsten wird die folgende Programmzeile bearbeitet.

Bedienen wir uns wieder des Programms TEST/BAS von der LINEINPUT#-Funktion mit einer kleinen Änderung. Auch dieses Programm ist mit SAVE"TEST/BAS",A im ASCII-Format auf der Diskette abzulegen und mit RUN zu starten.

```
10 CLS
20 CLEAR 500:DIMA$(100)
30 I=1
40 OPEN"I",1,"TEST/BAS"
50 LINEINPUT#1,A$(I)
60 IF EOF(1) THEN GOTO 80
70 I=I+1:GOTO50
80 CLOSE
90 FORA=1TOI
100 PRINTA$(A)
110 NEXTA
```

Im Gegensatz zum Beispiel von LINEINPUT# ist es bei diesem Programm nicht notwendig, die Länge der einzulesenden Datei TEST/BAS (in diesem Fall Anzahl der Zeilen) zu kennen, da in der Zeile 60 beim Auffinden des Dateiendes eine Programmverzweigung stattfindet und keine weiteren Leseversuche durchgeführt werden.

Damit ist das Kapitel über die Handhabung sequentieller Dateien beendet.

Es folgen nun noch 3 Anwendungsbeispiele zu diesem Abschnitt.

Programm zur Modifizierung von ASCII-Dateien:
(z.B. Umwandlung von PRINT in LPRINT)

Programmlisting: PROGMOD/BAS

```
10 CLS: CLEAR 1000
20 PRINT STRING$(64,42);
30 PRINT "****";
40 PRINT TAB(20) "Programm-Modifizierer";
50 PRINT TAB(61) "****";
60 PRINT STRING$(64,42);
70 PRINT
80 LINEINPUT "Quellprogrammname (ASC) "; QN$
90 LINEINPUT "Zielprogrammname (ASC) "; ZN$
100 LINEINPUT "Ersetze => "; A1$
110 LINEINPUT "durch   => "; A2$
120 OPEN "I",1,QN$
130 IF EOF(1) THEN RUN
140 OPEN "O",2,ZN$
150 IF EOF(1) THEN CLOSE:PRINT:INPUT "<ENTER>";Q$:RUN
160 LINEINPUT#1,A$
170 PRINTA$
180 GOSUB 1000
190 IF X=1 THEN PRINTA$ : X=0
200 PRINT#2,A$
210 GOTO 150
1000 A1%=INSTR(A$,"")+1
1010 IF LEN(A$)-LEN(A1$)+LEN(A2$)>255 THEN RETURN
1020 A%=INSTR(A1$,A$,A1$)
1030 IF A%=0 THEN RETURN ELSE X=1
1040 A$=LEFT$(A$,A%-1)+A2$+MID$(A$,A%+LEN(A1$))
1050 A1%=A%+LEN(A2$)
1060 GOTO 1010
```

Nach Erstellung der Bildschirmmaske erfolgt in den Zeilen 80 und 90 die Abfrage nach Quell- und Zieldatei (ursprüngl. und modifiziertes Programm). Diese Namen müssen ohne Blanks eingegeben werden und die zu behandelnden Programme im ASCII-Format abgespeichert sein.

Die Modifizierung wird in den Zeilen 100 und 110 angegeben. In den Zeilen 120 und 140 werden anschließend die sequentiellen Quell- und Zieldateien geöffnet. Zugleich ermittelt die EOF-Funktion, ob die Quelldatei evtl. leer ist. Trifft dies zu, kommt es zu einem Neustart des Programms, ohne daß die Zieldatei angelegt wird.

Wenn alles ordnungsgemäß abläuft, werden in Zeile 160 über LINE INPUT#1 die Daten (bzw. die Programmzeilen) eingelesen, anschließend gelistet und im Unterprogramm ab 1000 bei Bedarf geändert.

In der Zeile 1000 wird dabei nach dem ersten Blank hinter der Zeilennummer gesucht, um diese von der Modifikation auszunehmen, damit es möglich ist, Ziffern zu verändern, ohne dabei Zeilennummern zu zerstören.

Wurde eine Zeile im Unterprogramm modifiziert, erscheint sie auf dem Bildschirm nochmals in geänderter Form unmittelbar unter der Originalzeile.

Sollte durch die Modifizierung eine Zeilenlänge von 255 Zeichen überschritten werden, erfolgt in der Zeile 1010 ein Rücksprung aus dem Unterprogramm und die Zeile bleibt unverändert.

Das Programm läßt sich in vielfacher Hinsicht verändern. Durch ein DELETE 170 werden z.B. nur noch die modifizierten Zeilen angezeigt. Ferner könnten Sie eine Abfrage einbauen, bevor eine Änderung ausgeführt wird. Zeilen, die durch die Modifikation mehr als 255 Zeichen erhalten würden und daher unverändert bleiben, werden extra angezeigt, wenn das Ende der Zeile 1010 in

```
1010 ...THEN PRINTA$:RETURN
```

umgeändert wird.

Programm zum Formatieren eines Druckerlistings:

Programmlisting: FORMLIST/BAS

```
10 CLS: CLEAR 1000: L=10: R=75: I=8: Z=0: S=1
30 PRINT STRING$(64,42);
40 PRINT "****";
60 PRINT TAB(21) "Listing-Formatierer";
80 PRINT TAB(61) "****";
90 PRINT STRING$(64,42);
110 PRINT: PRINT "Linker Rand ="; L
120 PRINT "Rechter Rand ="; R
130 PRINT "Einrueckung ="; I
140 PRINT "Zeilen/Seite ="; 60
150 PRINT
170 LINEINPUT "Programmname (ASC): "; N$
200 OPEN "I", 1, N$
210 IF EOF(1) THEN RUN
220 GOSUB 3010
230 GOSUB 1000
240 IF EOF(1) THEN GOSUB 4010: RUN
250 GOTO 230
1000 LINEINPUT #1, Z$
1010 LPRINT TAB(L) USING "####"; VAL(Z$);
1020 Z$=MID$(Z$, INSTR(Z$, " ") + 1)
1030 IF LEN(Z$) <= R - I - L THEN P=LEN(Z$): GOTO 1050
1040 GOSUB 2000
1050 LPRINT TAB(I+L) LEFT$(Z$, P): Z=Z+1
1055 Z$=MID$(Z$, P+1)
1060 IF Z=60 THEN GOSUB 4010: GOSUB 3010
1070 IF LEN(Z$)=0 THEN RETURN
1080 GOTO 1030
2000 P=1
2010 PZ=INSTR(P+1, Z$, ":")
2020 IF PZ <> 0 AND PZ < (R-I-L) THEN P=PZ : GOTO 2010
2030 IF P > 1 THEN RETURN
2110 IF P=1 THEN P=R-I-L
2120 RETURN
3000 IF EOF(1) AND LEN(Z$) = 0 THEN RETURN
3010 LPRINT STRING$(2,10);
3020 LPRINT TAB(L)N$;
3030 IF S > 1 THEN LPRINT STRING$(62-L-LEN(N$),32); USING "Seite
##"; S;
3040 S=S+1
3050 LPRINT STRING$(4,10);
3060 RETURN
4000 IF Z=0 THEN RETURN
4010 LPRINT STRING$(66-Z,10);
4020 Z=0: RETURN
```

Zeile 10 beinhaltet die Druckparameter wie linker und rechter Rand, Einrückung und Seitennummerierung. Diese festeingestellten Parameter werden auf dem Schirm nochmals angezeigt, bevor die Abfrage nach dem zu listenden Programm erfolgt.

Anschließend kontrolliert die EOF-Funktion, ob die angegebene Datei auch Daten enthält. Sind keine Daten vorhanden, also EOF(1) erreicht, wird das Programm neu gestartet.

Ansonsten beginnt nun das Auflisten des gewünschten Programms im Unterprogramm ab Zeile 3000 mit dem Druck des Seitenkopfes, worin Programmname und Seitennummerierung enthalten sind.

Als nächstes wird das Unterprogramm ab Zeile 1000 abgearbeitet. Nachdem eine Programmzeile eingelesen ist, erfolgt eine Formatierung der Programmzeilennummer und ihre Ausgabe auf dem Drucker. Die Angaben in 1020 bewirken, daß die Zeilennummer vom übrigen Text abgetrennt und ein neues Z\$ ohne Zeilennummer ermittelt wird. Wenn es sich um eine kurze Programmzeile handelt, kommt sofort 1050 zur Ausführung, die die Anweisungen zum eingerückten Druck der Textzeile enthält. Zwischenzeitlich kontrolliert Zeile 1060 ständig, ob die festgelegte maximale Zeilenzahl/Seite erreicht ist und führt bei Bedarf die Unterprogramme ab 4000 und 3000 aus. (Kontrolle ob noch Text kommt und evtl. erneuter Seitenkopfdruck.)

Ist die zu druckende Programmzeile länger als eine Druckzeile, wird sie im Unterprogramm ab 2000 in einzelne Segmente aufgliedert, wobei Trennungen nur nach abgeschlossenen Befehlen, also hinter Doppelpunkten, unternommen werden. Die Untersuchung daraufhin erfolgt in Zeile 2010. Befindet sich in einer Zeile kein Doppelpunkt, wird beim Erreichen der max. Druckzeilenlänge aufgetrennt.

Bei Gebrauch des Programms ist zu beachten, daß der Befehl LINEINPUT# kein Zeilenende erkennt, wenn unmittelbar zuvor ein Line Feed (Pfeil nach unten) steht. Daher werden in solchen Fällen mehrere Zeilen zusammen eingelesen.

Sollte ein solch zusammengefasster Block länger als 255 Zeichen werden, so wird nur bis zum 255. eingelesen und ein Zeilenende angefügt. Der verbleibende Rest wird beim nächsten LINEINPUT verarbeitet. Hierbei kann eine Zeile ohne Zeilennummer entstehen bzw. die Zeilennummer nicht erkannt werden, was zu unsinnigen Trennungen und Formatierungen führt.

Linefeeds im Text werden auch ausgedruckt, was die Formatierung zerstört (keine Einrückung).

Programm zur Erstellung von bildschirmorientierten Listings:

Programmlisting: LIST64/BAS

```
10 CLS: CLEAR 1000: L=10: R=74: Z=0: S=1
30 PRINT STRING$(64,42);
40 PRINT "****";
60 PRINT TAB(21) "Listing-Formatierer";
80 PRINT TAB(61) "****";
90 PRINT STRING$(64,42);
110 PRINT: PRINT "Linker Rand ="; L
120 PRINT "Rechter Rand ="; R
140 PRINT "Zeilen/Seite ="; 60
150 PRINT
170 LINEINPUT "Programmname (ASC): "; N$
200 OPEN "I", 1, N$
210 IF EOF(1) THEN RUN
220 GOSUB 1000
230 IF EOF(1) THEN GOSUB 4000: RUN
240 GOTO 220
1000 LINEINPUT #1, Z$
1010 IF Z=0 THEN GOSUB 3000
1020 LPRINT TAB(L) LEFT$(Z$,64)
1030 Z=Z+1
1040 IF Z=60 THEN GOSUB 4000
1050 Z$=MID$(Z$,65)
1060 IF LEN(Z$)=0 THEN RETURN
1070 GOTO 1010
3000 LPRINT STRING$(2,10);
3010 LPRINT TAB(L)N$;
3020 IF S>1 THEN LPRINT STRING$(62-L-LEN(N$),32); USING "Seite #
#"; S;
3030 S=S+1
3040 LPRINT STRING$(4,10);
3050 RETURN
4000 IF Z=0 THEN RETURN
4010 LPRINT STRING$(66-Z,10);
4020 Z=0: RETURN
```

Dieses Programm erstellt Ihnen Programmlistings auf einem angeschlossenen Drucker, die dem Ausgabeformat auf dem Bildschirm entsprechen.

In Zeile 10 werden die Druckparameter festgelegt. Nach der Frage nach dem Programmnamen in 170 wird das Programm als sequentielle Datei geöffnet und mit EOF(1) auf Inhalt überprüft. Enthält die Datei Daten, erfolgt die Bearbeitung der Programmzeilen von der Diskette im Unterprogramm ab Zeile 1000. Zeile 1010 veranlaßt gegebenenfalls den Druck eines Seitenkopfes. Zeile 1020 druckt 64 Zeichen der Programmzeile aus. 1030 erhöht den Druckzeilenzähler und 1040 sorgt für den Seitenvorschub, falls die 60. Druckzeile gedruckt wurde.

Die Zeilen 1050 - 1070 bewirken die Weiterverarbeitung bis zum Programmzeilenende.

Allgemeine Dateien mit direktem Zugriff (Random- oder auch Field-Item-Files)

Die Programmierung dieser Dateien ist etwas komplizierter als die von sequentiellen Dateien, bietet aber dafür auch einige entscheidende Vorteile.

Ein Unterschied besteht im Format, in dem die Daten auf der Diskette abgelegt werden. Bei einer sequentiellen Datei werden sie im ASCII-Format und bei Random Dateien in einer komprimierten Form abgespeichert.

Sequentielle Dateien können entweder nur zur Eingabe oder nur zur Ausgabe von Daten geöffnet werden. Bei den Dateien mit direktem Zugriff ist es hingegen möglich, wahlweise Daten zu schreiben oder zu lesen, ohne sie zwischenzeitlich schließen und dann wieder öffnen zu müssen.

Während bei der Suche nach bestimmten Daten in einer sequentiellen Datei immer alle Einträge überprüft werden, sind sie bei einer Random Datei direkt verfügbar. Wenn Daten innerhalb einer Random Datei verändert werden, ist es nicht mehr notwendig, die komplette Datei einzulesen und anschließend auf der Diskette abzulegen. Sie können direkt auf die zu ändernden Einträge zugreifen, diese nach ihrer Modifizierung wieder an die alte Stelle innerhalb der Datei zurückzuschreiben und dabei die alten Einträge zu löschen.

Eine sequentielle Datei wird immer als Ganzes betrachtet und ihr Inhalt (Variablen) ist nur durch Kommata, Carriage Returns, etc. voneinander getrennt. Eine Datei mit direktem Zugriff ist dagegen aus sogenannten Sätzen aufgebaut. Sie stellen die Grundelemente dieser Dateiart dar. Ein Satz ist die Mindestlänge einer Random Datei mit 256 Byte. Er bekommt eine Nummer zugeordnet, die dem Rechner als Kennzeichen dient. Bei Ein- und Ausgaben von Daten muß nur noch die Satznummer angegeben werden und kann der Rechner direkt auf den entsprechenden Teil der Random Datei zugreifen.

Die gewünschten Sätze werden zur Veränderung in einen Puffer des Rechners geladen. Die Pufferlänge ist demnach ebenfalls 256 Byte. Jeder Puffer wird wie bei den sequentiellen Dateien mit einer Puffernummer versehen. Innerhalb der Puffer erfolgt der Aufbau der neuen, auf Diskette zurückzuschreibenden Sätze. Dazu ist ein Puffer in einzelne Felder aufgeteilt, in denen die Variablen abgelegt sind. Diese Aufteilung nennt man auch Fielding. Wenn ein Satz innerhalb einer Random Datei nicht mit der vollen Länge genutzt wird, bleibt der Rest frei. Die Gesamtlänge einer Random Datei ist lediglich durch das zur Verfügung stehende Speichermedium, sprich Diskettespeicherkapazität, begrenzt.

Zusammenfassend kann man also sagen, daß die Einsatzmöglichkeiten von Random Dateien erheblich vielseitiger sind als die der sequentiellen Dateien. Durch den direkten Zugriff auf einzelne Teile der Datei haben Sie eine bessere Kontrolle über das, was gelesen oder abgespeichert werden soll. Nach einiger Zeit werden Sie sehen, daß die Datenspeicherung schneller und sicherer vollzogen wird als bei sequentiellen Dateien. Das soll jedoch keine Abwertung der sequentiellen Dateien sein, denn auch diese haben in der Programmierung ihren festen Platz und bieten ihre Vorteile. Die Kunst des Programmierers ist es nur, den richtigen Dateityp zum richtigen Zeitpunkt einzusetzen.

SYNTAX: OPEN "Modus", #Puffernummer, "Dateiname.Kennwort"

WIRKUNG: Um eine neue Random Datei anzulegen oder eine vorhandene zu verändern oder zu lesen, muß sie mit Hilfe des OPEN-Kommandos geöffnet werden. Über den "Modus" teilen Sie dem Rechner mit, in welcher Form der Zugriff auf die Datei erfolgen soll.

"R" (RANDOM) als "Modus zeigt dem Rechner an, daß eine Ein- oder Ausgabe mit den Befehlen GET bzw. PUT auf eine Random Datei erfolgen soll.

"D" Dateiinterner, direkter Zugriff auf eine bereits existierende Datei, ohne Erweiterung.

Die Ausdrücke "#Puffernummer" und "Dateiname.Kennwort" haben die gleiche Bedeutung wie bei den sequentiellen Dateien.

BEISPIELE: OPEN "R", 2, "RANDOM/DAT:1"

Die Random-Datei RANDOM/DAT in Laufwerk 1 wird geöffnet und mit der Puffernummer 2 versehen. Ist der Eintrag RANDOM/DAT noch nicht im Inhaltsverzeichnis der Diskette, wird er dort eingetragen und die Datei neu eröffnet. Sie steht anschließend sowohl zur Eingabe als auch zur Ausgabe von Daten zur Verfügung.

OPEN "R", 5, "TESTDAT/DAT.KENNWORT"

Auf den Disketten in den angeschlossenen Laufwerken wird nach der Datei mit dem Namen TESTDAT/DAT und dem Kennwort "KENNWORT" gesucht. Die erste, diesen Angaben entsprechende gefundene Datei wird geöffnet und der Puffernummer 5 zugewiesen. Findet der Rechner keinen derartigen Eintrag in den Inhaltsverzeichnissen der Disketten, wird eine neue Datei unter diesem Namen auf der Diskette in Laufwerk 0 eröffnet. Bei späteren Zugriffen auf diese Datei muß immer das Kennwort "KENNWORT" mit angegeben werden.

OPEN "D", 1, "RANDOM/DAT:1"

Eine bereits existierende Datei RANDOM/DAT auf der Disk in Laufwerk 1 wird geöffnet und der Puffernummer 1 zugewiesen. Sie kann durch das OPEN "D" nicht erweitert werden. Es ist jedoch möglich, einen vorhandenen Satz zu editieren oder zu überschreiben. Bei fehlendem Eintrag im Inhaltsverzeichnis kommt die Meldung "Datei nicht gefunden" und es wird keine Datei unter diesem Namen auf der Diskette in Laufwerk 1 installiert.

SYNTAX: FIELD,Puffernummer,Blocklänge AS Stringvariablel,...
=====

WIRKUNG: Dieser Befehl wird dazu benutzt, um den Random-Puffer in einzelne Segmente aufzuteilen, denen wiederum verschiedenen Daten (Variablen) zugewiesen werden. Innerhalb einer FIELD-Anweisung können mehrere Variablen definiert werden.

Der Aufbau der FIELD-Anweisung ist vom Inhalt des Satzes abhängig, der abgespeichert werden soll. Die Blocklängen geben die Anzahl der Bytes an, die für die Stringvariablen auf der Diskette und im Puffer zur Verfügung stehen. Zwischen "Blocklänge", "AS" und "Stringvariable" sollte jeweils ein Leerzeichen stehen. Als Blocklänge kann auch ein numerischer Ausdruck eingesetzt werden, der jedoch in jedem Falle in Klammern stehen muß, um eine fehlerhafte Interpretation der FIELD-Anweisung zu verhindern.

BEMERKUNG: Innerhalb einer Datei ist es auch möglich, mit unterschiedlichen Feldtypen zu arbeiten. Dazu ist der Puffer lediglich durch ein weiteres FIELD-Statement neu aufzuteilen.

Es ist sinnvoll, speziell wenn mehrere Feldaufteilungen innerhalb eines Programms durchgeführt werden müssen, die dazu notwendigen FIELD-Statements in Unterprogrammen abzulegen, die dann bei Bedarf vom Hauptprogramm abgerufen werden.

Weiterhin ist zu beachten, daß Puffer- und Satzlänge zwar 256 Byte betragen, aber vom BASIC aus nur Zeichenketten bis 255 Byte zugelassen werden.

Sämtliche Variablen werden beim Abspeichern in eine Random-Datei wie Zeichenketten behandelt.

Zahlen benötigen dabei als Integerzahl 2 Byte, als Gleitkommazahl mit einfacher Genauigkeit 4 Byte und als Gleitkommazahl mit doppelter Genauigkeit 8 Byte Speicherplatz. Die Blöcke müssen daher innerhalb der FIELD-Anweisung diesen Längen entsprechend definiert werden.

BEISPIEL: Eine Adresse soll als Random-Datei auf der Diskette angelegt werden.

Der Name sei unter N\$, die Straße unter S\$, die Postleitzahl unter P% und der Ort unter O\$ abgelegt. Bei der Postleitzahl handelt es sich um eine Integerzahl. Für Name und Straße und Ort legen wir eine Blocklänge von 20 Byte (20 Zeichen) fest. Die Postleitzahl als Integerzahl benötigt 2 Byte. Dementsprechend erscheinen also innerhalb der FIELD-Anweisung die Angaben

```
FIELD 1,20 AS N$,20 AS S$,2 AS P%,20 AS O$
```

die von den 256 Byte des Puffers nur 62 belegen und den Rest "verschenken".

Da wir jedoch nur sehr ungern Speicherplatz verschenken und Disketten auch heute noch nicht ganz billig sind, müssen wir uns eines kleinen Tricks bedienen, um (fast) den gesamten Speicherplatz eines Satzes ausnutzen zu können.
Dies geschieht unter Zuhilfenahme von Untersätzen.

Bleiben wir bei dem Beispiel mit der abzuspeichernden Adresse, die nur 62 von 256 Byte eines Satzes belegt und somit 194 Byte verschenkt.

Da nur 64 Byte benutzt werden, hätten im Prinzip noch 3 weitere Adressen mit jeweils 64 Zeichen innerhalb des Speichers Platz.

Viermal 62 Byte ergeben 248 Byte, womit der Puffer eines Satzes bis auf einen Rest von 8 Byte komplett genutzt wäre.

Um es nun zu ermöglichen, diese vier Adressen innerhalb eines Satzes abzulegen, wird ein physikalischer Satz (256 Byte) in 4 Untersätze (logische Sätze) aufgeteilt.

Diese Untersätze erhalten die Nummern 0 bis 3, wobei am Anfang des ersten physikalischen Satzes der erste Untersatz (also die Nummer 0) zu finden ist.

Zu Beginn des zweiten physikalischen Satzes steht demzufolge der fünfte Untersatz.

Es muß nun nur noch die Nummer eines logischen Satzes bekannt sein, um die Nummer des physikalischen Satzes und die Position des darin enthaltenen Untersatzes zu berechnen.

PSN = physikalische Satznummer

LSN = logische Satznummer

ALS = Anzahl der logischen Sätze pro physik. Satz

USN = Untersatznummer

$$PSN = INT ((LSN - 1) / ALS) + 1$$
$$USN = LSN - ALS * ((LSN - 1) / ALS) - 1$$

Um nun den ersten physikalischen Satz in seine 4 Untersätze aufzuteilen, muß natürlich jedesmal erneut eine spezielle FIELD-Anweisung durchlaufen werden, wenn auf einen bestimmten Untersatz zugegriffen werden soll.

In unserem Beispiel sehen die 4 FIELD-Anweisungen dazu folgendermaßen aus:

```
FIELD 1,0*62ASDV$,20ASN$,20ASS$,2ASP$,20ASO$
```

```
FIELD 1,1*62ASDV$,20ASN$,20ASS$,2ASP$,20ASO$
```

```
FIELD 1,2*62ASDV$,20ASN$,20ASS$,2ASP$,20ASO$
```

```
FIELD 1,3*62ASDV$,20ASN$,20ASS$,2ASP$,20ASO$
```

oder allgemein für den Untersatz mit der Nummer USN:

```
FIELD 1,USN*62ASDV$,20ASN$,20ASS$,2ASP$,20ASO$
```

Mit Hilfe dieser FIELD-Anweisung können die Untersätze gefunden werden. Der Trick beruht auf der Verwendung einer Dummy-Variablen (DV\$), in die jeweils der Teil des physikalischen Satzes eingelesen wird, der nicht zum gewünschten logischen Satz gehört.

SYNTAX: LSET Puffer-Stringvariable = Stringvariable

=====

WIRKUNG: Dieses Kommando wird dazu benötigt, um einer Puffer Stringvariablen eine Zeichenkette zuzuweisen, die linksbündig ins Feld übernommen wird. D.h. die Zeichenkette beginnt links und füllt das Feld nach rechts hin auf.

ANWENDUNG: Mit diesem Statement ist die Möglichkeit gegeben, den Puffer-Stringvariablen eines Feldes Daten zuzuweisen.

BEMERKUNG: Wenn eine Kette länger als das Feld ist, dem sie zugewiesen wird, gehen die überzähligen Zeichen auf der rechten Seite der Stringvariablen verloren. Sollte die Kette kürzer als die Angabe innerhalb des FIELD-Kommandos sein, werden die fehlenden Stellen durch Leerzeichen aufgefüllt.

HINWEIS: Durch das Statement LSET wird zwar der folgenden Puffer-Stringvariablen ein neuer Inhalt zugeteilt, aber die im FIELD-Kommando zugewiesene Länge bleibt unverändert.

BEISPIEL: siehe RSET (entsprechend linksbündig)

SYNTAX: RSET Puffer-Stringvariable = Stringvariable

=====

WIRKUNG: Dieses Kommando wird dazu benötigt, um einer Puffer Stringvariablen eine Zeichenkette zuzuweisen, die rechtsbündig ins Feld übernommen wird. D.h. die Zeichenkette beginnt rechts und füllt das Feld nach links hin auf.

ANWENDUNG: siehe LSET

BEMERKUNG: Ähnl. wie bei LSET gehen überzählige Zeichen auf der linken Seite verloren und fehlende Stellen werden durch Leerzeichen aufgefüllt.

BEISPIEL: Innerhalb einer FIELD-Anweisung wird der Variablen NA\$ z.B. eine Länge von 10 Byte zugeteilt. Nach einem RSET NA\$="Meier" hätte die Puffer-Stringvariable NA\$ den Inhalt ".....Meier".

SYNTAX: MKI\$ (Integervariable oder -zahl)

=====

WIRKUNG: Diese Funktion wandelt eine Integerzahl oder -variable in eine 2-Byte lange Zeichenkette um.

ANWENDUNG: Da Zahlen innerhalb einer Random-Datei nur als String abgespeichert werden können, ist die MKI-Funktion notwendig, um Integerzahlen oder -variablen den Puffer-Stringvariablen zuordnen zu können.

HINWEIS: Die Integerzahl muß im Bereich -32768 bis +32767 liegen.

BEISPIEL: A\$=MKI\$(A%)

Die Integervariable A% wird in einen 2-Byte langen String A\$ umgewandelt. Um A\$ in den Dateipuffer zu übertragen muß LSET oder RSET verwendet werden.

BEMERKUNG: Wenn Sie nach Ausführung des obigen Beispiels ein PRINT A\$ eingeben, werden Sie feststellen, daß die Integervariable A% nicht mehr erscheint. Dies liegt an der komprimierten Form, in der sie nun als Zeichenkette A\$ vorliegt.

SYNTAX: MKS\$ (Variable oder Zahl mit einfacher Genauigkeit)

WIRKUNG: Diese Funktion wandelt eine Variable oder Zahl mit einfacher Genauigkeit in eine 4-Byte lange Zeichenkette um.

ANWENDUNG: Da Zahlen innerhalb einer Random-Datei nur als String abgespeichert werden können, ist die MKS-Funktion notwendig, um Zahlen oder Variablen mit einfacher Genauigkeit den Puffer-Stringvariablen zuordnen zu können.

BEISPIEL: A\$=MKS\$(A!)

Die Variable A! (mit einfacher Genauigkeit) wird in einen 4-Byte langen String A\$ umgewandelt. Um A\$ in den Datenpuffer zu übertragen, muß entweder LSET oder RSET benutzt werden.

BEMERKUNG: Wenn Sie nach Ausführung des obigen Beispiels ein PRINT A\$ eingeben, werden Sie feststellen, daß die Variable A! nicht mehr erscheint. Dies liegt an der komprimierten Form, in der sie nun als Zeichenkette A\$ vorliegt.

SYNTAX: MKD\$ (Variable oder Zahl mit doppelter Genauigkeit)
=====

WIRKUNG: Diese Funktion wandelt eine Variable oder Zahl mit doppelter Genauigkeit in eine 8-Byte lange Zeichenkette um.

ANWENDUNG: Da Zahlen innerhalb einer Random-Datei nur als String abgespeichert werden können, ist die MKD-Funktion notwendig, um Zahlen oder Variablen mit doppelter Genauigkeit den Puffer-Stringvariablen zuordnen zu können.

BEISPIEL: A\$=MKD\$(A#)

Die Variable A# (mit doppelter Genauigkeit) wird in einen 8-Byte langen String A\$ umgewandelt. Um A\$ in den Datenpuffer zu übertragen, muß entweder LSET oder RSET benutzt werden.

BEMERKUNG: Wenn Sie nach Ausführung des obigen Beispiels ein PRINT A\$ eingeben, werden Sie feststellen, daß die Variable A# nicht mehr erscheint. Dies liegt an der komprimierten Form in der sie nun als Zeichenkette A\$ vorliegt.

SYNTAX: CVI (2-Byte-Zeichenkette)

WIRKUNG: Eine 2-Byte lange Zeichenkette aus dem Datenpuffer wird mit dieser Funktion in einen Integerwert umgewandelt.

ANWENDUNG: Da Integerzahlen oder -variablen innerhalb einer Random-Datei, unter Verwendung von MKI, nur als Strings abgespeichert werden können, müssen diese Zeichenketten zur weiteren Verwendung nach ihrem Einlesen wieder in die entsprechenden Integerwerte umgesetzt werden.

HINWEIS: Der Versuch, eine Puffer-Stringvariable, die kürzer als 2 Byte ist, wieder in einen Zahlenwert umzuwandeln, bringt die Fehlermeldung "Unerlaubter Funktionsaufruf".
Wenn die Zeichenkette länger als 2 Byte ist, werden die zusätzlichen Bytes unterschlagen, was zu einem unkorrekten Wert führt.

BEISPIEL: A%=CVI(A\$)

Die 2-Byte lange Puffer-Stringvariable A\$ wird in die Integervariable A% umgewandelt.

SYNTAX: CVS (4-Byte-Zeichenkette)

WIRKUNG: Eine 4-Byte lange Zeichenkette aus dem Datenpuffer wird mit dieser Funktion in einen Zahlenwert mit einfacher Genauigkeit umgewandelt.

ANWENDUNG: Da Zahlen oder Variablen mit einfacher Genauigkeit innerhalb einer Random-Datei, unter Verwendung von MKS, nur als Strings abgespeichert werden können, müssen diese Zeichenketten zur weiteren Verwendung nach ihrem Einlesen wieder in entsprechende Werte mit einfacher Genauigkeit umgesetzt werden.

HINWEIS: Der Versuch, eine Puffer-Stringvariable, die kürzer als 4 Byte ist, wieder in einen Zahlenwert umzuwandeln, bringt die Fehlermeldung "Unerlaubter Funktionsaufruf".
Wenn die Zeichenkette länger als 4 Byte ist, werden die zusätzlichen Bytes unterschlagen, was zu einem unkorrekten Wert führt.

BEISPIEL: A!=CVS(A\$)

Die 4-Byte lange Puffer-Stringvariable A\$ wird in die Variable A! mit einfacher Genauigkeit umgewandelt.

SYNTAX: CVD (8-Byte-Zeichenkette)

WIRKUNG: Eine 8-Byte lange Zeichenkette aus dem Datenpuffer wird mit dieser Funktion in einen Zahlenwert mit doppelter Genauigkeit umgewandelt.

ANWENDUNG: Da Zahlen oder Variablen mit doppelter Genauigkeit innerhalb einer Random-Datei, unter Verwendung von MKD, nur als Strings abgespeichert werden können, müssen diese Zeichenketten zur weiteren Verwendung nach ihrem Einlesen wieder in entsprechende Werte mit doppelter Genauigkeit umgesetzt werden.

HINWEIS: Der Versuch, eine Puffer-Stringvariable, die kürzer als 8 Byte ist, wieder in einen Zahlenwert umzuwandeln, bringt die Fehlermeldung "Unerlaubter Funktionsaufruf".
Wenn die Zeichenkette länger als 8 Byte ist, werden die zusätzlichen Bytes unterschlagen, was zu einem unkorrekten Wert führt.

BEISPIEL: A#=CVD(A\$)

Die 8-Byte lange Puffer-Stringvariable A\$ wird in die Variable A# mit doppelter Genauigkeit umgewandelt.

SYNTAX: PUT Puffernummer, Satznummer

WIRKUNG: Dieses Statement kopiert den Inhalt des Random-Puffers auf die durch die Puffernummer definierte geöffnete Datei auf der Diskette. Der Schreib/Lesekopf wird dabei auf die angegebene Satznummer positioniert.

ANWENDUNG: Der PUT-Befehl dient dazu, die im Random-Puffer vorbereiteten Datensätze auf der Diskette abzuspeichern. Dabei bezeichnet die Puffernummer die Datei, die dazu eröffnet worden ist, die Daten aufzunehmen, und die Satznummer gibt die Position der Datenspeicherung auf der Diskette an. Wenn die Satznummer nicht extra angegeben wird, erfolgt die Ausgabe der Daten auf die nächst höhere Satznummer, (bzw. auf Satz 1 bei der ersten Ausgabe.)

BEMERKUNG: Jeder Satz innerhalb der geöffneten Datei hat eine Länge von 256 Byte. Das PUT-Statement schreibt immer alle 256 Byte auf die Diskette, auch dann, wenn nur ein Teil des Puffers durch die FIELD-Anweisung belegt wurde. Die restlichen Bytes bleiben in diesem Fall frei. Die Anzahl der möglichen Sätze auf einer Diskette hängt von ihrer Speicherkapazität ab.

BEISPIEL: PUT 1,2

Diese Anweisung bedeutet, daß die Daten aus dem Puffer Nummer 1 auf den zweiten Satz einer geöffneten Datei auf der Diskette abgelegt wird.

HINWEIS: Gibt man bei der ersten Ausgabe eines Satzes sofort eine Satznummer größer als 1 an, so werden auch alle Sätze von 1 bis zur angegebenen Nummer auf der Diskette angelegt. Wenn dabei durch eine zu hohe Satznummer der maximalen Speicherplatz der Diskette überschritten wird, kommt es zur Fehlermeldung "Diskette voll".

SYNTAX: GET Puffernummer, Satznummer

=====

WIRKUNG: Die Anweisung überträgt einen angegebenen Satz von einer geöffneten Datei in den zur Datei gehörenden Puffer.

ANWENDUNG: Das Kommando wird dazu benötigt, um einen auf einer Diskette innerhalb einer Random-Datei stehenden Satz wieder in einen durch FIELD aufgeteilten Random-Puffer zu übertragen. Wie bei der PUT-Anweisung ist es auch hier nicht unbedingt notwendig, die Satznummer anzugeben, wenn immer auf die nächst höhere Nummer zugegriffen werden soll. Sollte noch keine Satznummer definiert worden sein, wird zuerst auf Satz 1 zugegriffen.

BEISPIEL: GET 1,4

Diese Anweisung bedeutet, daß aus einer geöffneten Datei mit der Puffernummer 1 der vierte Satz von der Diskette in den Datenpuffer eingelesen wird. Der Datenpuffer muß vorher mit Hilfe der FIELD-Anweisung entsprechend vorbereitet werden.

SYNTAX: CLOSE Puffernummer, Puffernummer,...

WIRKUNG: Alle offenen Dateien, durch die Angabe der Puffernummern definiert, werden geschlossen. Die Variablen des Daten-Puffers sind nicht mehr verfügbar. Sollte die Datei in ihrer Länge verändert worden sein, wird ein entsprechender Eintrag im Inhaltsverzeichnis vorgenommen.

ANWENDUNG: Jede Datei, die durch OPEN geöffnet wurde, muß durch CLOSE wieder geschlossen werden, da sonst Daten verloren gehen können oder das Dateiende nicht mehr im Inhaltsverzeichnis eingetragen wird, was zur Unlesbarkeit einer Diskette führen kann. Schließen Sie also immer alle offenen Dateien, bevor Sie Veränderungen am Programm oder einen Diskettenwechsel vornehmen.

BEMERKUNG: Ein CLOSE ohne Parameterangabe schließt alle offenen Dateien. Ein PUT oder GET auf eine geschlossene Datei ist nicht mehr möglich.

BEISPIELE: siehe CLOSE bei sequentiellen Dateien

SYNTAX: LOF (Puffernummer)

=====

WIRKUNG: Mit dieser Funktion (Length Of File) wird die letzte Satznummer einer geöffneten Datei ermittelt.

ANWENDUNG: Die Anweisung kommt z.B. zum Einsatz, wenn an eine Datei, deren Länge unbekannt ist, noch weitere Sätze angefügt oder alle Sätze eingelesen werden sollen. Daher finden Sie die LOF-Funktion meist nach einem PRINT-Statement oder innerhalb einer FOR-NEXT-Loop.

BEMERKUNG: LOF ist nur dann Null, wenn in eine Datei noch kein Eintrag über die PUT-Anweisung unternommen wurde oder der Eintrag im Inhaltsverzeichnis nicht erfolgen konnte, (z.B. Diskettenwechsel ohne vorheriges CLOSE).

BEISPIEL: 10 OPEN "R",1,"VERSUCH/DAT":FIELD 1,20 AS A\$
20 LSET A\$="Teststring":PUT 1,250
30 PRINT LOF (1)" Sätze hat diese Datei.":CLOSE 1

Das Beispiel wird 250 als Wert für LOF ausgegeben, da in der PUT-Anweisung 250 Sätze erstellt werden.

SYNTAX: EOF (Puffernummer)

=====

WIRKUNG: Da die EOF-Funktion bei Random-Dateien genauso arbeitet wie bei sequentiellen Dateien, soll an dieser Stelle nicht mehr näher darauf eingegangen werden.

BEMERKUNG: Wenn eine geöffnete Random-Datei einen LOF-Wert von Null ergibt, ist EOF gleich -1. Das bedeutet gleichzeitig, daß das Ende der Datei angetroffen ist. Bei LOF-Werten ungleich Null ist EOF immer Null und zeigt damit an, daß das Dateiende noch nicht erreicht ist.

SYNTAX: LOC (Puffernummer)

=====

WIRKUNG: An dieser Stelle soll nur die älteste Art der LOC-Funktion erklärt werden. Weitere Angaben finden Sie im Abschnitt über besondere Dateistrukturen. Über die LOC-Funktion können Sie die Nummer des zuletzt mit GET oder PUT bearbeiteten Satzes einer geöffneten Random-Datei ermitteln. Die dabei ausgegebene Zahl kann Integer-Werte von 1 bis 32767 annehmen.

ANWENDUNG: Oft ist es bei der Erstellung von Dateien sinnvoll, die zuletzt bearbeitete Satznummer abrufen zu können.

BEISPIEL:

```
10 OPEN "R", 2, "VERSUCH/DAT"
20 FIELD 2, 20 AS A$
30 I=I+1
40 B$=STR$(I)+". Satz"
50 LSET A$=B$
60 PUT 2
70 IF LOC(2) < 20 GOTO 30 ELSE GET 2, 10
80 PRINT A$
90 PRINT "Satznummer :"; LOC(2)
100 CLOSE 2
```

Dieses Programm schreibt zunächst unter Verwendung der LOC-Funktion in Zeile 70 insgesamt 20 Sätze in die Datei "VERSUCH/DAT". Wenn LOC (2) gleich 20 ist, fährt das Programm fort und liest die 10. Eintragung.

Es erfolgt die Ausgabe des gelesenen Satzes und seiner Satznummer, (ebenfalls durch LOC ermittelt), die die durch GET veränderte Position des Schreib/Lesekopfes anzeigt.

Da sich die innerhalb der vorigen Kapitel erklärten sequentiellen Dateien (PRINT/INPUT-Dateien) und Dateien mit direktem Zugriff nach traditioneller Art (GET/PUT mit Fielding) mit einer festen Satzlänge von höchstens 256 Byte oft als zu starr und unflexibel erwiesen haben, wurden im G-DOS die Zugriffsmöglichkeiten auf Random-Dateien erweitert. Auch diese Dateien werden mit Hilfe des OPEN-Statements für die Zugriffe geöffnet. Am Ende der OPEN-Zeile erfolgen jedoch noch Angaben zur Dateizugriffsform und Satzlänge.

Dateistrukturen:

Wir unterscheiden 4 verschiedene Dateistrukturen:

1. Sequentielle Dateien (PRINT/INPUT-Files)
2. einfache Random-Dateien (Field-Item-Files)
3. Random-Dateien als Fixed-Item-Files
4. Random-Dateien als Marked-Item-Files

Unterschiede:

Während alle Dateien, die Sie über die Statements PRINT und INPUT erstellen (OPEN"I" u. OPEN"O") ausschließlich sequentiell nutzen können, lassen sich die verschiedenen Filetypen in der Random-Form sowohl sequentiell als auch in beliebiger Reihenfolge sowohl lesend, als auch schreibend bearbeiten. Sequentielle Dateien liegen auf der Diskette im ASCII-Format vor und benötigen aus diesem Grunde sehr viel Speichermedium. Bei allen Dateien mit direktem Zugriff werden die Daten vor der Speicherung komprimiert und die dabei erzeugten binären Werte auf der Diskette abgelegt.

Die bei einer sequentiellen Speicherung erforderliche und recht lästige Trennung von Zeichenketten durch das besondere Trennungszeichen ";"; ist bei Random-Dateien nicht notwendig, da dies bereits durch das FIELD-Statement oder durch Längenangaben vollzogen wird.

Im Gegensatz zu den PRINT/INPUT-, Fixed- und Marked Item-Dateien müssen Sie die Daten von Field-Item-Dateien vor der Speicherung über LSET oder RSET in den Puffer schreiben und Zahlen (numerische Variablen) zuvor über MKI\$, MKS\$ und MKD\$ in Strings umwandeln. Diese in Strings umgewandelten Zahlen sind nach dem Lesen aus einer Field-Datei mit Hilfe der Befehle CVI, CVS und CVD wieder in ihre entsprechenden numerischen Werte umzuwandeln.

Die Länge eines Satzes ist bei der sequentiellen Speicherung von Daten völlig beliebig. Field-Item-Files können dagegen nur mit Satzlängen von max. 256 Byte und Fixed- sowie Marked-Item-Dateien mit max. 4095 Byte beschrieben werden.

Bei der sequentiellen Datenspeicherung wird der Inhalt des Puffers unverändert auf die Diskette übertragen. Beim Einlesen der Daten werden lediglich evtl. vorlaufende Leerzeichen ignoriert und überlesen.

Bei den Field-Item-Files kommt es durch die Befehle LSET und RSET zu einer links- bzw. rechtsbündigen Speicherung der Zeichenketten in die jeweiligen Sätze. Die Speicherung der Zeichenketten innerhalb von Marked-Item-Dateien erfolgt dagegen immer linksbündig. Sollte dabei rechts noch freier Speicherplatz vorhanden sein, wird dieser durch Leerzeichen aufgefüllt, wobei die Strings exakt aus dem Puffer auf die Diskette übernommen werden. Sollte eine Zeichenkette jedoch die maximal vorgegebene Länge überschreiten, so wird sie einfach abgeschnitten und bekommt somit einen fehlerhaften Inhalt.

IGEL:

Im folgenden wird der Ausdruck IGEL immer wieder benutzt. Ein IGEL ist eigentlich die Abkürzung für "Item Group Expression List" und heißt soviel wie Bestandteilliste.

In diesem IGEL sind die Variablen aufgezählt, die in einen Satz der Datei geschrieben oder aus ihm gelesen werden sollen. Die genaue Beschreibung befindet sich unter den Befehlen GET und PUT.

RBA:

RBA ist die Abkürzung für "Relativ Byte Adress". Es ist eine Zahl zwischen 0 und 16777199 und bezeichnet die Position eines Bytes innerhalb einer Datei. RBA-Werte werden zur Positionierung innerhalb einer Datei benutzt oder geben das Dateiende an.

Die verschiedenen Dateitypen:

Fixed-Item-Files:

In einem Fixed-Item-File wird der Aufbau der Daten nicht in der Datei selbst, sondern im Programm festgelegt. Die Bestandteile der Datei sind also fest und der Zugriff kann nur erfolgen, wenn der Aufbau der Daten bekannt ist. Ist die Datei in Sätze aufgebaut, so sind alle Sätze gleich lang, wobei die maximale Satzlänge 4095 Bytes beträgt und im OPEN-Befehl festgelegt wird.

Die Bestandteile der Datei liegen ohne Trennzeichen dicht beieinander. Die Anzahl der Satzteile und ihre Art werden innerhalb von Variablenlisten beim GET- und PUT-Statement festgelegt. Ein Zugriff auf die Sätze eines Fixed-Item-Files erfolgt nach dem OPEN mit dem Typ "FI" oder "FF". Dabei werden mittels eines GET-Statements die Daten direkt in die Variablen eines IGELs gelesen oder über PUT die Variablen auf der Diskette abgelegt.

Fixed-Item-Dateien müssen wie alle anderen Dateiarten über den Befehl CLOSE geschlossen werden.

Marked-Item-Files:

Alle Sätze und Satzteile innerhalb eines Marked-Item-Files beginnen mit einem oder mehreren Steuerzeichen, die zur Markierung dienen.

Diese Steuerzeichen erzeugt der Rechner während der Abspeicherung der Daten selbständig. Der Platz für die Steuerzeichen muß jedoch bei der Ermittlung der Längen mit berücksichtigt werden. Die Sätze eines Marked-Item-Files können entweder alle gleichlang sein oder auch alle eine unterschiedliche Länge aufweisen. Die maximale Länge ist jedoch auf 4095 Bytes festgelegt, wobei darin auch sämtliche Steuer und Markierungszeichen enthalten sind. Bei Sätzen mit unterschiedlicher Länge dürfen beliebig viele Satzteile auch mit unterschiedlicher Art enthalten sein. Wenn eine feste Satzlänge vorgegeben wird, dürfen in einem Satz nicht mehr Daten einschließlich aller Markierungen stehen, als dessen Länge erlaubt.

Alle Bestandteile einer Marked-Item-Datei liegen wie bei Fixed-Item-Dateien dicht zusammen, sodaß kein ungenutzter Speicher-raum auf der Diskette verschwendet wird. Allerdings wird zusätzlicher Platz durch die Markierungsbytes benötigt.

Ein Zugriff auf die Sätze eines Marked-Item-Files kann mit Hilfe der Filetypen "MU", "MI" und "MF" erfolgen, wobei wie bei den Fixed-Item-Files auch hier die Daten direkt den Variablen zugewiesen bzw. Variablen sofort auf die Diskette geschrieben werden können. Nach Beendigung des Zugriffes auf die Datei ist sie mit CLOSE zu schließen.

Der benötigte Speicherplatz:

Bei der Berechnung der Satzlänge sind für die Variablentypen folgende Werte zu berücksichtigen:

| Variablentyp | Fixed-Item | Marked-Item |
|--------------|--------------|---|
| Integer | 2 | 2+1=3 |
| Real | 4 | 4+1=5 |
| Double | 8 | 8+1=9 |
| String | dessen Länge | bei Strings bis 127 Zeichen: Länge + 1 bei über 127 Zeichen: Länge + 2 |

Die Länge eines Strings wird bei Fixed-Item-Files immer im IGEL angegeben, bei Marked-Item-Files entweder im IGEL oder es wird die tatsächliche Stringlänge angenommen.

Bei "MU"-Dateien, also Dateien mit variabler Satzlänge, ist zusätzlich noch 1 Byte als Trennzeichen zwischen den einzelnen Sätzen reserviert.

Die im OPEN angegebene Satzlänge ist bei "MU"-Files das erlaubte Maximum. Kürzere Sätze sind möglich und belegen entsprechend weniger Speicherplatz. "MU"-Dateien belegen genausoviel Speicherplatz wie für Daten und Markierungsbytes notwendig sind.

Dateien mit fester Satzlänge belegen für jeden gespeicherten Datensatz gleichviel Speicherplatz. Der für die gesamte Datei benötigte Speicherplatz beträgt also Anzahl der Sätze mal Satzlänge.

Beachten Sie aber, daß G-DOS den Platz auf den Disketten in Einheiten von 5 mal 256 Byte verwaltet. Eine Datei mit nur einem Byte Inhalt belegt also auf der Diskette insgesamt 1280 Bytes!

"MI"- und "FI"-Files haben überhaupt keine einzelnen Sätze. Sie bilden einen einzigen Satz, der solange ist, wie die gesamte Datei.

SYNTAX: OPEN"Modus",#Puffer-Nr., "Dateiname", "Typ", Satzlänge
=====

WIRKUNG: Um mit den Random-Dateien in den verschiedensten Arten arbeiten zu können, sind den im vorigen Kapitel bereits beschriebenen Befehlen noch weitere Angaben hinzuzufügen.

Wie bei den traditionellen Random-Dateien mit einer Satzlänge von 256 Bytes muß auch bei den besonderen Dateistrukturen die Datei zunächst einmal geöffnet werden, damit irgend ein Zugriff über GET oder PUT überhaupt erst möglich wird.

"Modus", #Puffer-Nr. und "Dateiname" haben die gleiche Bedeutung wie im Kapitel bereits erklärt wurde.

Neu hinzugekommen ist jedoch die Angabe des Dateityps, mit der Sie dem Rechner mitteilen, in welcher der 5 möglichen Arten die Random-Datei bearbeitet werden soll.

Bei der Angabe "Typ" handelt es sich um einen String oder einer Stringvariablen mit dem Inhalt "FF", "FI", "MF", "MI" oder "MU".

Wenn der "Typ" bei der Eröffnung einer Random-Datei angegeben wird, darf im Anschluß kein Fielding durchgeführt werden. Dafür müssen jedoch alle folgenden GET- und PUT-Statements eine Variablenliste (IGEL) oder eine Zeilennummer, bei der eine Variablenliste beginnt, enthalten.

Eine weitere neue Angabe innerhalb der OPEN-Anweisung ist die Satzlänge. Sie muß z.B. angegeben werden, wenn zwei "Typ"-Anweisungen in einem OPEN-Statement vorkommen. Die Satzlänge ist immer eine Zahl zwischen 1 und 4095.

Wenn man die Datei-Typen "FF" und "MF" verwendet, so wird mit Angabe der Satzlänge gleichzeitig die Länge aller Sätze innerhalb der geöffneten Datei definiert.

Bei "MU"-Files ist hier die maximalen Länge eines Satzes anzugeben. Da sie standardmäßig auf 4095 festgelegt ist, muß bei "MU"-Files nicht unbedingt eine Längenangabe erfolgen.

Wenn man den "MF" oder "MU"-Typ benutzen, so ist darauf zu achten, daß bei der Längenangabe der Sätze auch die für die Markierung notwendigen Bytes berücksichtigt werden.

Die fünf verschiedenen Filetypen:

1. "MU"-Files (markiert unformatiert)

"MU"-Files bestehen aus Sätzen mit markierten Bestandteilen. Die Satzteile können jedoch unterschiedliche Längen besitzen. Die maximalen Längen sind innerhalb der OPEN-Zeile anzugeben. Standardmäßig ist die Länge auf 4095 definiert. Eine Längenangabe ist also nicht unbedingt notwendig.

Von allen 5 File-Typen dürfte dieser wohl am einfachsten zu benutzen sein, da er die PRINT/INPUT-Files ersetzen soll.

Bei diesem File mit variabler Satzlänge kann jeder Satz verändert oder mit einem anderen Satz ausgetauscht werden. Dabei ist jedoch darauf zu achten, daß der auszutauschende Satz gleichlang oder kürzer als der alte ist. Wenn er kürzer ist, wird der neue Satz linksbündig an die Stelle des alten geschrieben, wobei die restlichen Stellen bis zum nächsten Startbyte eines Satzes mit Nullen aufgefüllt werden.

Der Vorteil des MU-Files liegt darin, daß keine Trennzeichen zwischen den Variablen stehen müssen, (wie beispielsweise bei den PRINT/INPUT-Files), außerdem ist es nicht notwendig Variablen immer in Zeichenketten umzuwandeln, wie es im Kapitel 7.2 erklärt wurde.

Auch beim Lesen der Daten werden Zahlen wieder unmittelbar den entsprechenden Variablen zugewiesen, (kein CVI, CVS und CVD notwendig).

Aufgrund der variablen Satzlengthen bei diesem Filetyp ist es notwendig, jeden einzelnen Bestandteil der Datei zu markieren, damit sich der Rechner innerhalb der Datei zurecht findet. Natürlich benötigen auch diese Markierungsbytes auf der Diskette ihren Platz und dürfen daher bei der Längenbestimmung der Sätze nicht außer acht gelassen werden.

MU-Files können über OPEN"O", OPEN"I", OPEN"E" und OPEN"R" geöffnet werden.

Durch ein OPEN"O" läßt sich die Datei erzeugen und mit Hilfe von PUT-Anweisungen beschreiben. Es ist dabei gleich ob die Datei vorher schon existiert oder nicht.

Eine weitere Möglichkeit, eine MU-Datei zu bearbeiten, liefert das OPEN"E"-Statement (Extend), mit dessen Hilfe die Datei sequentiell erweitert werden kann. Für den Fall, daß die Datei dabei noch nicht existiert, wird sie neu angelegt. Sequentielles Lesen einer MU-Datei ist mit OPEN"I" möglich. Soll ein MU-File randommäßig mit OPEN"D" eröffnet werden, muß sie vorher auf der Diskette existieren.

2. "MF"-Files (markiert formatiert)

Diese Files bestehen aus Gruppen von Datenbytes (Sätze) und enthalten markierte Bestandteile. Alle Sätze innerhalb eines MF-Files haben die gleiche Länge. Diese muß innerhalb der OPEN-Zeile definiert werden (Satzlänge).

Wie der MU-File läßt sich auch der MF-File verändern.

Werden neue Daten in ein MF-File geschrieben, so müssen diese nicht unbedingt vom gleichen Typ wie die alten sein, ebenso hat eine Veränderung der Länge der neuen Datengruppen keinerlei Auswirkungen. Es ist darauf zu achten, daß im OPEN-Statement die exakte Satzlänge angegeben wird. Bei der Bearbeitung der Datei mit GET oder PUT überprüft der Rechner anhand der IGETL die Satzlangenangabe und zeigt bei Überschreitung einen Fehler an.

Da die Datei in gleich lange Sätze aufgeteilt ist, kann die für die Ein/Ausgabe in den Anweisungen PUT und GET einzusetzende Satznummer leicht berechnet werden. G-DOS ist dazu zwar selbst in der Lage, aber zum besseren Verständnis soll die Formel an dieser Stelle einmal erläutert werden.

Die Multiplikation der Byteanzahl je Satz mit der um eins verringerten Nummer des Satzes ergibt die relative Byte Adresse (RBA) des Satzes. Die 1 wird hier subtrahiert, weil der erste Satz zwar die Nummer 1 trägt, aber die niedrigste Byteadresse Null ist. In einer Datei mit Sätzen von jeweils 10 Bytes Länge liegt der erste Satz bei der RBA $(1-1)*10 = 0$, der zweite Satz fängt dementsprechend bei $(2-1)*10 = 10$ und der 3. Satz bei der RBA = 20 an, da $(3-1)*10=20$.

Sie sollten jedoch beim Arbeiten mit der RBA sehr vorsichtig sein, denn wenn der angegebene Wert falsch ist, kann es Ihnen wiederfahren, daß ein Eintrag an einer falschen Position vorgenommen und damit Daten zerstört werden.

3. "MI"-Files (markierter Inhalt)

Ein "MI"-File enthält markierte Bestandteile und wird nicht in Sätze mit fest definierten Längen aufgeteilt. Er enthält Datengruppen, deren Länge der Benutzer bestimmen muß. Die Form eines "MI"-Files kann später nicht mehr verändert werden. Schon aus diesem Grunde ist der "MI"-File in seiner Anwendung sehr beschränkt einsatzfähig.

Die Positionierung muß immer über die Angabe der RBA erfolgen, da innerhalb der Datei keine Start- oder Füllbytes anzutreffen sind. Alle Daten hängen also unmittelbar aneinander. Um mit "MI"-Files arbeiten zu können ist es also notwendig, die genaue Position der Markierungsbytes zu kennen.

4. "FF"-Files (frei formatiert)

Diese Files bestehen aus festen Sätzen, die wiederum in feste Bestandteile aufgeteilt sind. Alle Sätze innerhalb eines "FF"-Files haben also die gleiche Länge. Diese muß innerhalb der OPEN-Zeile definiert werden.

Beim "FF"-File finden wir keine Markierungsbytes. Zuständig für den Aufbau eines Satzes ist ausschließlich die IGEL. Sie haben also darauf zu achten, daß die Daten eines "FF"-Files nicht falsch zugeordnet werden. Es kann durchaus passieren, daß der Rechner einer 10-Byte langen Stringvariablen ihre 10 Bytes zuweist, obwohl es sich dabei z.B. um zuvor abgelegte numerische Variablen handelt. Es ist also immer darauf zu achten, daß beim erneuten Zugriff auf eine "FF"-Datei die IGEL in ihrem Format mit der ursprünglichen übereinstimmt.

Weiterhin kann mittels einer "FF"-Datei jedes einzelne Byte eines Satzes verändert und nur dieses allein an die richtige Position zurückgeschrieben werden ohne die anderen Daten des Satzes zu berühren.

Bei der Verwendung von "FF"-Files dürfen keine Ausdrücke oder Formeln als Variablen verwendet werden. Bei Zeichenketten ist die Länge in Klammern vorweg anzugeben.

Verwendung:

Beispielsweise Lesen einer Datei, die mit FIELD-Statements erzeugt wurde oder Bearbeitung einer beliebigen Datei (z.B. ein CMD-Programm).

5. "FI"-Files (freier Inhalt)

Dieser Filetyp enthält feste Bestandteile, ist jedoch nicht in Sätze mit fest definierten Längen aufgeteilt, ähnlich wie ein "MI"-File. Die Daten werden quasi in Form einer vom Benutzer in ihrer Länge zu bestimmenden Zeichenkette auf der Diskette abgelegt. Der "FI"-File enthält jedoch keine Markierungsbytes, und seine Form kann später auch wieder verändert werden.

Daraus geht schon hervor, daß sein Einsatzbereich etwas flexibler ist als der eines "MI"-Files. Wie beim "MI"-File muß auch hier die Positionierung immer durch den Benutzer über die Angabe der RBA erfolgen, da alle Daten ohne Start- und Füllbytes in der Datei wie eine große Zeichenkette aneinandergereiht werden.

SYNTAX: GET Parameterliste

bzw.

PUT Parameterliste

=====

WIRKUNG: Das GET-Kommando dient dazu, einen bestimmten Satz oder Satzteil einer durch OPEN geöffneten Datei in den zur Datei gehörenden Puffer zu übertragen. PUT arbeitet umgekehrt und schreibt Daten aus dem Puffer in eine geöffnete Datei. Dieser Puffer ist in einzelne Segmente aufgeteilt. (Im vorigen Abschnitt lernten Sie bereits einmal das GET- und PUT-Kommando kennen, wobei die Aufteilung des Puffers über das FIELD-Statement vollzogen wurde.)

ANWENDUNG: Die Parameterliste kann folgende Angaben enthalten:

1. GET Puffernummer
2. GET Puffernummer, Positionierung
3. GET Puffernummer, Positionierung, , IGEL
4. GET Puffernummer, Positionierung, IGEL-Zeilennr.

bzw.

1. PUT Puffernummer
2. PUT Puffernummer, Positionierung
3. PUT Puffernummer, Positionierung, , IGEL
4. PUT Puffernummer, Positionierung, IGEL-Zeilennr.

Da bei den 5 verschiedenen Dateitypen, die innerhalb dieses Kapitels erläutert werden, kein FIELD-Statement erforderlich ist, muß es also möglich sein, auch mit anderen Mitteln die Daten eines Files in eine entsprechende Variable zu schreiben oder den Inhalt einer Variablen an der entsprechenden Position auf der Diskette abzulegen.

Die Aufteilung eines Files bzw. eines Puffers erfolgt direkt über die beiden Statements GET und PUT. Wenn Sie die obigen vier Möglichkeiten der GET- und PUT-Anweisung mit den bisherigen Statements vergleichen, werden Sie feststellen, daß noch einige Parameter hinzugekommen sind.

Bevor wir näher auf den neuen Parameter "Positionierung" eingehen, müssen noch zwei neue Begriffe eingeführt werden:

1. letzte Satzadresse
2. letzte Byteadresse

Diese beiden Werte, relative Byte-Adressen, sind zur Durchführung einer Positionierung sehr wichtig. Nach einem Zugriff auf eine Datei über GET oder PUT merkt sich der Rechner diese Werte.

Die letzte Byteadresse gibt die relative Position eines bestimmten Files an.
Die letzte Satzadresse gibt die relative Position des ersten Bytes eines Satzes innerhalb eines Files an.
Beide Werte gehen durch ein erneutes OPEN verloren.

Bei PRINT/INPUT-Files wird automatisch die letzte Satzadresse auf den Satzanfang gesetzt. Eine letzte Byteadresse ist bei PRINT/INPUT-Files nicht vorhanden.

Über die Anweisungen zur Positionierung innerhalb der Statements GET und PUT erfolgen auch Veränderungen der letzten Byte- und Satzadresse.

Die letzte Satzadresse wird benötigt, wenn man in einer Datei über die Angabe "#" (siehe dort) positioniert. Die letzte Byteadresse ist dagegen zur File-Positionierung mit "\$" (siehe dort) notwendig.

Positionierung:

Die Angabe der Position erfolgt sowohl bei GET als auch bei PUT und hat in beiden Fällen die gleiche Bedeutung.

Sie teilen dem Rechner mit Hilfe dieses Ausdrucks mit, an welcher Stelle der geöffneten Datei eine Veränderung der Daten vollzogen werden soll.

Diese Positionierung mag zwar etwas umständlich sein, hat jedoch großen Vorteil, daß auch Dateien mit einem komplizierteren Aufbau geschaffen werden können.

Bei den Dateitypen mit variablen Satzlengthen müssen Sie zur Positionierung wissen, wo sich welcher Satz innerhalb des Files befindet. D.h. die Kenntnis der Relativen Byte Adresse (RBA) ist Voraussetzung zur Positionierung. Wie wir bereits wissen ist die RBA nichts anderes als die Nummer des Bytes der Datei, bei dem eine Änderung des Inhalts beginnen soll. Das erste Byte einer Datei hat dabei stets die Nummer Null.

Auf diesem Byte Null befinden wir uns, wenn eine Datei z.B. über

```
OPEN "R",1,"TESTDAT","FI"
```

geöffnet wurde.

Gibt man nun keinerlei nähere Anweisungen zur Positionierung innerhalb der folgenden GET- oder PUT-Kommandos, wird automatisch der erste Satz (also beginnend bei der RBA=Null) bearbeitet.

Dies ist bei allen Formen der OPEN-Anweisung der Fall. Lediglich ein OPEN"E" würde bei einem GET ohne Positionsangabe zu einer Fehlermeldung führen, da dadurch versucht wird, hinter der EOF-Marke Daten zu lesen.

HINWEIS: Eine Programmunterbrechung durch einen Fehler führt nicht zur Schließung der geöffneten Datei. Unachtsamkeiten können ernsthafte Folgen nach sich ziehen.

Bedeutung der verschiedenen Positionierungswerte:

Nichts Bei einer Null oder keiner Angabe wird bei Dateien vom Typ "MU", "MF" oder "FF" ein Zugriff auf den nächsten Satz, ausgehend von der letzten Satzadresse erfolgen. Ansonsten verändert sich die momentane Position nicht, und der Zugriff beginnt dort, wo das letzte GET oder PUT endete. Durch jedes OPEN wird die letzte Satzadresse gelöscht und damit auf Null positioniert.

Ausnahme: OPEN"E" positioniert auf EOF

* Während sich normalerweise nach einem GET oder PUT die Position automatisch verändert, indem auf den nächsten Satz weitergeschaltet wird, kann mit Hilfe der Angabe "*" auf einen Teil der Datei mehrfach zugegriffen werden. Die Position bleibt also unverändert.

Bei der Verwendung dieses Positionierungstyps muß dem Rechner die letzte Satzadresse bekannt sein, da in diesem Fall auf den damit definierten Satz nochmals zugegriffen wird. Ist keine Satzadresse gespeichert, kommt es zur Ausgabe einer Fehlermeldung. Bei Verwendung von "FI" und "MI"-Dateien werden entsprechend die gleichen Daten nochmals bearbeitet.

\$ Diese Angabe hat bei allen 5 Filetypen die gleiche Bedeutung. Sie ermöglicht einen weiteren Zugriff auf die beim letzten GET oder PUT bearbeiteten Bytes oder Variablengruppen eines Satzes. Voraussetzung zur Anwendung von "\$" ist wieder das Vorhandensein der letzten Satzadresse, andernfalls erfolgt eine Fehlermeldung.

% Das Zeichen "%" hat keinen direkten Einfluß auf die Daten innerhalb einer Datei, da keinerlei Werte übertragen werden. Es ist auch kein Öffnen des Files notwendig. "%" dient lediglich dazu, bei den File-Typen "FI" und "FF", ähnlich wie bei einer FIELD-Anweisung Datenfelder in ihrem Format aufzubauen. Man spricht auch von einem "Pseudo Fielding".

& Eine Anwendung des "&"-Zeichens hat keinerlei Auswirkung auf die Positionierung und ist zudem nur in Verbindung mit dem PUT-Statement sinnvoll.
Es wurde damit die Möglichkeit geschaffen, den gesamten aktuellen Inhalt des Datenpuffers auf der Diskette abzulegen. Wenn der Datenpuffer zuvor keinerlei Änderung erfahren hat, wird das Statement ignoriert.
Ein PUT mit der Positionierung "&" sollte speziell bei wichtigen Daten und bei Dateien, die über längere Zeit geöffnet bleiben müssen, zur Datensicherung ausgeführt werden.

&& Ein PUT mit "&&" als Positionierungswert veranlasst den Rechner einen neuen EOF-Wert ins Inhaltsverzeichnis einzutragen. Diese Eintragung erfolgt ansonsten nur bei der Schließung des Files durch CLOSE.
Vor dem Eintrag des neuen EOF-Wertes kommt es automatisch zur Abspeicherung des aktuellen Datenpuffers (siehe auch "&").

!RBA Bei dieser Positionierungsart können Sie auf eine beliebige Stelle innerhalb des Files zugreifen, und zwar durch Angabe der Relativen Byte-Adresse (RBA).
Zugriffe auf Marked-Item-Files durch Angabe der RBA sind nur gestattet, wenn RBA auf ein Markierungs-Byte weist. Auf "MI" und "FI"-Dateien kann ausschließlich über die Angabe der RBA wahlfrei zugegriffen werden.

BEISPIEL: PUT 1,!120,,(5)A\$;

positioniert auf die RBA 120 und schreibt 5 Zeichen der Variablen A\$ nach File 1.

HINWEIS: Falsche Angaben bezüglich der RBA können eine Datei zerstören.

!% Bei diesen Positionierungszeichen wird quasi das gleiche wie bei "!RBA" ausgeführt, mit dem Unterschied, daß RBA = EOF gesetzt wird und die Positionierung somit immer auf das Dateiende erfolgt. "!%" dient dadurch besonders zur Erweiterung eines Files.

!\$RBA Mit dieser Anweisung haben Sie die Möglichkeit, bereits vor der eigentlichen Datenübertragung auf die vorgesehene Stelle eine Positionierung vorzubereiten.
Es findet keine Datenübertragung statt. Aus diesem Grunde muß auch kein IGEL oder eine IGEL-Zeilenummer angegeben werden. Die Positionierung wird erst beim nächsten PUT-, GET-, PRINT- oder INPUT-Statement ohne Positionsangabe ausgeführt.

BEISPIELE: PUT 1,!\$120

positioniert auf die RBA 120, ohne Daten zu übertragen.

PUT 1,,(5)A\$;

führt die Datenübertragung ohne Positionsveränderung aus und schreibt 5 Zeichen der Variablen A\$ in File 1.

!\$% Das "\$" am Ende dieser Anweisung deutet bereits darauf hin, daß sie etwas mit dem EOF der Datei zu tun hat. Anstelle der RBA wird in diesem Fall EOF benutzt und die gleiche Operation durchgeführt wie bereits unter "\$RBA" erklärt wurde.

!#RBA Dieses Zeichen ist nur innerhalb eines PUT-Statements sinnvoll und hat eine Änderung des EOF-Wertes zur Folge. Das EOF-Zeichen wird auf die angegebene RBA gesetzt. Ansonsten bleibt der Inhalt eines mit "!#RBA" bearbeiteten Files unverändert. Der Eintrag des neuen EOF-Wertes erfolgt jedoch erst beim Schließen der Datei mit Hilfe der CLOSE-Anweisung oder durch PUT 1,&&.

Satznummer Die Bedeutung der Satznummer wurde bereits erklärt. Bei den Dateien mit besonderer Struktur ist sie nur für Files vom Typ "MF" und "FF" relevant. Bei der Satznummer muß es sich um einen Integerwert zwischen 1 und 32767 handeln. Nach Angabe der Satznummer erfolgt automatisch die Positionierung auf die erste RBA des angegebenen Satzes.

BEMERKUNG: Bei Positionsangaben sind keine Ausdrücke, die die Funktion LOC (x), LOF (x), oder EOF (x) enthalten, erlaubt.

IGEL:

Bei einem IGEL (Item Group Expression List) handelt es sich um eine Liste von beliebigen Variablen oder Ausdrücken, die innerhalb einer geöffneten Datei bearbeitet werden.

Es ist damit möglich, auf die Anweisungen FIELD, LSET und RSET zu verzichten, denn die entsprechenden Angaben zu den Puffer-Variablen erfolgen direkt bei der PUT- oder GET-Anweisung.

Hinter einem normalen PUT oder GET wird, durch Kommata getrennt, eine Liste von Variablen angegeben. Sie kann auch über mehrere Zeilen gehen, wobei auch bei Folgezeilen die Kommata nicht vergessen werden dürfen, entweder am Anfang der Folgezeile oder am Ende der fortzusetzenden Zeile. Die letzte Zeile muß mit einem Semikolon enden.

Es sind auch numerische Variablen als IGEL-Ausdruck zulässig, was zu einer erheblichen Programmiererleichterung führt, da die Konvertierung der Daten automatisch durchgeführt wird.

Bei Fixed-Item-Files, z.B. "FF", müssen die Variablen mit einem Typ-Kennzeichen enden, d.h. mit % für Integer-Variable, ! für einfachgenaue, # für doppeltgenaue Variable oder \$ für Textvariable.

Bei Textvariablen muß in diesem Fall die Länge in Klammern vor der Variablen angegeben werden.

Bei Dateien mit markiertem Inhalt ist diese explizite Kennzeichnung nicht notwendig, in einem IGEL-Ausdruck sind auch Rechenoperationen erlaubt.

IGEL-Zeilenummer:

Um den IGEL übersichtlich gestalten zu können, z.B. am Programmende), ist es möglich, über die Angabe einer IGEL-Zeilenummer eine Zeile innerhalb des Programms anzuspringen, bei der die Aufzählung von Variablen, die bearbeitet werden sollen, beginnt. Es ist darauf zu achten, daß das Programm niemals diese IGEL-Zeilen durchläuft. Die Variablen können wie bei einem IGEL auch über mehrere Zeilen verteilt werden.

IGEL-Ausdrücke:

Die Angaben innerhalb des IGEL unterliegen gewissen Vorschriften, die nun näher erläutert werden sollen.

Folgende Angaben sind im IGEL möglich:

1. Ausdruck
2. (Länge) Text-Variable
3. (Länge)\$
4. (Länge)#
5. Nichts

Mit einem Ausdruck ist hier z.B. die Angabe einer Zeichenkette, eines Zahlenwerts oder einer entsprechenden Variablen gemeint.

Bei der Verwendung von Fixed-Item-Files ist darauf zu achten, daß die Variablen mit ihren Typbezeichnungen, also \$, %, ! oder #, angegeben werden.

Die Längenangabe vor einer Variablen darf nur bei Strings erfolgen. Sie gibt bei Marked-Item-Files die maximale Anzahl der zu lesenden oder abzulegenden Zeichen einer Zeichenkette an und kann zwischen 1 und 255 betragen. Sollte die Variable weniger Zeichen beinhalten als angegeben sind, werden nur die wirklich vorhandenen Zeichen bearbeitet. Dies ist auch dann der Fall, wenn auf eine Längenangabe verzichtet wird.

Bei Fixed-Item-Files hat in jedem Fall eine Längenangabe zum Ein- und Auslesen von Zeichenkettenvariablen zu erfolgen.

Soll in einem Fixed-Item-File eine bestimmte Anzahl von Bytes bei einem GET ignoriert, d.h. nicht gelesen oder bei einem PUT nicht überschrieben werden, läßt sich dies durch die Angabe der Anzahl der entsprechenden Bytes, gefolgt von einem Dollar-Zeichen bewerkstelligen.

BEISPIEL: GET 1,,(5)\$

Die nächsten 5 Bytes von File 1 werden überlesen.

Der Ausdruck (Länge)# hat eine ähnliche Bedeutung wie (Länge)\$. Wenn er bei Fixed-Item-Files benutzt wird, erfolgt bei GET-Statements ein Überlesen der angegebenen Byteanzahl. Bei PUT-Anweisungen kommt es dagegen zu einem der Länge entsprechenden Eintrag von Bytes mit dem Inhalt 0. (Länge)# ist auch bei Marked-Item-Files anwendbar.

Läßt man einen IGEL-Ausdruck weg, durch zwei aufeinanderfolgende Trennzeichen, kann man beim Lesen von Daten aus einem Marked Item File einzelne Werte der Datei ignorieren. Das geht deshalb nur bei Marked-Item-Files, weil Typ und Länge der Daten in der Datei stehen, das BASIC also weiß, wieviel Bytes es überlesen muß. Bei Fixed-Item-Files ist (Länge)\$ zu verwenden.

BEISPIEL: PUT 1,10,,(100)A\$, (20)B\$,A%,B!,C#;

Die Variable A\$ hat eine Länge von 100 und B\$ von 20 Bytes. A\$ und B\$ müssen nicht durch LSET oder RSET in den Datenpuffer übernommen werden. Ebenso ist es nicht notwendig die Zahlenvariablen A%, B! und C# mittels der Funktionen MKI, MKS und MKD in Strings umzuwandeln, um sie in die Datei zu übernehmen. Sie werden direkt in der Variablenliste angegeben.

Die Variablen dürfen innerhalb der Zeilen auch über
REMARKS dokumentiert werden.

```
:  
:  
500 PUT 1,1,1010  
:  
:  
1000 STOP: REM Kein Programmlauf ab Zeile 1000  
1010 REM Variablenliste  
1020 "Zeichenkette",: REM Strings  
1030 (20)A$, : REM Stringvariable  
1040 1.234567 REM Zahlenwert  
1050 ,C# REM Zahlenvariable  
1060 ; : REM Ende der Variablenliste  
:  
:
```

Bei diesem Beispiel werden der String "Zeichenkette", der Zahlenwert 1.234567, sowie die Variablen A\$ und C# in den ersten Satz der über Puffer 1 geöffneten Random-Datei abgespeichert. Zu beachten ist, daß alle Teile des IGEL durch Komma voneinander getrennt sind, auch bei Folgezeilen. Das Semikolon schließt den IGEL ab. Unerklärliche IGEL-ITEM-Syntaxfehler sind meist auf ein vergessenes Trenn- oder Abschlußzeichen zurückzuführen.

SYNTAX: LOC (Puffernummer)Parameter

=====

WIRKUNG: Diese Funktion wurde zwar schon in den vorigen Kapiteln angesprochen, hat aber bei den Dateien mit besonderer Struktur zusätzliche Möglichkeiten. Um die Positionierung innerhalb von Dateien zu erleichtern, wurde die bisher bekannte Funktion "LOC (Puffernummer)" um weitere vier Arten erweitert. Während die bisherige LOC-Funktion lediglich die zuletzt bearbeitete Satznummer ausgibt, können Sie nun durch Hinzufügen eines speziellen Zeichens den Rechner dazu veranlassen, genauere Informationen über die Position in der gerade bearbeiteten Datei auszugeben.

1. LOC (Puffernummer)\$

Mit dieser Abfrage können Sie bei in Sätzen aufgeteilten Files feststellen, ob die augenblickliche Positionierung größer, gleich oder kleiner EOF ist. Wenn der Beginn des nächsten Satzes oder die Position bei gespeicherter Satzadresse größer oder gleich EOF ist, ergibt die Funktion den Wert -1 (IF-Statement = True).

Sollte der Anfang des folgenden Satzes oder die aktuelle Filepositionierung kleiner als EOF sein wird der Wert 0 ausgegeben (IF-Statement = False).

Bei nicht satzsegmentierten Files ("MI" und "FI") wird nur auf die aktuelle Positionierung hin untersucht.

2. LOC (Puffernummer)%

Mit dieser Funktion haben Sie die Möglichkeit sich die RBA, die der EOF-Position entspricht, ausgeben zu lassen.

3. LOC (Puffernummer)!

Sollte die Datei in Sätze aufgeteilt sein, wird diese Funktion die Startadresse des folgenden Satzes als RBA ausgeben.

Wenn keine Satznummern erlaubt sind, gibt die Funktion die momentane Positionierung als RBA aus.

4. LOC (Puffernummer)#

Diese Funktion bewirkt bei satzsegmentierten Files die Ausgabe der gespeicherten Satzadresse im RBA-Format. Sollte keine Satzadresse gespeichert sein, kommt es zu einer Fehlermeldung.

Anhang

Das Hilfsprogramm DIRCHECK/CMD

Eine genaue Auflistung aller Aufzeichnungen einer Diskette können Sie durch Aufruf des auf der System-Diskette befindlichen Hilfsprogrammes DIRCHECK/CMD erhalten. Dieses Programm wird wie alle durch den Zusatz /CMD gekennzeichneten Programme, nur durch Eingabe des Namens aufgerufen, falls sich das System im DOS befindet.

Mit Hilfe von DIRCHECK/CMD kann das Inhaltsverzeichnis gesäubert, jedoch nicht repariert werden. Weiterhin ist es möglich, das Inhaltsverzeichnis geschützt zu schreiben, was zum Austausch von Disketten zwischen Genie I/II Expander- und FC-Anlagen notwendig ist.

Nach dem Programmstart erfolgt die Frage

DRUCKER-AUSGABE ?

Wenn Sie hier mit J antworten, wird das Inhaltsverzeichnis auf einem angeschlossenen Drucker ausgegeben, bei der Eingabe von N erscheint es auf dem Bildschirm.

Anschließend fordert das Programm die Angabe der Nummer des Laufwerks (0...3), in dem sich die zu bearbeitende Diskette befindet.

DIRCHECK/CMD testet das Directory und zeigt Fehler, falls welche gefunden werden, vor dem Inhaltsverzeichnis der an. Zuletzt wird noch die Anzahl der freien und getilgten Einheiten der überprüften Diskette ausgegeben. Die Angaben von numerischen Werten erfolgen bei DIRCHECK/CMD in hexadezimaler Schreibweise.

Dem Kommentar

FUNKTION BEENDET

folgt zum Abschluß ein Menue. Sie erhalten nun nach Eingabe der entsprechenden Abkürzungen die folgenden Möglichkeiten:

- N Programmende, Rücksprung ins DOS
- Y Eine weitere Diskette mit den gleichen Daten bezügl. Spurenanzahl, Schreibdichte und Lage des Inhaltsverzeichnisses soll überprüft werden
- I Neustart des Programms
- W Die Adressmarke des Inhaltsverzeichnisses wird gelesen, korrigiert und geschützt zurückgeschrieben.
- C Bei dieser Eingabe werden alle nicht benutzten Inhaltsverzeichniseinträge auf 0 gesetzt.

Während die Liste des Inhaltsverzeichnisses auf dem Bildschirm oder auf einem Drucker ausgegeben wird, haben Sie noch die Möglichkeit folgende Eingaben zu vollziehen:

Durch Drücken der BREAK-Taste wird die auf dem Schirm erscheinende Liste nach der nächsten Zeile oder Zeilengruppe unterbrochen.

NEW LINE bewirkt die Fortführung des Programms nach einer durch BREAK erfolgten Unterbrechung. Mit der Hochpfeil-Taste läßt sich das Listen bzw. Drucken abbrechen.

Anzeige des Inhaltsverzeichnisses der G-DOS 2.1a Systemdiskette
nach Aufruf des Hilfsprogramms DIRCHECK/CMD:

Befehlseingabe:DIRCHECK

DRUCKER-AUSGABE ? N

IN WELCHEM LAUFWERK IST DIE DISKETTE? 0

GDOS2.1A 21.03.84

| | | | | |
|--------------|-------|--------------|--------|---------|
| GDOS/SYS | SIP=6 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS6/SYS | SIP=7 | Ende: 35/0 | 1 Erw. | 35 Sek. |
| SYS14/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS22/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| BASIC/CMD | -- | Ende: 18/0 | 1 Erw. | 20 Sek. |
| INHALT/SYS | SIP=5 | Ende: 30/0 | 1 Erw. | 30 Sek. |
| SYS7/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS15/SYS | SIP=7 | Ende: 3/18 | 1 Erw. | 5 Sek. |
| SYS23/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS0/SYS | SIP=7 | Ende: 70/0 | 1 Erw. | 70 Sek. |
| SYS8/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS16/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS24/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| PROGMOD/BAS | | Ende: 2/50 | 1 Erw. | 5 Sek. |
| SYS1/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS9/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS17/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS25/SYS | SIP=7 | Ende: 10/0 | 1 Erw. | 10 Sek. |
| SYS2/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS10/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS18/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS26/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| KEY/CMD | | Ende: 9/215 | 1 Erw. | 10 Sek. |
| ASM/CMD | | Ende: 33/167 | 1 Erw. | 15 Sek. |
| SYS3/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS11/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS19/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS27/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS4/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS12/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS20/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS28/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS5/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS13/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS21/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| SYS29/SYS | SIP=7 | Ende: 5/0 | 1 Erw. | 5 Sek. |
| LIST64/BAS | | Ende: 2/82 | 1 Erw. | 5 Sek. |
| FORMLIST/BAS | | Ende: 3/86 | 1 Erw. | 5 Sek. |
| VIDEO/BAS | | Ende: 6/155 | 1 Erw. | 10 Sek. |
| DIRCHECK/CMD | | Ende: 15/0 | 1 Erw. | 15 Sek. |
| D50/CMD | | Ende: 0/80 | 1 Erw. | 5 Sek. |
| JOB/CMD | | Ende: 33/0 | 1 Erw. | 35 Sek. |
| ITOH/CMD | | Ende: 1/210 | 1 Erw. | 5 Sek. |
| DEMO/JOB | | Ende: 1/0 | 1 Erw. | 5 Sek. |

488 Freie Einheiten 0 Getilgte Einheiten

TASTE

N PROGRAMMENDE

Y NEUE DISKETTE OHNE ERNEUTE DRUCKERABFRAGE

I PROGRAMM NEU STARTEN

W ADRESSMARKE des INHALTSVERZEICHNISSES KORRIGIEREN

C INHALTSVERZEICHNIS SÄUBERN

=====
Mit Hilfe des Programms VL/CMD haben Sie die Möglichkeit, Maschinenprogramme von Kassette oder Diskette in den Speicher des Rechners einzulesen. Dabei werden die Ladeinformationen des eingelesenen Programms angezeigt.

Weiterhin läßt sich mit VL/CMD der Ladebereich eines Programms ändern, wodurch es nach dem Einlesen in einem anderen Speicherbereich zur Ausführung gebracht wird.

Die geänderten Programme können unter einem neuen Namen auf der Diskette abgelegt werden.

Hier nun die Funktionen des Programms im Einzelnen:

1. Nach Programmstart, (Eingabe von VL/CMD in der DOS-Befehlsebene), erscheint die Frage, ob sich die Quelldatei auf einer Diskette oder Kassette befindet.

Handelt es sich um ein Kassettenprogramm, so wird, sobald die Eingabe von C mit anschließendem NEW LINE erfolgt ist, in der oberen rechten Ecke des Bildschirms ein Stern "*" angezeigt.

Positionieren Sie nun den Recorder auf den Anfang des Programms und schalten Sie ihn auf Wiedergabe (PLAY). Sobald sich der Rechner auf das Band synchronisiert hat, erscheinen zwei weitere Sterne von denen der rechte blinkt. Sollte beim Einlesen des Programms ein Fehler auftreten, so wird dies durch die Anzeige eines "C" (für Checksum-Fehler) anstelle eines Sterns kenntlich gemacht. Weiterhin können ein "P" bei zusätzlich führenden Bytes oder ein "I" bei sonstigen führenden Bytes auf den Fehler aufmerksam machen.

Liegt das mit VL/CMD zu bearbeitende Programm bereits auf einer Disk vor, so ist die erste Frage mit einem D und anschließendem NEW LINE zu beantworten. VL/CMD fragt dann nach dem Filenamen, wobei auch die Extension mit anzugeben ist.

2. Nachdem das gewünschte Programm von Kassette oder Diskette eingelesen ist, erscheint die Anzeige des Ladebereiches im Speicher und die Startadresse, wobei alle Angaben in hexadezimaler Form ausgegeben werden.
Sollte das eingelesene Programm in den ROM- oder DOS-Bereich ragen, erscheint diesbezüglich ein entsprechender Hinweis.
3. Es erfolgt nun die Frage nach der neuen Ladeadresse. Soll das Programm unter dem gleichen Adressbereich abgespeichert werden, so ist die Frage einfach mit NEW LINE zu beantworten. Liegt eine neu angegebene Ladeadresse innerhalb des ROM- oder DOS-Bereiches, erfolgt wie unter Punkt 2 ein Hinweis auf diese Überschneidung.
Nach Angabe der neuen Ladeadresse besteht die Möglichkeit, einen normalerweise durch VL/CMD erzeugten Zusatz zum Verschieben des Programms zu verhindern. Das hat jedoch zur Folge, daß derartig abgespeicherte Programme nur noch mit Hilfe des DOS-Befehls "LOAD" von der Diskette eingelesen werden können, da der Sprung zur neuen Startadresse fehlt. Als Startadresse wird in diesem Falle eine Null eingetragen.

4. Wenn Sie nach Angabe einer neuen Ladeadresse die Frage "Startanweisung unterdrücken?" mit "N" beantworten, erhält das durch VL/CMD bearbeitete Programm einen kurzen Zusatz. Innerhalb dieses Zusatzprogrammteils liegt die neue Startadresse.
Es folgt nun die hexadezimale Anzeige des vom verschobenen Programm benutzten Speicherbereichs und seine neue Startadresse.
VL/CMD fordert Sie danach wiederum auf, eine neue Ladeadresse anzugeben. Sollte Ihnen der angegebene Speicherbereich des verschobenen Programms noch nicht zusagen, können Sie erneut die Ladeadresse verändern, ansonsten drücken Sie NEW LINE und beantworten die Frage, ob das DOS unterdrückt werden soll. (Bei einigen Programmen kann es in Verbindung mit dem DOS zu Konflikten kommen. Beantworten Sie in diesem Fall die Frage mit "N".)
5. Nun können Sie entscheiden, ob das bearbeitete Programm auf Kassette oder Diskette abgespeichert werden soll.
Bei Kassettenspeicherung fragt VL/CMD nach einem 6-stelligen Programmnamen unter dem das Programm später von der Kassette auch wieder eingeladen werden kann.
Dabei ist zu beachten, daß die erste Stelle des Programmnamens immer ein Buchstabe sein muß (A - Z). In den weiteren fünf Stellen dürfen auch Zahlen (0 - 9) enthalten sein.
Nach Angabe des Programmnamens drücken Sie, wenn der Recorder aufnahmebereit ist, die NEW LINE-Taste und die Abspeicherung beginnt.
Der Vorgang bei Diskettenspeicherung ist dem der Kassettenspeicherung sehr ähnlich. Auch hier wird ein Name für die Zieldatei verlangt, der in der üblichen Weise, (also achtstellig plus Extension und evtl. folgender Laufwerksnummer), angegeben werden muß. Nachdem die Ausgabe eines Programms beendet ist, fragt VL/CMD ob Sie noch eine weitere Abspeicherung des Programms vornehmen möchten. Antworten Sie mit "N", erscheint die Frage ob eine weitere Quelldatei bearbeitet werden soll. Ein "J" hat einen Neustart des Programms zur Folge, und ein "N" beendet VL/CMD mit einem Sprung in die DOS-Befehlsebene.

Das Hilfsprogramm JOB/CMD

=====

Bei JOB/CMD handelt es sich um ein Programm zur Erstellung einer JOB-Datei. Diese dient dazu, dem Benutzer immer wiederkehrende Tastatureingaben, z.B. setzen von HIMEM, Aufruf von BASIC oder Unterprogrammen etc., zu ersparen.

Hierzu dient als Beispiel die auf der Diskette bereits vorhandene Datei DEMO/JOB.

Geben Sie zu Ihrer Ausführung einfach

```
DO DEMO/JOB
```

in der DOS-Befehlsebene ein.

Das automatisch ablaufende DEMO/JOB führt eine Vielzahl von Kommandos aus, die normalerweise alle von Hand eingegeben werden müssen. Alle Befehle werden dazu von der Diskette aus der DEMO/JOB-Datei gelesen.

Mit JOB/CMD ist sowohl das Editieren einer bereits vorhandenen, als auch die Erstellung einer völlig neuen JOB-Datei möglich. Soll eine bereits auf Diskette vorhandene JOB-Datei verändert oder erweitert werden, so fragt das Programm zuerst nach dem Namen der Datei, wobei auch die Extension (/JOB) mit anzugeben ist.

Die gewünschten Befehle einer JOB-Datei müssen nacheinander, jeweils durch Drücken der NEW LINE-Taste getrennt, eingegeben werden. Zur besseren Übersicht erfolgt eine automatische Numerierung der Kommandos.

Mittels der BREAK-Taste läßt sich eine Neueingabe beenden.

JOB/CMD fragt sodann, ob editiert, abgespeichert, neu eingegeben oder ins DOS gesprungen werden soll.

Für den Fall, daß Sie Ihre JOB-Datei noch erweitern wollen, drücken Sie "E" und anschließend "A". Sie werden nach der Nummer des zu ändernden Befehls gefragt und können ihn ihrer Angabe verändern.

Wenn Sie eine Erweiterung der Befehlsliste wünschen, geben Sie einfach die nächste, nicht mehr in der Datei stehende Nummer an.

Vor dem Abspeichern fragt JOB/CMD nach dem Namen, unter dem die JOB-Datei auf der Diskette abgespeichert werden soll.

Die auf diese Weise erstellten Dateien werden mit Hilfe des DOS-Befehls DO aufgerufen und zur Ausführung gebracht.

Dieses ca. 2,5 KByte lange Programm ermöglicht Ihnen die Anzeige von vergrößerten Zeichen auf dem Bildschirm, die Ausgabe von akustischen Signalen auf dem Kassettenport und einen schnellen Aufbau von Blockgrafiken aus dem BASIC heraus.

SUPER/CMD ist ein selbstverschiebendes Treiberprogramm, das ab 5200H lädt, automatisch HIMEM verändert und BASIC aufruft.

Nach dem Programmstart von SUPER/CMD erscheint die Anzeige

```
BASIC + GELADEN
READY
>
```

und im unteren Bildschirmteil in invertierter Großschrift der Hinweis MIT TON.

Sie haben nun folgende Möglichkeiten mit dem Programm zu arbeiten:

1. Großschrift

Bei eingeschalteter Großschrift stehen Ihnen 5 Zeilen mit jeweils 21 Zeichen zur Verfügung.

Die Schrift kann sowohl in normaler, als auch in invertierter Darstellung ausgegeben werden.

Zur Steuerung der Großschrift werden die Control-Codes 16 bis 19 verwendet.

```
PRINT CHR$(16) schaltet die invertierte Darstellung ein
PRINT CHR$(17) schaltet die invertierte Darstellung aus
PRINT CHR$(18) schaltet die Großschrift ein
PRINT CHR$(19) schaltet die Großschrift aus
```

Eine invertierte Darstellung ist natürlich nur bei eingeschalteter Großschrift sichtbar.

2. Ausgabe von akustischen Signalen:

Hierzu wird der Befehl NAME aus dem BASIC auf eine Maschinenroutine im Speicher gelenkt, die dann entsprechend dem nachfolgenden Zeichen einen Ton über den Kassettenport ausgibt. Probieren Sie es einfach einmal.

Beispiel:

```
NAME A: NAME H: NAME (: NAME ?
```

3. Schnelle Blockgrafik

Hierbei ist es möglich, auf dem Bildschirm zwei Punkte der Blockgrafik durch eine Linie von Grafikblöcken verbinden zu lassen.

| | |
|-----------------------------|---|
| LINE(X1,Y1)-(X2,Y2)SET | zieht eine Linie zwischen den Punkten (X1,Y1) und (X2,Y2) |
| LINE(X1,Y1)-(X2,Y2)RESET | löscht eine Liene zwischen den Punkten (X1,Y1) und (X2,Y2) |
| LINE(X1,Y1)-(X2,Y2)SET,B | zeichnet ein Rechteck mit den Eckpunkten (X1,Y1) und (X2,Y2) |
| LINE(X1,Y1)-(X2,Y2)RESET,B | löscht ein Rechteck mit den Eckpunkten (X1,Y1) und (X2,Y2) |
| LINE(X1,Y1)-(X2,Y2)SET,BF | zeichnet ein ausgefülltes Rechteck mit den Eckpunkten (X1,Y1) und (X2,Y2) |
| LINE(X1,Y1)-(X2,Y2)RESET,BF | löscht ein ausgefülltes Rechteck mit den Eckpunkten (X1,Y1) und (X2,Y2) |

Beispiel:

```
10 CLS
20 LINE(0,0)-(127,47)SET:GOSUB50:LINE(0,0)-(127,47)RESET:
   GOSUB50
30 LINE(0,0)-(127,47)SET,B:GOSUB50:LINE(3,2)-(124,45)SET,BF:
   GOSUB50:LINE(3,2)-(124,45)RESET,BF:GOSUB50:
   LINE(0,0)-(127,47)RESET,B:GOSUB50
40 GOTO20
50 A$=INKEY$
60 IFA$=""THEN50ELSERETURN
```

Dieses Beispiel erzeugt nach Programmstart auf dem Bildschirm eine Diagonale von links oben nach rechts unten. Anschließend, nach Betätigung irgendeiner Taste, wird die Diagonale wieder gelöscht.

Es folgt die Erstellung eines Rechtecks, in welches anschließend ein weiteres, jedoch ausgefülltes Rechteck gezeichnet wird.

Die nächsten beiden Tasteneingaben haben das Löschen dieser beiden Rechtecke zur Folge und das Programm beginnt wieder von Neuem.

KEY ist ein Hilfsprogramm für das Genie III, das Ihnen die tägliche Arbeit mit dem Computer erheblich vereinfachen soll. KEY ersetzt die Tastatur-, Monitor- und Drucker-Programme im Gerät durch sehr viel leistungsfähigere Treiber:

Drucker:

- Ein serieller Drucker kann angesprochen werden.
- Die Druckausgabe erfolgt mit Überspringen der Falz.
- Zeilen- und Seitenlänge können eingestellt werden.
- Bei Zeilenüberschreitung wird automatisch eingerückt.
- Durch Tastendruck oder BASIC-Befehl kann der Druckertreiber für Graphikausgabe abgeschaltet werden.
- Zeilen- und Seitenvorschub lassen sich von der Tastatur aus steuern.

Monitor:

- Blinken und Wahl eines anderen Cursors erfolgen flackerfrei.

Tastatur:

- Wichtige BASIC-Befehle mit einem Tastendruck.
- Tastenwiederholung und verbesserte Entprellung für alle Tasten, auch im rechten Tastenblock.
- Tastenwiederholung auch bei abgeschalteten Interrupts.
- Durch einen BASIC-Befehl lassen sich diese Funktionen abschalten. Die Funktionstasten erhalten dann besondere Codes.
- Alles, was auf dem Videoschirm steht, läßt sich als Eingabe wiederverwenden. Sie haben mit KEY einen vollständigen und sehr komfortablen Bildschirmeditor.

Damit der Editor auch in der Befehlseingabe benutzt werden kann, wurde im Modul SYS1/SYS der ":" hinter dem Wort "Befehlseingabe" in einen Zeilenrücklauf geändert.

Die Abkürzung für COPY, das '>'Zeichen, läßt sich mit dem Editor nicht wieder einlesen, da es mit dem BASIC-Promptzeichen verwechselt wird, schreiben Sie COPY aus. Benutzen Sie den Editor getrost auch im BASIC AUTO-Modus, die Zeilennummer wird dann nicht übergeben (sinnvoll auch zum Duplizieren von Zeilen). Die BREAK-Taste sollte durch die G-DOS-Systemoption AG=J aktiviert sein, damit der Editor richtig funktioniert.

Die Tastatur-Routine reagiert nicht mehr auf den Befehl LC,N. Dieser betrifft nur noch den Monitor.

Start des Programms:

Starten Sie KEY/CMD durch Eingabe von KEY <NEW LINE> in der Befehlseingabe. KEY/CMD schiebt sich automatisch an das obere Speicherende und schützt sich selbst. Nach dem Start zeigt das Programm den belegten Speicherplatz in Hex-Schreibweise an. Mit dem HIMEM-Befehl lässt sich vor dem Aufruf von KEY/CMD Speicherplatz für andere Maschinenprogramme freihalten. Auf diese Art können auch die Programme wie ITOH/CMD oder STAR510/CMD weiterbenutzt werden. Sie müssen nur zuerst ITOH/CMD bzw. STAR510/CMD und dann KEY/CMD starten. Wenn Sie in der Befehlseingabe den Befehl AUTO KEY eingeben, haben Sie alle seine Möglichkeiten schon mit dem Start des Systems zu Verfügung. Soll JOB/CMD benutzt werden, ist KEY/CMD durch INFO T,M,D abzuschalten.

Die Tastatur:

Die Tastatur Ihres Genie III verhält sich in einigen Punkten anders als gewohnt. Die Tastenwiederholung reagiert auch auf den Numerikblock und die Funktionstasten. Einige Probleme entstanden bisher auch durch die Lock-Taste. Bei manchen Geräten erscheinen bei zu schnellem Tippen falsche Zeichen, z. B. die '00' zwischen 'S' und 'R', falls die Lock-Taste gedrückt ist. KEY behebt diesen Fehler. Die Funktionstasten erzeugen bei gelöster LOCK-Taste auch keine Kleinschreibung mehr. Die CTRL-Taste zusammen mit den Buchstabentasten erzeugt jetzt ASCII-Codes von 01H bis 1EH. Die Funktion von CTRL, wie sie im G-DOS-Handbuch beschrieben ist, übernimmt jetzt <SHIFT> <CTRL>.

Einige Codes (<CTRL> <L> = 0CH, <CTRL> <P> = 10H, <CTRL> <R> = 12H und <CTRL> <S> = 13H) können nicht mehr eingegeben werden, da sie für die Steuerung einiger Funktionen gebraucht werden.

Mit <SHIFT> <0> (Null) ist ein weiteres Zeichen, '_' (5FH) über die Tastatur erreichbar. <CLEAR> löscht nicht den Schirm, sondern schaltet den Editor ein. Um den Schirm zu löschen, drücken Sie bitte <SHIFT> <CLEAR>. Sollten Sie versehentlich den Editor gestartet haben, so können Sie ihn mit <BREAK> ohne Wirkung wieder verlassen.

Kurzbefehle (Short Hands) und Funktionstasten:

<P1>

Bei gedrückter LOCK-Taste können Sie mit <SHIFT> + Buchstabe ganze BASIC-Befehle auf Tastendruck eingeben. Es wurde der Versuch gemacht, die wichtigsten Befehle leicht zu merkenden Tasten zuzuordnen, z. B. <A> für 'ASC(' oder <L> für 'LOAD'. Die Zuordnungen sind fest vorgegeben.

Die Funktionstasten arbeiten wie gewohnt, das Dienstprogramm F# wurde nicht verändert. Allerdings hat der ganze rechte Tastaturblock eine automatische Wiederholungsfunktion. Auch hat die LOCK-Taste keine Wirkung mehr auf die Funktionstasten.

Bis auf Drucker-Fernsteuerung und Autorepeat lassen sich alle KEY-Funktionen abschalten. Setzen Sie dazu Bit 0 im Statusbyte auf Adresse 4018H auf Null, ohne die anderen Bits zu verändern. KEY verhält sich dann wie bei abgeschaltetem Cursor:

Shorthandbefehle werden zu Kleinbuchstaben, <CLEAR> erzeugt den Code 1FH statt den Editor zu starten, und die Funktionstasten erzeugen einen Code, der der Nummer der Taste entspricht:

<F2> erzeugt 2, <F8> den Code 8. Lediglich <F1> macht eine Ausnahme, die Taste liefert 1BH (27D), den Code für <ESC>. Um <F8> von <Linkspfeil> unterscheiden zu können, muß das Anwenderprogramm mit `IF PEEK(&H38A0) AND 1 THEN ...` zusätzlich die Tastaturmatrix direkt abfragen. Ähnlich könnte verfahren werden, wenn mit <SHIFT> <Fn> zusätzliche Funktionen realisiert werden sollen, es muß dann die Shift-Taste auf Adresse 3880H getestet werden.

Wie gesagt, gilt diese Umschaltung bei ausgeschaltetem Cursor immer, da die Ausgabe von ganzen Zeichenketten nur bei eingeschaltetem Cursor einen Sinn ergibt.

Der Editor:

<CLEAR> löscht nicht mehr den Schirm sondern eröffnet Ihnen die Möglichkeit, alles, was dort steht, als Eingabe wiederzuverwenden. Den Schirm können sie mit <SHIFT> <CLEAR> löschen. Daß Sie im Editormodus sind, zeigt Ihnen ein blinkender Cursor, der sich mit den Pfeiltasten in jede Richtung bewegen läßt.

<NEW LINE> oder <ENTER> übergibt die Zeile, in der der Cursor steht, genau so, als ob sie neu eingetippt worden wäre. Dabei ist es möglich, Zeilen die länger als 64 bzw. 80 Zeichen sind, zu übernehmen. Die maximale Zeilenlänge ist etwa 240. Längere Zeilen werden nicht akzeptiert. Der Editor kennt das aufrufende Programm, z.B. das DOS nicht. Deshalb scheinen manche Informationen abgeschnitten, da das DOS nur 80 Zeichen pro Eingabe annimmt! Ein führendes '>' oder '*' wird als Promptzeichen gewertet und nicht übergeben. Damit ist auch von der Benutzung von '>' als COPY-Kürzel abzuraten.

<SHIFT> <Rechtspfeil> setzt die Endemarkierung (ein Graphikzeichen) in die nächste Zeile.

<SHIFT> <Linkspfeil> löscht die Markierung wieder.

<P2>

<CTRL> <H> (wie 'Hack') setzt die Markierung an die Cursorposition, was einem Abhacken der Zeile entspricht.

Um Zeilen, die länger sind als eine Bildschirmzeile, mit <NEW LINE> zu übernehmen, muß der Cursor unbedingt auf der ersten Bildschirmzeile des zu übergebenden Bereichs stehen. Übergeben wird alles vom Anfang dieser Zeile bis zur Endemarkierung oder dem Ende der Zeile, wenn keine Markierung gesetzt ist. Wird eine Zeile durch Einfügungen länger als eine Bildschirmzeile, erscheint die Markierung automatisch. Vergessen Sie nicht vor <NEW LINE> den Cursor in die erste Bildschirmzeile zu bewegen.

<SHIFT> <BREAK> schaltet den Einfügemodus ein und aus. Das Cursorzeichen ändert sich zur Bestätigung der Umschaltung. Im Einfügemodus schieben nachfolgende Zeichen den Rest der Zeile nach rechts. Setzen Sie bei Bedarf vorher eine Endemarkierung.

<CLEAR> löscht das Zeichen an der Cursorposition und rückt den Rest der Zeile heran.

<SHIFT> <CLEAR> löscht eine zusammenhängende Folge von Leerzeichen. Auf diese Art lassen sich leicht Zeilen zusammenfügen.

<BREAK> bricht den Editor ab.

Die wichtigste Anwendung wird wohl die Editierung eines BASIC-Programms oder einzelner BASIC-Zeilen sein, die nicht einmal eine Zeilennummer haben müssen. Übertippen der Zeilennummer ist genauso möglich, wie Einfügen einer Zeilennummer vor ein direktes Statement. Vertippt man sich innerhalb einer Zeile, so hilft meist <CLEAR>, bevor man die Zeile komplett neu eingibt, nur weil der Fehler gerade ganz vorne war! Hüten Sie sich bitte vor Linefeeds in BASIC-Zeilen, da der Editor diese nicht erkennt und in Leerzeichen umsetzt.

Im BASIC-AUTO-Modus arbeitet der Editor einwandfrei, die Zeilennummer gehört dann nicht zur Eingabezeile. Mit AUTO und <CLEAR> lassen sich ähnliche Zeilen ohne Übertippen der Zeilennummer leicht duplizieren. Eine andere Anwendung des Editors ist die Wiederverwendung von G-DOS- oder Assemblerbefehlen. Besonders lange COPY-Befehle werden Sie zur Benutzung des Editors veranlassen.

Der Druckertreiber:

Am Ende einer jeden Druckseite erfolgt automatisch ein Seitenvorschub. Dadurch werden Programmlisten etc. sehr viel übersichtlicher, da die Falz nicht mehr bedruckt wird. Überlange Zeilen werden nach einer einstellbaren Anzahl von Zeichen umgebrochen. Der Rest der Zeile wird etwas eingerückt. Dadurch erscheinen BASIC Zeilennummern immer alleine am linken Papierrand, der Rest steht immer rechts davon.

Damit die Formatierung nicht durcheinandergerät, müssen Sie sich eine gewisse Disziplin im Umgang mit dem Drucker angewöhnen. Die Knöpfe am Drucker werden tabu, die Funktionen Zeilenvorschub und Neue Seite lassen sich in einfacher Weise von der Tastatur fernbedienen:

<CTRL> <L> (erst die CTRL-Taste festhalten, dann L drücken) erzeugt einen Zeilenvorschub (Line Feed), <CTRL> <P> einen Seitenvorschub (Page Feed).

Die Formatierung läßt sich abschalten, wenn z. B. Grafikausgabe gemacht werden soll. Dazu drücken Sie <CTRL>F<G>. Oben rechts auf dem Schirm muß ein G erscheinen. Danach ist die Drucker-schnittstelle völlig transparent. Jedes Zeichen wird ungeprüft an den Drucker weitergegeben. Ein weiterer Druck auf <CTRL>F<G> schaltet den Treiber wieder ein, signalisiert durch ein F (wie Format) oben rechts auf dem Schirm. Dabei werden die Zähler für die momentane Druckposition auf 0 gesetzt. Nachdem man am Drucker ein neues Blatt eingelegt oder die Walze verdreht hat, läßt sich die Formatierung so neu initialisieren, ohne die Werte für Zeilen- oder Seitenlänge zu verändern!

<CTRL> <S> wählt statt der parallelen (P oben rechts) die serielle (S) Druckerschnittstelle. Das dient hauptsächlich der Datenübertragung zu fremden Rechnern. Die Übertragung erfolgt auch im Graphikmodus nicht transparent: CR (0DH) wird immer durch LF (0AH) ergänzt.

Getestet wurde die Übertragung einer ASCII-Datei auf einen Rechner mit einem anderen Betriebssystem. Die Schnittstelle im Genie III wurde durch V24 initialisiert und zur Kontrolle mit diesem Programm ein Zeichen gesendet. Das ist zur vollständigen Initialisierung notwendig!. Danach wurde der Druckertreiber mit <CTRL><G><CTRL><S> initialisiert und mit dem G-DOS-PRINT-Befehl die Datei in den fremden Rechner "gedruckt". Der andere Rechner wollte danach nur noch das Endezeichen LAH bekommen, um die Übertragung zu beenden. Das ging am einfachsten mit einem FORM-Befehl unter G-DOS, in SYS28/SYS musste lediglich das Zeichen LAH mit DDE eingetragen werden. (Achtung: BASIC LPRINT CHR\$(&H1A) hängt sich auf!). Einen zweiten Drucker über die serielle Schnittstelle zu betreiben ist möglich, wurde aber noch nicht getestet.

<CTRL> <R> initialisiert den Druckertreiber komplett neu, d.h. sämtliche Parameter, die das Seitenformat betreffen, werden auf ihre Vorgabewerte gesetzt.

Ändern der Parameter:

Alle Parameter lassen sich nach der folgenden Tabelle durch POKE-Befehle einstellen. Hinter der Speicheradresse findet sich in Klammern die Position des entsprechenden Bytes in der Datei KEY/CMD und der Standardwert. Diese Werte werden bei jedem Start von KEY oder nach <CTRL> <R> in die Speicheradressen zurückgeladen. Diese Defaulteinstellungen lassen sich mit DDE leicht ändern, die Angaben sind hexadezimal.

4018H (06:03) Statusbyte
Bit 0 gesetzt: Shorthand, F-Tasten und Editor eingeschaltet
rückges.: alles ausgeschaltet
Bit 1 gesetzt: Druckformatierung ein
rückges.: Graphikmodus
Bit 2 gesetzt: serielle Schnittstelle
rückges.: Parallelschnittstelle
4028H (07:48) Länge einer Seite (72 Zeilen)
4029H (08:00) aktuelle Zeilennummer
402AH (09:41) Länge einer logischen Seite (65 Zeilen)
402BH (0A:50) Zeichen pro Zeile (80)
402CH (0B:05) Zeichen einrücken bei Zeilenumbruch (5)

Wenn Sie einen breiteren Drucker haben oder andere Zeilenabstände oder Schriftarten einstellen wollen, sollten Sie auch die entsprechenden Parameter ändern, oder den Treiber mit <CTRL> <G> von der Tastatur aus in den Graphikmodus setzen, sonst stimmt mit Sicherheit das Seitenformat nicht mehr.

Einige Tips zum Schluß:

G-DOS bietet die Möglichkeit, die Adressen für Tastatur-, Monitor- und Druckertreiber mit dem INFO-Befehl auf die Systemvorgaben zu setzen. Solange Sie nur den Monitor- oder Druckertreiber abgeschaltet haben, lassen sich die KEY-Routinen durch <CTRL> <R> wieder starten.

Der nächste Tip ist ein sich selbstveränderndes BASIC-Programm, das die Eigenschaft ausnutzt, daß KEY ganze Zeichenketten von der Tastatur aus eingeben kann, ohne daß eine Taste gedrückt wird. Der Zeiger dafür steht in 401AH, die Anzahl der Zeichen in 4019H. Der Text wird übergeben, sobald der Cursor wieder eingeschaltet wird.

```
10 REM ***** KEY-DEMO *****
20 CLEAR 300
30 A$="1000 REM --- NEUE ZEILE ---"
40 A$=A$+CHR$(13)+"LIST"+CHR$(13)
50 REM Statt LIST geht auch RUN oder GOTO ...
60 VP=VARPTR(A$)
70 FOR I=0 TO 2
80 POKE &H4019+I,PEEK(VP+I)
90 NEXT
100 END'Schaltet den Cursor wieder ein !
```

Da sich die Funktionen Editor, Shorthand und Funktionstasten abschalten lassen, ist KEY ein wertvolles Hilfsmittel bei der Programmierung kommerzieller Programme, die die Funktionstasten zur Steuerung der Programmfunktionen benutzen wollen. Wie oben schon einmal gesagt, erzeugen diese Tasten spezielle Codes, die sich leicht abfragen lassen. Außerdem sind die Wiederholfunktion des Numerikblocks und die einwandfreie Entprellung der Tastatur in kommerziellen Anwendungen wichtig.

Damit die Abschaltung schon mit dem Start des Programms aktiv ist, müssen Sie im File KEY/CMD das Bit 0 des Bytes 06 (Statusbyte) auf Null setzen. Da in Anwendungsprogrammen die Formatierung des Formulars normalerweise durch das Programm geschieht, sollten Sie auch den Druckertreiber abschalten (Bit 1 des Statusbytes = 0) oder Seiten- und Zeilenlänge (Bytes 09 und 0A) auf FF (255 dezimal) setzen. Diese Änderungen sollten im File und nicht erst nach dem Start durch POKE-Befehle gemacht werden, damit mit <CTRL> <R> nicht die Einstellungen zurückgesetzt werden können.

HINWEIS: Folgende Tastenkombinationen sind bei der Verwendung von KEY in Verbindung mit Genie IIIs geändert:

| | | |
|--------|-------------------------|-------------------------|
| CTRL R | entspricht | CTRL CLEAR |
| CTRL P | entspricht | CTRL ↑ (Pfeil aufwärts) |
| CTRL L | entspricht | CTRL ↓ (Pfeil abwärts) |
| CTRL G | entspricht | CTRL → (Pfeil rechts) |
| CTRL S | ist nicht implementiert | |

Das Hilfsprogramm ASM/CMD

=====

Bei dem Programm ASM/CMD handelt es sich um einen Disk-Editor-Assembler, mit dem es möglich ist, Assemblerprogramme in mnemonischer Schreibweise zu programmieren und zu editieren. Die erstellten Quellprogramme (Source Programme) können assembliert (Quellcode in Objectcode) und anschließend wahlweise auf Diskette oder Kassette (Genie I/II) abgespeichert werden. Während der Assemblierung erkennt ASM/CMD evtl. vorhandene Syntaxfehler und zeigt diese auf dem Schirm an.

Bei der Eingabe des Sourceprogramms in mnemonischer Schreibweise haben Sie sich an folgende Regeln zu halten:

Jede Zeile beginnt mit einer 6-stelligen Zeilennummer. Für Label und Opcodes werden ausschließlich Großbuchstaben akzeptiert. Ein Label darf nicht mit einer Ziffer beginnen und besteht aus maximal 6 Zeichen (A-Z, X, O, U, S, 0-9) wobei Registerbezeichnungen sind als Label untersagt sind. Bemerkungen werden durch ein Semikolon gekennzeichnet. Label, Opcodes und Bemerkungen sind durch ein TAB (09H), von der Tastatur durch Druck auf den Pfeil nach rechts erreichbar, oder mindestens einem Leerzeichen zu trennen.

Neben den gültigen Befehlen für den Z80 (gemäß ZILOG) sind noch nachfolgende Befehle als Sourcetext möglich:

| | | | | |
|-------|-----------|------|-------------|-----------------------------|
| 00100 | *LIST OFF | | | ;kein Listing ab hier |
| 00999 | *LIST ON | | | ;ab hier wieder listen |
| 00200 | | ORG | 5200H | ;Ladeadresse für Objectcode |
| 00210 | BEGINN: | EQU | \$ | ;\$ steht für Position |
| 09999 | | END | BEGINN | ;weist Startadresse zu |
| 00300 | | SLL | A | ;ergibt 0CBH 37H |
| 00310 | | SLL | (IY+0BH) | ;ebenfalls möglich und von |
| 00320 | | | | ;ZILOG nicht dokumentiert |
| 00400 | | DEFW | 1024 | ;ergibt 00H 04H |
| 00410 | | DW | -1 | ;wie DEFW |
| 00420 | | DEFB | '3' | ;ergibt 033H |
| 00430 | | DB | 0DH | ;wie DEFB nur ein Zeichen |
| 00440 | | DEFM | 'Textfolge' | ;Zeichenketten |
| 00500 | ADDIT | EQU | 15+0C0H | ;ergibt 0CFH |
| 00510 | SUBTR | EQU | ADDIT-4 | ;ergibt 0CFH-4=0CBH |
| 00520 | LOGAND | EQU | 8FH&SUBTR | ;ergibt 2-Byte: 0008BH |
| 00530 | LSHIFT | EQU | LOGAND>4 | ;ergibt 008B0H 4 Bits links |
| 00540 | RSHIFT | EQU | LSHIFT>-1 | ;ergibt 00458H 1 Bit rechts |
| 00600 | LABEL | DEFL | BEGINN+100 | ;mehrfach verwendbar |

Die Befehle des ASM/CMD:

A (Assemble)
=====

SYNTAX: A Dateiname/Optional/Option2...

Das Kommando führt zur Assemblierung der Quellprogramme im Textbuffer des Rechners, wobei die Eingabe des Dateinamens lediglich bei der Verwendung eines Kassettenrecorders notwendig ist. Unter diesem Namen, der eine Länge von max. 6 Buchstaben haben darf, wird später die Objectdatei auf der Kassette abgespeichert. Gibt man keinen Dateinamen an und speichert trotzdem auf Kassette ab, erhält die Datei automatisch den Namen ASSEM3.

Als Optionen sind folgende Abkürzungen möglich:

- NL (No Listing), das Programm wird nicht auf dem Schirm gelistet
- NO (No Objectcode), das Programm wird nicht assembliert, sondern nur auf Fehler untersucht
- NS (No Symbol), es wird keine Symboltabelle erstellt
- LP (Line Printer), Listing, Fehlermeldungen und Symboltabelle werden auf dem Drucker ausgegeben.
- WE (Wait Error), wenn ein Fehler während der Assemblierung erkannt wird, hält das Programm an und gibt eine Fehlermeldung aus. Durch Druck auf eine beliebige Taste, wird die Assemblierung bis zum Ende bzw. zum nächsten Fehler fortgesetzt. Durch Betätigung der C-Taste, nach einer Fehlermeldung, wird die Option WE abgeschaltet und bis zum Programmende assembliert. Ein Druck auf die BREAK-Taste hat in jedem Falle einen Rücksprung in die Kommandoebene des ASM/CMD zur Folge.

Nach Assemblierung eines fehlerfreien Sourcecodes erfolgt die Frage ob die Objectdatei auf Disk oder Tape geschrieben werden soll. Gibt man die Disk an, muß ein Dateiname angegeben werden, dessen Aufbau den üblichen Vorschriften unterliegt.

Falls Sie eine Ausgabe der Objectdatei auf Kassette wünschen, drücken Sie nach dem Erscheinen der Meldung "Start CASSETTE" eine beliebige Taste und Datenausgabe auf dem Rekorder beginnt. Die Objectdatei erhält dabei den zu Beginn angegebenen Namen.

B (Bye)
=====

SYNTAX: B

Die Eingabe des Buchstaben B in der Kommandoebene des ASM/CMD bewirkt einen Rücksprung in die DOS-Befehlsebene.

D (Delete)

=====

SYNTAX: D Zeilennummer1:Zeilennummer2

Nach Eingabe dieses Befehls wird der durch Zeilennummer1 und Zeilennummer2 angegebene Bereich des Textes im Quellbuffer gelöscht.

Gibt man nur eine Nummer an, so wird nur diese Zeile gelöscht.

SYNTAX: D.

Löscht die aktuelle (gerade bearbeitete oder angezeigte) Zeile. Der Punkt muß nicht unbedingt angegeben werden.

SYNTAX: D#:*

Löscht den gesamten Text im Puffer und führt ein Clear aus.

E (Edit)

=====

SYNTAX: E Zeilennummer

Dieser Befehl erlaubt es Ihnen eine Zeile im Textbuffer zu editieren oder modifizieren.

Innerhalb des Editkommandos sind weitere Unterkommandos möglich.

A (Again)

Der Cursor springt wieder an den Zeilenanfang und alle zuvor durchgeführten Veränderungen der Zeile werden wieder rückgängig gemacht.

nC (Change)

Rechts von der momentanen Cursorposition an werden n Zeichen verändert. Bei der Eingabe eines C ohne n wird nur ein Zeichen verändert.

nD (Delete)

Rechts von der momentanen Cursorposition an werden n Zeichen gelöscht. Bei der Eingabe eines D ohne n wird nur ein Zeichen gelöscht.

E (End)

Dieses Kommando beendet die Editierung und bringt den Rechner in die Kommandoebene des ASM/CMD zurück. Alle zuvor ausgeführten Änderungen werden in der editierten Zeile eingetragen.

H (Hack)

Das H-Kommando bedeutet für den Editor, daß ab der momentanen Cursorposition die Zeile gelöscht und anschließend in den Eingabemodus gesprungen werden soll.

Sie sollten das H-Kommando nicht dazu benutzen, um eine Zeile komplett zu löschen. Es muß nach Ausführung des Kommandos mindestens noch ein Zeichen als Zeileninhalt vorhanden sein. Ansonsten kann es bei der späteren Ausführung des Programms Probleme geben.

I (Insert)

Mit Hilfe dieses Subkommandos können Sie an der momentanen Cursorposition weitere Zeichen einsetzen, ohne daß die anderen Teile der Zeile verändert werden müssen.

nKc (Kill)

Das Kommando löscht ab Cursorposition alle Zeichen bis zum n-ten Vorkommen des Zeichens c und setzt den Cursor dann an diese Stelle.

Wenn n nicht angegeben wird, sucht der Rechner das erste Zeichen c, löscht die Zeile bis dorthin und setzt den Cursor an diese Position.

Wenn das Zeichen c in der Zeile nicht vorhanden ist, wird ab Cursorposition bis zum Ende der Zeile alles gelöscht.

L (List)

Listet die Zeile auf dem Bildschirm und bringt den Cursor wieder an den Zeilenanfang.

Q (Quit)

Dieses Kommando bringt den Rechner dazu, wieder in die Kommandoebene des ASM/CMD zu springen, ohne die evtl. in der Zeile ausgeführten Änderungen zu übernehmen.

nSc (Search)

Das Kommando sucht in der Zeile ab Cursorposition nach dem n-ten Vorkommen des Zeichens c und setzt den Cursor an diese Position. Wird n nicht angegeben, sucht der Rechner nach dem ersten auftretenden Zeichen c.

Wenn das Zeichen c nicht gefunden wird, springt der Cursor an das Ende der Zeile.

X

Der Cursor springt an das Ende der Zeile und befindet sich anschließend im Eingabemodus.

Backspace (Pfeil nach links)

Bei Betätigung dieser Taste wird der Cursor zurück (nach links) bewegt.

Wenn sich der Editor im Eingabemodus befindet werden dabei die mit Backspace überschriebenen Zeichen gelöscht.

SHIFT-Hochpfeil

Wenn diese Tasten gleichzeitig betätigt werden, verläßt der Rechner den Eingabemodus, befindet sich jedoch weiter in in der Editierungsebene. Die Cursorposition bleibt unverändert.

NEW LINE

Mit NEW LINE wird eine editierte Zeile mit ihren Änderungen in den Textspeicher übernommen. (Wirkung wie bei Subkommando E)

nSPACE (Leertaste)

Bei jeder Betätigung der SPACE-Taste wird ein weiteres Zeichen ab Cursorposition in der editierten Zeile angezeigt. Wenn Sie eine Zahl n vor dem Druck auf die Leertaste eingeben, werden n Zeichen der gerade editierten Zeile ausgegeben.

F (Find)

SYNTAX: F Zeichenkette

Mit diesem Befehl haben Sie die Möglichkeit ab aktueller Zeilennummer innerhalb des Quellprogramms nach einer Zeichenkette suchen zu lassen, die eine Länge von maximal 16 Zeichen haben kann.

Wenn Sie keinen String angeben, benutzt das Programm die zuletzt bei einem F-Befehl eingegebene Zeichenkette.

Beim Antreffen des angegebenen Strings im Programmtext wird die entsprechende Zeile angezeigt.

Für den Fall, daß die Zeichenkette nicht im Programm enthalten ist, erfolgt die Meldung "nichts gefunden".

H (Hardcopy)

SYNTAX: H Zeilennummer1:Zeilennummer2

Der durch die Zeilennummer1 und Zeilennummer2 definierte Teil des Quellprogramms wird mit dem H-Befehl auf einem angeschlossenen Drucker ausgegeben.

Wenn Sie nur die Ausgabe einer bestimmten Zeile wünschen, genügt die Angabe der Zeilennummer1.

Vor Beginn der jeweiligen Ausgabe auf dem Drucker fragt ASM/CMD "Drucker okay?". Nach Betätigung einer beliebigen Taste beginnt dann die Ausgabe.

SYNTAX: H.

Die Ausgabe der aktuellen Zeile erfolgt nach Eingabe dieses Befehls.

SYNTAX: H

Diese Anweisung bringt 15 Zeilen, beginnend mit der aktuellen, auf dem Drucker zur Ausgabe.

SYNTAX: H†:*

Der gesamte Inhalt des Textpuffers wird auf dem Drucker gelistet.

I (Insert)

SYNTAX: I Zeilennummer,Zeilenabstand

Das I-Kommando wird dazu benutzt, um weitere Textzeilen in das Quellprogramm einzufügen. Wenn mehrere Zeilen hintereinander eingesetzt werden sollen, kann man noch den Zeilenabstand angeben. Sollte zwischen den bereits vorhandenen Zeilen des Quellprogramms kein Platz für weitere Eingaben sein, die durch die Zeilennummer angegebene Zeile bereits existieren oder die nächste Zeilennummer größer als 65529 sein, so erscheint die Meldung "Zeilenabstand zu kurz".

Wenn der Zeilenabstand nicht angegeben wird, übernimmt das Programm den diesbezüglich zuletzt eingegebenen Wert.

Um aus dem I-Modus herauszukommen, ohne daß der evtl. bereits geschriebene Text der Zeile in den Quellpuffer übernommen wird, ist die BREAK-Taste zu drücken.

LT (Load Tape)

=====

SYNTAX: LT=Dateiname

Die Datei mit dem angegebenen Namen, der eine Länge von maximal 6 Buchstaben haben darf, wird auf Grund dieses Kommandos von der Kassette eines angeschlossenen Recorders eingelesen.

Sollte noch Text im Quellpuffer vorhanden sein, so weist das Programm durch die Meldung "TEXT IM PUFFER" darauf hin und fragt anschließend "Wollen Sie den Sourcecode erweitern?(J/N)". Wenn Sie mit N antworten, wird der im Puffer vorhandene Text gelöscht. Sodann erfolgt der Hinweis "Start CASSETTE".

Durch Druck auf eine beliebige Taste startet nun der Recorder und die angegebene Datei wird geladen.

Antworten Sie dagegen mit J, bleibt der Inhalt des Textpuffers bestehen und wird durch die angegebene Datei erweitert. Dabei ist darauf zu achten, daß sich keine Überschneidungen bezüglich der Zeilennumerierung ergeben. Abhilfe können Sie in diesem Fall mit Hilfe des N-Kommandos schaffen.

Während ein File von Kassette geladen wird, erscheint der bekannte blinkende Stern in der oberen rechten Ecke des Bildschirms.

Es sei an dieser Stelle ausdrücklich darauf hingewiesen, daß mit Hilfe des L-Kommandos ausschließlich Sourceprogramme (also Dateien im Quellcode) von der Kassette in den Speicher geladen werden können.

LD (Load Diskfile)

=====

SYNTAX: LD=Dateiname

Dieses Kommando dient dazu, Sourcefiles von der Diskette einzuladen. Der dabei anzugebende Dateiname unterliegt in seiner Zusammensetzung den üblichen Vorschriften.

Sollte zum Zeitpunkt dieses Kommandoaufrufs noch Text im Quellpuffer sein, so erfolgt wie beim Laden eines Files von Kassette der Hinweis "TEXT IM PUFFER" und anschließend die Frage "Wollen Sie den Sourcecode erweitern?(J/N)". Diese Frage ist wie unter dem LT-Kommando bereits beschrieben zu beantworten.

Wenn sich keine Datei unter dem angegebenen Namen auf der Diskette befindet, erscheint die Meldung "Datei nicht im Inhaltsverzeichnis" und ASM/CMD erwartet erneute Befehle.

Mit Hilfe des LD-Kommandos lassen sich ausschließlich Sourceprogramme von der Diskette in den Speicher des Rechners einladen.

N (Number)

SYNTAX: N Startnummer,Abstand

Dieses Kommando dient dazu einen im Puffer befindlichen Source-code neu zu numerieren. Das kann z.B. notwendig werden, wenn zwei Files über die Kommandos L oder LD aneinandergesetzt werden sollen und sich ohne Umnummerierung eine Überschneidung der Zeilennummern ergeben würde.

Wenn keine Startnummer angegeben wird, ist die erste Nummer die 00100. Alle weiteren Zeilen werden entweder im angegebenen Abstand oder bei fehlender Angabe im aktuellen Abstand umnummeriert.

P (Print)

SYNTAX: P Zeilennummer1:Zeilennummer2

Der durch die Zeilennummer1 und Zeilennummer 2 definierte Teil des Quellprogramms wird mit dem P-Befehl auf dem Bildschirm ausgegeben.

Wenn Sie nur die Ausgabe einer bestimmten Zeile wünschen, genügt die Angabe der Zeilennummer1.

SYNTAX: P.

Nach Eingabe dieses Befehls erfolgt die Ausgabe der aktuellen Zeile.

SYNTAX: P

Diese Anweisung bringt 15 Zeilen, bei der aktuellen beginnend, auf dem Bildschirm zur Anzeige.

SYNTAX: P#:*

Der gesamte Inhalt des Textpuffers wird auf dem Bildschirm ausgegeben.

R (Replace)

SYNTAX: R Zeilennummer,Zeilenabstand

Das R-Kommando ist in seiner Wirkung dem I-Kommando sehr ähnlich. Soll eine im Textpuffer vorhandene Zeile durch eine neue Zeile ersetzt werden, so ist deren Zeilennummer anzugeben. Der Zeileninhalt wird gelöscht und steht für neue Eingaben bereit. Wenn die angegebene Zeilennummer im Textpuffer noch nicht vorhanden ist, verhält sich das R-Kommando wie eine I-Kommando. Ist kein Zeilenabstand definiert, übernimmt der Rechner den des letzten I-, R- oder N-Kommandos.

T (Type)

=====

SYNTAX: T Zeilennummer1:Zeilennummer2

Das T-Kommando hat im Prinzip die gleiche Funktion wie das H-Kommando, indem es den Rechner dazu veranlaßt, den Textpuffer auf einem angeschlossenen Drucker auszugeben.

Der einzige Unterschied besteht darin, daß die Zeilennummern nicht mit ausgegeben werden (siehe auch H-Kommando).

Auch hier fragt ASM/CMD zunächst "Drucker okay?" und beginnt mit der Ausgabe erst nach Betätigung einer beliebigen Taste.

WT (Write Tape)

=====

SYNTAX: WT=Dateiname

Die Datei mit dem angegebenen Namen, der eine Länge von maximal 6 Buchstaben haben darf, wird auf Grund dieses Kommandos auf die Kassette eines angeschlossenen Rekorders geschrieben. Die Abspeicherung beginnt nicht unmittelbar nach dem WT-Kommando, sondern erst nach Betätigung einer beliebigen Taste als Antwort auf den Hinweis "Start CASSETTE".

Mit Hilfe des W-Kommandos können ausschließlich Sourceprogramme (also Dateien im Quellcode) auf der Kassette abgelegt werden.

WD (Write Diskfile)

=====

SYNTAX: WD=Dateiname

Dieses Kommando dient dazu, Sourcefiles auf einer Diskette abzulegen. Der Dateiname unterliegt in seiner Zusammensetzung den üblichen Vorschriften.

Sollte sich auf der Diskette noch kein File unter dem angegebenen Namen befinden, erscheint folgende Meldung auf dem Schirm:

=> neue Datei:

zur Eröffnung CREATE mit <C> aufrufen

Drücken Sie nun die C-Taste und der Inhalt des Textpuffers wird unter dem angegebenen Namen als neue Datei auf der Disk angelegt.

Sollte bereits eine Datei mit dem angegebenen Namen auf der Disk vorliegen, so weist der Rechner darauf hin und fragt, ob die Datei überschrieben werden soll.

Druckertreiberprogramme ITOH/CMD und STAR/CMD

=====

Diese beiden Programme ermöglichen den Ausdruck von Bildschirm-blockgrafik auf den Itoh-Druckern EG-3085 und EG-3100 bzw. auf den Star-Druckern DP510, Gemini und Delta.

Die Programme werden automatisch vor den normalen Druckertreiber "geschaltet" und setzen die Zeichen von ASC 128 bis ASC 191 in Steuerzeichen und Daten um, die die Drucker veranlassen, die gewünschten Zeichen mittels Ihrer höchauflösenden Grafik auszugeben. —

Ladevorgang der Programme:

In der DOS-Befehlsebene geben Sie einfach

ITO/CMD bzw. STAR/CMD

ein und drücken die NEW LINE-Taste.

Die Programme relocieren sich selbstständig unter die bestehende HIMEM-Grenze, die vorher evtl. aus dem DOS, BASIC oder von Programmen gesetzt wurde.

Dieser Vorgang zerstört ein evtl. im ungeschützten Speicher vorhandenes BASIC- oder Maschinenprogramm. Programme oberhalb von HIMEM bleiben unbeeinflusst.

Die Druckertreiberprogramme schützen sich selbst (vor BASIC) durch Verkleinern der HIMEM-Grenze.

Oberhalb von HIMEM darf natürlich nicht gePOKEd werden, da dies eine Zerstörung des Druckertreibers zur Folge hätte.

Nach dem Programmstart erfolgt eine entsprechende Ausgabe auf dem Bildschirm und anschließend die Rückkehr in die DOS-Befehlsebene.

insf 1305

S 30-1

=====

Dieses Programm dient dazu, den im Genie enthaltenen CRTC vom Typ HD46505 zu programmieren. CRTC ist die Abkürzung für Cathode Ray Tube Controller. Darunter hat man ein hochintegriertes Bauteil zu verstehen, das sämtliche Bildschirmausgaben des Genie III organisiert. Dabei ist die Ausgabeform des Bildes auf dem Monitor in großem Maße frei programmierbar.

Es ist z.B. möglich das gesamte Bild in horizontaler und vertikaler Richtung zu verschieben oder die Anzahl der angezeigten Zeilen und deren Länge (Zeichen pro Zeile) zu ändern. Ferner ist die Vertikal- und Horizontalsynchronisation des Bildes über den CRTC leicht verstellbar.

Um die Ausgabeform des Bildes auf dem Monitor zu variieren, muß der Inhalt der Register 0 bis 7 des CRTC entsprechend verändert werden. Diese Register haben dabei folgende Bedeutung:

Horizontal Total Register (R0)

Das Register R0 enthält die Gesamtzahl der Zeichen in einer Zeile, einschließlich der nicht dargestellten im Randbereich und beeinflusst damit auch die Zeilen- und Bildfrequenz. Änderungen dieses Registerinhalts können zum Verlust der Synchronisation führen.

Horizontal Displayed Register (R1)

Dieses Register definiert die auf dem Schirm angezeigten Zeichen pro Zeile.

Horizontal Sync Position Register (R2)

Durch Programmierung dieses Register wird die horizontale Synchronisationsposition definiert. Dadurch wird es möglich das Bild horizontal zu versetzen. Wenn der Wert des Registers erhöht wird, verschiebt sich das Bild nach links. Bei seiner Verringerung verschiebt es sich dementsprechend nach rechts.

Sync Width Register (R3)

Mit dem Register R3 läßt sich die Länge des Zeilensynchronimpulses programmieren. Dabei wird die horizontale Impulslänge mit den unteren 4 Bit und die vertikale Impulslänge mit den oberen 4 Bit programmiert. Änderungen sind hier nur bei Verwendung spezieller Datensichtgeräte sinnvoll.

Vertical Total Register (R4)

Durch Programmierung des Registers R4 definiert man die vertikale Gesamtzahl der Textzeilen, einschließlich der nicht genutzten im Randbereich. Änderungen beeinflussen hier die Bildfrequenz und können zum Synchronisationsverlust führen.

Vertical Total Adjust Register (R5)

Dieses Register dient dem Feinabgleich der Bildfrequenz durch Hinzufügen einzelner (bis 31) Elektronenstrahlzeilen zum festgelegten Raster.

Vertical Displayed Register (R6)

Dieses Register wird dazu benötigt, um die Anzahl der angezeigten Zeilen auf dem Bildschirm programmieren zu können.

Vertical Sync Position Register (R7)

Durch Programmierung dieses Registers wird die vertikale Synchronisationsposition definiert. Dadurch wird es möglich, das Bild vertikal zu versetzen. Wenn der Wert des Registers erhöht wird, verschiebt sich das Bild nach oben. Bei seiner Verringerung verschiebt es sich dementsprechend nach unten.

Um mit VIDEO/BAS zu arbeiten, geben Sie innerhalb der BASIC-Ebene RUN"VIDEO/BAS" ein. Das Programm meldet sich auf dem Bildschirm mit folgender Tabelle:

Programm zur Voreinstellung der Monitorparameter
TCS 3.Mai 1984 Vers. 1.3

| Register | -C- | ##,N | ##,X | ##,V | ##,H | ##,S |
|----------------|-----|------|------|------|------|------|
| 0 Gesamtbreite | 98 | 98 | 99 | 78 | 95 | 104 |
| 1 Spalten | 64 | 64 | 80 | 64 | 64 | 64 |
| 2 Hor.-Sync | 82 | 82 | 82 | 61 | 74 | 78 |
| 3 Sync.-Weiten | 66 | 66 | 46 | 35 | 75 | 34 |
| 4 Ges.-Höhe | 22 | 22 | 24 | 35 | 25 | 25 |
| 5 Vert.-Adj. | 5 | 5 | 0 | 2 | 12 | 0 |
| 6 Zeilen | 16 | 16 | 24 | 32 | 16 | 24 |
| 7 Vert.-Sync. | 19 | 19 | 24 | 35 | 20 | 24 |

Welche Einstellung ändern (C,X,V,H,S) (E=Ende)

Die innerhalb dieser Tabelle sichtbaren Einstellungen des Videocontrollers können vom DOS aus mit Hilfe des ##-Kommandos aufgerufen werden (siehe auch DOS-Befehl ##).

Wenn Sie eine Einstellung zur Änderung gewählt haben, läßt sich nach Angabe der Registernummer durch Tastendruck auf ";" und "-" der Registerinhalt jeweils um 1 erhöhen bzw. verringern. Der Inhalt des jeweiligen Registers wird dann rechts neben der Tabelle angezeigt.

Mit der @-Taste haben Sie die Möglichkeit die Register zu wechseln und E beendet eine Änderung.

Direkt sichtbar werden Änderungen der Werte unter C, da sie sich unmittelbar auf den Controller beziehen.

Nach Abschluß der Änderungen durch E erfolgt die Frage, ob die Werte auf die Diskette eingetragen werden sollen. Wenn Sie die Einstellung von C verändert haben und die Frage nach dem Eintrag auf Diskette mit J beantworten, werden die Registerinhalte von C nach ##,N übernommen.

Um die Auswirkungen von verschiedenen Registerinhalten des CRTC auf die Bildschirmausgabe besser kennenzulernen, sollten Sie einmal die Werte unter C verändern, ohne sie zurückzuschreiben.

System-Files

=====

Auf einer G-DOS Diskette befinden sich außer den mit Hilfe von "DIR 0" oder "I 0" angezeigten Programmen noch eine Vielzahl unsichtbarer Dateien. Diese unsichtbaren System-Module beinhalten das gesamte DOS. Die sogenannten SYS-Files werden mit

DIR 0 /SYS oder I 0 /SYS

angezeigt.

G-DOS-Module:

GDOS/SYS oder BOOT/SYS:

In diesem File sind alle für den Systemstart notwendigen Daten enthalten. Beim Einschalten des Gerätes oder Betätigung von RESET wird Sektor 0 von GDOS/SYS vom ROM geladen und initialisiert die weitere Speicherbelegung durch das DOS. Da einige Bootstrap-Loader nicht in der Lage sind den Sektor 0 als Startsektor zu lesen, sind Sektor 0 und 1 identisch.

INHALT/SYS oder DIR/SYS:

Dieser File enthält sämtliche Informationen über die auf der Diskette befindlichen Files und wird wie GDOS/SYS bereits beim Formatieren einer neuen Diskette eingetragen.

SYS0/SYS:

Hierbei handelt es es sich um das einzige residente Modul. Alle anderen SYS-Files werden nur bei Bedarf eingeladen. SYS0/SYS besorgt die Ein- und Ausgabe auf der Floppy, sowie das Laden weiterer Files.

Bei der Initialisierung wertet SYS0/SYS mit Hilfe eines kurzzeitig eingelesenen Initialisierungsteils die Systemparameter und Laufwerksdefinitionen aus. Ferner wird mit seiner Hilfe das G-DOS-Logo auf dem Bildschirm ausgegeben und ein evtl. vorhandener AUTO-Start ausgeführt.

In SYS0/SYS befindet sich die RST28-Routine zum Nachladen anderer Systemmodule und die RST38-Routine zur Interruptsteuerung (Funktion 'Uhr' und Autorepeat-Funktion der Tasten).

Die Systemeinsprünge \$PRINT, \$LPRINT, \$MULT, \$DIV und \$DELIM werden von SYS0/SYS bewerkstelligt.

SYS1/SYS:

Mit Hilfe dieses Files werden die DOS-Kommandos erkannt und die Befehlseingabe neu initialisiert. SYS1/SYS beinhaltet die Einsprungstellen der LIB-Befehle 'R', 'LIB', '/', ';' und 'CLS'. Von den Systemeinsprünge werden \$DOS, \$DOSCMD, \$EXFILE, \$DEFEXT und \$DOSCAL durch SYS1/SYS ausgeführt.

SYS2/SYS:

Seine Funktion besteht darin, Files zu generieren oder zu öffnen, Platz für Dateien zu belegen, Einträge ins Inhaltsverzeichnis vorzunehmen, Kennworte zu verarbeiten, Anwenderprogramme zu laden und zu starten, sowie die Befehle 'LOAD' und 'N' zu bearbeiten.

Ferner liegen die Systemeinsprünge für \$INIT, \$OPEN, \$LOAD und \$RUN in SYS2/SYS.

SYS3/SYS:

Dieses Modul beinhaltet die LIB-Befehle 'KILL', 'JKL', 'BL', 'UHR', '&', 'V+', 'BREAK' und 'LC'. SYS3/SYS schließt und löscht Files und enthält den größten Teil der PURGE-Funktion. Außerdem liegen die Systemeinsprünge für \$CLOSE, \$KILL, \$INTENQ, und \$WREOF in SYS3/SYS.

SYS4/SYS:

SYS4/SYS enthält keinerlei LIB-Befehle, dafür aber die Texte aller DOS-Fehlermeldungen. Demzufolge liegt auch der Einsprung für \$DERROR in diesem Modul.

SYS5/SYS:

Da SYS5/SYS den DEBUG-Monitor beinhaltet, führt auch der Systemaufruf von \$DEBUG in dieses Modul. LIB-Befehle sind in SYS5/SYS nicht enthalten.

SYS6/SYS:

In diesem Modul werden die LIB-Befehle 'NDF', 'COPY' und 'APPEND' bearbeitet.

SYS7/SYS:

Die LIB-Befehle 'ZEIT', 'DATUM', 'AUTO', 'ATTRIB', 'PROT', 'DUMP' und 'HIMEM' kommen mit SYS7/SYS zur Ausführung. Ferner werden 'S', 'PD' und 'PURGE' von SYS7/SYS initialisiert.

SYS8/SYS:

Dieses Modul beinhaltet die Bearbeitung der LIB-Kommandos 'I', ('DIR') und 'FREE'.

SYS9/SYS:

Mittels SYS9/SYS erfolgt die Auswertung der LIB-Befehle 'M>', '!', 'DO', 'CONT', 'B2', 'PAUSE', 'STMT' und 'BOOT'.

Weiterhin liegen die Systemeinsprünge von \$NAMENQ und \$NAMDEQ und die Erkennung von Tastatureingaben, die zum Abbruch einer JOB-Dateibearbeitung führen können, innerhalb dieses Moduls.

SYS14/SYS:

Diese Systemmodul hat die Funktion die LIB-Befehle 'O', 'CREATE', 'E', 'LIST', 'PRINT', und '@' zu bearbeiten.

SYS15/SYS:

SYS15/SYS hat den Disk-Daten-Editor 'DDE' zum Inhalt.

SYS16/SYS:

Der größte Teil des PD-Kommandos liegt innerhalb dieses Systemmoduls.

SYS17/SYS:

Dieser File wertet die LIB-Befehle 'AIK' und 'S' aus.

SYS23/SYS:

Dieses Modul enthält die LIB-Befehle '##', '64' und '80'.

SYS24/SYS:

Innerhalb dieses Moduls liegen beim G-DOS 2.1a die Routinen des V24-Befehls und bei G-DOS 2.1c die des SIO-Befehls.

SYS25/SYS:

Das Modul SYS25/SYS dient zur Ausführung des Befehls 'F#'.

SYS26/SYS:

Mit Hilfe dieses System-Files erfolgt die Umschaltung auf das Bildschirmformat 64*16.

SYS27/SYS:

Mit Hilfe dieses System-Files erfolgt die Umschaltung auf das Bildschirmformat 80*24.

SYS28/SYS:

Dieses Modul hat die Aufgabe, die beiden LIB-Befehle 'DR' und 'FORM' zu bearbeiten.

SYS29/SYS:

Das Modul SYS29/SYS führt die Befehle 'INFO', 'PORT' und 'DISK' aus.

Disk-BASIC-Module:

BASIC/CMD:

Dieses, für die Arbeit mit Disk-BASIC unentbehrliche Modul, beinhaltet beim Genie I/II und IIs die Ergänzungen des ROM-BASIC und beim Genie III, IIIs und Speedmaster das gesamte BASIC, da bei diesen Rechnern bekanntlich nur ein Boot-ROM zum Starten des Diskettenlaufwerks enthalten ist. BASIC/CMD ist ein residentes Programm und muß sich nicht unbedingt auf der Systemdiskette befinden. Es kann also auch, wie ein normales Programm, auf der Datendiskette abgelegt sein.

SYS10/SYS:

Dieses Modul hat die Funktion die Disk-BASIC-Befehle 'OPEN', 'GET' und 'PUT' zu bearbeiten.

SYS11/SYS:

Wenn ein BASIC-Programm mit Hilfe des Befehls 'RENUM' umnummeriert werden soll, so ist SYS11/SYS notwendig.

SYS12/SYS:

Dieses Modul dient zur Erstellung einer Reference-Liste über den Befehl 'REF'.

SYS13/SYS:

Über SYS13/SYS erfolgt die Ausgabe von BASIC-Fehlermeldungen.

SYS18/SYS:

Direkte Kommandos werden mit Hilfe dieses Files bearbeitet.

SYS19/SYS:

Die BASIC-Befehle 'LOAD', 'RUN', 'MERGE', 'SAVE' und 'CMD" F DELETE' kommen über dieses Modul zur Ausführung.

SYS20/SYS:

Hier werden verschiedene BASIC-Statement bearbeitet (Runtime-Interpreter). Dazu befindet sich dieses Modul normalerweise beim Ablauf eines BASIC-Programms immer im Speicher des Rechners.

SYS21/SYS:

Dieses System-Modul bearbeitet das Kommando 'CMD" O".

Die System-Module SYS24/SYS bis SYS27/SYS sind nur beim G-DOS 2.1a und 2.1c, also bei Verwendung der Rechner Genie III und IIIs belegt.

Da von den BASIC-Modulen nur BASIC/CMD resident ist, kann es natürlich vorkommen, daß bei Start eines BASIC-Programms, das Laufwerk 0 nochmals anläuft, um sich den entsprechend benötigten SYS-File nachzuladen. Das bedeutet, daß sich unter Disk-BASIC immer eine G-DOS-Diskette mit allen Disk-BASIC-Modulen im Laufwerk 0 befinden sollte.

Weitere Hinweise zu den System-Modulen:

Es ist natürlich möglich einen großen Teil dieser SYS-Files unter Verzicht auf die darin enthaltenen Befehle zu löschen. Sie können sich auf diese Art ein Minimalbetriebssystem schaffen, mit dem es beispielsweise nur noch möglich ist CMD-Files zu starten.

Auf einer solchen Minimalsystemdiskette müssen jedoch in jedem Falle die System-Module

- GDOS/SYS
- INHALT/SYS
- SYS0/SYS
- SYS1/SYS
- SYS2/SYS
- SYS3/SYS
- SYS4/SYS (wenn ein DOS-Fehler noch ausgegeben werden soll)
- SYS8/SYS (wenn das Inhaltsverzeichnis gelesen werden soll)

enthalten sein.

Wenn Sie Benutzer eines einzelnen Diskettenlaufwerks sind und sich deshalb bekanntlich auf jeder Diskette ein DOS befinden muß, können Sie auf diese Art einiges an Diskettenspeicherplatz einsparen.

Zur Erstellung einer Minimalbetriebssystemdiskette muß zunächst eine normale Kopie einer G-DOS-Diskette geschaffen werden. Anschließend werden alle nicht benötigten SYS-Files auf der Kopie gelöscht.

Wichtig:

Wenn einmal ein Systemfile auf einer Diskette gelöscht wurde, so läßt er sich nicht einfach von einer anderen Systemdiskette zurückkopieren, da der Eintrag eines SYS-Files an einer für ihn spezifischen Stelle im Inhaltsverzeichnis erfolgen muß.

Struktur des Inhaltsverzeichnisses

=====

Das Inhaltsverzeichnis liegt beim G-DOS in der Mitte einer Diskette (z.B. auf Spur 20 bei einer 40 Spur Diskette SS/SD), damit die Zugriffszeit so kurz wie möglich bleibt. Die Länge des Inhaltsverzeichnisses ist unterschiedlich und hängt von der Art ab, wie die Diskette beschrieben wird (Schreibdichte etc.). Wie die gesamte Diskette ist natürlich auch das Inhaltsverzeichnis in einzelne Sektoren aufgegliedert. Diesen Sektoren kommen jedoch völlig unterschiedliche Bedeutungen zu.

1. GAT-Sektor (Granule Allocation Table)

In diesem Sektor, es ist der erste des Inhaltsverzeichnisses, finden wir den Plan für die Speicherbelegung, d.h. der Sektor wird vom DOS benötigt, um den Dateien ihren Speicherplatz zuweisen zu können.

Im Genie-DOS kann eine Diskette maximal in 192 Blöcke zu je 8 Einheiten (Granules) mit 5 Sektoren zu je 256 Bytes aufgeteilt werden. Daraus folgt, daß mit dem Inhaltsverzeichnis maximal $192 \cdot 8 \cdot 5 \cdot 256$ KByte = 1920 KByte organisiert werden könnten.

Im GAT-Sektor sind weiterhin ein evtl. vorhandenes Kennwort (Byte 00CEH - 00CFH), der Name der Diskette (Byte 00D0H 00D7H), das Datum der Diskette (Byte 00D8H - 00DFH) und ein evtl. eingetragener und durch RESET über das Kommando "AUTO" auszuführender Befehl zu finden (Byte 00E0H - 00FFH). Sollte kein Autostart vorgesehen sein, finden Sie nur ein Return (0DH).

2. HIT-Sektor (Hash Index Table)

In diesem zweiten Sektor des Inhaltsverzeichnisses finden Sie den sogenannten Hashcode eines jeden eingetragenen Dateinamens. Es handelt sich dabei um die Verschlüsselung von Dateinamen in Codes von jeweils einem Byte.

Alle Einträge ungleich Null innerhalb dieses Sektors entsprechen einem Dateinamen.

Die Tabelle ist in 8 Reihen zu jeweils 32 Bytes aufgebaut und jedes Byte innerhalb einer Reihe entspricht dem Sektor des Eintrags der Datei im Inhaltsverzeichnis.

Da die ersten beiden Sektoren für GAT- und HIT-Table benutzt werden, bleiben die letzten beiden Bytes einer Reihe frei.

(Dies folgt aus der Tatsache, daß das Inhaltsverzeichnis immer eine durch 5 teilbare Sektorenzahl haben muß.)

Lediglich in der ersten Reihe wird innerhalb des letzten Bytes die Anzahl der die Mindestgröße des Inhaltsverzeichnisses überschreitenden Sektoren abgespeichert. (Die Mindestgröße ist auf 10 Sektoren, 2 Einheiten festgelegt.)

Über die Hash-Codes in der Tabelle werden die Einträge der Dateinamen im Inhaltsverzeichnis sehr schnell gefunden. Beim Aufruf eines Programms oder einer Datei findet zunächst eine Umwandlung des Namens in den entsprechenden Hash-Code statt. Anschließend wird dieser im HIT-Sektor gesucht. Da verschiedene Namen den gleichen Hash-Code besitzen können, findet beim Antreffen eines übereinstimmenden Codes mit den echten Einträgen der Dateinamen im Inhaltsverzeichnis eine Überprüfung statt.

Dabei ergibt sich aus der relativen Position des Hash-Codes im Sektor ein Zeiger auf die entsprechende Datei, (DEC = Directory Entry Code), der wiederum bei allen folgenden Dateizugriffen als Verbindung zwischen File Control Block (FCB) und Inhaltsverzeichnis dient.

Sollte der Dateiname, auf den der Zeiger weist, nicht mit dem angegeben übereinstimmen, wird nach dem nächsten übereinstimmenden Hash-Code gesucht. Aus diesem Grunde kann der Aufruf einer nicht im Inhaltsverzeichnis stehenden Datei relativ viel Zeit in Anspruch nehmen bis eine entsprechende Fehlermeldung auf dem Bildschirm erscheint.

3. FDE-Sektoren (File Directory Entries)

Außer dem ersten (GAT-Table) und zweiten (HIT-Table) Sektor handelt es sich bei allen weiteren Sektoren des Inhaltsverzeichnisses um FDE-Sektoren.

Diese Sektoren sind wieder in je 8 Reihen mit einer Länge von jeweils 32 Bytes aufgeteilt.

Die Belegung eines FDEs erkennt man am Inhalt des Bits 4. Wenn in Bit 4 eine Null steht, ist der FDE frei.

Durch KILL oder PURGE werden nur dieses Bit 4 des FDE, sowie die Einträge im HIT- und GAT-Sektor gelöscht. Derart gelöschte Files können also in jedem Fall wieder zum Leben erweckt werden.

Wir unterscheiden bei den Einträgen in den FDE-Sektoren zwischen FPDE- und FXDE-Dateien. Wenn Bit 7 eines FDE-Eintrags gleich Null ist, handelt es sich um den Ersteintrag eines Files (FPDE = File Primary Directory Entry). Falls Bit 7 gleich Eins ist, liegt dagegen ein Erweiterungseintrag eines Files vor (FXDE = File Extension Directory Entry).

4. Datei-Ersteintrag (FPDE = File Primary Directory Entry)

Diesen Bytes kommt im einzelnen folgende Bedeutung zu:

| | | |
|---------|----------|--|
| Byte 00 | Bit 7=0 | entspricht einem Ersteintrag |
| | Bit 7=1 | entspricht einem Erweiterungseintrag |
| | Bit 6=1 | zeigt Systemfile an |
| | Bit 5 | keine Bedeutung |
| | Bit 4=1 | wenn ein Eintrag besteht |
| | Bit 3=1 | zeigt unsichtbaren File an |
| Byte 01 | Bits 2-0 | geben die Zugriffsstufe auf die Datei an (siehe "ATTRIB") |
| | Bit 7=0 | der Datei kann bei Bedarf mehr Platz zugeordnet werden |
| | Bit 7=1 | sperrt die Dateiplatz-Freigabe |
| | Bit 6=0 | Dateierweiterung ist möglich |
| | Bit 6=1 | Dateierweiterung ist gesperrt |
| | Bit 5=1 | zeigt an, daß die Datei bereits einmal bearbeitet wurde |
| | Bits 4-0 | keine Bedeutung oder Tag des Datums bei Dateieröffnung (G-DOS 3.0) |

- Byte 02 Bits 7-4 keine Bedeutung oder Jahr des Datums bei Dateieröffnung (G-DOS 3.0)
Bits 3-0 keine Bedeutung oder Monat des Datums bei Dateieröffnung (G-DOS 3.0)
- Byte 03 Es handelt sich um das EOF-Byte und kann Werte zwischen 0 und 255 annehmen. Der Wert Null entspricht dabei einer EOF-Position von 256.
- Byte 04 Innerhalb dieses Bytes ist die logische Satzlänge zu finden. Diese dient zur Berechnung der Anzahl der Sätze einer Datei und kann Werte zwischen 0 und 255 annehmen. Null entspricht einer Satzlänge von 256 Bytes. Bei Dateizugriffen wird der Wert nicht benutzt.
- Bytes 05-0C Diese 8 Bytes bieten dem Namen der Datei Platz. Der Name wird linksbündig eingetragen und evtl. durch Leerzeichen auf der rechten Seite ergänzt.
- Bytes 0D-0F Aus diesen 3 Bytes geht der Dateityp hervor. Wie der Name, so wird auch der Dateityp linksbündig eingetragen und evtl. mit Leerzeichen auf der rechten Seite aufgefüllt.
- Bytes 10-11 Diese beiden Bytes enthalten den Code des Hauptkennwortes, wobei dem aus maximal 8 Zeichen langen Kennwort das Doppelbyte zugeordnet wird.
- Bytes 12-13 Hier liegt der Code des Bearbeitungskennwortes, dem ebenso aus den maximal 8 möglichen Zeichen dieses Doppelbyte zugeordnet wird.
- Bytes 14-15 Es handelt sich um die relative Sektornummer des letzten Sektors der Datei. Dies ist der Sektor, der das EOF-Byte enthält. Beachten Sie, daß einer Datei erst dann ein Sektor zugeordnet wird, wenn das EOF-Byte größer als Null ist.
Die Bytes 14-15 werden nur beim Neueintrag einer Datei oder durch ein CLOSE verändert. Es kann also bei der Ausführung eines Resets und geöffneten Dateien die Diskette unlesbar werden, da der EOF-Eintrag nicht ordnungsgemäß verändert wurde und noch den alten Inhalt enthält.
- Byte 16 Das Byte gibt bei Dateierweiterungen den Block des Speicherbelegungsplans an, der die noch folgenden Daten beinhaltet. Dabei können Werte zwischen 0 und 191 auftreten. Sollte das Byte 16 gleich FF sein, so bedeutet dies, daß keinerlei Erweiterungen folgen.

- Byte 17 Bits 7-5 Diese Bits geben die Einheit innerhalb des durch Byte 16 definierten Blocks bei einer Dateierweiterung an.
 Bits 4-0 Aus diesen Bits geht die Anzahl der Einheiten hervor, die der ersten Einheit unmittelbar folgen und noch zur Datei gehören.
- Bytes 18-1D jeweils wie Bytes 16-17
- Byte 1E Dieses Byte enthält eine Markierung aus der hervorgeht, ob für den File ein Erweiterungseintrag vorliegt. FE bedeutet, daß das Byte 1F auf den Erweiterungseintrag zeigt. FF sagt dagegen aus, daß kein Erweiterungseintrag vorhanden ist.
- Byte 1F Bits 7-5 Reihe, in der der zum File gehörende Erweiterungseintrag zu finden ist.
 Bits 4-0 Erweiterungseintrag enthaltende logische Sektornummer minus 2 im Inhaltsverzeichnis.

5. Datei-Erweiterungseintrag (FXDE = File extended Directory Entry)

Der FXDE wird immer dann benötigt, wenn eine Datei mehr als vier unterschiedliche Bereiche auf einer Diskette benutzt. Im FXDE sind dann die Adressen der Bereiche, die größer als vier sind, abgelegt.

Der FXDE ist wie folgt aufgebaut:

- Byte 00 Bit 7=1 Dadurch erfolgt die Kennzeichnung eines Erweiterungseintrags (siehe auch FPDE Byte 00).
 Bit 6 Null
 Bit 5 Null
 Bit 4=1 Daraus geht hervor, daß der FXDE belegt ist (siehe auch FPDE Byte 00).
 Bit 3-0 Null
- Byte 01 In diesem Byte finden wir einen Rückverweis auf den vorangegangenen FPDE oder FXDE.
- Bytes 02-15 Null, unbenutzt
- Bytes 16-1F Definition wie bei FPDE

6. Datensteuerblock FCB (File Control Block)

Beim FCB handelt es sich um einen 32 Byte langen Block im RAM, durch den erst die Ein/Ausgabe über Diskette möglich ist. Bei jeder Schreib- oder Leseoperation muß über den FCB eine Verbindung zwischen dem Programm und dem DOS hergestellt werden.

In dem Moment, wo im DOS durch eine OPEN die Verbindung mit einer Datei hergestellt wird, bekommt der Rechner alle weiteren, für die Datenübertragung wichtigen Informationen aus dem FCB (Positionierung, Filenamen, Adressen, um nur einige Aufgaben zu nennen).

Im folgenden Abschnitt soll nun die Bedeutung der 32 Bytes des FCB näher erläutert werden. Da einige Bytes in ihrer Bedeutung mit den entsprechenden Bytes des FDE übereinstimmen, sollen sie an dieser Stelle nicht nochmals erklärt werden.

- Byte 00 Bit 7=1 Markierung, daß Datei geöffnet ist
=0 Markierung, daß Datei geschlossen ist
Bits 6-2 nicht benutzt
Bit 1 Ein- und Ausgabe auf der Diskette ohne Dateieintrag
Bit 0=1 Die Sektoren des File werden lesegeschützt. Dabei wird die Adressmarke des Inhaltsverzeichnisses geschrieben (DAM).
- Byte 01 Bit 7=1 Die Übertragung der Daten erfolgt byteweise.
=0 Die Übertragung der Daten erfolgt sektorweise in Blöcken von jeweils 256 Bytes.
Bit 6=1 Der Eintrag des EOF-Wertes wird auf der Diskette nur bei verändertem EOF-vollzogen.
=0 Bei jedem Schreibzyklus auf der Diskette wird auch der EOF-Wert neu eingetragen.
Bit 5=1 Der Sektor wurde noch nicht gelesen
=0 Der Sektor wurde bereits gelesen
Bit 4=1 Die im Puffer stehenden veränderten Daten wurden noch nicht auf die Disk übertragen. Ein CLOSE hat die Übertragung des Datenpuffers auf die Diskette zur Folge.
=0 Im Datenpuffer wurden keinerlei Veränderungen vorgenommen.
Bit 3=1 Die Länge des FCBs ist damit auf 32 Bytes festgelegt.
Bits 2-0 Hierin ist die Zugriffstufe auf den File festgelegt (Werte von 0 bis 7).
- Byte 02 Bits 7-5 Gleiche Bedeutung wie Byte 01 des FPDEs.
Bits 4-0 unbenutzt oder Tag des Datums (G-DOS 3.0)
- Bytes 03-04 Diese Bytes enthalten die Adresse des Puffers für den FCB-Sektor. Der Puffer dient zur Zwischenspeicherung der Daten des FCB-Sektors.
- Byte 05 Hier finden wir die relative Schreib-/Lese-position innerhalb eines Sektors, d.h. das Byte zeigt bei der Ein- bzw. Ausgabe von einzelnen Bytes auf das nächste zu bearbeitende Zeichen.
- Byte 06 Bits 7-3 nicht benutzt
Bits 2-0 Nummer des Laufwerks beim OPEN
- Byte 07 In diesem Byte finden wir den Dateieintragszeiger (DEC), der auf den entsprechenden Code im HIT-Sektor weist. Es handelt sich also um die Verbindung zwischen FCB und den Einträgen im Inhaltsverzeichnis.
- Byte 08 EOF-Byte

Byte 09 Das Byte enthält die logische Satzlänge und wird zur Berechnung der Satzanzahl einer Datei herangezogen, wobei die Null einer Satzlänge von 256 Bytes entspricht.

Bytes 0A-0B In diesem beiden Bytes steht, welches Byte bei der nächsten Ein- oder Ausgabe von Daten bearbeitet wird. Es handelt sich um die RBA des jeweiligen Files.

Bytes 0C-0D Die beiden Bytes bilden die RBA der EOF-Position des Files.

Bytes 0E-1F siehe Bytes 16-1F des FPDE

Systemeinsprungtabelle

=====

| Adresse | Bezeichnung | Beschreibung der Routine |
|---------|-------------|---|
| 402DH | \$RET | Rückkehr zum DOS ohne Fehleranzeige |
| 4030H | \$EXIT | Fehlerausgang zum DOS ohne Fehlermeldung |
| 4400H | \$DOS | kehrt zur Befehlseingabe zurück |
| 4403H | MENTRY | 2-Byte Einsprungsadresse nach \$LOAD |
| 4405H | \$DOSCMD | ruft Maschinenprog. oder DOS-Befehl auf |
| 4408H | \$NERROR | Fehlermeldung, wenn NZ-Status |
| 4409H | \$DERROR | Fehlermeldung ausgeben |
| 440DH | \$DEBUG | Systemmonitor aufrufen (DEBUG) |
| 4410H | \$ENQUE | Interruptroutine einfügen |
| 4413H | \$DEQUE | Interruptroutine herausnehmen |
| 4416H | \$RESEL | wählt aktuelles Laufwerk erneut an |
| 4419H | \$DOSCAL | wie \$DOSCMD, mit Rückkehr zum Programm |
| 441CH | \$EXFIL | prüft und überträgt Dateinamen in FCB |
| 441FH | MGDOS0 | systemintern reserviertes Merkbyte |
| 4420H | \$INIT | initialisiert Datei und eröffnet FCB |
| 4423H | MGDOS1 | systemintern reserviertes Merkbyte |
| 4424H | \$OPEN | eröffnet FCB für existierende Datei |
| 4427H | MGDOS2 | systemintern reserviertes Merkbyte |
| 4428H | \$CLOSE | schließt FCB |
| 442BH | MGDOS3 | systemintern reserviertes Merkbyte |
| 442CH | \$KILL | löscht Dateieintrag |
| 442FH | MGDOS4 | systemintern reserviertes Merkbyte |
| 4430H | \$LOAD | lädt Maschinenprogramm ohne Ausführung |
| 4433H | \$RUN | lädt und startet ein Maschinenprogramm |
| 4436H | \$RDSEC | liest einen Sektor |
| 4439H | \$WRSEC | schreibt einen Sektor |
| 443CH | \$WRSECV | schreibt einen Sektor und prüft Inhalt |
| 443FH | \$POS0 | setzt FCB auf Satznummer Null |
| 4442H | \$POSBC | setzt FCB auf Satznummer aus Reg. (bc) |
| 4445H | \$POSDEC | decrementiert FCB-Sektornummer |
| 4448H | \$POSEOF | positioniert FCB auf Dateiende |
| 444BH | \$ALLOC | reserviert für Datei Platz a. d. Diskette |
| 444EH | \$POSRBA | positioniert FCB a. rel. Byte-Adresse |
| 4451H | \$WREOF | Dateiende ins Inhaltsverzeichnis eintr. |
| 4454H | \$DELIM | prüft Delimiter bei Parametern |
| 4457H | MGDOS5 | systemintern reserviertes Merkbyte |
| 4458H | \$FILLER | vergleicht Füllwort/Parameter |
| 445BH | \$DRVSEL | selektiert gewünschtes Laufwerk |
| 445EH | \$DSKMNT | prüft ob Diskette in Laufwerk ist |
| 4461H | \$NAMENQ | erweitert Befehlsvorrat um gelad. Routine |
| 4464H | \$NAMDEQ | löscht Befehl aus erweitertem Befehlssatz |
| 4467H | \$PRINT | Text auf dem Monitor ausgeben |
| 446AH | \$LPRINT | Text auf dem Drucker ausgeben |
| 446DH | \$CONTIM | Uhrzeit in das Format hh:mm:ss umwandeln |
| 4470H | \$CONDAT | Datum in das Format tt.mm.jj umwandeln |
| 4473H | \$DEFEXT | ergänzt Typbezeichnung |
| 447FH | MGDOS6 | systemintern reserviertes Merkbyte |
| 4480H | DOSFCB | systeminterner FCB von 32 Bytes Länge |

Nur für G-DOS 3.0:

| | | |
|-------|---------|---|
| 4475H | \$MULT | multipliziert A*HL= AHL |
| 4479H | \$DIV | dividiert HL/A=HL Rest A |
| 447CH | \$HEXDE | erzeugt hexadezimale Darstellung von DE |

Beschreibung der Restart-Befehle

Die Restart-Befehle (RST) sind Unterroutinen in Z80-Maschinensprache (ähnlich wie CALL-Routinen). Ihr Vorteil liegt in ihrem geringen Speicherplatzbedarf. Während CALL-Befehle 3 Bytes für ihre Ausführung beanspruchen, kommt ein RST-Befehl mit einem Byte aus.

Im G-DOS sind folgende Restarts als Unterroutine belegt:

- RST 00 - springt auf Speicherstelle 0 und entspricht einem Kaltstart
- RST 08 - Syntaxcheck
Die durch HL adressierte Speicherstelle wird mit dem Byte, das dem RST 08 folgt, verglichen. Sind beide gleich, wird RST 10 angesprungen, ansonsten der Syntax-Error.
- RST 10 - lädt die durch HL+1 adressierte Speicherstelle in den Akku, ohne Blanks und Linefeeds zu berücksichtigen. Bei Ziffern ist das Carry-Flag, bei \$:\$ oder 00 das Zero-Flag gesetzt
- RST 18 - vergleicht das HL mit dem DE-Register.
HL größer DE Zero-Flag=0; Carry-Flag=0
HL gleich DE Zero-Flag=1; Carry-Flag=0
HL kleiner DE Zero-Flag=0; Carry-Flag=1
- RST 20 - testet den Typ des X-Registerinhaltes in der Speicherstelle 40APH
Integer : C=1, P=1, S=1
Single : C=1
Double : P=1
String : Z=1, C=1, P=1

(Z=Zero-, C=Carry-, P=Parity-, S=Sign-Flag)
- RST 28 - wird benötigt, um ein Systemmodul zu laden. Dabei steht im Akku der Lade-code. RST 28 kommt auch beim Drücken der BREAK-Taste zur Ausführung.
- RST 30 - ist zum Aufruf eines DEBUG-Programms notwendig. Nach Beendigung des DEBUG-Programms erfolgt die Rückkehr.
- RST 38 - diese Routine wird alle 25 Millisekunden ausgeführt, sobald sich die Z80-CPU im Interruptmode 1 (IM 1) befindet

Fehlercode-Tabelle:

Die hier aufgeführte Fehlercodetabelle gilt für alle G-DOS-Versionen.

| Code | Fehlerbeschreibung |
|-----------|---|
| Dez. Hex. | |
| 1 00 | kein Fehler |
| 2 01 | schlechte Dateidaten |
| 3 02 | Suchfehler beim Lesen |
| 3 03 | verlorene Daten beim Lesen |
| 4 04 | Prüfzahlfehler beim Lesen |
| 5 05 | Daten-Satz beim Lesen nicht gefunden |
| 6 06 | Leseversuch auf markierten Satz |
| 7 07 | Leseversuch auf System-Satz |
| 8 08 | Bauteil nicht erreichbar |
| 9 09 | Fehlercode undefiniert |
| 10 0A | Suchfehler beim Schreiben |
| 11 0B | verlorene Daten beim Schreiben |
| 12 0C | Prüfzahlfehler beim Schreiben |
| 13 0D | Daten-Satz beim Schreiben nicht gefunden |
| 14 0E | Schreibfehler auf Diskettenlaufwerk |
| 15 0F | Diskette schreibgeschützt |
| 16 10 | Peripherie nicht erreichbar |
| 17 11 | Lesefehler Inhaltsverzeichnis |
| 18 12 | Schreibfehler Inhaltsverzeichnis |
| 19 13 | Dateiname unerlaubt |
| 20 14 | Spurnummer zu hoch |
| 21 15 | Funktion unter DOS-Call unerlaubt |
| 22 16 | Fehlercode undefiniert |
| 23 17 | Fehlercode undefiniert |
| 24 18 | Datei nicht in Inhaltsverzeichnis |
| 25 19 | Datei verwehrt Zugriff |
| 26 1A | Inhaltsverzeichnis voll |
| 27 1B | Diskette voll |
| 28 1C | Ende der Datei angetroffen |
| 29 1D | hinter Ende der Datei |
| 30 1E | Inhaltsverz. voll, kann Datei nicht erweitern |
| 31 1F | Programm nicht gefunden |
| 32 20 | unzulässiges oder fehlendes Laufwerk |
| 33 21 | kein Bauteilplatz erreichbar |
| 34 22 | Formfehler beim Laden |
| 35 23 | Speicherplatz defekt |
| 36 24 | Ladeversuch auf ROM |
| 37 25 | Ladeversuch - Zugriff verwehrt |
| 38 26 | Datei nicht offen |
| 39 27 | unzulässige Initialisierungsdaten auf System-Disk |
| 40 28 | unzulässige Spurnanzahl |
| 41 29 | unzulässige logische Dateinummer |
| 42 2A | unzulässige DOS-Funktion |
| 43 2B | unzulässige Funktion unter Verkettung |
| 44 2C | Inhaltsverzeichnis nicht korrekt |
| 45 2D | schlechte FCB-Daten |
| 46 2E | System-Programm nicht gefunden |
| 47 2F | schlechte Parameter |

| | | |
|----|----|--|
| 48 | 30 | kein Dateiname |
| 49 | 31 | Diskette falscher Satz-Typ |
| 50 | 32 | Lesefehler BOOT |
| 51 | 33 | Fataler DOS-Fehler |
| 52 | 34 | Syntax oder Trennzeichen oder Endzeichen unerlaubt |
| 53 | 35 | Datei existiert schon |
| 54 | 36 | Befehl zu lang |
| 55 | 37 | Diskette verwehrt Zugriff |
| 56 | 38 | Keine Mini-DOS-Funktion |
| 57 | 39 | Erzwungene Beendigung der Funktion |
| 58 | 3A | Abweichung bei Vergleich |
| 59 | 3B | unzureichender Speicherplatz |
| 60 | 3C | nichtkompatible Laufwerke oder Disketten |
| 61 | 3D | ADE=N Attribut - kann Datei nicht erweitern |
| 62 | 3E | Kann Datei beim Lesen nicht erweitern |

Vergleichsliste: Genie-DOS <-----> NEWDOS80.2

=====

Mini-Befehlseingabe:

MINI-NEWDOS/80 READY

M>
;
/
!

MDCOPY
MDRET
MDBORT

Befehlseingabe:

NEWDOS/80 READY

& (J,N) -
? bzw. LIB
I bzw. DIR (I,S,A,B,/typ)
> bzw. COPY (QPDN=,ZPDN=,KDWA,
QKW=,NZKW=,ZZND,AZN=,QN=,
BZN,NZN=,BZD,SQD,IVU,SBIV=,
AEIV=,EDK,AZKW=,FRD,BEA,NVD,
IDL=,XDL=,FRAG, ---
O (VON=,BIS=,HIMEM=)
@ (KEINE,MO,TA,DR,ST=,NL)
AIK
ATTRIB (BKW=,HKW=,BEA=,PROT=,
KEIN,START,LESEN,ANDERN,
NAME,KILL,NULL)
B2
BL (J,N)
CREATE (ANZ=,ADE=,ADF=,LOG=)
CONT (J,N,D)
DATUM (tt.mm.jj)
E
LC

N
NDF (SPUR=,STOP=,MAG)
PD (SP=,SEK=,SWZ=,EIB=,
SBIV=,AEIV=)
PROT (DATUM=,BKL,ZU,AUF)
S
UHR
V+ (J,N)
ZEIT (hh:mm:ss)

DEBUG (Y,N)
LIB
DIR (I,S,A,U,/ext)
COPY (SPDN=,DPDN=,NDMW,
SPW=,NDPW=,DDND, ODN=,SN=,
KDN,NDN=,KDD,USD,BDU,DDSL=,
DDGA=,CBF,ODPW=,USR,UPD,DFO,
ILP=,XLP=,CFWO,UBB
CLEAR (START=,END=,MEM=)
ROUTE (CLEAR,DO,KI,PR,MM=,NL)
WRDIRP
ATTRIB (ACC=,UPD=,UDF=,PROT=,
LOCK,EXEC,READ,WRITE,
(RE)NAME,KILL,FULL)
BASIC2
BLINK (Y,N)
CREATE (REC=,ASE=,ASC=,LRL=)
CHNON (Y,N,D)
DATE (MM/DD/YY)
ERROR
LCDVR
LC
RENAME
FORMAT (PFST=,PFTC=,RWF)
PDRIVE (TC=,SPT=,TSR=,GPL=,
DDSL=,DDGA=)
PROT (DATE=,RUF,LOCK,UNLOCK)
SYSTEM
CLOCK
VERIFY (Y,N)
TIME (hh:mm:ss)

• nur bei Genie-DOS:

nur bei NEWDOS80.2:

DISK
DR
FORM
INFO
LF
PORT
V24
Z
F#

80
64
DDE

FORMS
SETCOM

Register:
 =====

| | | | |
|---------------------------|----------------|---------------------------|---------------|
| I | 193 | CMD"Dosbefehl" | 71,249 |
| ## | 190 | CMD"E" | 72,231 |
| E | 188 | CMD"F",DELETE | 74,232 |
| EH | 228 | CMD"F-ERASE" | 75,239 |
| EO | 229 | CMD"F-KEEP" | 75,233 |
| / | 193 | CMD"F-POPN" | 75,235 |
| 0 | 181 | CMD"F-POPR" | 75,236 |
| 123 | 191 | CMD"F-POPS" | 74,237 |
| 64 | 183 | CMD"F-SASZ" | 75,238 |
| 64-Zeichen-Format | 183,190 | CMD"F-SS" | 76,240 |
| 80 | 184 | CMD"F-SWAP" | 76,234 |
| 80-Zeichen-Format | 184,190 | CMD"J" | 73,241 |
| : | 193 | CMD"O" | 72,242 |
| > | 32,187 | CMD"R" | 40,73,246 |
| ? | 38,189 | CMD"S" | 74,247 |
| # | 185 | CMD"S=Dosbefehl" | 74 |
| | | CMD"T" | 40,73,248 |
| ADE | 119 | CMD-Kommando | 71,230 |
| ADF | 119 | CONT | 102 |
| AEIV | 78,118,155,158 | COPY | 31,105 |
| AIK | 86 | CPU | 83 |
| ANZ | 119 | CREATE | 119 |
| APPEND | 87 | CRTC | 325 |
| ASCII-Format | 42,213 | CVD | 68,277 |
| ASM/CMD | 316 | CVI | 68,275 |
| ATTRIB | 88 | CVI | 275 |
| AUP | 162 | CVS | 68,276 |
| AUTO | 45,91 | Contoller-Typ | 156 |
| AUTO-Befehl | 217 | Cursor | 94,183,310 |
| AUTO-Start | 45 | | |
| AZRW | 118 | DATUM | 40,120,162 |
| AZN | 114 | ODE | 98 |
| Abkürzungen | 216 | DEBUG | 188,191,194 |
| Abspeichern | 42,213 | DEF FN | 52,204 |
| Adressmarke korrigieren | 303 | DEPUSR | 54,205 |
| Adressmarke | 86 | DELETE | 74,232 |
| Arbeitskopien | 79 | DFG | 191 |
| Arbeitsspeicher | 19 | DI | 218 |
| Aufzeichnung | 18 | DIR | 36,121 |
| Ausnullen | 181 | DIR/SYS | 327 |
| | | DIRCHECK/CMD | 303 |
| B2 | 92 | DISK | 124 |
| BASIC | 202 | DO | 126,307 |
| BASIC * | 46 | DOS | 29 |
| BASIC-Interpreter | 83 | DOS-Befehle | 38,81 |
| BASIC-Prompt | 47 | DOS-Befehlsebene | 85,247 |
| BASIC/CMD | 200,330 | DOS-Fehler | 231 |
| BEA | 117 | DR | 128,185 |
| BKL | 162 | DTR | 171 |
| BL | 94 | DU | 218 |
| BOOT | 95 | DUMP | 129 |
| BOOT/SYS | 327 | Datei schließen | 254,280 |
| BREAK | 96 | Datei-Ersteintrag | 333 |
| BZD | 114 | Datei-Erweiterungseintrag | 335 |
| BZN | 114 | Dateien | 55,250 |
| Baudrate | 170,176 | Dateiende | 260,282 |
| bearbeitete Dateien | 117 | Dateieröffnung | 254 |
| Bearbeitungskennzeichen | 88,90,162 | Dateiliste | 117 |
| Befehlsfolgen | 91 | Dateilänge | 281 |
| Befehlsliste | 38,147 | Dateiname | 42,55,105,250 |
| Befehls wiederholung | 164 | Dateistrukturen | 284 |
| besondere Dateistrukturen | 284 | Dateitypbezeichnungen | 55 |
| Betriebssystem | 83 | Dateitypen | 285 |
| Bildschirm-Kontrollcodes | 183 | Dateiumbenennung | 150 |
| Bildschirmausgaben | 325 | Dateiunterschiede | 284 |
| Bildschirminhalt | 140 | Daten lesen | 258,279 |
| Bildschirmmeldungen | 173 | Datenaufzeichnung | 20 |
| Bildschirmparameter | 190 | Datenausgabe | 255,278 |
| Bildschirmpositionen | 184 | Datenblöcke | 56 |
| Blockgrafik | 177,179,309 | Datensatz | 56 |
| Blocklänge | 64 | Datensteuerblock | 335 |
| Booten | 27 | Datenverifizierung | 175 |
| Byteadresse | 292 | Datenzuweisung | 268,270 |
| | | Datenübertragung | 292 |
| CLOSE | 57,254,280 | Datum | 40,214 |
| CLS | 101 | Datumkonvertierung | 241 |
| CMD"C" | 71,230 | Direkt-Zugriff-Datei | 55,64,266 |

| | | | |
|----------------------------|-------------------------|----------------------------|------------------------------|
| Direkte Sortierung | 242 | Fixed-Item-Files | 205 |
| Disk-BASIC | 25, 29, 85, 199 | Flieskoma-Variable | 67 |
| Disk-BASIC-Module | 330 | Floppy-Controller | 84, 86 |
| Disk-Editor-Assembler | 316 | Floppy-Disk | 20 |
| Disketten-Daten-Editor | 90 | Floppy-Laufwerk | 19 |
| Diskettenbehandlung | 23 | Floppybetrieb | 17 |
| Diskettenformate | 124 | Formatieren | 34, 113, 117, 151 |
| Diskettenlaufwerke | 84 | Formatierungsmuster | 152 |
| Diskettenwechsel | 113 | Frei formatiert | 291 |
| Diskettenzugriff | 99 | Freie Dateien | 116 |
| Doppelseitige Aufzeichnung | 21 | Freier Inhalt | 291 |
| Doppelte Genauigkeit | 274, 277 | Freier Speicherplatz | 132 |
| Doppelte Schreibdichte | 21 | Funktionen | 204 |
| Doppeltgenaue-Variable | 67 | Funktionstasten | 135, 311 |
| Double Sided | 21 | | |
| Double-Step | 156 | G-DOS | 25 |
| Drehzahl | 79 | G-DOS-Befehle | 189 |
| Drucker | 310 | G-DOS-Module | 327 |
| Druckerausgabe | 128, 140, 161, 170, 185 | GAT-Sektor | 332 |
| Druckerpositionierung | 145 | GDOS/SYS | 327 |
| Druckersteuerzeichen | 133 | GET | 66, 279, 292 |
| Druckertreiber | 313, 324 | GOSUB-Ebenen | 74 |
| Dummy-Variable | 54 | Ganzzahl-Variable | 67 |
| Duplizieren | 48, 218 | Grafikausdruck | 324 |
| | | Granule Allocation Table | 332 |
| E | 131 | | |
| EDIT-Befehl | 218 | HIMEM | 136, 181, 202 |
| EDK | 116 | HIT-Sektor | 332 |
| EIB | 78, 155, 158 | Hardware | 19 |
| EOP | 56, 58, 63, 260, 282 | Hash Index Table | 332 |
| ERROR | 206 | Hexadezimalumwandlung | 228 |
| Echtzeituhr | 178 | | |
| Editieren | 47, 217 | I | 36, 138 |
| Editor | 312 | IDL | 117 |
| Einfache Genauigkeit | 273, 276 | IGEL | 285, 296 |
| Einfache Schreibdichte | 21 | IGEL-Ausdrücke | 297 |
| Einfügen | 48, 218 | IGEL-Zeilenummer | 297 |
| Einheiten | 78 | INFO | 139 |
| Einheiten im Block | 158 | INHALT/SYS | 327 |
| Einheiten im Inhaltsverz. | 78, 158 | INP-Signale | 160 |
| Einschalteinweise | 27 | INPUT# | 58, 60, 258 |
| Einseitige Aufzeichnung | 21 | INSTR | 52, 207 |
| Einzel-Datei-Kopie | 116, 192 | ITALIC-Schrift | 179 |
| Einzeilschrittausführung | 194, 240 | ITOH/CMD | 324 |
| End Of File | 260, 282 | IVU | 115 |
| Expansioninterface | 86 | Indexloch | 20, 79 |
| Extension | 250 | Indeziffern | 55 |
| | | Indirekte Sortierung | 242 |
| F# | 135 | Inhaltsverzeichnis | 36, 115, 118, 121, 137, 303 |
| FC-Adapter | 26 | Inhaltsverzeichnis säubern | 303 |
| FCB | 335 | Inhaltsverzeichnisstruktur | 332 |
| FDE-Sektoren | 333 | Initialisierungsdaten | 165 |
| FF-Files | 285, 291 | Initialisierungsstatus | 139 |
| FI-Files | 285, 291 | Integer | 275 |
| FIELD | 64, 268 | Integervariable | 272 |
| FLOPPY5/8 | 26 | Interfacetyp | 77 |
| FMT | 113, 117 | Interruptkette | 246, 248 |
| FN | 52 | Inverse Schrift | 177, 179 |
| FOR-NEXT-Schleifen | 74, 235 | Item Group Expression List | 285, 296 |
| FORM | 133 | | |
| FORMLIST/BAS | 263 | J | 113 |
| FPDE | 333 | JKL | 140, 191 |
| FRAG | 116 | JOB-Datei | 102, 126, 154, 173, 193, 307 |
| FRD | 116 | JOB-Unterbrechung | 102 |
| FREE | 38, 132 | JOB/CMD | 307 |
| FXDE | 335 | | |
| Fehler | 131 | KDMA | 113 |
| Fehlercode-Tabelle | 206, 338 | KEY/CMD | 310 |
| Fehlerdiagnose | 51 | KILL | 39, 142, 208 |
| Fehlermeldungen | 201, 206 | KW | 162 |
| Fehlersuche | 227 | Kanal | 170 |
| Feldaufteilung | 64, 268 | Kassettenbefehle | 170 |
| Fettschrift | 179 | Kassettenbetrieb | 40 |
| Field-Item-Datei | 266 | Kassettenprogramme | 305 |
| Fielding | 64 | Kassettenrekorder | 17, 84, 248 |
| File Control Block | 335 | Kennwort | 88, 90, 114, 118, 162, 250 |
| File Directory Entries | 333 | Kleinschrift | 144 |
| File Extended Direc. Entry | 335 | Kontrolldaten | 162 |
| File Primary Direc. Entry | 333 | Kopfpositionierung | 157 |
| File-Anzahl | 202 | Kopieren | 31, 187, 192 |
| Files | 55, 255 | Kurzbefehle | 311 |
| Pilotypen | 289 | | |

| | |
|------------------------|-------------------|
| LC | 144 |
| LF | 145 |
| LIB | 38,147 |
| LINE INPUT | 53,209 |
| LINE INPUT# | 58,62,259 |
| LIST | 148 |
| LIST64/BAS | 265 |
| LOAD | 43,149,210 |
| LOC(x) | 70,283,298 |
| LOP(x) | 69,281 |
| LOG | 119 |
| LSET | 65,270 |
| Laden | 43,210 |
| Laufwerknummer | 251 |
| Laufwerktyp | 77,157 |
| Length Of File | 281 |
| Level II BASIC | 25,27,29,92,201 |
| Listen | 47,216 |
| Literaturhinweise | 343 |
| Löschen | 39,48,142,163,208 |
| | |
| M> | 192 |
| MAG | 151 |
| MERGE | 44,219,222 |
| MF-Files | 286,290 |
| MI-Files | 286,290 |
| MIDS | 53,211 |
| MKDS | 67,274 |
| MKIS | 67,272 |
| MKSS | 67,273 |
| MO | 185 |
| MU-Files | 286,289 |
| Marked-Item-Files | 286 |
| Markiert formatiert | 290 |
| Markiert unformatiert | 289 |
| Markierter Inhalt | 290 |
| Maschinenroutinen | 54 |
| Maschinenunterprogramm | 205,215 |
| Mini-COPY | 192 |
| mini-DOS | 191 |
| Minidiskette | 22 |
| Minimal-Betriebssystem | 331 |
| Monitorparameter | 325 |
| | |
| N | 41,113,150 |
| NAME | 162 |
| NDF | 34,151 |
| NPMT | 113 |
| NL | 185 |
| NOPR | 171 |
| NOWAIT | 171 |
| NVD | 117 |
| NZKW | 114 |
| NZN | 114 |
| Normaldiskette | 22 |
| Numerieren | 48,217 |
| | |
| OPEN | 57,252,267,288 |
| OPEN"D" | 267 |
| OPEN"E" | 252 |
| OPEN"I" | 58,252 |
| OPEN"O" | 252 |
| OPEN"R" | 64,267 |
| Oktalumwandlung | 229 |
| | |
| PAUSE | 154 |
| PD | 77,155 |
| PD-Nummer | 115 |
| PD-Tabelle | 107 |
| PORT | 160 |
| PR | 171 |
| PRINT | 161 |
| PRINT# | 58,255 |
| PRINT# USING | 58,256 |
| PRINT/INPUT-Files | 252 |
| PROGMOD/BAS | 261 |
| PROT | 162 |
| PURGE | 39,163 |
| PUT | 66,278,292 |
| Parameter of Drives | 77 |
| Parität | 170 |
| Passwort | 106 |

| | |
|-----------------------|---------|
| Pop Return | 75 |
| Pop Stack | 74 |
| Ports | 139,160 |
| Positionierung | 66,293 |
| Programmierhilfen | 47,216 |
| Programmstart | 212 |
| Programmunterbrechung | 191 |
| Programmverbindung | 219 |
| Programmverschiebung | 305 |
| Protokoll | 171 |
| Puffernummer | 252 |

| | |
|----------------------|---------|
| QKW | 114 |
| QN | 114 |
| QPDN | 106,115 |
| Quelldiskette | 31 |
| Quelldisketten-Datum | 115 |

| | |
|---------------------|---------|
| R | 41,164 |
| RAM | 19,83 |
| RBA | 285 |
| REP | 49,221 |
| REP \$ | 49,221 |
| REP * | 49,221 |
| RENEW | 224 |
| RENUM | 225 |
| RENUM U | 49,227 |
| ROM | 19 |
| RSET | 65,271 |
| RTS | 171 |
| RUN | 43,212 |
| Random Files | 64 |
| Random-Datei | 266 |
| Referenzliste | 49,221 |
| Register | 194,325 |
| Relativ Byte Adress | 285 |
| Restart-Befehle | 337 |
| Restaurieren | 224 |

| | |
|-----------|----------------|
| S | 165 |
| SAVE | 42,213 |
| SBIV | 78,118,155,158 |
| SEK | 77,155,157 |
| SIO | 170 |
| SP | 77,155,157 |
| SPUR | 151 |
| SQD | 115 |
| ST | 185 |
| STAR/CMD | 324 |
| STMT | 173 |
| STOP | 151 |
| SUPER/CMD | 308 |
| SW2 | 77,155,157 |
| SYS0/SYS | 327 |
| SYS1/SYS | 327 |
| SYS10/SYS | 330 |
| SYS11/SYS | 330 |
| SYS12/SYS | 330 |
| SYS13/SYS | 330 |
| SYS14/SYS | 329 |
| SYS15/SYS | 329 |
| SYS16/SYS | 329 |
| SYS17/SYS | 329 |
| SYS18/SYS | 330 |
| SYS19/SYS | 330 |
| SYS2/SYS | 328 |
| SYS20/SYS | 330 |
| SYS21/SYS | 330 |
| SYS23/SYS | 329 |
| SYS24/SYS | 329 |
| SYS25/SYS | 329 |
| SYS26/SYS | 329 |
| SYS27/SYS | 329 |
| SYS28/SYS | 329 |
| SYS29/SYS | 329 |
| SYS3/SYS | 328 |
| SYS4/SYS | 328 |
| SYS5/SYS | 328 |
| SYS6/SYS | 328 |
| SYS7/SYS | 328 |
| SYS8/SYS | 328 |
| SYS9/SYS | 328 |

| | |
|------------------------------|------------|
| Satz | 64 |
| Satzadresse | 292 |
| Satznummer | 70 |
| Schreib/Lesekopf | 19 |
| Schreibdicke | 21 |
| Schreibschutz | 20 |
| Schreibschutzkerbe | 79 |
| Sektoren | 20,77,157 |
| Sequentielle Dateien | 55,58,252 |
| Serielle Schnittstelle | 170,176 |
| Short Hands | 311 |
| Single Sided | 21 |
| Single Step | 76 |
| Speicher löschen | 181 |
| Speichergröße | 202 |
| Speicherkapazität | 21,38 |
| Speicherschutz | 136 |
| Spur | 20,77,157 |
| Spurwechselzeitfaktor | 77 |
| Startblock Inhaltsverz. | 78,158- |
| Starten | 43 |
| Steckverbindungen | 79 |
| Steppermotor | 19 |
| Stopbits | 170 |
| String Area Size | 75 |
| Stringbereich | 75,238 |
| Stromausfall | 79 |
| System-Files | 327 |
| Systemdiskette | 31,83 |
| Systemeinsprungtabelle | 336 |
| Systemparameter | 165 |
| Systemzeit | 178 |
| Sätze | 55 |
| TA | 185 |
| TD | 77,155,157 |
| TI | 77,155 |
| TIMES | 41,214 |
| TT.MM.JJ | 113,120 |
| Tastatur | 310 |
| Teilformatierung | 152 |
| Tonausgaben | 308 |
| Treiberadressen | 139 |
| Trennzeichen | 55,59 |
| UHR | 40,174 |
| USR | 54,215 |
| Uhrzeit | 40,214 |
| Umleitungen | 185 |
| Ummumerieren | 49,225 |
| Unsichtbare Dateien | 36,88 |
| Unterprogramme | 236 |
| Untersätze | 269 |
| Übertragungsgeschwindigkeit | 17 |
| Übertragungsformat | 176 |
| V+ | 175 |
| V24 | 176 |
| VIDEO/BAS | 325 |
| VL/CMD | 305 |
| Variablen | 75 |
| Variablen löschen | 233,239 |
| Variablenaustausch | 76,234 |
| Vereinigen | 44 |
| Vergleichsliste G-DOS-NEWDOS | 342 |
| Vergrößerte Zeichen | 308 |
| Videospeicher | 190 |
| Vorhandene Dateien | 117 |
| WAIT | 171 |
| Warmstart | 95 |
| Warteschleife | 154 |
| Wortlänge | 170 |
| XDL | 118 |
| XON | 171 |
| Z | 177 |
| ZEIT | 40,178 |
| ZL | 179 |
| ZPDN | 106,115 |
| ZU | 162 |

| | |
|--------------------|-------------|
| ZZND | 114 |
| Zeichengenerator | 179 |
| Zeichensatz | 139,177,179 |
| Zeilen löschen | 218,232 |
| Zeitanzeige | 174 |
| Zieldiskette | 31 |
| Zieldiskettendatum | 114 |
| Zieldiskettensname | 114 |
| Zugriffstufen | 88 |