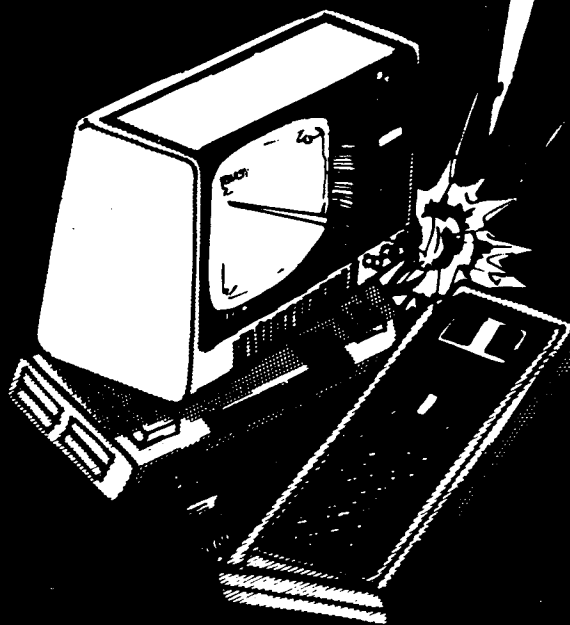


H.C. Pennington

TRS-80[®]

DISK

& andere Geheimnisse



H.C. Pennington

TRS - 80 DISK

... und andere Geheimnisse

übersetzt aus dem Englischen von

Günther Daubach

mit Zeichnungen des Autors



1. Auflage 1982

Alle Rechte, auch die der Übersetzung vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlages reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Der Verlag übernimmt keine Gewähr dafür, daß die in diesem Buch enthaltenen Informationen fehlerfrei und frei von Schutzrechten Dritter sind.

Printed in U.S.A.

Copyright 1982 by Interface Age Verlagsgesellschaft,
München, Vohburgerstr. 1

Gesamtherstellung: Interface Age, Cerritos, California, U.S.A.

Radio Shack und TRS-80 sind registrierte Warenzeichen
der Fa. TANDY Corporation

ISBN 3-88623-007-4

I N H A L T

Vorwort	7
Einführung	9
Hexadezimal - Binär - Dezimal	11
Das Arbeiten mit "SUPERZAP"	16
Andere Hilfsprogramme	29
Betriebssysteme	35
Organisation von Disketten	37
Das Directory	41
Schutzworte und sonstige Trivialitäten	57
Verfahren zur Wiedergewinnung von Daten	58
Dateistrukturen und -arten	60
Wiedergewinnung von Daten	71
Beheben von Fehlern bei "ELECTRIC PENCIL"	85
Korrigieren der "GAT"- und "HIT"-Sektoren	90
Was Sie sonst noch können	90
Glossar	94
Level II BASIC Tokens	99
Wartung der Diskettenlaufwerke	100
Murphy und die Gesetze der konstanten Gemeinheit	101
Programm "SEARCH" zum Suchen von Daten auf einer Diskette	103
TRSDOS 2.2 Directory	109
NEWDOS 2.1 Directory	114
VTOS 3.2 Directory	119

Vorwort zur englischen Ausgabe

Ich werde wohl nie meine erste Begegnung mit Harward C. Pennington, dem Autor dieses Buches vergessen. Bei einem Treffen der örtlichen TRS-80 Anwender-Gruppe sah ich zufällig einen unserer Radio Shack Manager. Er schien etwas außer Atem. Bei genauerer Betrachtung stellte ich fest, daß er von einem Diskettenanschlußkabel stranguliert wurde. Wer ihn da so quälte war niemand anders als Harv Pennington.

Harv hat aber inzwischen von solch drastischen Maßnahmen Abstand genommen, um die Antworten auf die vielen Fragen zu finden, die sich zu TRSDOS und anderen Hilfsprogrammen ergeben, ob sie von Radio Shack oder von jemand anderem vertrieben werden. Er ist nun von roher Gewalt zu Forschung und Schriftstellerei übergegangen. Die Ergebnisse seiner Untersuchungen sind in diesem Buch, "TRS-80 Disk und sonstige Geheimnisse" niedergelegt.

Lohnt sich denn nun die Ausgabe für dieses Buch? Um mit einem von Harvs Ausdrücken zu antworten: "zum Teufel, ja!" (Sie werden im Buch noch mehr solcher Kraftausdrücke finden.) Aber ernsthaft, lieber TRS-80 Anwender, das Buch ist nicht nur seinen Preis wert, es ist hervorragend und das aus zwei Gründen: Es bietet Informationen über die Organisation von TRS-DOS Disketten und Dateiverwaltung, wie man sie sonst nirgendwo findet! Zum anderen ist es verfügbar, wenn Sie es brauchen, nämlich jetzt!

Das Buch beschreibt, wie Disketten organisiert sind, wie freie Speicherbereiche zugeordnet werden und die Verfahren, wie man Disketten-Dateien untersuchen und ändern kann. Es gibt nicht nur eine allgemeine Information zu diesen Themen, sondern es zeigt klare Wege auf, wie Diskettenfehler behoben werden können, so z.B. verlorene Dateien, Fehler in Electric Pencil und andere Gemeinheiten.

Vor Ihnen liegt ein klar aufgebautes Buch, vollgepackt mit guten Informationen. Mein Rat ist: kaufen Sie es, gebrauchen Sie es und nähern Sie sich Mr. Pennington nicht, wenn er ein Diskettenkabel in der Hand hält.

William Barden Jr.

Vorwort zur deutschen Ausgabe

Der Mikroprozessor hat in den letzten Jahren weltweit seinen Einzug in fast alle Gebiete der Technik gehalten. Überrascht mußte man feststellen, daß sich aber auch im Bereich der Freizeitbeschäftigung neue Möglichkeiten aufboten, nachdem Kleinrechner zu erschwinglichen Preisen auf den Markt kamen.

Ein typischer Vertreter dieser Rechner ist der TRS-80 der Fa. TANDY Radio Shack. Die Anzahl der weltweit bisher verkauften Geräte beweist, daß TANDY eine wirkliche Marktlücke gefunden hat.

Der beste Rechner nützt jedoch recht wenig, wenn der Anwender nicht auch Literatur und Programme dazu erwerben kann. Das erst macht den Unterschied zwischen einem teuren Spielzeug und einem nutzbringenden Helfer im täglichen Leben aus. Gerade in den USA, dem Ursprungsland dieses Rechners, findet man eine Fülle von Literatur über den TRS-80, wie auch das Buch "TRS-80 Disk and Other Mysteries". Dem Verfasser dieses Buches gehört eine ganz besondere Anerkennung, da er in mühevoller Kleinarbeit bis in die hintersten Winkel und Ecken der Disketten vorgedrungen ist, um uns Anwendern eine Hilfestellung bei der Beseitigung von Diskettenproblemen zu geben.

Aber selbst, wenn Sie, lieber Leser -was ich Ihnen ganz besonders wünsche- nie Probleme mit Disketten

haben, bietet dieses Buch viele nützliche Informationen für Ihre Arbeit mit dem TRS-80.

Diese deutsche Ausgabe soll helfen, die Sprachbarriere abzubauen, die vielen Benutzern doch Schwierigkeiten bereitet. Selbst mehrjähriger Anwender des TRS-80, habe ich versucht, eine Übersetzung zu schreiben, die nicht wortwörtlich den Originaltext wiedergibt, sondern die zum Verständnis der Sachverhalte beiträgt, eine Eigenschaft, die ich bei manch anderen Übersetzungen aus dem Englischen leider vermisste.

Die technische Sprache bringt es aber mit sich, daß manche englischen Begriffe auch in unseren Sprachgebrauch übernommen werden. Hier halte ich es für besser, die englischen Ausdrücke weiterzuverwenden, als ein schwer zu verstehendes deutsches Wort zu benutzen, zumal Ihr Rechner sich mit Ihnen (oder besser Sie mit Ihrem Rechner ?) ja auch in "Computerchinesisch" unterhält.

Am Ende des Buches finden Sie ein Glossar, in dem die wichtigsten englischen Fachausdrücke dieses Buches zusammengestellt und erläutert sind.

Nun bleibt nur noch, Ihnen viel Freude beim Lesen des Buches zu wünschen. Mögen Sie möglichst viele Informationen für Ihre Beschäftigung mit dem TRS-80 gewinnen.

Leverkusen im August 1981,


Günther Daubach



Einführung

Programmierung ist eine Arbeit, die ich erst seit kurzer Zeit betreibe, aber ich habe mich ihr ganz verschrieben und es scheint mir, daß ich einige Erfahrungen gemacht habe, die andere gerne mit mir teilen möchten.

Sicherlich wurde auch Ihnen gesagt, daß sie bestimmte Dinge mit dem TRS-80 nicht machen können, wie zum Beispiel den automatischen Start eines BASIC-Programms beim Einschalten oder daß Sie die "BREAK"-Taste nicht sperren können, ohne die Ein-Ausgaberoutinen durcheinanderzubringen oder daß die LIST- und LLIST-Anweisungen nicht zu blockieren sind. Das alles ist falsch, denn all diese interessanten Dinge sind möglich. Es ist mir gelungen, die oben beschriebenen Funktionen mit nur geringem Aufwand zu realisieren. Die einzige Grenze der Möglichkeiten ist Ihr eigenes Vorstellungsvermögen.

Es ist nun einmal so, daß es keinen sicheren Weg gibt, ein Programm vollständig zu schützen, denn irgendjemand wird auch die obskurste und versteckteste Methode erkennen und sie entschlüsseln, genau wie ich es in diesem Buch beschreiben will.

Nachfolgend finden Sie das Ergebnis stundenlangen Starrens auf den Bildschirm, endloser Ausdrücke und einer geraumen Zeit, die mit Quervergleichen verbracht wurde. Nun, nachdem Sie sicher nach Lesen der vorangehenden Zeilen eine gewisse Hochachtung vor meinen Bemühungen gewonnen haben, werden wir sofort zum Thema kommen.

Eines muß jedoch gesagt werden: die gewonnenen Erkenntnisse hätten ohne das Programm "SUPERZAP" nie erreicht werden können. Dieses Programm stammt von der APPARAT Inc., Denver Colorado und ist in dem Diskettenbetriebssystem NEWDOS enthalten. Sie werden selbst feststellen, daß SUPERZAP eine

unbezahlbare Hilfe ist, in die tiefsten Tiefen einer Diskette vorzudringen.

Die folgenden Personen haben alle in irgend einer Weise am Entstehen dieses Buches mitgeholfen; hiermit möchte ich mich vor ihnen verbeugen:

Bill Barden
Jim Farvour
Ron Mackle
Jim Lauletta
C.I.
Michael Shrayner
Dick Schubert
Stu Nims
Dennis Fagan
Bob Thorpe.

Ihnen allen sage ich Dank aus dem innersten Herzen meiner CPU!

Einige freundliche Worte über den TRS-80

Im Großen und Ganzen ist der TRS-80 eine feine Maschine, wirklich, ich mag ihn. Vor einigen Jahren hätte ein Computer mit dieser Leistungsfähigkeit sicher einige hunderttausend Dollar gekostet, Sie hätten eine Klima-Anlage installieren und einen Stab von Personal einstellen müssen, um das Ding zu betreiben. Ganz gewiß verdient TANDY Corp. jedes nur mögliche Lob, für die Entwicklung, Fertigung und den Vertrieb dieser geheimnisvollen Maschine.....TANDY Corporation, mein besonderer Gruß gilt Ihnen!

...was es sonst noch zu sagen gibt

Die Firma TANDY hat mit dem TRS-80 seit seinem Erscheinen zweifellos einen großen Erfolg gehabt. Es fragt sich nur, ob der Erfolg nicht noch größer sein könnte, wenn TANDY mehr

Anwenderunterstützung in Form von Literatur und Software bieten würde. Wie sich zeigt, gibt es aber in vielen Ländern clevere Programmierer und Verfasser technischer Schriften, die diese Informationslücke auffüllen. Nicht zuletzt durch sie ist der TRS-80 zu einem der vielseitigsten Systeme geworden.

Ein typisches Beispiel dafür ist die Fa. APPARAT Corporation. Hier sind zwei clevere Burschen so mutig gewesen und haben das TRS-DOS 2.1 untersucht, überprüft und von Fehlern befreit. Das Ergebnis war NEWDOS, das bereits in der ersten Auflage praktisch fehlerfrei war. Die Fehler, die noch entdeckt wurden, haben sie sofort berichtet und auch die Besitzer von vorangehenden Versionen nicht vergessen, indem sie ihnen die Korrekturen mitteilten.

Über dieses Buch

Wenn Sie dieses Buch erstmals lesen, könnten Sie den Eindruck gewinnen, daß die Hauptaufgabe beim TRS-80 darin besteht, irgendwelche Fehler zu beheben, die durch ihn hervorgerufen werden. Dem ist nicht so! Es vergeht kein Tag, an dem ich nicht etwas Nützliches und Produktives auf dem Rechner erledige. Manchmal auftretende Fehler nehmen nur einen kleinen Teil des Arbeitsalltages in Anspruch. Nur, wenn Sie diese Fehler nicht berichtigen können, dann beginnen sie, Ihre Computer-Erfahrungen negativ zu beeinflussen.

Es ist mein Wunsch, daß Sie durch dieses Buch den Nutzen Ihres TRS-80 um einhundert Prozent steigern mögen.

H.C. Pennington

1.0 Buchstabe oder Zahl ?

Bei unserer Reise durch die Diskettenwelt werden wir häufig HEXADEZIMALE Zahlen verwenden. Es folgt daher an dieser Stelle eine kurze Beschreibung des hexadezimalen Zahlensystems. Falls Sie noch keine Erfahrungen zu diesem Thema haben sollten, sei Ihnen empfohlen, eines der vielen Grundlagenbücher zu studieren.

Der Computer führt all seine Gedanken in BINÄRZAHLEN aus. Da wir Menschen von Hause aus nicht binär denken, übernimmt der Computer die Arbeit des Umwandelns und versteht und liefert unsere Information in DEZIMAL. Allerdings sind Dezimalzahlen zu lang, wenn wir eine große Anzahl davon auf dem Bildschirm darstellen wollen und außerdem kann ein Computer sehr viel schneller binäre in hexadezimale Zahlen umwandeln als in die dezimale Darstellung.

Hexadezimal wird oft mit HEX abgekürzt und auch mit "H" gekennzeichnet. Es gibt auch andere Verfahren, Zahlen als hexadezimal zu kennzeichnen, aber davon später.

1.1 Binär

Sie haben zehn Finger und jemand, der die Zahlenwissenschaft studiert hat, wird uns sicher sagen, daß das der Grund für die Verwendung unseres Dezimalsystems ist. Für jeden Finger haben wir ein besonderes Symbol, nämlich:

Abb. 1.1

Die zehn Dezimalsymbole

0 1 2 3 4 5 6 7 8 9

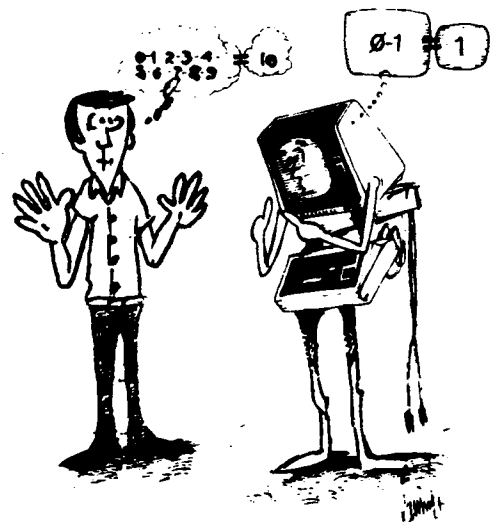
Andererseits hat ein Computer nur zwei "Finger" ("ein" und "aus") und daher "denkt" er natürlich im Zweiersystem und benötigt auch nur zwei Symbole, um Zahlen darzustellen:

Abb. 1.2

Die zwei Binärsymbole

0 1

Zur Darstellung beliebiger Zahlen verwenden wir ein System, das Wert darauf legt, in welcher REIHENFOLGE und WO in einer Zahl die einzelnen Ziffern stehen.



Wenn wir Menschen beim Zählen die Zahl 9 erreichen, müssen wir mit unserem Ziffernvorrat wieder von vorne beginnen. Wir setzen links eine "1" ein und beginnen rechts wieder mit der Ziffer "0". Durch das Schreiben einer "1" auf die linke Position ordnen wir ihr auch einen anderen Wert zu, der davon abhängt, WIEVIEL Stellen weiter links die Eins plaziert ist.

Abb. 1.4

10 = (100 10)	4 = (111)
9 = (100 11)	3 = (11)
8 = (100 11)	2 = (11)
7 = (100 1)	1 = (1)
6 = (100)	0 = ()
5 = (100)	

Aus der obigen Abbildung kann man den Zusammenhang zwischen Dezimalzahlen und dem Wert erkennen, den sie darstellen. Es bereitet uns Menschen oft Schwierigkeiten, "NULL" als Zahl anzusehen, dabei ist "Null" die ERSTE Zahl in der Reihe und sollte immer so betrachtet werden, d.h. wir beginnen mit dem Zählen bei der Zahl "Null". Wenn Sie die Finger Ihrer Hände zählen und dabei mit "Null" beginnen, werden Sie nur bis zur "Neun" gelangen. Genauso zählt auch ein Computer, er beginnt IMMER mit "Null"!

Nun können wir auch ohne Schwierigkeit untersuchen, wie ein Computer mit seinen zwei "Fingern" zählen kann.

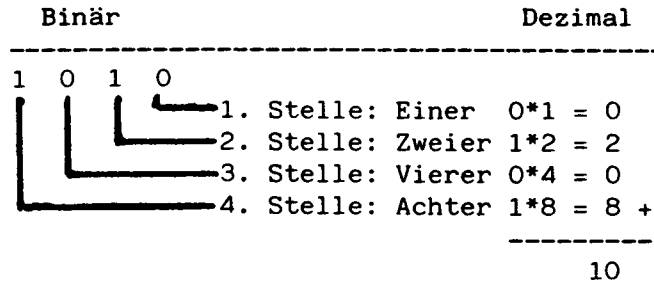
Daß der Computer nur zwei "Finger" hat ist dadurch bedingt, daß es sich um eine digitale Maschine handelt, die technisch nur zwei unterschiedliche Bedingungen erkennen kann, nämlich "EIN" und "AUS". Sie werden sich aus dem Vorangehenden daran erinnern, daß ein Mensch beim Erreichen der "9" mit einer neuen Stelle weiterzählt und insgesamt zehn Ziffernsymbole zur Verfügung hat. Der

Computer besitzt jedoch nur zwei Ziffernsymbole und muß aus diesem Grund bereits nach Erreichen der "1" mit einer neuen Stelle fortfahren. Anders ausgedrückt, der Ziffernvorrat ist bereits beim Erreichen der Zahl "1" erschöpft und es muß eine "1" in die nächsthöhere Stelle übernommen werden, um in der niedrigen Stelle wieder mit der "0" weiterzuzählen.

Abb. 1.5

Binärzahl		Dezimalzahl
1010	= (100 10)	= 10
1001	= (100 11)	= 9
1000	= (100 11)	= 8
111	= (100 1)	= 7
110	= (100)	= 6
101	= (100)	= 5
100	= (100)	= 4
11	= (11)	= 3
10	= (11)	= 2
1	= (1)	= 1
0	= ()	= 0

Lassen Sie uns die Binärzahl 1010 etwas genauer untersuchen. Zunächst hat sie vier Stellen. Jede Stelle entspricht einer Zweierpotenz. Wir können diese Dualzahl in eine Dezimalzahl umwandeln, indem wir die Stellen jeweils mit dem Stellenwert multiplizieren und die Einzelergebnisse addieren.



Durch die Addition unserer Multiplikationsergebnisse haben wir eine Binärzahl (zur Basis 2) in eine Dezimalzahl (zur Basis 10) umgewandelt.

1.2 Hexadezimal

Nun werden wir uns mit Hexadezimalzahlen (zur Basis 16) herumschlagen. Der Computer benötigt eine Methode, um große Zahlen auf möglichst kleinem Raum darzustellen. Binärzahlen sind

leicht in hexadezimale Zahlen umzuwandeln (zumindest für den Computer). Das Hexadezimalsystem verwendet 16 Zeichen, um die 16 Ziffern darzustellen und wenn wir mit dem Ziffernvorrat zu Ende sind, schreiben wir eine "1" in die linke Stelle und beginnen wieder von vorne in der Reihe der Zeichen, also genauso wie bei Binär- und Dezimalzahlen.

Die folgende Abbildung zeigt die Hexadezimalziffern und ihre entsprechenden Dezimalwerte:

Abb. 1.7

HEX	=	Dezimal	HEX	=	Dezimal
0	=	0	8	=	8
1	=	1	9	=	9
2	=	2	A	=	10
3	=	3	B	=	11
4	=	4	C	=	12
5	=	5	D	=	13
6	=	6	E	=	14
7	=	7	F	=	15

Irgendjemand hat einmal entschieden, für die zusätzlichen Hexadezimalziffern Buchstaben zu verwenden (fragen Sie mich nicht, wer), da Ziffern und Buchstaben auf jeder normalen Tastatur sowieso vorhanden sind. Das führt dazu, daß wir auch Zahlen wie "1A" oder "FF" erhalten. Sie werden sehen, die Verwendung von Hexadezimalzahlen ist nicht so unbequem, wie es zuerst scheinen mag.

Es folgt die Darstellung von Hexadezimal-, Binär- und Dualzahlen. Die Tabelle ist jedoch etwas verkürzt, da ja die Zahlen von "0" bis "9" in hexadezimal und dezimal gleich sind.

Abb. 1.8

HEX	Dezimal	Binär
20 = (1010101010101010)	= 32	= 100000
1A = (1010101010101010)	= 26	= 11010
15 = (1010101010101010)	= 21	= 10101
10 = (1010101010101010)	= 16	= 10000
F = (1010101010101010)	= 15	= 1111
A = (1010101010101010)	= 10	= 1010
5 = (1010101010101010)	= 5	= 101

Wie Sie sehen, sind wir mit unseren Ziffern am Ende, wenn wir den Wert "F" erreichen und wir müssen dann, wie bei jedem anderen Zahlensystem auch, eine neue Stelle "aufmachen", indem wir eine "1" in diese neue Stelle bringen und in die "Einerstelle" eine "0" schreiben.

Wenn Sie also die 10 hex, 10 dezimal oder 10 binär sehen, wissen Sie, daß ich von drei unterschiedlichen Werten "Eins-Null" spreche.

Wir müssen uns noch etwas anschauen, was wir im Verlauf des Buches öfters brauchen werden:

Manchmal möchten wir eine Binärzahl in ihren Hex-Wert umwandeln. Das ist leichter, als Sie vielleicht annehmen werden. Betrachten wir das Folgende:

$$FF \text{ (Hex)} = 11111111 \text{ (Binär)}$$

Es sieht kompliziert aus, ist es aber nicht, es ist sogar sehr einfach! Die Art, wie man ein Problem lösen sollte ist die Aufteilung in handliche Abschnitte. Hier ist es nicht anders. Zuerst nehmen wir die Binärzahl, die in diesem Fall aus acht Bit oder einem Byte besteht und teilen sie in zwei Vierergruppen, genannt "Nybble". Wenn wir also das obige Beispiel nehmen, ergibt sich:

$$FF \text{ (Hex)} = 1111 \ 1111$$

Nun erinnern Sie sich bitte an die Stellenwerte von Dualzahlen. Zur Erinnerung hier nochmal eine Abbildung, die die Zusammenhänge zeigt:

Abb. 1.9

- 8. Stelle = 128
- 7. Stelle = 64
- 6. Stelle = 32
- 5. Stelle = 16
- 4. Stelle = 8
- 3. Stelle = 4
- 2. Stelle = 2
- 1. Stelle = 1

Wenn wir jede Vierergruppe (Nybble) nehmen, können wir den Hex-Wert leicht ausrechnen, da es einfach ist, sich die Hex-Zahlen von "0" bis "F" zu merken. Addieren wir auf der rechten Seite die Binärwerte, so kommen wir zu folgendem Ergebnis:

$$\begin{array}{r}
 8 * 1 = 8 \\
 4 * 1 = 4 \\
 2 * 1 = 2 \\
 1 * 1 = 1 + \\
 \hline
 15 \text{ Dezimal} = F \text{ (HEX)}
 \end{array}$$

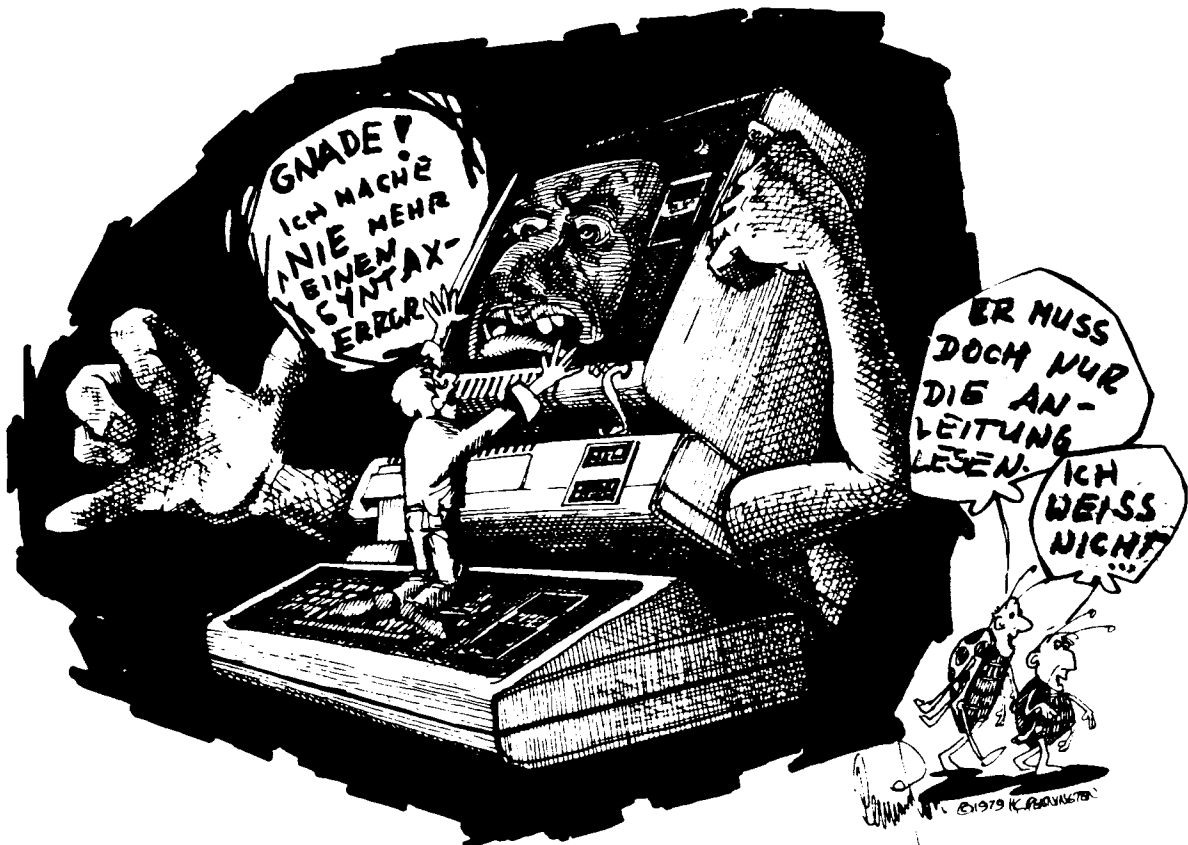
Das Gleiche nochmal für die linke Seite, die Teilergebnisse kombinieren....und wir sind fertig!

Um sicher zu werden, wollen wir das noch einmal mit einer anderen Binärzahl versuchen. Suchen wir den Hex-Wert für den Binärwert 00101101. Die Nybbles sind also 0010 1101.

linke Seite	rechte Seite
8 * 0 = 0	8 * 1 = 8
4 * 0 = 0	4 * 1 = 4
2 * 1 = 2	2 * 0 = 0
1 * 0 = 0 +	1 * 1 = 1 +
<hr/>	
2 dez.	13 dez.

2 dezimal = 2 HEX 13 dezimal = D HEX
linke + rechte Seite = 2D HEX

Der Alptraum



2.0 Das Arbeiten mit "SUPERZAP"

SUPERZAP ist in mehrfacher Hinsicht einzigartig. Einmal hat es seine eigenen Disketten-Ein/Ausgabe Routinen und benötigt nicht das DOS in Laufwerk 0, um Wunder zu wirken. Außerdem kann es alles von Diskette lesen, was nur lesbar ist, unabhängig vom Programmschutz. Nicht zuletzt können wir fast alle Daten wiedergewinnen, die unter DOS nicht mehr lesbar sind.

Zusätzlich hat es eine "BACKUP"-Routine, die viele Leseversuche macht, wenn sie auf einen fehlerhaften oder beschädigten Sektor trifft, bevor sie abbricht und dann erlaubt sie Ihnen noch beliebig viele, weitere Versuche.

Halt, es gibt noch mehr Möglichkeiten! SUPERZAP kann Sektoren kopieren, verschieben, und beliebige Bytes auf der Diskette oder im Speicher ändern, Daten von einem Sektor auf einen anderen übertragen und Sektoren mit Null überschreiben.

Nachdem wir nun einen Überblick bekommen haben, wollen wir uns jetzt alle Funktionen und Befehle der Reihe nach anschauen.

2.1 SUPERZAP-Funktionen

Ich würde empfehlen, daß Sie sich vor den Computer setzen, SUPERZAP starten, und jede Funktion und jeden Befehl ausprobieren, so wie hier beschrieben. Auf diese Art werden Sie sich schnell mit SUPERZAP vertraut machen.

ANM.. geben Sie alle Anweisungen ohne Anführungszeichen ein!

"DD" oder "NULL" - DISPLAY DISK
SECTOR (Anzeige von Disketten-Sektoren).

Diese Funktion werden Sie häufiger brauchen, als jede andere. Sie werden sich sehr oft Sektoren ansehen, oder etwas darin ändern wollen. "NULL" bedeutet in diesem Fall "drücke ENTER". Da diese Funktion sooft verwendet wird, entschied sich Cliff, der Autor von SUPERZAP dazu, uns das dauernde Drücken von "DD" zu ersparen. (Vielen Dank, Cliff!) Nachdem Sie "ENTER" oder "DD" eingegeben haben, antwortet der Computer wie in Abb. 2.2 dargestellt.

Abb. 2.2

```
RELATIVE DISK # (0 - 3)?  
TRACK # (HEX) (0 - 22)?  
SECTOR # (0 - 9)?
```

Geben Sie auf "RELATIVE DISK #" die Laufwerksnummer ein, mit der Sie arbeiten wollen. Sie werden bemerken, daß auf "TRACK #" nur eine Zahl zwischen 0 und 22 angenommen wird (0 bis 27 bei der 40-Spur-Version). Sie müssen alle Zahlen in Hexadezimal eingeben. Bei Sektornummern ist das leicht; es gibt 10 Sektoren von 0 bis 9. Suchen Sie sich ein Laufwerk, eine Spur und einen Sektor aus und sehen Sie sich ihn an. Wenn Sie genug gesehen haben, drücken Sie "X" und es erscheint wieder das Menue. Es gibt noch andere Dinge, als einfaches Anschauen, die man in dieser Funktion durchführen kann, aber dazu später mehr.

"PD" - PRINT DISK SECTORS (Drucken von Disketten-Sektoren)

Diese Funktion ist fast genau - ich betone fast - genau das Gleiche, wie die "DD"-Funktion, außer daß die Inhalte von Sektoren auf dem Drucker ausgegeben werden und Sie nichts ändern können. Diese Funktion fragt nach einem zusätzlichen Parameter, nämlich der Sektor-Anzahl. Geben Sie die Anzahl der zu druckenden Sektoren ein, drücken Sie ENTER und treten Sie zurück!

ANM.: Geben Sie die Anzahl der Sektoren dezimal ein!

Diese Funktion schlägt Radio Shacks DISKDUMP/BAS auf der ganzen Linie. Es druckt eine "geschützte" Datei aus, ohne die gewohnte Anzeige "FILE ACCESS DENIED". Sollten Sie sich entschließen, den Ausdruck anzuhalten, drücken Sie SOLANGE auf die Taste "H", bis Sie den Erfolg merken.

"DM" - DISPLAY MAIN MEMORY (Anzeige des Speicherinhaltes).

Diese Funktion macht genau dasselbe mit dem RAM, wie "DD" mit der Diskette. Sie werden allerdings nach der RAM-Adresse (in Hex) gefragt und nicht nach "DISK", "TRACK" und "SECTOR". Später, wenn wir weitere Funktionen von "DD" besprechen, treffen diese für "DM" ebenso zu. Durch Drücken von "X" gelangen Sie wieder in das Menue.

-----W A R N U N G-----
Änderungen, die mit "MODnn" eingegeben werden, wirken sich hier sofort auf den Speicher aus !

"PM" - PRINT MAIN MEMORY (Ausdruck des Speicherinhaltes)

Diese Funktion entspricht "PD", nur daß hier der Inhalt des RAM- oder ROM-Speichers ausgedruckt wird. Durch Drücken und Festhalten der Taste "H" kann auch hier der Ausdruck abgebrochen werden, genau wie bei "PD".

"VERIFY DISK SECTORS" (Überprüfen von Disketten-Sektoren)

Diese Funktion erkennt Sektoren mit Schreibschutz, solche mit Paritäts- und physikalischen Fehlern.

Sie können zusätzlich eine "PAUSE"-Option wählen, um den Prüfungsvorgang immer dann anzuhalten, wenn ein "lesegeschützter" Sektor gefunden wird. Damit können Sie sich solche Sektoren für spätere Arbeiten notieren.

Diese Funktion ist ausgesprochen nützlich, um herauszufinden, wo auf einer unzuverlässigen Diskette schlechte Sektoren sind, deren Daten Sie wiedergewinnen wollen. Sie müssen die Anzahl der abzusuchenden Sektoren als Dezimalzahl eingeben.

"ZERO DISK SECTORS" (Überschreiben von Sektoren mit Null)

Von Zeit zu Zeit werden Sie bestimmte Sektoren mit lauter Nullen überschreiben wollen. Wenn Sie das Byte für Byte durchführen müßten, wäre das eine sehr umständliche Sache. Ist ein Sektor, den Sie überschreiben wollen, "lesegeschützt", werden Sie gefragt, ob dieser Schutz bestehen bleiben soll. Antworten Sie entsprechend mit "Y" (YES-ja) oder "N" (NO-nein). Diese Funktion benötigt die Eingabe einer Sektorzahl in dezimal.

"COPY DISK SECTORS" (Sektoren kopieren)

Mit diesem Bonbon können Sie einen einzelnen Sektor oder eine ganze Gruppe von Sektoren von einer Stelle zu einer anderen oder von einer Diskette auf eine andere kopieren. Wenn wir eine Datei rekonstruieren müssen, die von DOS über die gesamte Diskette verstreut wurde, werden Sie sich glücklich schätzen, daß Sie diese Funktion besitzen. Was normalerweise ein fast unmögliches Unterfangen wäre, ist hiermit so einfach, daß Sie sich wundern werden, wie schnell das alles geht.

Normalerweise kopiert diese Funktion die Sektoren in aufsteigender Reihenfolge, nach Spur und Sektor. Wenn allerdings der niedrigste Sektor, auf den Sie kopieren wollen, innerhalb des Bereiches liegt, aus dem Sie kopieren möchten, so wird automatisch in absteigender Folge gearbeitet. Das alles erfolgt selbständig ohne Ihr Eingreifen.

Dadurch können Sie eine Anzahl von Sektoren auch auf eine Stelle kopieren, die innerhalb des Bereiches liegt, aus dem Sie kopieren wollen.

Der "Leseschutz" von Sektoren bleibt beim Kopieren erhalten; die gewünschte Anzahl der Sektoren müssen Sie als Dezimalzahl eingeben.

"DISK BACKUP" (Disketten duplizieren)

Erstaunlich...diese Funktion ist wirklich erstaunlich! Es wird einfach ein Diskettenduplikat erzeugt, aber was für eines! Die Funktion arbeitet Sektor für Sektor und ist daher seeehr laaangsam. Aber sicher! Sie kann noch Sektoren lesen, wenn die BACKUP-Funktion des DOS schon lange "abgestürzt" ist. Sie macht etwa ein Dutzend Versuche, um einen "schlechten Sektor" zu lesen und gibt Ihnen, wenn alles nichts geholfen hat, eine

Fehlermeldung entsprechend Abb. 2.3.

Abb. 2.3

```
SECTOR READ ERROR
DRIVE 1, TRACK 05, SECTOR 0
SYSTEM ERROR CODE 04
PARITY ERROR
REPLY 'R' FOR RETRY,
OR 'X' TO CANCEL FUNCTION?
```

Nun haben Sie mehrere Möglichkeiten: (1) drücken Sie "X" und vergessen Sie das Ganze, (2) drücken Sie "S" und überspringen Sie den fehlerhaften Sektor (vergessen Sie nicht, sich eine Notiz für spätere Verwendung zu machen!) oder drücken Sie "R" und starten Sie einen neuen Versuch, den Sektor zu lesen. Oft haben Sie mit dem "R"-Kommando dann doch noch Erfolg.

Nach jedem Schreiben erfolgt automatisch ein Prüfllesen des betreffenden Sektors, um sicherzustellen, daß die Datenübertragung fehlerfrei war.

-----ACHTUNG-----
Die Diskette, auf die kopiert werden soll, muß vorher FORMATIERT werden. "Quellen-" und "Ziel-" Diskette können NICHT im selben Laufwerk sein, mit anderen Worten, Sie benötigen für diese Funktion ZWEI Laufwerke. Die "Ziel-" Diskette wird NICHT auf ihren Namen und Inhalt überprüft, alle Daten, die auf dieser Diskette stehen, werden durch die Funktion überschrieben.

"COPY DISK DATA" (Kopieren von Diskettendaten).

Diese Funktion entspricht "COPY DISK SECTORS", jedoch werden hier einzelne BYTES kopiert und zwar von einem bis zu 65.536 Stück. Das ist eine weitere Super-Funktion, um verlorengegangene Daten zu regenerieren.

Auch hier treffen dieselben Regeln für die Reihenfolge zu, wie bei "COPY DISK SECTORS". Sie brauchen sich darüber aber keine Gedanken zu machen.

Der "Leseschutz-Status" bleibt unverändert. Geben Sie die Anzahl der zu kopierenden Bytes als HEXADEZIMALZAHL ein!

2.2 SUPERZAP-Befehle

Wenn Sie sich in der "DISPLAY DISK SECTORS-" oder "DISPLAY MAIN MEMORY-" Funktion befinden, fragt das Programm ständig die Tastatur ab und reagiert auf eines der folgenden Kommandos, die Sie ohne "ENTER" eingeben müssen:

- "X" - Abbruch der Gesamtfunktion
- "R" - Anzeige des Sektors oder Speicherblocks wiederholen
- "J" - Neustart der Gesamtfunktion
- "K" - (nur bei "DD") wie "J", jedoch Anzeige von Spuren und Sektoren auf dem gleichen Laufwerk, wie vorher
- "H" - (nur bei "PD" und "PM") Abbruch des Ausdrucks
- "+" oder ";" - Vorwärtsrollen um einen Sektor oder Speicherblock
- "-" oder "=" - Rückwärtsrollen um einen Sektor oder Speicherblock

2.3 Spezielle Befehle

Die folgenden Kommandos funktionieren nur bei "DISPLAY DISK SECTORS" und "DISPLAY MAIN MEMORY". Sie werden dazu verwendet, um den Inhalt des Speichers oder die Diskette Byte für

Byte zu modifizieren. Die Kommandos sind:

- "MODnn" - Modifizieren des Bytes im momentan angezeigten Bereich, "nn" ist eine zweistellige Hexadezimalzahl, für das relative Byte, das geändert werden soll (Siehe Beisp. 1).
- "ENTER" - NACH Änderung eines oder mehrerer Bytes bewirkt ENTER, daß diese Daten auf Diskette geschrieben werden.
- "Leertaste" - Die momentane Stelle wird nicht geändert, die Position für die Änderung wird um eine Stelle weitergeschaltet.
- "Rechtspfeil" - wie "Leertaste"
- "Linkspfeil" - Die momentane Stelle wird nicht geändert, die Position für die Änderung wird um eine Stelle zurückgeschaltet.
- "SHIFT-Rechtspfeil" - Weiterschalten der Änderungsposition um vier Stellen.
- "SHIFT-Linkspfeil" - Zurückschalten der Änderungsposition um vier Stellen.
- "Aufwärtspfeil" - Änderungsposition um eine Zeile zurück.
- "Abwärtspfeil" - Änderungsposition um eine Zeile weiter.
- "SCOPY" - (nur bei "DD") Übertragen des angezeigten Sektors auf eine andere Stelle.

2.4 Spezielle Symbole

Während der Änderung von Bytes im Speicher oder auf der Diskette erscheinen auf dem Bildschirm einige spezielle Symbole, um die Zeile und die Lage des Bytes zu kennzeichnen, das Sie gerade bearbeiten (Änderungsposition).

Diese Zeichen sind: "M", "+", "-", "*" und "/". Das "M" markiert eine Zeile und befindet sich zwischen der ersten, sechsstelligen Spalte und der ersten vierstelligen Spalte mit Daten.

Die Zeichen "+", "-", "*" und "/" erscheinen unmittelbar links neben der Vierergruppe, in der eine Änderung vorgenommen werden kann. Das "+"-Zeichen kennzeichnet die erste Stelle, das "-" die zweite, der "*" die dritte und "/" die vierte Stelle. Mit dieser Anzeige sind Sie darüber informiert, welche Stelle Sie als nächste verändern können.

2.5 SUPERZAP Anzeige-Format

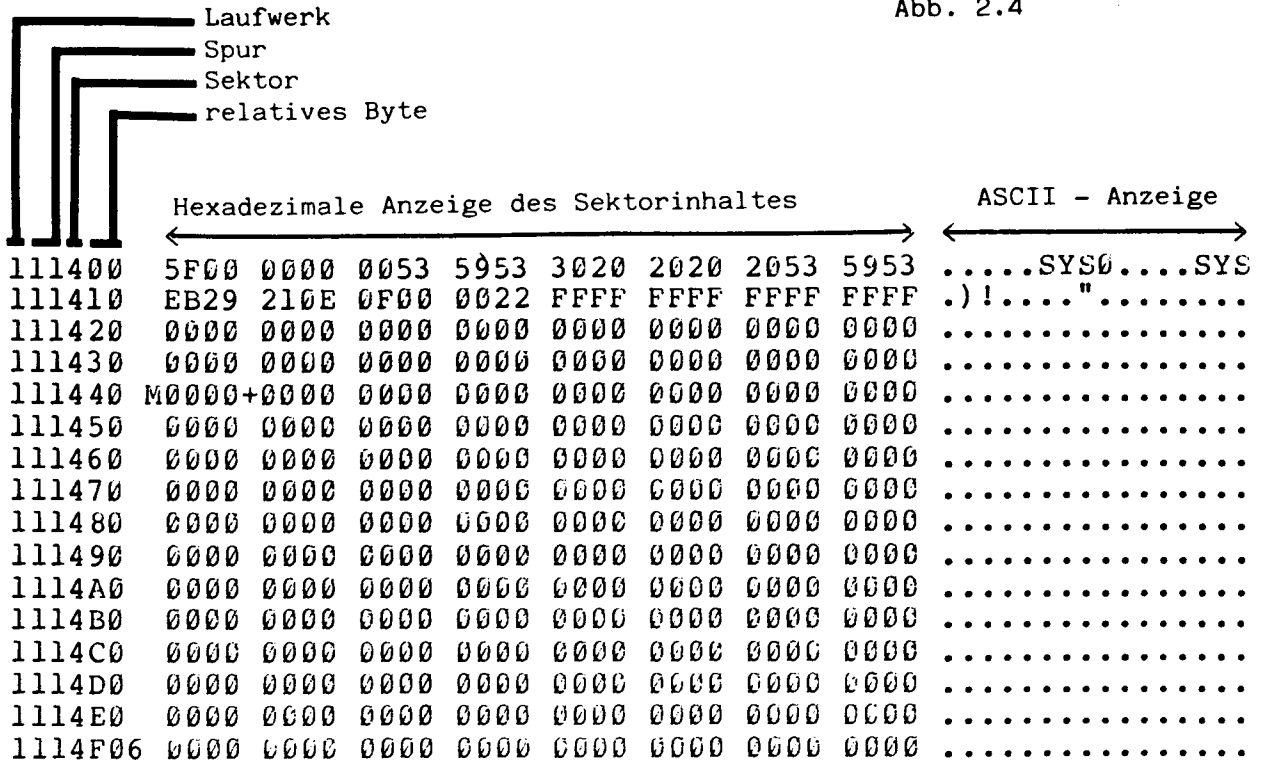
Bevor wir weitergehen und eine richtige Demonstration mit einigen Beispielen kennenlernen, müssen wir zuerst verstehen, wie SUPERZAP Daten und Informationen anzeigt. Abb. 2.4 bringt die typische Darstellung eines Sektors auf dem Bildschirm. Die sechs Stellen, ganz links enthalten folgende Parameter (von links nach rechts):

Position 1: das verwendete Laufwerk
Position 2 und 3: die gelesene Spur
Position 4: der relative Sektor
 innerhalb der Spur
Position 5 und 6: relative Byte-
 Nummer innerhalb des
 Sektors.

In der unteren Zeile erscheint ggf. eine weitere Stelle an Position 7. Hier bedeutet eine "6", daß es sich um einen Sektor mit "Leseschutz" handelt.

Rechts von diesen sechs Stellen finden Sie einen Block von 32 Stellen, aufgeteilt in Vierergruppen. Jedes Stellenpaar stellt ein Byte als Hexadezimalzahl dar. Rechts davon finden Sie die gleichen Daten nochmal, aber in ASCII-Darstellung. Punkte kennzeichnen dabei "nicht druckbare Zeichen". Was ein "nicht druckbares Zeichen" ist? Nun, es ist einfach ein gültiges ASCII-Codezeichen, für das es jedoch kein Symbol gibt (z.B. Steuerzeichen).

Ein Punkt kann aber auch eine Leerstelle anzeigen. Es besteht ein feiner Unterschied zwischen Leerzeichen und "nicht druckbaren Zeichen". So sind z.B. in BASIC-Programmen "nicht druckbare Zeichen" Kenner für die nächste Programmzeile, Zeilen-Nummern und Befehlskürzel (darüber später mehr). Manchmal sind solche Hexadezimalwerte gleichzeitig "druckbare Zeichen", die dann die Anzeige eines Symbols auf dem Bildschirm verursachen. Es scheint dann so, als ob sich irgendwelcher Unsinn in Ihr Programm eingeschlichen hätte, aber Sie sollten nicht verzweifeln, alles ist in Ordnung! Andererseits wird ein "Leerzeichen" durch eine Hex-Zahl 20 dargestellt. Das ist der besondere Fall, bei dem "nichts" doch etwas ist. Darauf sollten Sie achten und sich nicht verwirren lassen.



Typische "SUPERZAP"-Anzeige eines Sektors, wie sie auf Ihrem Bildschirm erscheint. Der hier dargestellte Sektor gehört zum Directory (Spur 11, Sektor 4).

2.6.1 Beispiel 1: "MOD nn"

Um ein oder mehrere Bytes in einem bestimmten Sektor zu ändern, wählen Sie zunächst die Funktion "DD" aus (oder "ENTER"). Beantworten Sie die Fragen nach Laufwerk, Spur und Sektor entsprechend. Nachdem der Sektor auf dem Bildschirm dargestellt ist, geben Sie "MOD42" ein.

Anmerkung: Während Sie diese Eingabe machen, erscheint keine zusätzliche Anzeige auf dem Bildschirm, solange bis Sie den vollständigen Befehl eingegeben haben.

Nun erscheint in unserem Beispiel in der fünften Zeile von oben und vor dem ersten Viererblock ein "M" und vor dem zweiten Viererblock in dieser Reihe sehen Sie ein "+". (Siehe Abb. 2.4)

Wir sind nun bereit, die angezeigten Daten zu ändern. Sie können jedes beliebige gültige Hexadezimalzeichen eingeben. Jedesmal, wenn Sie eine Taste drücken, ändert sich die entsprechende Stelle auf dem Bildschirm und das Zeichen vor dem Viererblock verändert sich ebenfalls. Diese Symbole dienen als Anzeige, in welcher Position innerhalb des Viererblocks die nächste Änderung stattfindet.

Wenn Sie eine Stelle überspringen wollen, drücken Sie entweder die Leertaste oder den Pfeil nach rechts. Das Indikator-Symbol wird sich ändern, und jeweils vor den nächsten Viererblock springen, nachdem Sie entweder vier Änderungen oder vier Leerzeichen eingegeben haben.

Wenn alle Änderungen eingegeben sind, drücken Sie "ENTER" und diese nun geänderten Daten werden in den entsprechenden Sektor auf die Diskette geschrieben. Nach Beendigung des Schreibvorgangs folgt die Frage:

```
MODIFICATIONS COMPLETE.  
REPLY ENTER TO CONTINUE?
```

(Abb. 2.5)

Nach Drücken von "ENTER" wird der betreffende Sektor wieder von der Diskette gelesen und erneut angezeigt. Sie können nun nochmals die veränderten Daten überprüfen und eventuell korrigieren. Dabei kann jeder Sektor sofort geändert werden, wie es Ihnen Freude macht.

Wenn Sie nun die ";"-Taste drücken, schalten Sie weiter zum nächsthöheren Sektor; mit "-" gelangen Sie zum vorhergehenden Sektor.

Durch "R" rufen Sie die letzte Hauptfunktion erneut auf, in diesem Fall also wieder die Anzeige des Sektors, dessen Daten Sie zu Anfang eingegeben hatten.

Mit "K" ist es möglich, eine andere Spur und einen anderen Sektor einzugeben, ohne daß Sie erst wieder zurück in das Menü springen müssen.

"J" entspricht der Funktion "K", hier können Sie zusätzlich aber auch ein anderes Laufwerk ansprechen.

Drücken von "X" führt immer sofort zum Menü zurück.

Mit "Q" läßt sich die "MOD"-Funktion abbrechen, OHNE daß die Daten auf der Diskette verändert werden.

Nun, nachdem Sie die wichtigsten Funktionen kennengelernt haben, sollten Sie sich von Ihrer Systemdiskette eine Kopie anfertigen (BACKUP) und selbst ausprobieren, was diese Funktionen bewirken.

*****WARNUNG*****

Ihre Versuche sollten Sie unbedingt auf einer Kopie machen, damit Ihnen keine wichtigen Daten oder Programme verlorengehen !

2.6.2 Beispiel 2: "SCOPY"

"SCOPY" erlaubt Ihnen, einen ganzen Sektor von einer Stelle der Diskette an eine andere Stelle oder auf eine andere Diskette zu bringen, wenn Sie sich in der "DD"-Betriebsart befinden.

Nehmen wir einmal an, Sie hätten versucht, einen Sektor zu lesen und erhalten als Fehleranzeige: "BUFFER MAY CONTAIN ALL OR SOME OF SECTOR'S DATA" (Puffer kann einige oder alle Daten des Sektors fehlerfrei enthalten). Sie erkennen nun aus der Anzeige, daß bestimmte Daten tatsächlich fehlerfrei sind und möchten diese für spätere "Rettungsaktionen" sichern. Geben Sie dazu "SCOPY" ein (auf dem Bildschirm erscheint erst dann eine Reaktion, wenn der gesamte Befehl eingegeben ist). Nun erhalten Sie folgende Meldung:

```
DRIVE 1, TRACK 12, SECTOR 9  
IS TO BE COPIED TO  
RELATIVE DISK (0 - 3)?  
TRACK      HEX (0 - 22)?  
SECTOR    (0-9)
```

(Abb. 2.6)

Nachdem Sie die Laufwerksnummer, die Spur und den Sektor eingegeben haben, prüft das Programm diesen Sektor, ob er fehlerfrei ist. Ist auch dieser Zielsektor unzuverlässig, erhalten Sie eine neue Fehlermeldung, ist alles in Ordnung, wird sofort mit der Datenübertragung begonnen. Danach werden Sie aufgefordert, "ENTER" zu drücken, um den neuen nun übertragenen Sektor anzusehen.

Wenn Sie versuchen, Daten einer Diskette zu retten, ist es eine gute Idee, einige Sektoren als "Puffer-Sektoren" vorzusehen, in die Sie mit "SCOPY" vorübergehend Daten übertragen können. Natürlich müssen Sie sich unbedingt merken, wo diese Sektoren liegen und welche Daten Sie dahin übertragen haben. Eine weitere gute Möglichkeit ist es, eine vollständige Datei (file), die Sie bearbeiten möchten, in einen Pufferbereich zu übertragen, dort Ihre Änderungen vorzunehmen und wenn alles stimmt, die Datei aus dem Pufferbereich wieder an die richtige Stelle zu übertragen.

2.6.3 Beispiel 3: "COPY DISK DATA" (Kopieren von Disketten-Daten)

Diese Funktion erlaubt es Ihnen, ganze Datenblöcke zu verschieben. Sie können sowohl ein einzelnes Byte als auch eine beliebig lange Folge von Bytes auf eine andere Stelle kopieren.

Angenommen, wir müssen einen Directory-Eintrag von einem Sektor in einen anderen übertragen, aber auch noch die Lage dieser Daten innerhalb des Sektors verschieben. Die folgenden Abbildungen und Texte zeigen Ihnen die entsprechenden Meldungen und Eingaben und den Zustand der Sektoren vor- und nachher:

Abb. 2.7

```

011400 5F00 0000 0053 5953 3020 2020 2053 5953 .....SYS0....SYS
011410 EB29 210E 0F00 0622 FFFF FFFF FFFF FFFF .)!.....".....
011420 0000 0000 0000 0000 0000 0000 0000 0000 .....
011430 0000 0000 0000 0000 0000 0000 0000 0000 .....
011440 0000 0000 0000 0000 0000 0000 0000 0000 .....
011450 0000 0000 0000 0000 0000 0000 0000 0000 .....
011460 0000 0000 0000 0000 0000 0000 0000 0000 .....
011470 0000 0000 0000 0000 0000 0000 0000 0000 .....
011480 1000 0027 0644 4F53 4E4F 5445 5350 434C ...'.DOSNOTESPCL
011490 9642 9642 0400 0020 FFFF FFFF FFFF FFFF .B.B.....
0114A0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114B0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114C0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114D0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114E0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114F06 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

"Quellen"-Sektor (Spur 11, Sektor 4)

Das ist der Sektor der die Daten, beginnend ab dem relativen Byte 80 (HEX) enthält, die wir kopieren wollen. Wir wollen 32 Bytes in einen anderen Sektor übertragen.

"Ziel"-Sektor, vorher (Spur 1, Sektor 3)

Das ist der Sektor, in den wir die Daten übertragen und zwar beginnend ab dem relativen Byte "E0" in die nächsten 32 Bytes.

```

001300 E5E5 E5E5 E5E5 E5E5 E5E5 E5D5 E5E5 E5E5 .....
001310 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001320 E5E5 E5E5 E5E5 E5D5 E5E5 E5E5 E5E5 E5E5 .....
001330 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001340 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001350 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001360 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001370 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001380 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001390 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013A0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013B0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013C0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013D0 E5E5 E5E5 E5E5 E5E5 E5E5 E4E5 E5E5 E5E5 .....
0013E0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013F06 E5E5 E5E5 E5E5 E4E5 E5E5 E5E5 E5E5 E5E5 .....

```

Abbildung 2.9 zeigt das "SUPERZAP"-Menü und die Funktionseingabe.

Abb. 2.9

```

INPUT ONE OF THE FOLLOWING INSTRUCTIONS
'DD' OR NULL - DISPLAY DISK SECTOR
'PD' - PRINT MAIN MEMORY
'DM' - DISPLAY MAIN MEMORY
'PM' - PRINT MAIN MEMORY
'VERIFY DISK SECTORS'
'ZERO DISK SECTORS'
'COPY DISK SECTORS'
'DISK BACKUP'
'COPY DISK DATA'
? COPY DISK DATA <ENTER>

```

Die nächste Frage verlangt die Eingabe der "Quelle", des "Ziels" und der Anzahl der Bytes, wie in Abbildung 2.10 gezeigt.

Abb. 2.10

```

PROVIDE SOURCE BASE INFORMATION
RELATIVE DISK # (0 - 3)? 0
TRACK # (HEX) (0 - 22)? 11
SECTOR # (0 - 9)? 4
RELATIVE BYTE # IN SECTOR (HEX, 00-FF)? 80

PROVIDE DESTINATION BASE INFORMATION
RELATIVE DISK # (0 - 3)? 0
TRACK # (HEX) (0 - 22)? 1
SECTOR # (0 - 9)? 3
RELATIVE BYTE # IN SECTOR (HEX, 00-FF)? E0
BYTE COUNT (HEX)? 20
    
```

Wenn die obigen Werte eingegeben sind, um die Daten von einer Position auf eine andere zu kopieren, erhalten wir das Ergebnis, wie in Abbildung 2.11 gezeigt. Vergessen Sie nicht, die Eingabe der Byte-Anzahl erfolgt **hexadezimal** !

Abb. 2.11

```

001300 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001310 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001320 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001330 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001340 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5D5 E5E5 .....
001350 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001360 E5E5 E5E5 E5E5 E5E5 E5D5 E5E5 E5E5 E5E5 .....
001370 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001380 E5E5 E5E5 E5D5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
001390 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013A0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013B0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013C0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013D0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
0013E0 1000 0027 0044 4F53 4E4F 5445 5350 434C ...'.DOSNOTESPCL
0013F0 9642 9642 0400 0020 FFFF FFFF FFFF FFFF .B.B.....
    
```

"Ziel"-Sektor nach Übertragung der Daten (Spur 1, Sektor 3)

Wie Sie sehen können, sind die Daten nun auch in diesem Sektor, aber beginnen an einer anderen Stelle.

2.7 SUPERZAP 3.0

Wenn man meint, man hätte das beste und neueste Programm...schon kommt jemand und verbessert es nochmals. So ist es auch mit SUPERZAP. Das Menue von SUPERZAP 3.0 zeigt nun eine neue Funktion an: "DFS" - DISPLAY FILE' S SECTORS (Anzeige der Sektoren einer Datei).

Damit ist es nun möglich, auf eine Datei zuzugreifen, OHNE wissen zu müssen, wo sie sich physikalisch auf der Diskette befindet. "DFS" arbeitet ähnlich wie "DD", nur daß Sie jetzt einen Dateinamen (FILENAME) und den bzw. die gewünschten relativen Sektoren der Datei eingeben müssen, die Sie bearbeiten wollen.

Um die Funktion aufzurufen, geben Sie "DFS (ENTER)" ein. Sie werden dann gefragt: "FILESPEC". Antworten Sie mit dem File-Namen und beachten Sie, daß ggf. auch ein Schutzwort (PASSWORD) mit angegeben werden muß. Die nächste Meldung fragt Sie nach dem gewünschten relativen Sektor (in HEX); beachten Sie, der erste Sektor ist Sektor "0"!

Das Anzeigeformat unterscheidet sich etwas von dem der "DD"-Funktion und ist in Abb. 2.12 dargestellt. "DFS" verwendet den üblichen wahlfreien (RANDOM) Datenzugriff von BASIC.

Abb. 2.12

"F" zeigt an, daß die "DFS"-Funktion aktiviert ist
 relativer Sektor
 relatives Byte

F00000	FFF4	6832	0093	3A20	4D41	494E	2F44	4953	...	2...:MAIN/DIS
F00010	4B20	4D45	4D4F	5259	2044	554D	502F	4D4F	K.MEMORY.DULP/MC	
F00020	4449	4659	2052	4F55	5449	4E45	2E20	2056	DIFY.ROUTINE...V	
F00030	4552	5349	4F4E	2032	2E30	0000	6964	008D	ERSION.2.0.....	
F00040	2031	3034	3030	0029	6996	0041	24D5	C93A	.10400.)...A\$::	
F00050	208F	2041	24D5	2222	20CA	2031	3430	3A20	...A\$. "...150::	
F00060	203A	9520	4258	D5F6	2841	2429	3A20	9200	...BX..(A\$):...	
F00070	4E69	C800	8F20	4258	20D4	D534	3820	D220	N.....BX...48...	
F00080	4258	D6D5	3537	20CA	2042	58D5	4258	CE34	BX..57...BX.BX.4	
F00090	383A	2092	0072	69FA	008F	2042	58D4	D536	8:.....BX..6	
F000A0	3520	D220	4258	D6D5	3730	20CA	2042	58D5	5...BX..70...BX.	
F000B0	4258	CE35	353A	2092	0080	692C	0142	58D5	BX.55:.....,BX.	
F000C0	CE42	583A	2092	00AF	695E	0193	3A20	2A2A	.BX:.....:..**	
F000D0	2A2A	2A2A	2A2A	2A20	5641	5249	4142	4C45	*****.VARIABLE	
F000E0	2041	4C4C	4F43	4154	494F	4E20	494E	4849	.ALLOCATION.INHI	
F000F0	4249	5445	4400	D569	9001	4432	2528	3129	BITED.....D2%(1)	

Die übrigen Verbesserungen von SUPERZAP 3.0 liegen in der Betriebsart "MOD" (Änderung von Disketten- daten). Der neue Befehl lautet hier:

"ZTnn" (ZERO BYTES)

Überschreiben von nn Bytes mit dem Wert "00", beginnend ab der momentanen Position. Dabei ist nn eine HEX-Zahl kleiner oder gleich "FF".

Der Befehl funktioniert wie "MODnn" dahingehend, daß auf dem Bildschirm erst dann eine Reaktion erfolgt, wenn das Kommando "ZT" eingegeben ist. Nachdem Sie "ZT" eingegeben haben, erscheint die Anzeige "ZT" in Spalte 7 und sobald Sie den HEX-Wert eintippen, wird dieser Wert auch in Spalte 7 angezeigt (Siehe Abb. 2.13).

(Abb. 2.13)

```

F00000Z FFF4 6832 0093 3A20 4D41 494E 2F44 4953 ...2...:MAIN/DIS
F00010T 4B20 4D45 4D4F 5259 2044 554D 502F 4D4F K.MEMORY.DUMP/MO
F000208 4449 4659 2052 4F55 5449 4E45 2E20 2056 DIFY.ROUTINE...V
F00030F 4552 5349 4F4E 2032 2E30 0000 6964 008D ERSION.2.0.....
F00040 2031 3034 3030 0029 6996 0041 24D5 C93A .10400.)...A$...:
F00050 208F 2041 24D5 2222 20CA 2031 3530 3A20 ...A$. ""...150::
F00060 203A 9520 4258 D5F6 2841 2429 3A20 9200 ...BX..(A$):...
F00070 M4E69 C800 8F20 4258 20D4+D534 3820 D220 N....BX...48...
F00080 4258 D6D5 3537 20CA 2042 58D5 4258 CE34 BX..57...BX.BX.4
F00090 383A 2092 0072 69FA 008F 2042 58D4 D536 8:.....BX..6
F000A0 3520 D220 4258 D6D5 3730 20CA 2042 58D5 5...BX..70...BX.
F000B0 4258 CE35 353A 2092 0080 692C 0142 58D5 BX.55:.....,.BX.
F000C0 CE42 583A 2092 00AF 695E 0193 3A20 2A2A .BX:.....:.*
F000D0 2A2A 2A2A 2A2A 2A20 5641 5249 4142 C45 *****.VARIABLE
F000E0 2041 4C46 4F43 4154 494F 4E20 494E 4849 .ALLOCATION.INHI
F000F0 4249 5445 4400 D569 9001 4400 2528 8129 BITED.....D2%(1)

```

Sie erkennen aus der Abbildung, daß das "MOD"-Symbol (M) vor dem relativen Byte "7A" steht und daß "ZT" in Spalte 7 auf "ZT8F" gesetzt ist. Damit werden ALLE Bytes ab dem "MOD"-Symbol bis zum relativen Byte "8F" mit "00" überschrieben, wie in Abb. 2.14 gezeigt.

Diese Funktion ist wirklich eine feine Sache, man muß nun nicht mehr endlose Reihen von "00" eingeben, um z.B. einen "Directory"-Eintrag zu löschen.

Wenn Sie nun aber aus Versehen die falsche Anzahl von Bytes eingegeben haben und Sie die Funktion unterbrechen möchten... Nichts leichter als das, drücken Sie auf eine beliebige, unzulässige Taste (z.B. "P") und die Anzeige antwortet gemäß Abb. 2.15.

Beachten Sie hier die Spalte 7, unten und Sie erkennen den Ausdruck "CHECK". Sie können nun keine weiteren Eingaben vornehmen, bis Sie nicht die Fehlersituation durch (SHIFT) "*"

Abb. 2.14

F00000Z	FFF4	6832	0093	3A20	4D41	494E	2F44	4953	...2...:MAIN/DIS
F00010T	4B20	4D45	4D4F	5259	2044	554D	502F	4D4F	K.MEMORY.DUMP/MO
F000208	4449	4659	2052	4F55	5449	4E45	2E20	2056	DIFY.ROUTINE...V
F00030F	4552	5349	4F4E	2032	2E30	0000	6964	008D	ERSION.2.0.....
F00040	2031	3034	3020	0029	6996	0041	24D5	C93A	.10400.)...A\$...:
F00050	208F	2041	24D5	2222	20CA	2031	3530	3A20	...A\$.""...150:..
F00060	203A	9520	4258	D5F6	2841	2429	3A20	9200	...BX..(A\$):...
F00070	M4E69	C800	8F20	4258	20D4	0000	0000	0000	N.....BX.....
F00080	0000	0000	0000	0000	0000	0000	0000	0000
F00090	+383A	2092	0072	69FA	008F	2042	58D4	D536	8:.....BX..6
F000A0	3520	D220	4258	D6D5	3730	20CA	2042	58D5	5...BX..70...BX.
F000B0	4258	CE35	353A	2092	0080	692C	0142	58D5	BX.55:.....,BX.
F000C0	CE42	583A	2092	00AF	695E	0193	3A20	2A2A	.BX:.....:..**
F000D0	2A2A	2A2A	2A2A	2A20	5641	5249	4042	4C45	*****.VARIABLE
F000E0	2041	4C4C	4F43	4154	494F	4E20	494E	4849	.ALLOCATION.INHI
F000F0	4249	5445	4400	D569	9001	4432	2528	3129	BITED.....D2%(1)

Abb. 2.15

F00000Z	FFF4	6832	0093	3A20	4D41	494E	2F44	4953	...2...:MAIN/DIS
F00010T	4B20	4D45	4D4F	5259	2044	554D	502F	4D4F	K.MEMORY.DUMP/MO
F000208	4449	4659	2052	4F55	5449	4E45	2E20	2056	DIFY.ROUTINE...V
F00030F	4552	5349	4F4E	2032	2E30	0000	6964	008D	ERSION.2.0.....
F00040	2031	3034	3030	0029	6996	0041	24D5	C93A	.10400.)...A\$...:
F00050	208F	2041	24D5	2222	20CA	2031	3530	3A20	...A\$.""...150:..
F00060	203A	9520	4258	D5F6	2841	2429	3A20	9200	...BX..(A\$):...
F00070	M4E69	C800	8F20	4258	20D4+D534	3820	D220	D220	N.....BX...48...
F00080	4258	D6D5	3537	20CA	2042	58D5	4258	CE34	BX..57...BX.BX.4
F00090	383A	2092	0072	69FA	008F	2042	58D4	D536	8:.....BX..6
F000A0	3520	D220	4258	D6D5	3730	20CA	2042	58D5	5...BX..70...BX.
F000B0C	4258	CE35	343A	2092	0080	692C	0142	58D5	BX.55:.....,BX.
F000C0H	CE42	583A	2092	00AF	695E	0193	3A20	2A2A	.BX:.....:..**
F000D0E	202A	2A2A	2A2A	2A20	5641	5249	4142	4C45	*****.VARIABLE
F000E0C	2041	4C4C	4F43	4154	494F	4E20	494E	4849	.ALLOCATION.INHI
F000F0K	4249	5445	4400	D569	9001	4432	2528	3129	BITED.....D2%(1)

gelöscht haben. Dadurch wird der gesamte Befehl aufgehoben und Sie können neu beginnen.

"CHECK" funktioniert genauso bei "MODnn" und zeigt Ihnen eine unzulässige Eingabe an.

Mit "SUPERZAP 3.0" können Sie nun auch Disketten mit 80 Spuren lesen, so daß allen "Superzappern", auch solchen mit den "Mammutlaufwerken" geholfen ist. Ebenso ist es möglich, von einer Diskette mit höherer Spurnzahl auf eine mit kleiner Spurnzahl zu kopieren.



3.0 Andere Hilfsprogramme

Neben SUPERZAP gibt es noch eine ganze Reihe weiterer Hilfsprogramme, die Sie in die Hand bekommen können und die teilweise als Ersatz für SUPERZAP dienen mögen. Jedoch kenne ich kein anderes Programm, das so universell und einfach zu verwenden ist, besonders wenn es um die "Rettung" verlorengegangener Daten geht. Die anderen Programme, die hier erwähnt werden sollen sind:

RSM-2D (Small Systems Software)
MONITOR 3 (ACS)
DEBUG (Radio Shack)
DIRCHECK (Apparat)
LMOFFSET (Apparat)

Es werden sicher viele fragen: "kann ich denn nicht statt SUPERZAP das wirklich nette Programm verwenden, das ich von MICRO-SUPER-80-SOFT-TRON in Kleinkleckersdorf gekauft habe?" Dazu muß ich sagen "und wenn Sie mich schlagen, ich kann nicht alles kennen!" Ich bin gerne bereit, jedes Programm zu prüfen, das mir zur Verfügung steht, aber im Moment kenne ich eben nur die angegebenen.

3.1 "RSM-2D"

"RSM-2D" kommt von Small Systems Software. Es ist gut geschrieben und ohne Fehler. Die Dokumentation ist nicht die beste, aber es gibt genügend schlechtere.

RSM-2D gehört zu einer Familie von Monitorprogrammen für den TRS-80 und basiert auf weit verbreiteten Programmen für S-100 Systeme. Die "2D"-Version erlaubt das direkte Lesen und Schreiben von Disketten-Sektoren. Es enthält auch eine Routine zur Ausgabe auf dem Drucker über das ebenfalls von SSS vertriebene RS-232 Interface. Die Ausgabe über die Parallelschnittstelle ist natürlich auch möglich.

Die beiden Befehle, die zusätzlich bei der Diskettenversion vorhanden sind, lauten "L" (LOAD - Lesen) und "S" (SAVE - Schreiben). "L" lädt einen bestimmten Sektor in einen RAM-Speicherbereich und mit "S" kann man den Inhalt eines Speicherbereiches auf die Diskette in einen Sektor übertragen.

Will man RSM-2D zur "Rettung" von Daten einsetzen, so ist das zwar möglich, aber recht mühsam. Das rührt daher, daß Sie abwechselnd im RAM und auf Diskette arbeiten müssen. Darüberhinaus haben Sie nicht den Vorteil einer formatierten Anzeige, die Ihnen gleichzeitig die Daten in HEX und in ASCII in einer Form Sektor für Sektor bringt. Sie müssen sich außerdem stets daran erinnern, wo die Grenzen eines Sektors im RAM liegen, um Daten richtig lesen und schreiben zu können.

Das Programm ist zuverlässig und Sie werden keine Schwierigkeiten außer den gezeigten Unbequemlichkeiten bekommen.

3.2 "MONITOR 3"

Auch hier handelt es sich um ein gut geschriebenes Monitor-Programm, aber es hat keine ausreichenden Möglichkeiten, der Datenübertragung von und auf Diskette, die bei der Aufbereitung von Daten von Nutzen sind. Ich habe zwar auch schon einen "MONITOR 4" angeboten gesehen, kenne aber dieses Programm nicht und kann daher auch nicht sagen, ob damit Diskettenoperationen möglich sind.

3.4 "DEBUG"

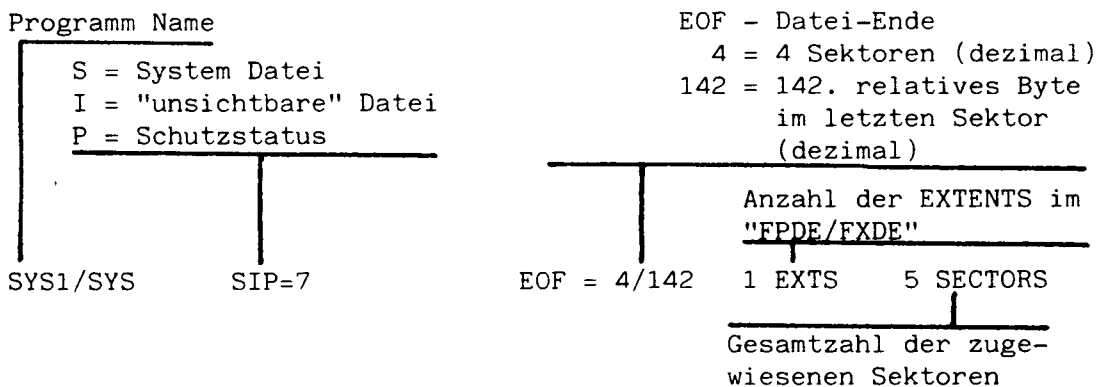
Das ist das normale Monitorprogramm, das Radio Shack mit TRS-DOS ausliefert und es gehört zweifellos zu den besseren Programmen von Radio Shack. Daher verdient es Beachtung, obwohl es keine Möglichkeiten zum Lesen und Schreiben von Disketten bietet. Somit ist es nicht geeignet, um Disketten zu bearbeiten, man kann aber sehr gut ein Maschinenprogramm im RAM bearbeiten.

(Anm. d. Ü.: Das neue DEBUG für TRS-80 Model III besitzt nun auch diese Möglichkeiten)

3.4 "DIRCHECK"

Hier handelt es sich um ein Hilfsprogramm, das Apparat mit seinem NEWDOS ausliefert. Es ist ein unschätzbare Hilfsmittel zum Überprüfen einer "Directory" auf Fehler. Darüberhinaus druckt es eine alphabetische Liste der Directory-Einträge, das "END OF FILE" (EOF - Marke für Datei-Ende), die Anzahl der Zusatzblöcke (EXTENTS), die eine Datei belegt und die Anzahl der belegten Sektoren (statt GRANULES - Halbspuren). In Abb. 3.2 sehen Sie einen Musterausdruck von "DIRCHECK", wie er aussehen kann, wenn Fehler in der Directory vorliegen. In Abb. 3.1 finden Sie die Erläuterungen zu den einzelnen Informationen.

Abb. 3.1



NEWDOS+ 07/15/79

64 BAD "HIT" SECTOR BYTE
 BASIC/CMD 84 PRIMARY ENTRY HAS BAD CODE IN "HIT" SECTOR
 00 ***** GRANULE FREE, BUT ASSIGNED TO FILE(S)
 00 BOOT/SYS
 1E ***** GRANULE LOCKED OUT, BUT FREE
 1F ***** GRANULE LOCKED OUT, BUT FREE
 20 ***** GRANULE FREE, BUT ASSIGNED TO FILE(S)
 84 BASIC/CMD
 36 ***** GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE
 37 ***** GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE
 65 ***** GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
 83 SUPERZAP/PCL
 C7 DISKORG/PCL

BASIC/CMD	I	EOF = 6/231	2 EXTS	10 SECTORS
BOOT/SYS	SIP=6	EOF = 19/119	2 EXTS	5 SECTORS
COPY/CMD	IP=6	EOF = 4/253	1 EXTS	5 SECTORS
DIR/SYS	SIP=5	EOF = 10/0	1 EXTS	10 SECTORS
DIRCHECK/CMD		EOF = 12/136	3 EXTS	15 SECTORS
DISKORG/PCL		EOF = 18/211	2 EXTS	20 SECTORS
FORMAT/CMD	IP=6	EOF = 14/8	1 EXTS	15 SECTORS
SYS0/SYS	SIP=7	EOF = 12/93	1 EXTS	15 SECTORS
SYS11/SYS	QIP=7	EOF = 4/142	1 EXTS	5 SECTORS
SYS12/SYS	SIP=7	EOF = 4/236	1 EXTS	5 SECTORS
SYS13/SYS	SIP=7	EOF = 3/9	1 EXTS	5 SECTORS
SYS2/SYS	SIP=7	EOF = 4/52	1 EXTS	5 SECTORS
SYS3/SYS	SIP=7	EOF = 4/76	1 EXTS	5 SECTORS
SYS4/SYS	SIP=7	EOF = 4/186	1 EXTS	5 SECTORS
SYS5/SYS	SIP=7	EOF = 4/203	1 EXTS	5 SECTORS
SYS6/SYS	SIP=7	EOF = 13/33	1 EXTS	5 SECTORS
SUPERZAP		EOF = 21/38	4 EXTS	25 SECTORS

43 FREE GRANULES 0 LOCKED-OUT GRANULES

NEWDOS DIRECTORY CHECK & LIST COMPLETED

Neben dem Namen jeder als fehlerhaft erkannten Datei erkennen Sie in Abb. 3.2 eine Zahl. Dieser Wert entspricht dem "DEC" (Directory Entry Code - Directory Schlüssel) für den betreffenden Datei-Namen. Mehr zu dem Begriff "DEC" finden Sie in Kapitel 6.

Es ist leicht einzusehen, daß diese Zusammenfassung aller Fehler in der "GAT" (Granule Allocation Table - Tabelle der belegten Halbspuren) und in der "HIT" (Hash Index Table - Tabelle mit Suchschlüsseln für alle Eintragungen im Directory) eine große Hilfe beim Aufspüren von Fehlern ist.

Sie sollten in regelmäßigen Abständen "DIRCHECK" bei allen Ihren Disketten anwenden, um rechtzeitig zu erkennen, ob ein hinterlistiger Fehler bereits auf der Lauer liegt, um Ihre wichtigen Daten und Programme durcheinanderzubringen.

Falls ein Fehler im Programm "BOOT/SYS" vorliegt, die Directory-Spur nicht mehr lesegeschützt ist oder ein Paritätsfehler in "BOOT/SYS" bzw. im Directory vorliegt, bricht "DIRCHECK" mit folgender Meldung ab:

```
FUNCTION TERMINATED DUE TO ERROR  
(Funktion wegen Fehler abgebrochen).
```

Sie sind aber immer noch in der Lage, alle Sektoren mit "SUPERZAP" zu lesen und müssen die vorhandenen Fehler korrigieren, bevor Sie "DIRCHECK" anwenden können. Das entsprechende Verfahren ist in Kapitel 10 erläutert.

Die folgenden Fehler werden von "DIRCHECK" erkannt und angezeigt:

3.4.1 BAD "HIT" SECTOR BYTE (Fehlerhaftes Byte in der "HIT")

In einem Sektor der "HIT" existiert ein "HASH"-Code, obwohl dort keiner vorhanden sein dürfte. Die Zahl ganz links gibt die relative Lage dieses fehlerhaften Code-Bytes im "HIT"-Sektor an. Ersetzen Sie den widersprüchlichen Code durch "00". Die Zahl neben dem Datei-Namen ist der "DEC" (Siehe Kapitel 6, Abb. 6.13).

3.4.2 PRIMARY ENTRY HAS BAD CODE IN "HIT" SECTOR (Haupteintrag hat einen fehlerhaften Code in der "HIT")

In der "HIT" existiert ein "HASH"-Code, der nicht zum entsprechenden Directory-Eintrag passt. Die Zahl ganz links gibt wieder die relative Lage dieses Codes in der "HIT" an. Ersetzen Sie diesen Code durch den richtigen Wert.

3.4.3 GRANULE FREE BUT ASSIGNED TO FILE(S) (Halbspur nicht belegt, jedoch einer Datei zugeordnet)

Eine Halbspur ist als "belegt" gekennzeichnet, obwohl keine Datei diese Halbspur benutzt. Die Zahl links gibt die relative Lage des fehlerhaften Eintrages in der "GAT" (GRANULE ALLOCATION TABLE) an. Ersetzen Sie den widersprüchlichen Code durch den richtigen.

3.4.4 GRANULE ALLOCATED BUT ASSIGNED TO MULTIPLE FILES (Halbspur belegt, aber mehreren Dateien zugeordnet)

Mehr als eine Datei verwenden dieselbe Halbspur (5 Sektoren langer Abschnitt). Der letzte "SAVE"- oder "PUT"-Befehl kann Daten ÜBER die vorherigen Daten geschrieben haben.

Stellen Sie fest, welche Datei diese Halbspur zuletzt verwendet hat. Kopieren Sie diese Datei auf eine andere Diskette (COPY). Löschen Sie diese Datei nun auf der alten Diskette (KILL). Laden Sie die vorangehende Datei, die dieses "Granule" benutzt hat, berichtigen Sie die fehlerhaften Daten und speichern Sie die Datei auf einer anderen Diskette. Berichten Sie dann die "GAT".

Die Zahl links gibt die relative Lage des fehlerhaften Eintrages in der "GAT" an.

3.4.5 GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE (Halbspur als belegt gekennzeichnet, aber keiner Datei zugeordnet)

Ein "Granule" ist als "belegt" gekennzeichnet, wird aber von keiner Datei benutzt. Korrigieren Sie den widersprüchlichen Code in der "GAT". Die Zahl ganz links gibt wieder die relative Lage dieses Codes in der "GAT" an.

3.4.6 GRANULE LOCKED OUT BUT FREE (Ein "Granule" ist gesperrt, aber unbenutzt)

Ein "Granule" ist als gesperrt gekennzeichnet, und kann vom System nicht benutzt werden. Korrigieren Sie den widersprüchlichen Eintrag in der "GRANULE LOCK-OUT TABLE" (Tabelle mit Codes für gesperrte Halbspuren). Die Zahl links kennzeichnet die relative Lage des Codes in der LOCK-OUT Tabelle.

3.5 "LMOFFSET"

Der eigentliche Zweck dieser Routine ist es, ein Programm zu laden und auszuführen, das "normalerweise" nicht zusammen mit DOS geladen werden kann, da sich die Adressbereiche überschneiden.

"LMOFFSET" gibt Ihnen zunächst an, wohin das Programm lädt und welches die Adresse für den Programmstart ist. Abb. 3.4 zeigt die Folge der Ein- und Ausgaben von "LMOFFSET".

Dieses Programm sagt Ihnen etwas über eine Datei aus, es sagt Ihnen nicht, wo auf der Diskette sich diese Datei befindet. Es hilft Ihnen jedoch dabei, ein Maschinenprogramm im RAM-Speicher zu lokalisieren, so daß an ihm Änderungen vorgenommen werden können, bevor es auf Diskette zurückgeschrieben wird.

Es hilft Ihnen auch, wenn Sie ein Programm disassemblieren wollen und Sie dazu den Adressbereich wissen müssen, den das Programm belegt.

-----ACHTUNG-----
Wenn Sie "LMOFFSET" in dieser Weise gebrauchen, brechen Sie das Programm ab, wenn Sie die Frage: "NEW LOAD BASE ADDRESS (HEX) ?" erhalten (durch RESET). "LMOFFSET" ergänzt sonst einen sogenannten "Anhang", der bewirkt, daß das Programm nun in einen anderen Bereich als den gewünschten lädt !!

Mit NEWDOS und einem Drucker können Sie durch die "JKL"-Funktion die angezeigten Daten für spätere Verwendung ausdrucken oder machen Sie sich entsprechende Notizen.

```

APPARAT LOAD MODULE OFFSET PROGRAM, VERSION 1.1
SOURCE FROM DISK OR TAPE? REPLY "D" OR "T"? D
SOURCE FILESPEC?BASIC/CMD
MODULE LOADS TO 4D00-6431
MODULE OVERLAPS DOS RAM (4000-51FF)
MODULE LOAD WILL OVERLAP "CMD" PROGRAM AREA (5200-6FFF)
ENTRY POINT = 5BAD
NEW LOAD BASE ADDRESS (HEX)?

```

4.0 Betriebssysteme

Es folgt nun eine kurze Übersicht über verschiedene Betriebssysteme für den TRS-80, die zum Zeitpunkt des Entstehens dieses Buches verfügbar waren. Mit diesem Kapitel werde ich mich nicht sehr lange aufhalten und bedenken Sie bitte, daß ich hier meine Meinung ausdrücke.

4.1 "TRSDOS 2.1"

Mit Ausnahme einiger weniger ganz unglücklicher Seelen, die mit TRSDOS 2.0 begonnen haben, ist TRSDOS 2.1 das Betriebssystem, mit dem die meisten von uns ihre ersten Erfahrungen gemacht haben und durch das sich eine gewisse Haßliebe zwischen den Benutzern und dem TRS-80 entwickelt hat. Wenn man die kurze Lebensdauer von TRSDOS 2.0 berücksichtigt, kann man sagen daß TRSDOS 2.1 das erste Disketten-Betriebssystem für den TRS-80 war.

TRSDOS 2.1 brachte viele Probleme mit sich, obwohl das von Radio Shack eigentlich nie so richtig zugegeben wurde. Es ist brauchbar für die meisten einfachen Anwendungen und einige ernsthafte Dinge, wenn Sie darauf vorbereitet sind, daß Ihnen manchmal eine Datei verlorengehen kann.

Das "Retten" von Daten ist ohne Besonderheiten möglich.

4.2 "TRSDOS 2.2"

TRSDOS 2.2 stellt eine riesige Verbesserung gegenüber 2.1 dar. Die meisten Fehler sind behoben, obwohl immer noch Fehler auftreten können. Der größte Mangel ist meiner Meinung, daß es nicht die vielseitigen Hilfsprogramme, wie bei NEWDOS enthält.

Bei der "Datenrettung" müssen Sie beachten, daß unter 2.2 nach dem Löschen einer Datei mit "KILL" der gesamte Directory-Eintrag mit Nullen überschrieben wird. Es gibt nicht einen bekannten Grund, warum diese Eigenschaft eingebaut wurde. Nachdem es unter TRSDOS 2.2 kein Programm gibt, um sich den Inhalt von Disketten anzusehen, vermute ich, daß Radio Shack die vielen "Superzapper" daran hindern will, zuviel herauszufinden. Wenn Sie allerdings etwas "retten" wollen, so ist das nicht unmöglich, aber ein ziemlicher Umstand, da Sie die ganze Diskette nach der gewünschten Datei absuchen müssen.

Alleine schon aus diesem Grund würde ich dieses DOS nicht für eine ernsthafte Anwendung einsetzen, bei der unter Umständen die Notwendigkeit bestehen könnte, eine "ge-KILL-te" Datei wieder zum Leben zu erwecken.

Die Wiedergewinnung von Daten läuft ansonsten ganz normal ab, sofern es sich um formatierte Datendisketten oder Systemdisketten handelt.

4.3 "VTOS 3.0"

Das ist die Version des TRSDOS 2.2 von Randy Cook, dem Autor von TRSDOS 2.1 und ich habe Grund zur Annahme, daß er auch hinter TRSDOS 2.2 steckt. VTOS hat einige gute Funktionen, aber ist nach meiner Ansicht sehr problematisch zu gebrauchen, da es gegen "BACKUP" geschützt ist. In der Version, die ich getestet habe, arbeiteten einige Funktionen nicht so, wie in der Dokumentation beschrieben. Ich bin sicher, daß das inzwischen behoben ist. Insgesamt ist das Programm gut und die zugrundeliegenden Konzepte sind hervorragend. Bisher habe ich es noch nicht so viel benutzt, um eventuelle Fehler zu erkennen.

Wenn Sie Daten oder Dateien wiedergewinnen wollen, die auf eine VTOS Systemdiskette geschrieben wurden, werden Sie über die notwendigen Verfahren nicht sehr erfreut sein.

Das liegt an der Tatsache, daß Sie die Daten nicht auf eine andere Diskette übertragen können!

Trotz aller guten Eigenschaften des Programms kann ich es aus den obigen Gründen nicht für ernsthafte Anwendungen empfehlen. Die "Rettung" von Daten auf einer VTOS-Diskette ist sehr schwierig. Sie müssen dazu eine neue Diskette formatieren und darauf mit der "BACKUP"-Funktion von "SUPERZAP" eine Kopie der fehlerhaften Diskette herstellen. Sie werden nicht in der Lage sein, den Sektor 4 der Spur 0 zu kopieren, sie müssen diesen Sektor überspringen, wenn SUPERZAP versucht, ihn zu lesen. Nachdem Sie auf dieser Kopie Ihre Korrekturen vorgenommen haben, ist es notwendig, daß Sie diese Diskette wieder auf eine VTOS Systemdiskette übertragen, die Sie wiederum von der Originaldiskette herstellen, die Ihnen Mrs. COOKs Sohn, Randy geliefert hat.

VTOS 3.0 arbeitet nicht, wenn der Sektor 4 in Spur 0 formatiert ist! So hat Randy Cook seine Software vor Raubkopien geschützt. Das ist zwar eine sehr gute Idee, erschwert die Sache für uns aber sehr stark. Gut zu verwenden ist VTOS 3.0 für die Anwender, die nur den Computer einschalten wollen und automatisch ein bestimmtes Programm ausführen möchten.

4.4 "NEWDOS 2.1"

Es funktioniert! Bisher sind keine Fehler in der derzeitigen Ausgabe bekannt geworden und es führt alle Funktionen aus, von denen Radio Shack sagt, daß sie nicht möglich sind. NEWDOS berichtigt jeden bekannten Fehler von TRSDOS und besitzt insgesamt über 200 Erweiterungen, Berichtigungen oder Verbesserungen.

NEWDOS ist sowohl auf den Programmierer als auch auf den Anwender ausgerichtet. In der NEWDOS+ Ausgabe sind die hilfreichen Programme wie "SUPERZAP", "DIRCHECK", "LMOFFSET" und andere enthalten. Diese Hilfsprogramme unterstützen den Anwender in großem Maße und sind sehr wichtig, wenn Sie Daten wiedergewinnen wollen.

Die "Rettung" von Daten ist bei Daten- und Systemdisketten ohne große Probleme möglich.

4.5 Zukünftige Betriebssysteme

Nun schlägt die Stunde den Wahrsagern! Ich habe keine zuverlässigen Informationen darüber, was Randy Cook oder Radio Shack planen. Vermutlich konzentriert sich Radio Shack im Moment auf die Entwicklung des neuen TRS-80 Model II.

Randy Cook ist offensichtlich nicht mehr länger mit R.S. zusammen, da er seine eigene Firma, Virtual Technology eröffnet hat und mag vielleicht an einem neuen VTOS arbeiten. Randy Cook kennt natürlich den TRS-80 sehr gut und es dürfte nicht verwundern, wenn er in diesem Bereich weiter aktiv bleibt.

Mit Apparat habe ich einen engen Kontakt und kenne daher einige ihrer zukünftigen Pläne. Zur Zeit ist NEWDOS als 35- und 40-Spur Version lieferbar und eine 77-Spur Version wird bald fertig sein.

Ein "SUPERDOS" ist in Arbeit, das Ihnen den Hut vom Kopf reißen wird. Ich hatte Gelegenheit, einige der neuen Funktionen zu sehen, besonders die neuen Möglichkeiten bei der Bearbeitung von Dateien. Dieses DOS kann auch gleichzeitig mit unterschiedlichen Laufwerken (35, 40 oder 77 Spuren) arbeiten. Ohne weitere Einzelheiten zu nennen, muß man sagen, daß "SUPERDOS" ein Riesenschritt vorwärts sein wird.

(Anm. d. Ü.: H.C. Pennington spricht hier von NEWDOS 80, das inzwischen lieferbar ist.)

-----ACHTUNG-----

Beim Schreiben dieses Buches ist ein neuer Fehler in TRSDOS 2.2 bekannt geworden:

Wenn Dateien auf zwei unterschiedlichen Laufwerken geöffnet werden (mit OPEN) und eine der beiden Dateien wird abgeschlossen (mit CLOSE), so ist es möglich, daß diese Datei gelöscht wird!

Alle nachfolgenden CLOSE-Befehle werden richtig behandelt.

Dieser Fehler tritt nur zeitweilig auf.

5.0 Organisation von Disketten

Im Handbuch des TRSDOS 2.1 wird uns gesagt, daß wir 67 "GRANULES" (Halbspuren) an freiem Speicherplatz auf einer formatierten Diskette zur Verfügung haben und etwas weniger auf einer Diskette mit DOS. Hier folgt eine Übersicht über die verschiedenen Disketten:

Spuren:

TRSDOS 2.1	35
NEWDOS 2.2	35 oder 40
VTOS 3.0	35
SUPERDOS 1.0	18 bis 80
(Anm.: mit SUPERDOS 1.0 können Sie Laufwerke unterschiedlicher Spurzahl mischen.)	

Sektoren pro Spur	10
Sektoren pro Diskette (35 Spuren)	350
(40 Spuren)	400
(77 Spuren)	770

Sektoren pro "Granule"	5
Bytes pro Sektor	256
Nutzbare Bytes/Sekt. (TRSDOS 2.1)	255
(TRSDOS 2.2)	256
(NEWDOS 2.1)	255
(VTOS 3.0)	256
(SUPERDOS 1.0)	256

Bytes/Diskette (35 Spuren)	89.600
(40 Spuren)	102.400
(77 Spuren)	197.120

Nutzbare Bytes/Diskette (35 Spuren)	83.060
-------------------------------------	--------

Nutzbare Sektoren für die Datenspeicherung (35 Spuren)	335
"Granules"/Diskette (35 Spuren)	70
Nutzbare "Granules" bei einer Diskette ohne DOS (35 Spuren)	67

Sie können die obigen Angaben mit wenig Mühe nachrechnen. Jede Spur, von denen uns normalerweise 35 Stück zur Verfügung stehen, hat 10 Sektoren mit je 256 Byte. Damit erhalten wir 350 Sektoren auf einer Diskette und $350 \times 256 = 89.600$ Byte Speicherkapazität.

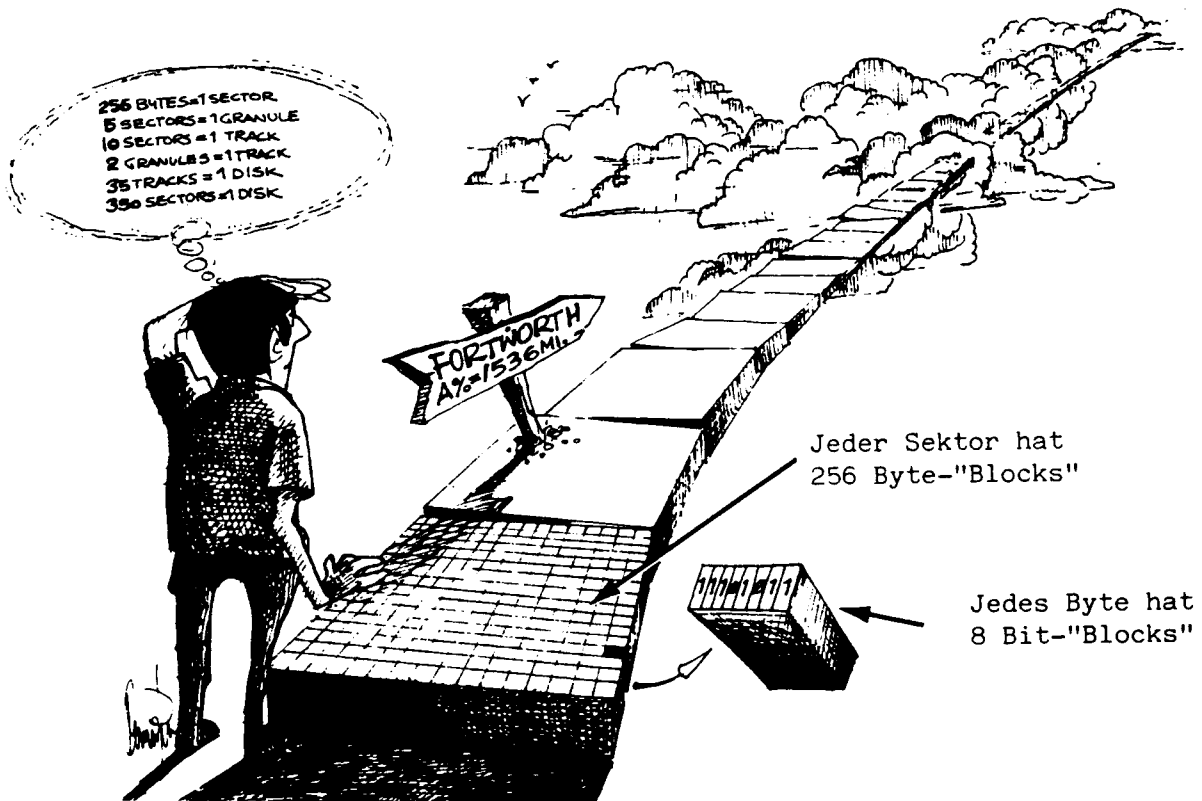
Der Urlader (BOOT/SYS) und das Directory belegen auf jeder Diskette 15 Sektoren. "BOOT/SYS" liegt physikalisch auf Spur 0 in den Sektoren 0 bis 4. Die Directory liegt in der Spur 11 (HEX) oder 17 (dez.) und belegt die gesamte Spur (Sektor 0...9).

Die TRSDOS-Systemprogramme beanspruchen einen großen Teil der Speicherkapazität und lassen uns nur noch 58.880 Bytes frei. Diese Systemprogramme können unter TRSDOS 2.1 nicht gelöscht werden. Als Ergebnis daraus wird dem BASIC-Programmierer wertvoller Speicherplatz weggenommen, der von Routinen belegt ist, die er nie braucht. Außerdem kann es passieren, daß er z.B. unerwartet in

DEBUG landet. Einige Kapitel später werden Sie wissen, wie man ein Schutzwort (PASSWORD) entfernt und können dann die unerwünschten Systemprogramme löschen (KILL). Wenn Sie allerdings NEWDOS verwenden, haben Sie diese Probleme nicht. Es entfällt nicht nur der unerwartete "Absprung" in DEBUG, sondern Sie haben auch mehr Speicherplatz zur Verfügung.

Zurück zum Geschäft! In diesem Buch finden Sie sehr oft den Begriff "relatives Byte". Stellen Sie sich dazu einmal die Diskette aus 350 Blöcken (Sektoren) aufgebaut vor, die alle aneinandergereiht sind (Siehe Abb. 5.1).

Abb. 5.1



Zehn Blöcke bilden jeweils eine Spur. Jeder Block besteht aus 256 kleineren Blöcken, den Bytes. Das erste Byte in der oberen linken Ecke ist das relative Byte 0 eines Sektors.

Wenn Sie bis zum 16. Byte weiterzählen, gelangen Sie in die rechte obere Ecke unseres 16 x 16 Byte-Blocks. Dieses Byte ist das 15. relative Byte. Das 16. relative Byte finden Sie in der nächsten Zeile, ganz links, das 31. relative Byte in dieser Zeile ganz rechts. Dieser Wahnsinn setzt sich fort bis in die rechte untere Ecke, in der wir das relative Byte 255 finden.

Um die Sache noch weiter zu zerlegen, können wir sagen, daß jeder Byte-Block aus acht kleineren Elementen besteht, den Bits. Jedes Bit kann nur zwei unterschiedliche Werte speichern, "0" oder "1".

Stellen Sie sich nun weiter vor, daß jeweils zehn der großen Sektor-Blöcke eine Spur bilden. Diese Spuren sind von "0" bis "22" (HEX) nummeriert. Einige dieser Spuren werden System-Programmen fest zugeordnet. In Abb. 5.2 finden Sie eine Übersicht der einzelnen Spuren und der Bereiche, die bestimmten Programmen fest zugeordnet sind. Alle Programme mit dem Kenner "/SYS" sind Programme dieser Art. "FORMAT/CMD", "BASIC/CMD" und "BACKUP/CMD" sind zwar ebenso wichtig, jedoch nicht an einen festen Platz auf der Diskette gebunden.

Tatsache ist, daß nur ganz bestimmte, wenige Bereiche auf der Diskette bestimmte Informationen enthalten MÜSSEN. Diese Bereiche sind "BOOT/SYS", "SYSO/SYS" und "DIR/SYS". "BOOT/SYS" muß immer auf Spur 0 in den Sektoren 0...4 liegen, "SYSO/SYS" wird auf Spur 0, Sektor 5 erwartet und "DIR/SYS" steht fest auf Spur 11 (HEX), ab Sektor 0.

"DIR/SYS" kann prinzipiell auch auf eine andere Stelle gebracht werden, muß aber immer "lesegeschützt" sein. Wenn das Directory an anderer Stelle liegt, benötigt "SAVE" eine Weile mehr Zeit, bis die neue Lage von "DIR/SYS" gefunden ist, da es normalerweise Spur 11 (HEX) anspricht. Eventuell funktioniert die Sache und die Daten werden an die richtige Stelle gebracht. Man kann den Vorgang beschleunigen, wenn man in das relative Byte "02" in "BOOT/SYS", Spur 0, Sektor 0 die Spurnummer (HEX) schreibt, unter der nun "DIR/SYS" zu finden ist.

"BOOT/SYS" ist eigentlich kein richtiges Programm sondern nur eine "Tabelle", die automatisch geladen wird, wenn man das System einschaltet oder "RESET" betätigt. Man findet auch die Bezeichnung "IPL" (Initial Program Load - Programm-Urlader). Das ist der Computerjargon für den Begriff "gib Gas, Otto!"

In Abb. 5.2 sehen Sie die Übersicht einer typischen System-Diskette (TRSDOS 2.1). Sie werden erkennen, daß die Systemprogramme in Gruppen zusammengefasst sind. Das ist nicht unbedingt erforderlich, Sie können Systemprogramme (außer BOOT/SYS, SYSO/SYS und DIR/SYS) beliebig auf der Diskette verteilen.

Bei NEWDOS muß außerdem SYS13/SYS, falls vorhanden, an einer bestimmten Stelle liegen.

Programme wie z.B. BASIC/CMD oder FORMAT/CMD können auf Ihrer Diskette durchaus an einer anderen Stelle stehen, als hier gezeigt, besonders dann, wenn Sie diese Programme von einer anderen Diskette mit "COPY" kopiert haben.

Die Belegung von Speicherplatz auf der Diskette erfolgt immer in Gruppen zu fünf Sektoren, den sogenannten "Granules".

Übersicht der TRSDOS 2.1 Diskette

Spurnummer HEX - dez.	Granule HEX	Inhalt der Spur
		Sektoren 0...4 Sektoren 5...9
		-----Granule----- -----Granule-----
0	- 0	0 & 1 : <----BOOT/SYS----> : <----SYS0/SYS----> :
1	- 1	2 & 3 : <----SYS0/SYS----> : <----SYS0/SYS----> :
2	- 2	4 & 5 : <--FORMAT/CMD----> : <--FORMAT/CMD----> :
3	- 3	6 & 7 : <--FORMAT/CMD----> : <--BACKUP/CMD----> :
4	- 4	8 & 9 : <--BACKUP/CMD----> : <--BACKUP/CMD----> :
5	- 5	A & B : <-----FREE-----> : <-----FREE-----> :
6	- 6	C & D : <-----FREE-----> : <-----FREE-----> :
7	- 7	E & F : <-----FREE-----> : <-----FREE-----> :
8	- 8	10 & 11 : <-----FREE-----> : <-----FREE-----> :
9	- 9	12 & 13 : <-----FREE-----> : <-----FREE-----> :
A	- 10	14 & 15 : <-----FREE-----> : <-----FREE-----> :
B	- 11	16 & 17 : <-----FREE-----> : <-----FREE-----> :
C	- 12	18 & 19 : <-----FREE-----> : <-----FREE-----> :
D	- 13	1A & 1B : <-----FREE-----> : <-----FREE-----> :
E	- 14	1C & 1D : <-----FREE-----> : <-----FREE-----> :
F	- 15	1E & 1F : <-----FREE-----> : <-----FREE-----> :
10	- 16	20 & 21 : <----SYS1/SYS----> : <----SYS2/SYS----> :
11	- 17	22 & 23 : <----DIR/SYS----> : <----DIR/SYS----> :
12	- 18	24 & 25 : <----SYS3/SYS----> : <----SYS4/SYS----> :
13	- 19	26 & 27 : <----SYS5/SYS----> : <----SYS6/SYS----> :
14	- 20	28 & 29 : <----SYS6/SYS----> : <----SYS6/SYS----> :
15	- 21	2A & 2B : <--BASIC/CMD----> : <--BASIC/CMD----> :
16	- 22	2C & 2D : <--BASIC/CMD----> : <--BASIC/CMD----> :
17	- 23	2E & 2F : <-----FREE-----> : <-----FREE-----> :
18	- 24	30 & 31 : <-----FREE-----> : <-----FREE-----> :
19	- 25	32 & 33 : <-----FREE-----> : <-----FREE-----> :
1A	- 26	34 & 35 : <-----FREE-----> : <-----FREE-----> :
1B	- 27	36 & 37 : <-----FREE-----> : <-----FREE-----> :
1C	- 28	38 & 39 : <-----FREE-----> : <-----FREE-----> :
1D	- 29	3A & 3B : <-----FREE-----> : <-----FREE-----> :
1E	- 30	3C & 3D : <-----FREE-----> : <-----FREE-----> :
1F	- 31	3E & 3F : <-----FREE-----> : <-----FREE-----> :
20	- 32	40 & 41 : <-----FREE-----> : <-----FREE-----> :
21	- 33	42 & 43 : <-----FREE-----> : <-----FREE-----> :
22	- 34	44 & 45 : <-----FREE-----> : <-----FREE-----> :

TRSDOS 2.1 und 2.2 weisen immer gleich zwei Granules zu. Das ist der Grund, warum Sie so schnell eine volle Diskette bekommen, wenn Sie viele unterschiedliche, aber kleine Programme oder Datenblöcke auf einer Diskette speichern.

NEWDOS weist immer nur ein Granule gleichzeitig zu.

Sie können das überprüfen, indem Sie ein einzeiliges BASIC Programm auf die Diskette schreiben. Bevor Sie das erledigen, sehen Sie sich mit "SUPERZAP" die "GAT" (Granule Allocation Table - Tabelle der zugewiesenen Granules) an. Führen Sie dann den "SAVE" durch und schauen Sie sich die "GAT" erneut an. In Kapitel 6 werden wir die Bedeutung des "GAT"-Sektors erläutern, so daß Sie in der Lage sein werden, das hier gewonnene Ergebnis zu interpretieren.

Der Schlüssel zum Finden von Daten auf der Diskette ist das Directory (Disketten-"Inhaltsverzeichnis"). Selbst das Betriebssystem kann ohne Directory keine Information von der Diskette lesen.

Nun, nachdem wir bereits die Grundlagen der Diskettenorganisation kennen, werden wir uns das Directory genauer ansehen. Ich werde Ihnen zeigen, was jedes einzelne Byte bedeutet, was es bewirkt und wie man es benutzt, um irgendeine Information zu finden, genau wie es das Betriebssystem auch durchführt.

Das Directory liegt auf Spur 17 (11 HEX). Es wird von 10 Sektoren gebildet, die je 256 Byte enthalten. Damit hat das Directory eine Länge von 2.560 Byte, in denen Daten gespeichert werden können. Sie werden hier kaum ein unbenutztes Byte finden. Abb. 6.3 zeigt eine Übersicht der Directory.

Der kleinste Speicherbereich, der einer Datei zugeordnet werden kann, wird "Granule" oder "Halbspur" genannt. (Anm. d. Ü.: "Granule" ist eigentlich ein Maßbegriff aus der Zigarrenindustrie.) Wir können auch sagen:

5 Sektoren = 1 Granule
2 Granules = 1 Spur.

Wenn Sie das Kommando "FREE" eingeben, erhalten Sie eine Anzeige, ähnlich Abb. 6.1.

Damit haben wir dieses Thema auch erschlagen und wenden uns nun dem Directory zu. (Einen Ausdruck der verschiedenen Directory-Spuren von TRSDOS, NEWDOS und VTOS finden Sie im Anhang B.)

Wir werden nun jeden Sektor und jeden Eintrag einzeln besprechen.

"GAT Sektor" - Sektor 0

"GAT" ist die Abkürzung für "Granule Allocation Table" - Tabelle der zugewiesenen Granules. Dieser Sektor enthält die vollständige Information, die DOS benötigt, um den Dateien Speicherplatz zuweisen zu können. In diesem Sektor werden auch gesperrte (locked out) Spuren markiert.

Abb. 6.6 zeigt den "GAT"-Sektor mit einer Erläuterung der verschiedenen Abschnitte in diesem Sektor.

Sie werden aus Abb. 6.2 erkennen, daß die ersten 35 Byte den 35 Spuren entsprechen und Werte wie "FF", "FC", "FD" und "FE" enthalten können. Ein "FF" in einem dieser Bytes bedeutet, daß die Spur belegt ist. Bei "FE" ist das erste Granule frei, bei "FD" ist das zweite Granule unbesetzt und bei "FC" sind beide Granules verfügbar. (Siehe auch Abb. 6.4).

Abb. 6.1

DRIVE 0 --	TRSDOS	11/27/78	41 files	42 GRANS
DRIVE 1 --	TRSDOS	01/01/79	33 files	6 GRANS

GAT SECTOR

Abb. 6.2

```

011000  FFCF FCFC FCFC FCFF FEFE FCFD FCFC FCFC .....
011010  FCFF FCFC FFCF FEFD FCFD FDFC FCFC FEFC .....
011020  FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF .....
011030  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
011040  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
011050  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
011060  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC .....
011070  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC .....
011080  FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF .....
011090  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
0110A0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
0110B0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
0110C0  FFFF FFFF FFFF FFFF FFFF FF21 0000 E042 .....!...B
0110D0  5452 5344 4F53 2020 3034 2F30 312F 3739 NOTES...04/01/79
0110E0  0D0D FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....
0110F06 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF .....

```

Beginnend mit dem relativen Byte "60" (siehe Abb. 6.2) finden Sie eine Wiederholung der ersten drei Zeilen dieses Sektors. Dieser Bereich enthält die Information, welche Spuren gesperrt sind. Ab Byte "60" sehen Sie 35 mal "FC". Das heißt, alle 35 Spuren sind für das Betriebssystem verfügbar. Falls hier in einer Position ein "FF" steht UND entsprechend auch in der "GAT" ein "FF" zu finden ist, dann ist die betreffende Spur gesperrt.

Die drei Bytes, beginnend mit Byte "CB" enthalten "21 00 00". Was auch immer diese Codes bedeuten mögen, sie werden vom System nicht benutzt.

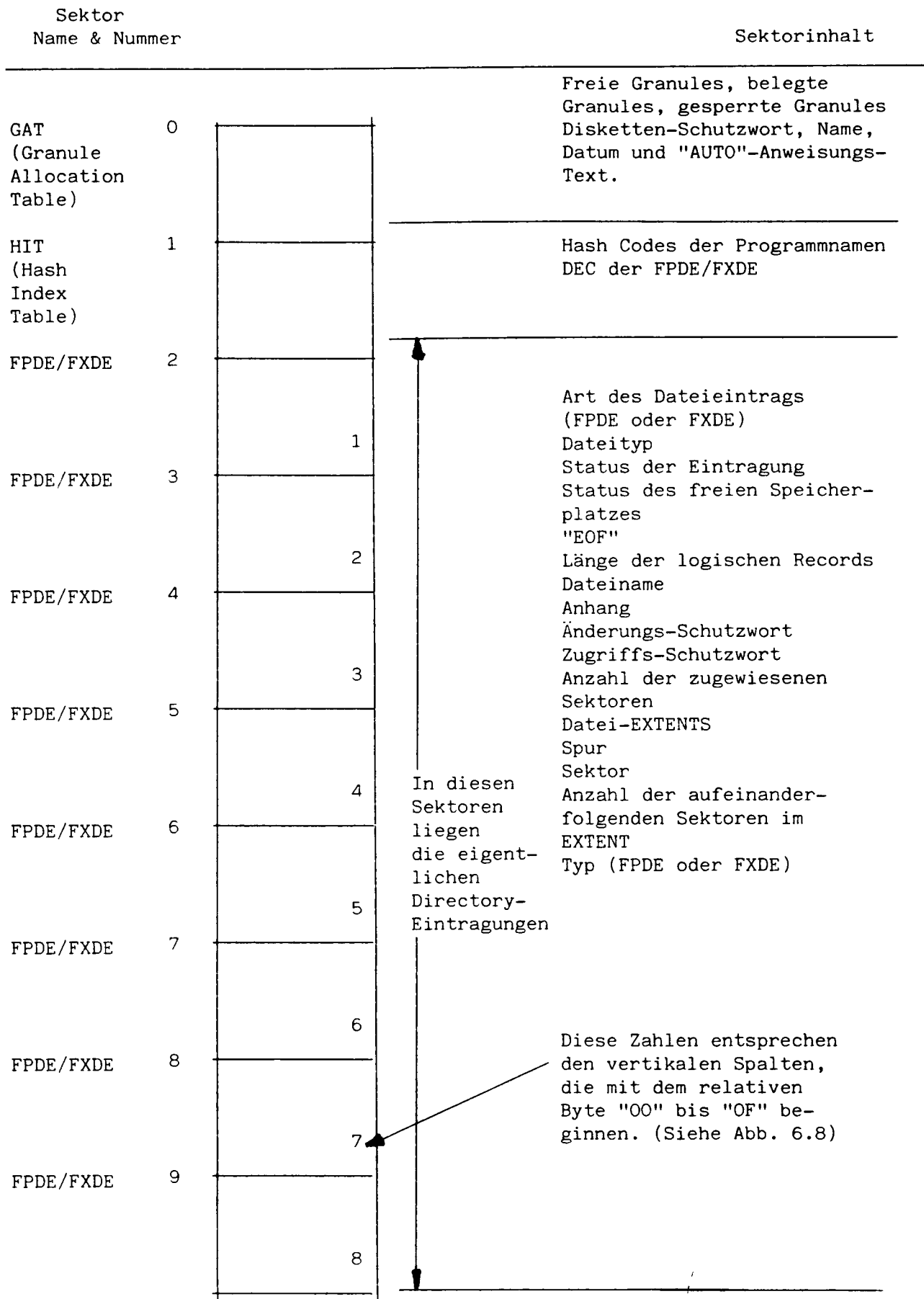
Im relativen Byte "CE" und "CF" findet man den "Hash-Code" für das Schutzwort der Diskette (Master Password).

Die nächste Zeile, beginnend mit dem Byte "DO" enthält den Diskettenamen und das Datum.

Die Bytes "EO" bis "FF" enthalten gegebenenfalls einen Text. Dieser stellt den Befehl der AUTO-Funktion dar. Wurde für die Diskette keine AUTO-Funktion festgelegt, enthält Byte "EO" den Wert "OD" (Wagenrücklauf). In diesem Fall wird beim Systemstart keine Funktion automatisch ausgeführt.

"HIT Sektor" - Sektor 1

"HIT" bedeutet "Hash Index Table". In diesem Sektor ist in verschlüsselter Form der Name jeder auf dieser Diskette gespeicherten Datei enthalten. Die Position des Hash-Codes in der "HIT" informiert weiterhin das Be-



Binär	HEX	Bedeutung
11111111	FF	1. & 2. Granule belegt
11111110	FE	2. Granule belegt
11111101	FD	1. Granule belegt
11111100	FC	1. & 2. Granule frei

triebssystem darüber, wo im Directory die vollständige Information über die betreffende Datei zu finden ist. Abb. 6.5 zeigt einen Ausdruck des "HIT"-Sektors und Abb. 6.8 bringt eine Übersicht dieses Sektors.'

Für jede gespeicherte Datei enthält die "HIT" ein Byte mit dem Hashcode. Sowohl der Code als auch die Lage dieses Bytes in der "HIT" ist wichtig.

Ein Hashcode ist eine Zahl, die nach einem bestimmten Schema aus dem Dateinamen gewonnen wird, indem jedem Buchstaben und Zeichen des Namens ein numerischer Wert zugeordnet wird. Diese Werte werden je nach Position des Zeichens im Namen mit einer bestimmten Zahl multipliziert und diese Teilergebnisse werden anschließend addiert, dividiert und gerundet.

Es gibt unzählige unterschiedliche Verfahren, wie man einen Hashcode erzeugen kann und ein Hashcode kann eine Länge von beliebig vielen Bytes haben, abhängig davon, wozu er gebraucht wird.

In unserem speziellen Fall ist der Hashcode ein Byte lang. Wir werden uns noch näher mit Hashcodes im Zusammenhang mit der "Datenrettung" befassen.

Am unteren Rand der "HIT"-Übersicht finden Sie Spalten, von 1 bis 8 nummeriert. Jede dieser senkrechten Spalten entspricht einem der acht Sektoren, die zum Speichern von Dateieinträgen in der Directory vorgesehen sind. Jede geradzahlig nummerierte waagerechte Zeile gibt mit ihrer Zeilennummer das relative Byte an, ab dem im betreffenden Sektor (entsprechend der senkrechten Spalte) ein Eintrag beginnt.

Nehmen wir als Beispiel die Abb. 6.5. Hier findet man einen Hashcode im relativen Byte "A2"; der Code lautet "8F". Dieser Code steht in der dritten senkrechten Spalte. Das bedeutet, der Directory-Eintrag ist im DRITTEN Sektor NACH dem "HIT"-Sektor zu finden, also im vierten relativen Sektor der Spur 17 (dez.).

Beachten Sie auch den Hashcode "A2" im relativen Byte "00" der "HIT". Dieser Code steht in der ERSTEN senkrechten Spalte. Damit ist der Datei-Eintrag im ERSTEN Sektor NACH der "HIT" zu finden, also im relativen Sektor 2 (es ist der Eintrag für "BOOT/SYS"). Neben diesem Code "A2" finden Sie "2C". Dieser Code entspricht einem Eintrag im zweiten Sektor nach der "HIT", also dem relativen Sektor 3 (Eintrag für "DIR/SYS"). Diese beiden letztge-

```

011100 A22C 2E2F 2C2D 2A2B 0000 0000 0000 0000 .,./,*+.....
011110 0000 0000 0000 0000 0000 0000 0000 0000 (. . . . .&.....
011120 2800 0000 00A7 26A6 0000 0000 0000 0000 .....
011130 0000 0000 0000 0000 0000 0000 0000 0000 .....
011140 F200 8900 0000 0000 0000 0000 0000 0000 .....
011150 0000 0000 0000 0000 0000 0000 0000 0000 .....
011160 0000 5600 00C5 0000 0000 0000 0000 0000 .....
011170 0000 0000 0000 0000 0000 0000 0000 0000 .....
011180 7900 AD00 0032 0000 0000 0000 0000 0000 .U.....
011190 0000 0000 0000 0000 0000 0000 0000 0000 .....
0111A0 F01D 8F00 009D 00B7 0000 0000 0000 0000 ....F2.....
0111B0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0111C0 0067 0000 0000 0000 0000 0000 0000 0000 .....
0111D0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0111E0 A3DB 0000 0000 00EE 0000 0000 0000 0000 .....
0111F06 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

nannten Codes stehen außerdem in der waagerechten Zeile, die mit dem relativen Byte "00" beginnt. Das bedeutet, daß die beiden zugehörigen Eintragungen jeweils im entsprechenden Sektor ab dem relativen Byte "00" beginnen. Der Code unseres ersten Beispiels steht in der waagerechten Reihe, die mit Byte Nummer "A0" beginnt. Demnach beginnt auch der zugehörige Eintrag im Sektor 4 ab dem relativen Byte "A0".

Sie finden in der ersten waagerechten Zeile zweimal den Hashcode "2C". Der eine steht, wie oben gesagt, für "DIR/SYS", der andere entspricht "SYS2/SYS". Sie erkennen daran, daß die Codes für unterschiedliche Dateinamen durchaus gleich sein können, sie müssen aber dennoch jeweils vom Dateinamen abgeleitet werden und in der richtigen Position der "HIT" stehen.

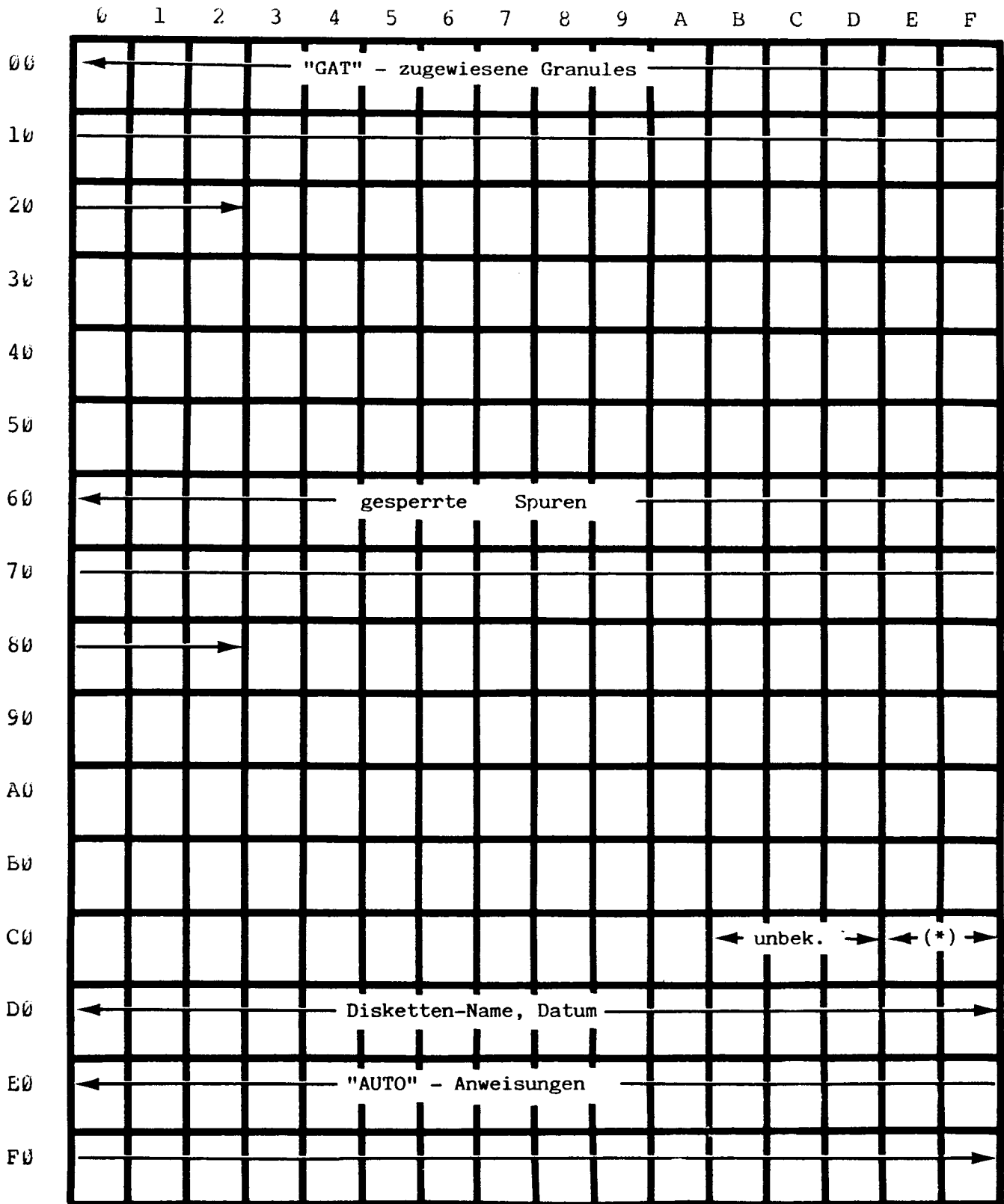
"FPDE/FXDE Sektoren" - Sektoren 2...9

"FPDE" bedeutet "File Primary Directory Entry" (Datei-Haupteintrag) und "FXDE" heißt "File Extended Directory Entry" (Erweiterter Dateieintrag). Diese Sektoren stellen das eigentliche Directory dar. (Siehe auch Abb. 6.10)

Hier werden Name, Schutzwort, Länge (in Sektoren), Lage des Dateiendes (EOF - End Of File) und die physikalische Lage auf der Diskette für jede Datei gespeichert.

Zusätzlich zu diesem "FPDE" werden hier auch die "FXDE" gespeichert. Ein "FXDE" wird immer dann erzeugt, wenn im "FPDE" nicht genügend Platz ist, um alle Informationen über eine Datei aufzunehmen, die DOS benötigt. Dieser "Platzmangel" rührt daher, daß jeder Directory-Eintrag nur 32 Byte lang sein darf. Wird mehr Platz benötigt, erzeugt DOS einen weiteren 32 Byte langen Eintrag, eben den "FXDE".

Übersicht des GAT-Sektors (Spur 11, Sektor 0)



(*) = Diskettenschutzwort

Tabelle der zugewiesenen Granules "GAT"

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
	01	03	05	07	09	0B	0D	0F	11	13	15	17	19	1B	1D	1F
20	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
30	20	22	24	26	28	2A	2C	2E	30	32	34	36	38	3A	3C	3E
	21	23	25	27	29	2B	2D	2F	31	33	35	37	39	3B	3D	3F
40	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	22	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
50	40	42	44	46	48	4A	4C	4E	50	52	54	56	58	5A	5C	5E
	41	43	45	47	49	4B	4D	4F	51	53	55	57	59	5B	5D	5F
60	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
70	60	62	64	66	68	6A	6C	6E	70	72	74	76	78	7A	7C	7E
	61	63	65	67	69	6B	6D	6F	71	73	75	77	79	7B	7D	7F
80	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
90	80	82	84	86	88	8A	8C	8E	90	92	94	96	98	9A	9C	9E
	81	83	85	87	89	8B	8D	8F	91	93	95	97	99	9B	9D	9F

Legende

- xx --- Spur, dezimal
- xx --- Spur, HEX
- xx --- 1. Granule (HEX) (*)
- xx --- 2. Granule (HEX) (*)
in dieser Spur

*) Granule - Code

- FF = beide Granules belegt
- FC = beide Granules frei
- FD = 1. Granule belegt
- FE = 2. Granule belegt

Übersicht des "HIT"-Sektors (Spur 11, Sektor 1)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	← Eintragungen, beginnend ab relativem Byte 00 →															
10																
20	← Eintragungen, beginnend ab relativem Byte 20 →															
30																
40	← Eintragungen, beginnend ab relativem Byte 40 →															
50																
60	← Eintragungen, beginnend ab relativem Byte 60 →															
70																
80	← Eintragungen, beginnend ab relativem Byte 80 →															
90																
A0	← Eintragungen, beginnend ab relativem Byte A0 →															
B0																
C0	← Eintragungen, beginnend ab relativem Byte C0 →															
D0																
E0	← Eintragungen, beginnend ab relativem Byte D0 →															
F0																

1 2 3 4 5 6 7 8 ← Diese Zahlen bedeuten die ersten acht Spalten dieser Tabelle und symbolisieren die acht Sektoren, in denen die eigentlichen Directory-Eintragungen gespeichert werden. (Siehe auch Abb. 6.3)

```

011400 5F00 0000 0053 5953 3020 2020 2053 5953 .....SYS0....SYS
011410 EB29 210E 0F00 0022 FFFF FFFF FFFF FFFF .)!....".....
011420 0000 0000 0000 0000 0000 0000 0000 0000 .....
011430 0000 0000 0000 0000 0000 0000 0000 0000 .....
011440 1000 009A 0045 4454 4153 4D20 2043 4D44 .....EDTASM..CMD
011450 9642 9642 2000 0D24 1A01 FFFF FFFF FFFF .B.B...$......
011460 1000 00B7 0054 5253 3233 3220 2020 2020 .....TRS232.....
011470 9642 9642 0300 1D20 FFFF FFFF FFFF .B.B.....
011480 0000 0000 0000 0000 0000 0000 0000 0000 .....
011490 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114A0 0000 00D9 0054 5249 4254 5241 5020 2020 .....TRIBTRAP...
0114B0 9642 9642 2100 1E22 2023 FFFF FFFF FFFF .B.B!.."#......
0114C0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114D0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114E0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0114F06 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

Es ist nun, glaube ich, an der Zeit, einmal zu definieren, was eigentlich mit einer Datei gemeint ist. Ein BASIC-Programm, mit "SAVE" gespeichert ist eine Datei. Ein Maschinenprogramm, gespeichert mit "DUMP", "TAPEDISK" oder "EDTASM" ist genauso eine Datei. Auch Daten, die unter BASIC mit "OPEN", "PRINT" und "PUT" gespeichert werden, sind Dateien. Tatsächlich kann man alles als "Datei" bezeichnen, was auf Diskette geschrieben wird. Damit sind nun hoffentlich alle Klarheiten beseitigt...oder ?

Jeder Directory-Sektor, beginnend ab dem relativen Sektor 2 kann bis zu acht Eintragungen ("FPDE" oder "FXDE") enthalten. Jede dieser Eintragungen belegt 32 Byte. Die jeweils erste Eintragung in einem Sektor ist für System-Dateien reserviert.

Sie können auch erkennen, daß die Eintragungen immer bei einem der folgenden relativen Bytes eines Sektors beginnen:

00, 20, 40, 60, 80, A0, C0 und E0.

Lassen Sie uns nun einen Directory-Eintrag genauer untersuchen. Wir greifen uns dazu den ersten, 32-Byte Block des Sektors 2 heraus.

Für die weitere Besprechung sehen Sie sich bitte Bild 6.11 an und beachten Sie, daß jedes einzelne Byte des hier dargestellten "FPDE" mit einer hexadezimalen Zahl (00 bis 1F) gekennzeichnet ist.

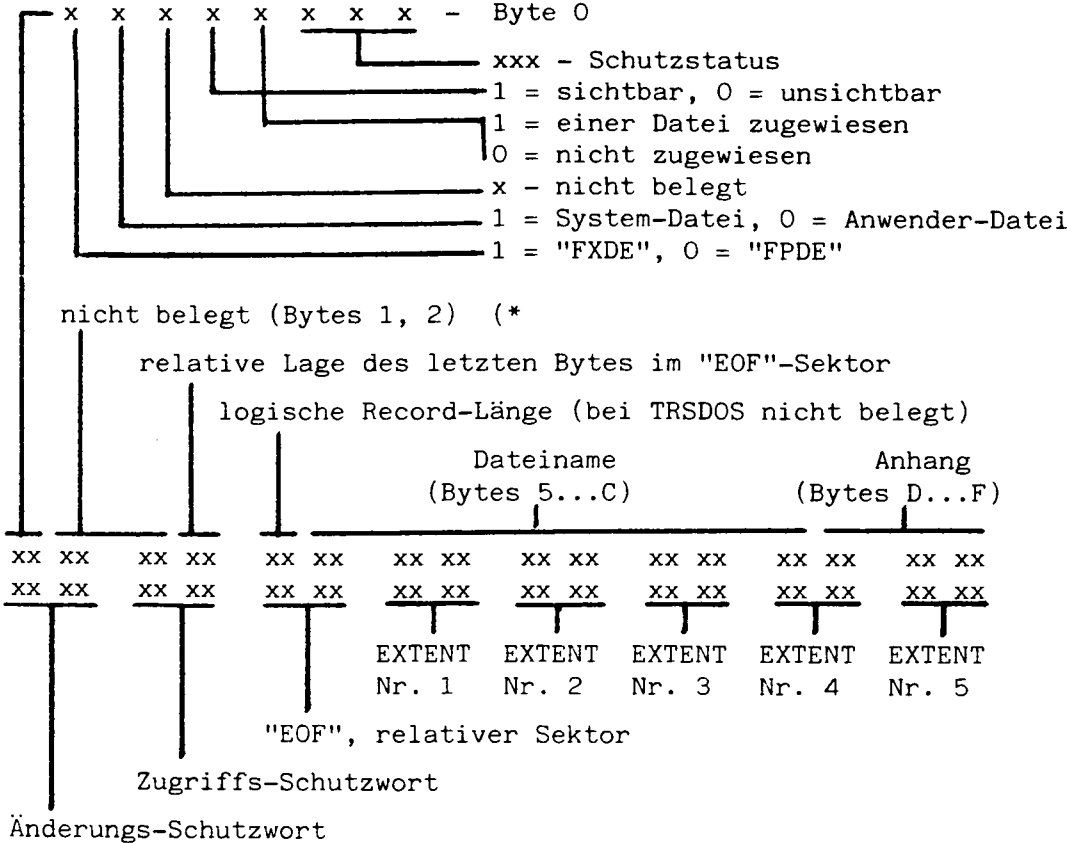
"FPDE" Byte 00

Dieses Byte kennzeichnet den Dateityp (System- oder keine System-Datei), die Attribute, den Zuordnungsstatus und die Schutzart. Hier hat jedes der acht Bit eine besondere Bedeutung. Dabei werden die drei rechten Bit (0...2) als eine Einheit betrachtet, die übrigen fünf Bit haben jeweils eine eigene Bedeutung:

Übersicht der "FPDE" - "FXDE" (Spur 11, Sektoren 2 ... 9)

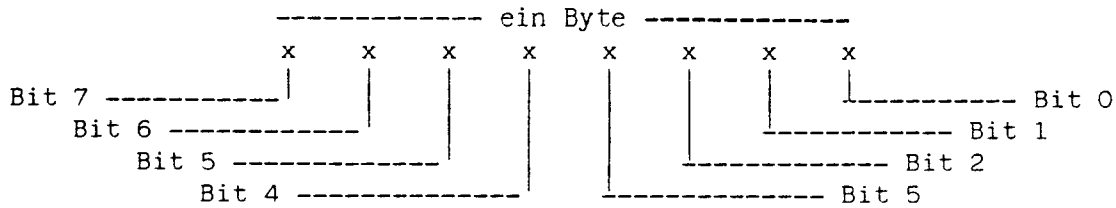
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	← 00	01	02	03	04	Directory-Eintrag 1				09	0A	0B	0C	0D	0E	0F
10	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	→ 1F
20	← 20	21	22	23	24	Directory-Eintrag 2				29	2A	2B	2C	2D	2E	2F
30	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	→ 3F
40	← 40	41	42	43	44	Directory-Eintrag 3				49	4A	4B	4C	4D	4E	4F
50	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	→ 5F
60	← 60	61	62	63	64	Directory-Eintrag 4				69	6A	6B	6C	6D	6E	6F
70	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	→ 7F
80	← 80	81	82	83	84	Directory-Eintrag 5				89	8A	8B	8C	8D	8E	8F
90	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	→ 9F
A0	← A0	A1	A2	A3	A4	Directory-Eintrag 6				A9	AA	AB	AC	AD	AE	AF
B0	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	→ BF
C0	← C0	C1	C2	C3	C4	Directory-Eintrag 7				C9	CA	CB	CC	CD	CE	CF
D0	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	→ DF
E0	← E0	E1	E2	E3	E4	Directory-Eintrag 8				E9	EA	EB	EC	ED	EE	EF
F0	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	→ FF

"FPDE" - Directory-Eintrag



*) Anm.: Byte 1 wird bei einem "FXDE" als "DEC" verwendet, der auf den zugehörigen "FPDE" zeigt.

Abb. 6.12



Bit 7: 1="FXDE", 0="FPDE"

Wenn Bit 7 "aus" ist, handelt es sich hier um einen "FPDE", also einen Haupteintrag. Ist es "ein", liegt ein "FXDE" vor, also ein Ergänzungseintrag zu einem anderen "FPDE". In diesem Fall gibt es in der "HIT" keinen Hashcode, der sich auf diesen Eintrag bezieht.

Bit 6: ist "1", wenn es sich um den "FPDE" einer System-Datei handelt.

Bit 5: nicht verwendet

Bit 4: 1 = "Invisible File", diese Datei wird normalerweise nach dem "DIR"-Befehl nicht angezeigt.

Bit 3: 1 = dieser Eintrag bezieht sich auf eine gültige Datei; in der "HIT" liegt ein Hashcode ungleich "00" vor. Ist das Bit = 0, dann wurde die Datei, zu der dieser Eintrag gehörte, bereits gelöscht. (Der Eintrag selbst wird außer bei TRSDOS 2.2 nicht mit "0" überschrieben, kann aber jederzeit von DOS mit einem neuen "FPDE" oder "FXDE" überschrieben werden.

Bit 2, 1 und 0: Diese drei Bits werden als eine Einheit interpretiert und geben den Schutzstatus der Datei an:

Bit	2-1-0	Schutzstatus
	1 1 1	kein Zugriff
	1 1 0	nur Ausführen
	1 0 1	Lesen u. Ausführen
	1 0 0	Schreiben, Lesen u. Ausführen
	0 1 1	nicht verwendet
	0 1 0	Umbenennen, Schreiben, Lesen u. Ausführen
	0 0 1	Löschen, Umbenennen, Schreiben, Lesen u. Ausführen
	0 0 0	keine Einschränkung.

"FPDE" Bytes 1 und 2

Wenn es sich bei dem Eintrag um einen "FPDE" handelt, dann sind diese Bytes unbenutzt. Handelt es sich jedoch um einen "FXDE", denn enthält Byte 1 den "DEC" (Directory Entry Code), der ZURÜCK auf den zugehörigen "FPDE" zeigt. Byte 2 wird nie verwendet und enthält immer den Wert "00".

"FPDE" Byte 3

Dies ist das "End Of File"- (EOF) Byte. Es gibt die relative Lage des letzten Bytes im letzten relativen Sektor einer Datei an. Wenn Sie z.B. eine Datei haben, die vier Sektoren lang ist und das EOF-Byte enthält den Wert "13", dann endet die Datei im vierten relativen Sektor mit Byte 13 (HEX).

"FPDE" Byte 4

Länge der logischen Records. Diese gute Idee ist leider im TRSDOS nicht realisiert. Solange es nicht genutzt wird, enthält es immer den Wert "00"; Sie können es aber beliebig setzen und evtl. für eigene Anwendungen nutzen, da es auf das System keinen Einfluß hat.

"FPDE" Bytes 5 bis C

Datei-Name. Diese acht Bytes enthalten den Dateinamen im ASCII-Code. Der Schrägstrich "/" ist nicht gespeichert. Sie können Dateinamen mit "SUPERZAP" leicht vertauschen oder ändern, aber beachten Sie, daß Sie in diesem Fall auch den Hashcode in der "HIT" ändern müssen!

"FPDE" Bytes D bis F

Dateinamen-Ergänzung. Hier werden die "Anhängsel" wie "/BAS" und "/CMD" gespeichert (ohne "/"). Sie können sie ebenfalls "ZAPPEN", aber auch

hier müssen Sie den zugehörigen Hashcode ändern.

"FPDE" Bytes 10 und 11

Update-Password. Hier ist in codierter Form das eventuell vorhandene Schutzwort gespeichert, das Sie bei beabsichtigten Dateiänderungen wissen müssen. Es handelt sich hier um einen Zweibyte-Hashcode, der erzeugt wird, wenn Sie die DOS-Funktion "ATTRIB" entsprechend anwenden. (Machen Sie sich über Passwords keine Sorgen, wir werden noch sehen, wie man sie umgehen kann.)

"FPDE" Bytes 12 und 13

Access Password. Hier wird, in gleicher Form wie oben, das Schutzwort für den Dateizugriff gespeichert, falls es vorhanden ist. Der Hashcode wird immer dann erzeugt, wenn Sie einen Dateinamen mit einem Schutzwort eingeben, z.B. bei

```
SAVE GEHEIM/BAS.SUPERZAP
```

In diesem Fall ist "SUPERZAP" das Schutzwort, dessen Code in die Bytes 12 und 13 geschrieben wird.

"FPDE" Bytes 14 und 15

Letzter relativer Sektor der Datei. Dieser Eintrag ist ein wenig trickreich. Das Verfahren ist aber einfach und logisch. Die beiden Bytes enthalten die Anzahl der Sektoren einer Datei in HEX. Aber Sie müssen zwei Regeln bei der Entschlüsselung beachten:

Regel 1: Wenn Byte 14 den Wert "00" enthält ("00" entspricht in dem Fall dem Dezimalwert "256"), dann enthalten die Bytes die tatsächliche relative Sektornummer.

Regel 2: Wenn Byte 14 einen beliebigen, anderen Wert hat, dann enthalten die Bytes die relative Sektornummer

PLUS eins.

Nun, was macht das für einen Sinn? Nehmen wir an, wir hätten eine kurze Datei mit genau 256 Bytes und wir speichern sie auf der Diskette. Die gesamte Datei passt also genau in einen Sektor, nämlich den relativen Sektor "0" dieser Datei. In diesem Fall enthalten die Bytes 14 und 15 den Wert "01" bzw. "00".

Nun nehmen wir eine etwas längere Datei an, sagen wir etwa 600 Bytes. Diese Datei benötigt etwas mehr als zwei Sektoren. Das bedeutet, die Datei endet im 2. relativen Sektor (0-1-2 gleich drei Sektoren). Mit anderen Worten, wir brauchen drei Sektoren, um die Datei zu speichern. Nach Regel 2 wird also eine "03" in Byte 14 gespeichert.

Aber das ist noch nicht alles. Lassen Sie uns noch eine Annahme machen. Sagen wir, wir hätten eine riesen-große Datei, die die Diskette ganz belegt, also 335 Sektoren lang wäre. Die größte Zahl, die wir aber in einem Byte unterbringen können ist "FF" oder "255" dezimal. Demnach benötigen wir ein zweites Byte, das Byte 15, um alle möglichen Sektor-zahlen zu speichern. In unserem Fall ergibt sich:

355 dezimal = 014F (HEX)
Byte 14 = "4F", Byte 15 = "01"

Es ist ziemlich offensichtlich, daß die beiden Bytes einfach vertauscht gespeichert werden. Zur Entschlüsselung müssen Sie die Sache also einfach "herumdrehen", die so erhaltene HEX-Zahl in eine Dezimalzahl umwandeln und Sie wissen die Anzahl der von dieser Datei belegten Sektoren.

"FPDE" Bytes 16-17, 18-19, 1A-1B, 1C-1D und 1E-1F

"EXTENT" 1, 2, 3, 4 und 5. Diese "EXTENTS" (Kenner für einen zusammenhängenden Datenblock)

enthalten jeweils die Spur, das Granule, die Anzahl zusammenhängender Granules (innerhalb des EXTENTS) und, falls erforderlich einen Zeiger auf den "FXDE".

Das alles liest sich recht kompliziert, Sie werden aber gleich sehen, es ist ganz einfach.

Bisher haben wir zwar alle Informationen, die wir brauchen, um den Namen, die Länge u.s.w. einer Datei festzustellen, aber wir wissen noch nicht, WO sie auf der Diskette gespeichert ist. Diese Information wird in den sogenannten "EXTENT"-Elementen aufgezeichnet.

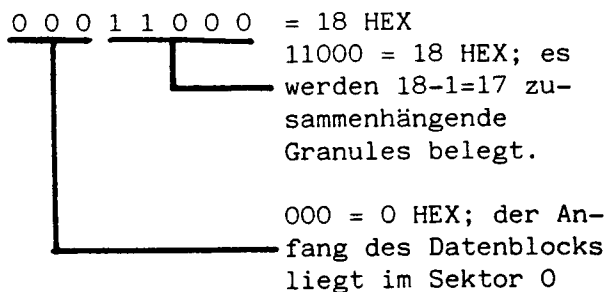
Nehmen wir als Beispiel ein EXTENT mit dem Wert

0718

an.

Das erste Byte gibt hexadezimal die Spurnummer an, hier also "07"

Das zweite Byte müssen wir wieder in seine einzelnen Bits zerlegen. Die rechten fünf Bits geben die Anzahl der lückenlos aufeinanderfolgenden Granules MINUS eins an. Die linken drei Bits sagen aus, ob der Datenblock ab dem ersten Granule der Spur beginnt (also ab Sektor 0) oder ob der Anfang im zweiten Granule (also ab Sektor 5) liegt. Hier folgen zwei Beispiele, die das verdeutlichen:



0 0 1 0 1 0 0 1 = 29 HEX
 01001 = 9 HEX; es werden 9-1= 8 zusammenhängende Granules belegt.
 001 = 1 HEX; der Anfang des Datenblocks ist um 1 Granule versetzt, liegt also im Sektor 5.

Zusammenfassend kann man feststellen:

Die Spur ist leicht zu erkennen, lesen Sie sie einfach ab.
 Wenn das zweite Byte des EXTENTS kleiner oder gleich 19 HEX ist, dann beginnt der Datenblock im Sektor 0, ist es größer oder gleich 20, dann beginnt der Datenblock im Sektor 5.

"FPDE" Kenner für Ende der EXTENTS

"Das alles ist ja gut und recht", werden Sie sagen, "aber wozu sind die vielen "FF"s nach den EXTENTS da?" Ganz einfach, damit wird dem System mitgeteilt: "es gibt keine weiteren EXTENTS mehr, du hast alle Informationen, die du brauchst, nun lies endlich die Datei!"

Wenn Sie weitere Daten zu einer Datei hinzufügen und DOS diese Daten nicht mehr an den letzten Datenblock der Datei anfügen kann, weil der dahinterliegende Speicherplatz schon von einer anderen Datei belegt ist, so sucht sich DOS einen freien Bereich aus der "GAT", legt die Daten in dem entsprechenden Bereich ab und erzeugt dann einen neuen "EXTENT".

Eine Datei kann im "FPDE" bis zu fünf EXTENTS haben...und das bringt uns zu den...

"FXDE"-Eintragungen.

Nun wird auch das Geheimnis um den "FXDE" (Directory Ergänzungseintrag) gelüftet. Ob ein "FXDE" zu einer Datei existiert, erkennen Sie im

"FPDE" dieser Datei und zwar im fünften EXTENT, daran, daß hier im ersten Byte "FE" steht. Das folgende Byte enthält den "DEC" (Directory Entry Code) für den "FXDE".
 Sie wollen wissen, was ein "DEC" ist? Sehen Sie sich bitte Abb. 6.13 an und passen Sie gut auf, wir schreiben morgen eine Prüfung darüber!

Nun sollten wir uns den "FXDE" im relativen Sektor 2 in Abb. 6.14 einmal ansehen:

Das erste Byte des "FXDE" wird genauso decodiert wie beim "FPDE". Der Wert "C7" im relativen Byte "41" ist ebenfalls ein "DEC", der auf den zugehörigen "FPDE" zeigt.

Fenwyler T. Murphy, ein Neffe DES Mr. MURPHY sagte mir, daß sich dahinter auch ein Anfangswert für einen Zufallsgenerator zur Erzeugung von Diskettenfehlern bei Schreiboperationen verbirgt, der den geheimen TRS-DOS-Befehl "DESDSK" (Destroy Disk - Diskette zerstören) aufruft.

Na, na, haben Sie geglaubt ich meine das ernst ?

Mit der obigen Information sollten Sie in der Lage sein, jede beliebige Datei auf der Diskette zu finden, solange das Directory intakt ist. Wenn das nicht mehr der Fall ist, dann müssen wir unsere "Datenrettungsverfahren" anwenden, die wir bald kennenlernen werden. Aber jetzt machen Sie erst einmal eine Pause!

Abb. 6.13

"FPDE"-Eintrag mit einem "DEC"-Zeiger auf einen "FXDE"-Eintrag.

```
0119C0 1000 002B 0044 4953 4B4F 5247 2050 434C ...+.DISKORG.PCL
0194D0 9642 9642 4200 2023 0124 0500 0701 FE40 .B.BB..$......B
```

Diese beiden Bytes in EXTENT 5 _____
"zeigen" auf den "FXDE"

"FE" gibt an, daß das nächste Byte ein "DEC" für
einen "FXDE" ist

40 (HEX) = 01000000 (binär)

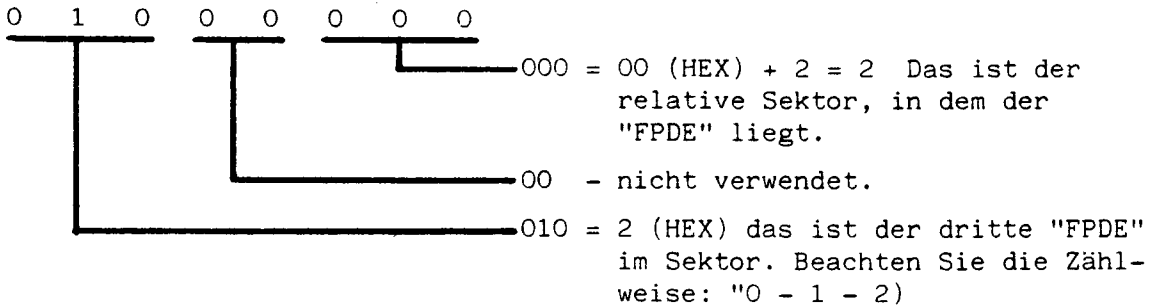


Abb. 6.14

```
011200 5E00 0000 0042 4F4F 5420 2020 2053 5953 .....BOOT....SYS
011210 EB29 210E 0500 0000 FFFF FFFF FFFF FFFF .)!.....
011220 0000 0000 0000 0000 0000 0000 0000 0000 .....
011230 0000 0000 0000 0000 0000 0000 0000 0000 .....
011240 90C7 0000 0000 0000 0000 0000 0000 0000 .....
011250 0000 0000 0000 0821 FFFF FFFF FFFF FFFF .....
011260 0000 0000 0000 0000 0000 0000 0000 0000 .....
011270 0000 0000 0000 0000 0000 0000 0000 0000 .....
011280 0000 0000 0000 0000 0000 0000 0000 0000 .....
011290 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112A0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112B0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112C0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112D0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112E0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0112F06 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

Directory-Sektor mit einem "FXDE" ab dem relativen Byte 40

7.0 Schutzworte und sonstige Trivialitäten

Viele machen ein großes Geschrei um Schutzworte - ich gebe zu, ich gehörte auch eine Weile dazu, aber einige Tage nachdem ich meine Diskette mit "SUPERZAP" bekommen hatte, existierte bei mir keine Diskette mit Schutzworten mehr.

Wenn Sie Kapitel 6 gelesen haben, dann wissen Sie, wo die Schutzworte codiert gespeichert werden. Als erstes werden wir nun einmal das Diskettenschutzwort angreifen.

Das Diskettenschutzwort

Der Hashcode für das Diskettenschutzwort wird im "GAT"-Sektor in den relativen Bytes "CD" und "CF" gespeichert. In Abb. 6.2 sehen wir den Code "EO42". Dieses Diskettenschutzwort wird beim DOS-Befehl "PROT :d (LOCK)" benötigt (:d ist eine Laufwerknummer).

Nach Eingabe des Befehls wird das Diskettenschutzwort ALLEN Anwenderdateien als Änderungs- und Zugriffs-Schutzwort zugeordnet. Die Systemdateien bleiben davon unberührt.

Umgekehrt kann man mit "UNLOCK" diesen Vorgang wieder aufheben. Dadurch werden in die Bytes mit den Schutzwortcodes der Dateien die Ziffern "9642" geschrieben.

Tatsächlich ist auch "9642" der Code für ein Schutzwort, nämlich für

" " (acht Leerzeichen),

die vom System ignoriert werden.

Das sehr oft verwendete Schutzwort "PASSWORD" hat den Code "EO42".

Nehmen wir einmal an, Sie haben eine Diskette, auf der ALLES geschützt ist und Sie haben das Diskettenschutzwort vergessen aber Sie müssen, aus wel-

chem Grund auch immer, auf die Dateien zugreifen. Wenden Sie folgendes Verfahren an:

(1) Lesen Sie mit "SUPERZAP" eine Diskette, deren Schutzwort Sie kennen.

(2) Notieren Sie sich den entsprechenden Hashcode.

(3) Entfernen Sie die Diskette und ersetzen Sie sie durch die "geheimnisvolle Unbekannte)

(4) Rufen Sie "DD" aus "SUPERZAP" auf.

(5) Zeigen Sie Spur 11 (HEX), Sektor 0 an.

(6) Mit "MODCE" ändern Sie die Bytes CE und CF auf den anfangs notierten Code um.

(7) Drücken Sie "BREAK" und rufen aus DOS die Funktion "PROT :d (UNLOCK)" auf.

(8) Nun sollten Sie mit "SUPERZAP" nochmals prüfen, ob bei allen Anwenderdateien das Schutzwort entfernt wurde.

(9) Gönnen Sie sich eine Pause, Sie haben es hervorragend gemacht !

Änderungs- und Zugriff-Schutzworte

Hier läuft die Sache fast genauso ab, nur daß Sie jetzt jedes Schutzwort einzeln bearbeiten. Mit der Information aus Kapitel 6 suchen Sie die Codes der beiden Schutzworte in den "FPDE" der entsprechenden Dateien. Dann schreiben Sie mit "SUPERZAP" in die Bytes 10 und 11 sowie 12 und 13 jeweils "9642", den Code für "ohne Schutzwort" und alles ist erledigt.

Viele Leute haben mich nach dem Berechnungsverfahren für den Hashcode gefragt.... Ich kenne es nicht und will es auch gar nicht wissen, denn es reicht mir, wenn ich mit "9642" ein

Schutzwort "vernichten" kann. Sie können mit dieser Methode ALLE Schutzworte entfernen, auch die von Systemdateien.

Sonstige Trivialitäten - Schutzstatus

Wie Sie aus Kapitel 6 wissen, wird der Schutzstatus einer Datei jeweils im ersten Byte eines jeden "FPDE" gespeichert (siehe auch Abb. 6.8).

Mit SUPERZAP können Sie nun natürlich auch den Schutzstatus einer Datei beliebig abändern.

Noch eine Trivialität - ein "Universal-Schutzwort"

Es geht die Sage um, daß eines der beiden folgenden Schutzworte IMMER passt, egal welches Schutzwort eine Datei oder Diskette wirklich hat:

"NV36"
"F3GUM".

Ich habe es noch nicht unter allen Bedingungen getestet, aber offensichtlich funktionieren diese beiden "Schutzworte" immer.

8.0 Verfahren zur Rettung von Daten

Ihr Erfolg bei der "Datenrettung" wird von Ihren Planungsfähigkeiten in großem Maße abhängen. Ob Sie eine zerstörte Datei erfolgreich wiedergewinnen, hängt davon ab, wie genau Sie sich vorher Gedanken über ihre Vorgehensweise gemacht haben.

Das Puzzle-Spiel

Das Verfahren zur Wiedergewinnung von verlorengegangenen Dateien ähnelt sehr einem Puzzle mit mindestens 5000 Teilen. Oft glauben Sie, jetzt endlich das richtige Teilchen gefunden zu haben und dann - durch eine Unachtsamkeit - ist alles wieder durcheinandergebracht.

Wenn Sie einem Profi zuschauen, dann wird der wahrscheinlich sagen: "alles ganz einfach, da ist der Fehler, wir schieben den Block auf diesen Sektor, machen hier einen Sektor frei, schieben den alten Sektor dahinein und kopieren alles zum Sektor XY." ZANG-BANG! Bevor Sie genau geschaut haben, ist alles erledigt. Es sieht so einfach aus, daß Sie glauben könnten "nichts leichter als das!" FALSCH ! Bedenken Sie den Unterschied zwischen Ihnen und einem Profi. Sie sollten schon erst ein wenig üben, dann werden Sie sicher auch bald ein "Superzapper" sein.

Die folgenden Punkte sollen Ihnen helfen, nicht die Übersicht zu verlieren, sie sind sozusagen der rote Faden durch das Dickicht der "Datenrettung":

1. Stellen Sie die Ursache des Problems fest.
2. Finden Sie die Lage der Datei und die Position der EXTENTS heraus.
3. Erzeugen Sie eine "Pufferspur", in der Sie Daten zwischenspeichern können.
4. Sehen Sie sich jeden Sektor der Datei an, NOTIEREN Sie sich die Sektoren, die fehlerhaft sind.
5. Fertigen Sie eine Checkliste an, in der Sie Ihre Vorgehensweise bei der Rekonstruktion der Datei aufschreiben.
6. Überprüfen Sie Ihren Plan nochmals.
7. Formatieren Sie eine weitere Diskette und halten Sie sie griffbereit, so daß Sie JEDERZEIT zusätzlichen Platz haben, um Daten zu speichern, falls erforderlich.
8. Arbeiten Sie möglichst immer mit einer Kopie der Diskette, auf der Sie eine Datei retten wollen.
9. Überprüfen Sie immer das Directory und stellen Sie sicher, daß Sie auch mit der richtigen Diskette arbeiten.
10. Machen Sie nie Vermutungen, probieren Sie alles vorher aus.
11. Wenn Sie einen Schritt ausgeführt haben, haken Sie ihn auf der Checkliste ab, so wissen Sie

immer, woran Sie sind und was als nächstes zu tun ist.

12. Überprüfen Sie die Ergebnisse einer Änderung bevor Sie die Daten wieder auf ihre ursprüngliche Position zurückbringen.
13. Legen Sie ggf. eine Übersicht an, aus der hervorgeht, welche Art von Daten sich in der Datei an welcher Stelle befindet.
14. Trinken Sie viel Flüssigkeit, nehmen Sie Aspirin und gönnen Sie sich viel Schlaf.

8.2 Die Verwendung von "SUPERZAP" bei Systemen mit nur einem Diskettenlaufwerk

"SUPERZAP" verwendet eigene Routinen zur Datenübertragung von und nach Diskette und benötigt daher keine Systemdiskette in Laufwerk 0. Nachdem "SUPERZAP" geladen ist, können Sie die Systemdiskette aus dem Laufwerk entfernen. Wenn Sie Sektorinhalte von einer Diskette zur anderen übertragen müssen, geht das mit der "SCOPY"-Funktion:

Rufen Sie "DD" auf und zeigen Sie den zu übertragenden Sektor an. Danach geben Sie "SCOPY" ein. Wenn Sie nach dem Ziel ("DESTINATION") gefragt werden, entfernen Sie die Originaldiskette und ersetzen Sie sie durch die "Zieldiskette". Beantworten Sie nun die Fragen und der betreffende Sektor wird auf diese Diskette kopiert. So können Sie theoretisch eine komplette Diskette duplizieren, wenn Sie die Geduld und Ausdauer besitzen sollten, die Disketten insgesamt 350 mal umzutauschen.

Im Allgemeinen müssen Sie jedoch nur einige Sektoren kopieren und dann ist dieses Verfahren durchaus sinnvoll.

Eine andere Technik ist, die gesamte Diskette mit DOS-BACKUP zu duplizieren und anschließend auf der Kopie außer der Datei, die Sie bearbeiten möchten, alle anderen Dateien zu löschen (KILL). Das schafft genügend

Raum zur Zwischenspeicherung von Daten.

8.3 Erzeugen einer "Pufferspur"

Das ist ganz einfach! Sehen Sie sich den "GAT"-Sektor an und finden Sie eine oder mehrere unbenutzte Spuren oder Granules und verwenden Sie diese, wenn Sie etwas zwischenspeichern müssen.

Wenn Sie fertig sind, brauchen Sie nicht einmal die Daten in der Pufferspur zu löschen, das System überschreibt diese Daten einfach, wenn es diesen Platz benötigt (er ist ja in der "GAT" immer noch als "frei" gekennzeichnet).

9.0 Dateistrukturen und -arten

Es gibt eine Anzahl unterschiedlicher Arten von Dateien, die auf einer Diskette gespeichert werden können. Jede Art hat ihr eigenes Format oder ihre eigene Struktur. Die Fähigkeit, eine Dateiart nur durch Anschauen eines HEX-Ausdruckes dieser Datei zu erkennen, kommt im Laufe der Zeit von ganz alleine mit etwas Übung. Die folgende Besprechung soll Ihnen helfen, die verschiedenen Dateien leichter zu erkennen und ihre Struktur zu verstehen.

9.1 Allgemeines

Durch Betrachten des Directory können Sie noch keine Aussage über die Art einer Datei machen. Eine Ausnahme stellen hier jedoch System-Dateien dar, da diese sowohl an definierten Stellen im Directory eingetragen sind und sie außerdem im "FPDE" einen Kenner besitzen, der sie als System-Dateien markiert. Die ersten beiden Positionen für "FPDE"s in jedem Directory-Sektor sind für System-Dateien reserviert. Bei allen anderen Eintragungen können Sie aus dem Directory nur eine Aussage machen, wenn Sie bereits wissen, von welcher Art eine Datei ist.

Alle Arten von Dateien werden auf die Diskette in Blöcken zu je 256 Bytes geschrieben. Wenn die Daten nicht ausreichen, um einen solchen Block oder Sektor ganz zu füllen, nimmt die Schreibroutine für den Rest einfach andere Daten aus dem Speicher. Mir ist nicht bekannt, nach welchen Regeln dieser Speicherbereich ausgewählt wird. Das ist der Grund, warum man das Ende einer Datei nicht unmittelbar "sehen" kann, weil dem Dateiende keine typischen "Leerbytes" wie z.B. "00" oder "FF" folgen.

9.2.1 BASIC Programme in ASCII-Darstellung

Wir beginnen damit, ein Stück unseres alten Freundes "SUPERZAP" anzusehen. Ich habe dieses Programm ausgewählt, da die meisten von Ihnen es ohnehin zur Verfügung haben.

Eine ASCII-Datei erscheint, wie Sie sehen können, genauso wie sie eingegeben wurde. Sie entsteht dann, wenn Sie ein BASIC-Programm mit dem Befehl SAVE "Programmname", A auf Diskette schreiben. Es gibt keine besonderen Codes oder Bytes. Das erste Byte in einer ASCII-Datei mit einem BASIC-Programm muß zu einer Zeilennummer gehören. Jede Zeile endet mit "0D" HEX (Wagenrücklauf). Dadurch "weiß" BASIC beim Laden des Programms, wann eine neue Zeile beginnt.

Im Beispiel der Abb. 9.1 finden Sie einen Wagenrücklauf u.a. im relativen Byte "EC". Versuchen Sie, was passiert, wenn Sie hier den Wert "20" (Leerzeichen) hinein"ZAP"pen und anschließend versuchen, das Programm zu laden und zu starten. Als nächstes können Sie versuchen, einige Zeilennummern zu ändern. Die ASCII-Codes für die Ziffern lauten in HEX:

0 = 30	5 = 35
1 = 31	6 = 36
2 = 32	7 = 37
3 = 33	8 = 38
4 = 34	9 = 39

Das Dateiende (EOF) wird im Directory-Eintrag gekennzeichnet. In der Datei selbst gibt es keine spezielle "Endemarke". Sie werden auch bemerken, daß der letzte Sektor der Datei zum Ende hin mit willkürlichen Daten gefüllt ist. Den Grund dafür kennen Sie ja bereits.

Mit einigen Experimenten werden Sie mit ASCII-Dateien bald vertraut sein.

ASCII-codierte
Zeilennummer

ASCII-Code für
"Leerzeichen"

Erstes Zeichen des Programmtextes

"EOR"-Merker (0D HEX)

F00000	3530	2052	454D	3A20	4D41	494E	2F44	4953	50.REM:.MAIN/DIS
F00010	4B20	4D45	4D4F	5259	2044	55D	502F	4D4F	K.MEMORY.DUMP/MO
F00020	4449	4659	2052	4F55	5449	4045	2E20	2056	DIFY.ROUTINE...V
F00030	4552	5349	4F4E	2032	2E30	0D31 3030		2047	ERSION.2.0.100.G
F00040	4F54	4F20	3130	3430	300D	3135 3020		4124	OTO.10400.150.A\$
F00050	3D49	4E4B	4559	243A	2049	4620	4124	3D22	=INKEY\$: .IF.A\$="
F00060	2220	5448	454E	2031	3530	3A20	2045	4C53	".THEN.150:..ELS
F00070	4520	4258	3D41	5343	2841	2429	3A20	5245	E.BX=ASC(A\$):.RE
F00080	5455	524E	0D32 3030		2049	4620	4258	203E	TURN.200.IF.BX.>
F00090	3D34	3820	414E	4420	4258	3C3D	3537	2054	=48.AND.BX<=57.T
F000A0	4845	4E20	4258	3D42	582D	3438	3A20	5245	HEN.BX=BX-48:.RE
F000B0	5455	524E	0D32 3530		2049	4620	4258	3E3D	TURN.250.IF.BX>=
F000C0	3635	2041	4E44	2042	583C	3D37	3020	5448	65.AND.BX<=70.TH
F000D0	454E	2042	583D	4258	2D35	353A	2052	4554	EN.BX=BX-55:.RET
F000E0	5552	4E0D	3330 3020		4258	3D2D	0D33 3530		URN.300.BX=-.350
F000F0	204C	5345	3A52	4554	5552	4E3A	5054	4F50	.LSE:RETURN:STOP

hier simulierter Fehler
(siehe auch Abb. 10.3)

||||||| = "EOR" - Kenner
===== = Zeilennummer

9.2.2 BASIC Programme in Binär- darstellung

Diese Dateien sind etwas schwieriger zu entziffern, da die Programmzeilennummern in komprimierter Binärdarstellung und die BASIC-Befehle als sogenannte "Tokens" (Befehlskürzel) gespeichert werden. Wenn Sie den Befehl SAVE "filename" geben, wird ein BASIC-Programm auto-

matisch in dieser Form auf die Diskette geschrieben.

In Abb. 9.2 sehen Sie den ersten Sektor des Programms "SUPERZAP" in komprimiertem Binärformat. Vergleichen Sie diesen Sektor mit der Abb. 9.1 und Sie werden auf Anhieb erkennen, daß nun wesentlich mehr Programmteile in einem Sektor gespeichert sind. Diese Art der Pro-

grammspeicherung ist sehr effizient was Platz und Geschwindigkeit beim Schreiben oder Lesen anbetrifft und sollte verwendet werden, wann immer es möglich ist.

Das erste Byte in JEDER Datei mit einem BASIC-Programm in komprimierter Form lautet "FF". Wenn hier irgendein anderer Wert steht, erhalten Sie die sicherlich altbekannte Fehlermeldung

"DIRECT STATEMENT IN FILE" (Datei enthält eine direkte Anweisung, also eine Zeile ohne Zeilennummer).

Nun sollten wir uns ansehen, woran BASIC "erkennt", was eine Zeilennummer ist. Wenn ein BASIC-Programm im RAM steht, enthält der jeweils erste Teil einer Programmzeile einen "Zeiger" auf die nächstfolgende Zeile. Dieser "Zeiger" wird zusammen mit den

Abb. 9.2

"FF" markiert BASIC-
Programm-Datei

"68F4" ist der "Zeiger"
auf die nächste Programm-
zeilennummer im RAM

"3200" ist die erste Zeilen-
nummer des Programms und steht
in der Reihenfolge LSB - MSB d.h.
ist zu lesen als "0032" (HEX) =
"50" (dezimal)

```

F00000  FFF4 6832 0093 3A20 4D41 494E 2F44 4953 ...2...:MAIN/DIS
F00010  4E20 4D45 4D4F 5259 2044 554D 502F 4D4F K.MEMORY.DUMP/MO
F00020  4449 4659 2052 4F55 5449 4E45 2E20 2056 DIFY.ROUTINE...V
F00030  4552 5349 4F4E 2032 2E30 0000 6964 008D ERSION.2.0.....
F00040  2031 3034 3030 0029 6996 0041 24D5 C93A .10400.)...A$:..
F00050  208F 2041 24D5 2222 20CA 2031 3530 3A20 ...A$.""...150:..
F00060  203A 9520 4258 D5F6 2841 2429 3A20 9200 ...:..BX..(A$):...
F00070  4E69 C800 8F20 4258 20D4 D534 3820 D220 N.....BX...48...
F00080  4258 D6D5 3537 20CA 2042 58D5 4258 CE34 BX..57...BX.BX.4
F00090  383A 2092 0072 69FA 008F 2042 58D4 D536 8:.....BX..6
F000A0  3520 D220 4258 D6D5 3730 20CA 2042 58D5 5...BX..70...BX.
F000B0  4258 CE35 353A 2092 0080 692C 0142 58D5 BX.55:.....,BX.
F000C0  CE42 583A 2092 00AF 695E 0193 3A20 2A2A .BX:.....:..**
F000D0  2A2A 2A2A 2A2A 2A20 5641 5249 4142 4C45 *****.VARIABLE
F000E0  2041 4C4C 4F43 4154 494F 4E20 494E 4849 .ALLOCATION.INHI
F000F0  4249 5445 4400 D569 9001 4432 2528 3129 BITED.....D2%(1)

```

= "EOR" - Merker

= Zeiger

==== = Zeilennummer

übrigen Daten dieser Zeile durch "SAVE" auf die Diskette übertragen. Am Ende jeder Zeile, unmittelbar vor Beginn des nächsten "Zeigers" kommt ein "END OF RECORD"-Merker (Merker für das Ende des Datensatzes, Ende der Programmzeile). Dieser "EOR"-Merker hat den Wert "00".

Im vorangehenden Absatz wurden EOF-Merker, der "Zeiger" und die Zeilennummer angesprochen. Dabei ist der "Zeiger" nicht so wichtig. Sie können jeweils beliebige Werte in diese Speicherplätze schreiben. Der "EOR"-Merker ist jedoch sehr wichtig, er muß den Wert "00" enthalten.

Versuchen Sie, was passiert, wenn Sie einen "EOR"-Merker z.B. mit "FF" überschreiben. Das Programm läßt sich zwar laden aber die Zeile, deren "EOR"-Merker Sie geändert haben, wird durcheinandergebracht und die höchste Zeile noch mit angefügt, da es für BASIC keine Möglichkeit gibt, zu erkennen, wo das Zeilenende liegt und wo eine neue Zeilennummer zu finden ist.

Als nächstes versuchen Sie einmal, einen "Zeiger" auf den Wert "FF" zu ändern. Was passiert ? Aha, das Programm lädt fehlerfrei da BASIC diese Unstimmigkeit selbst ausgleicht. Wenn sie das Programm anschließend wieder auf Diskette schreiben (SAVE), wird dieser Fehler selbständig behoben.

Sie können sogar eine neue Zeilennummer in ein Programm durch "ZAP"ppen schreiben: fügen Sie dazu folgende Codes an der richtigen Stelle ein:

```
" 00 FFFF llhh"
```

"llhh" drückt dabei die Zeilennummer aus und zwar in hexadezimaler Darstellung, ll=niedriges Byte, hh=höheres Byte. Beachten Sie, daß "ll" und "hh" "vertauscht" sind.

9.3 Dateien mit EDTASM-Quellencodes

Meines Wissens ist der Editor-Assembler von Apparat die einzige Version des "EDTASM" von Radio Shack mit einem Zusatz, um Daten auf Disketten zu schreiben. Falls noch andere Versionen existieren, kann es sein, daß diese ein anderes Format verwenden. Daher sollten Sie die folgenden Aussagen nur im Zusammenhang mit der "EDTASM"-Version von Apparat als gültig ansehen.

EDTASM-Dateien sind ganz normale ASCII-Dateien mit nur kleinen Unterschieden. Die ersten sieben Bytes stellen eine "Titelzeile" dar. Das erste Byte ist immer "D3". Die nächsten sechs Bytes sind die ersten sechs Zeichen des Programm-Namens (mir ist nicht bekannt, wozu das erforderlich ist).

Die Zeilennummern sind in ASCII-Darstellung mit der Ausnahme, daß der Wert 128 (dezimal) zum normalen ASCII-Codezeichen addiert wird. So wird aus dem Zeichen "0", dem das ASCII-Codezeichen "30" (HEX) oder 48 (dez) entspricht: $48 + 128 = 176$ (dez) oder "BO" (HEX).

(Anm. d. Ü.: man kann auch sagen, daß im ASCII-Zeichen zusätzlich das höchste Bit, nämlich Bit 7 gesetzt ist.)

In Abb. 9.3 hat z.B. die erste Zeile die Nummer "00100". Sie finden sie ab dem relativen Byte "07" und die Codezeichen lauten:

```
"BO BO B1 BO BO BO"
```

Schneiden Sie einfach die "B"s ab und Sie erhalten:

```
" 0 0 1 0 0 0"
```

Es ist eine recht einfache Sache, eine solche Datei durch ein BASIC-Programm mit "INPUT"-Anweisungen zu lesen, die Zeilennummern wieder in ihr normales Format umzuwandeln, um

sie anzeigen zu können, sie zu editieren und anschließend wieder in eine andere Datei mit "PRINT" zurückzuschreiben.

Die "EOR"-Marke ist hier ein Wagenrücklauf ("OD" HEX), genau wie in einer BASIC-ASCII-Programmdatei.

Vermutlich hat man die besondere Form der Codierung der Zeilennummern gewählt, um zu vermeiden, daß ver-

sehtentlich eine solche Datei als BASIC-Programm geladen wird.

Das Ende der Datei ist mit einem "EOF"-Merker (END OF FILE) gekennzeichnet. Wenn Sie das "EOF"-Byte aus dem "FPDE" herauslesen, werden Sie feststellen, daß der "EOF"-Merker in der Datei ein Byte vorher zu finden ist. Er lautet "1A" (HEX), ihm geht ein Wagenrücklauf ("OD" HEX) voraus.

Abb. 9.3

"D3" kennzeichnet ein "EDTASM"-
Quellenprogramm

Die ersten sechs Zeichen
des Dateinamens

Zeilennummer - ASCII-Code
plus 128 (dezimal)

Erstes Zeichen des
Quellenprogramms

"EOR"-Kenner

109500	D345	5045	5242	4FB0	B0B1	B0B0	2009	4F57	.APARBO.....OR
109510	4709	3041	4230	3048	0DB0	B0B1	B0B5	2042	G.0AB00H.....B
109520	4547	494E	0945	5155	0924	0DB0	B0B1	51B0	EGIN.EQU.\$.....
109530	2009	4C44	0948	4C2C	5354	4152	540D	B0B0	..LD.HL,START...
109540	B1B2	B020	094C	4409	2834	3031	3648	292CLD.(4016H),
109550	484C	0DB0	B0B1	B3B0	2009	4C44	0948	4C2C	HL.....LD.HL,
109560	5354	5249	4E47	0DB0	B0B1	B4B0	2009	4C44	STRING.....LD
109570	0928	4255	4646	4552	292C	484C	0DB0	B0B1	.(BUFFER),HL....
109580	B5B0	2009	4A50	0934	3032	4448	0DB0	B0B1JP.402DH....
109590	B6B0	2053	5441	5254	0945	5155	0924	0DB0	...START.EQU.\$..
1095A0	B0B1	B7B0	2009	5055	5348	0948	4C0D	B0B0PUSH.HL...
1095B0	B1B8	B020	094C	4409	484C	2C28	4255	4646LD.HL,(BUFF
1095C0	4552	290D	B0B0	B1B9	B020	094C	4409	412C	ER).....LD.A,
1095D0	2848	4C29	0DB0	B0B2	B0B0	2009	4350	0930	(HL).....CP.0
1095E0	4148	0DB0	E0B2	B1B0	2009	4A50	095A	2C53	AH.....JP,Z,S
1095F0	544F	434B	0DB0	E0B2	B2B0	2009	494E	4309	TOCK.....INC.

9.4 Dateien mit Objekt-Code

Diese Dateien sind sehr leicht zu erkennen, da sie nie einen Sinn ergeben. Die ASCII-Anzeige von "SUPERZAP" bringt ein Mischmasch von Leerzeichen und buntgewürfelten Symbolen.

Wir wollen hier nicht versuchen, zu zeigen, wie Maschinencode funktioniert, dazu gibt es eine ganze Reihe guter Bücher.

In der Abb. 9.4 sehen Sie als Beispiel ein kurzes Maschinenprogramm von etwas über 100 Byte Länge. Alle Dateien, die Maschinen- oder Objektcode enthalten, beginnen mit dem Kenner "01", der auch verstanden werden kann als Anweisung "Lade die folgenden Bytes ab der folgenden Adresse und zwar soundsoviel Stück ins RAM!"

Wenn Sie sich an den Anfang des

Abb. 9.4

Steuercode für Ladeprogramm "01" =
"Lade ab der folgenden Adresse"

Anzahl der zu ladenden Bytes

Zieladresse des Objekt-Codes,
LSB - MSB ((E8FD entspr. FDE8 HEX)

Der eigentliche Objektcode
beginnt hier

Startadresse des Programms
in der Folge LSB - MSB

204000	017B	E8FD	110F	0021	E8FD	1922	2640	C39A!..."&@..
204010	0A00	00F3	79FE	0D28	03FE	20D8	F5E0	C506(.....
204020	0937	F5F5	2101	FCCD	2102	2133	002B	7CB5	.7..!...!.!3.+..
204030	20FB	F11F	F530	1321	00FC	1813	0E48	AF0D0.!.'.....H..
204040	2802	18DB	3E0A	18D7	182F	C600	2101	1CCD	(...>.../..!...
204050	2102	0000	2133	002B	7CB5	20FB	10D4	1133	!...!3.+.....3
204060	00CB	4A28	0E21	00FC	CD21	021B	7AB3	207B	..J(!...!.....
204070	F1F1	FE0D	28C6	B728	C5C1	E1F1	C902	02E8(..(.....
204080	FDB5	B020	534E	4950	3109	504F	5009	4146SKIPL.POP.AF
204090	0909	3E46	4958	2054	4845	2053	5441	434B	..;FIX.THE.STACK
2040A0	0DB0	B4E6	B7E0	2009	504F	5020	0941	4609POP..AF.
2040B0	093B	4745	5420	5052	494E	5420	4348	4152	.;GET.PRINT.CHAR
2040C0	4143	5445	520D	B0B4	B6B9	B020	0943	5009	ACTER.....CP.
2040D0	4352	0909	3E43	4152	5241	4745	2052	4554	CR..;CARRAGE.RET
2040E0	5552	4E3F	0DB0	B4B7	B1E0	2009	4A52	095A	URN?.....JR.Z
2040F0	2C4E	554C	4C53	0909	3B59	4553	2C20	444F	,NULLS..;YES,.DO

= Steuerzeichen für das Ladeprogramm
 = Adresse, wohin zu laden ist
 = Anzahl der zu ladenden Bytes
 = Startadresse

Kapitels zurückerinnern, wissen Sie, daß der Rechner immer nur Datenblöcke mit einer Länge von 256 Bytes in einem Takt laden kann. Das gilt hier genauso. Wenn der Rechner 256 Bytes geladen hat, hält er erneut "Aus-schau" nach dem Merker "01" und falls er ihn findet, läd er weitere 256 Bytes und so fort.

Manchmal wird das Ladeprogramm auch angewiesen, einen Block zu lesen, der kürzer als 256 Bytes ist. In diesem Fall ist der Kenner für die Anzahl der Bytes nicht "00" (was hier 256 entspricht), sondern er hat Werte von "FF" abwärts.

(Anm. d. Ü.: Diese Zahl wird vom Ladeprogramm als Zähler verwendet und beim Laden nach jedem gelesenen Byte DEKREMENTIERT, bis der Wert "00" erreicht ist.)

In unserem Beispiel finden Sie in den relativen Bytes 2 und 3 die Werte

" E8 FD"

Hier ist die Anfangsadresse eingetragen, ab der die folgenden Bytes in das RAM zu schreiben sind. Beachten Sie, daß auch hier die Bytes "vertauscht" sind; die tatsächliche Adresse lautet also

" FD E8"

Als weitere Information ist gegebenenfalls eine Startadresse (Transfer Address) vorhanden. Dieser Wert sagt aus, ab welcher Adresse das Programm ausgeführt werden soll. Die Adresse muß nicht, wie in unserem Beispiel, mit der Anfangsadresse des Datenblocks übereinstimmen.

Einen "EOF"-Merker gibt es hier nicht. Das Ende der Datei ist ja ohnehin im "FPDE" eingetragen.

9.5 System-Dateien

System-Dateien sind im Prinzip normale Objektcode-Dateien, wie im vor-

angehenden Abschnitt besprochen, jedoch mit einigen Ergänzungen. Eine System-Datei kann im "FPDE" durch das erste Byte erkannt werden. (Byte "0", Bit "1" = 1). Sonst sollte eigentlich kein Unterschied sein.

Wohlgermerkt, "sollte" kein Unterschied sein. Aber die System-Dateien von TRS-DOS und von VTOS 3.0 sind doch anders. Abb. 9.5 zeigt eine solche Datei. Wenn Sie sich hingegen eine System-Datei ansehen, die Apparat in NEWDOS hinzugefügt hat, werden Sie keinen Unterschied zu normalen Objekt-Dateien feststellen.

Die erste Änderung ist, daß diese Dateien mit "0506" beginnen. Das sind Steuer-codes für das Ladeprogramm. Die Codes, die mir bekannt sind, führe ich hier auf, es kann aber durchaus noch weitere geben.

- 01 - Lade den folgenden Objekt-Code.
- 00 und 02...1F - Lade die folgenden n Bytes NICHT, wobei "n" ein Byte ist, das die Anzahl der zu überspringenden Bytes enthält.
- 0202 - Die folgenden zwei Bytes enthalten die Startadresse.

So bedeutet also:

- "0506" - "Überspringe die nächsten 6 Bytes"
- "1FA9" - "Überspringe die nächsten "A9" Bytes (= 169 dezimal)."

Wenn Sie in der Abb. 9.5 "1FA9" suchen und von dort 169 Bytes weitergehen, landen Sie beim relativen Byte "B2"...und was finden Sie da? Hurra, tatsächlich "01 08 0C 40" und das ist die Anweisung, die nächsten acht Bytes ab der Adresse 400C ins RAM zu laden.

(Anm. d. Ü.: ACHTUNG, die beiden Adressbytes werden mitgezählt, es

```

000500 0506 5359 5330 2020 1FA9 0D2A 202A 202A ..SYS0.....*.*.*
000510 204E 204F 2054 2049 2043 2045 202A 202A .N.O.T.I.C.E.*.*
000520 202A 0D2A 2050 524F 5052 4945 5441 5259 .*.*.PROPRIETARY
000530 2050 524F 4752 414D 202A 0D2A 2043 4F50 .PROGRAM.*.*.COP
000540 5952 4947 4854 2028 6329 2031 3937 3820 YRIGHT.(.)1978.
000550 202A 0D2A 2020 4259 2052 414E 444F 4C50 .*.*.BY.RANDOLP
000560 4820 434F 4F4B 2020 202A 0D2A 2020 4341 H.COOK...*.*.CA
000570 5252 4F4C 4C54 4F4E 2C20 5445 5841 5320 RROLLTON,.TEXAS.
000580 202A 0D2A 2041 4C4C 2052 4947 4854 5320 .*.*.ALL.RIGHTS.
000590 5245 5345 5256 4544 692A 6A2A 672A 752A RESERVED.*.*.*
0005A0 204E 204F 2054 2049 2043 2045 002A 002A .N.O.T.I.C.E.*.*
0005B0 202A 0D01 080C 40C3 A24B C3B4 4401 0B2D .*....@..K..D..-
0005C0 40C3 0044 3EA3 EFC3 BB44 0105 3E40 2101 @..D>....D..>@!.
0005D0 0001 144B 4000 0037 4537 4537 4537 4537 ...K@..7E7E7E7E7E
0005E0 4537 4537 4537 4501 1700 4311 1111 1111 E7E7E7E...C.....
0005F0 1111 1100 0100 0000 0002 0000 00C3 A04B .....K

```

sind zwar acht Bytes angegeben, die ersten beiden sind jedoch die Adresse; es werden also nur sechs Datenbytes gelesen und ins RAM übertragen.)

Wenn Sie nun sechs Bytes weitergehen, finden Sie "01 0B 2D 40". Das bedeutet, daß die nächsten 11 Bytes zu lesen sind und davon 9 Datenbytes ins RAM ab Adresse 402D zu übertragen sind.

Wenn Sie eine System-Datei löschen (KILL) und sich später entschließen, sie wieder auf die Diskette zu kopieren, ist wichtig, daß der "FPDE" an der gleichen Stelle steht, wie bei der Original DOS-Diskette! Der belegte Speicherbereich auf der Diskette MUSS im ersten EXTENT des "FPDE" festgelegt werden, Sie dürfen keine weiteren "EXTENTS" verwenden. Das heißt, die betreffende System-Datei muß auf der Diskette in einem zusammenhängenden Block stehen. Eine Ausnahme macht "SYS6/SYS". Diese Datei kann beliebig aufgeteilt werden. Die tatsächliche Lage der Systemdateien auf der Diskette ist frei wählbar, mit einigen Ausnahmen, die im folgenden Abschnitt gezeigt werden.

9.6 "BOOT/SYS" - "DIR/SYS" - "SYSO/SYS"

"BOOT/SYS" und "DIR/SYS" sind zwar "System-Dateien", enthalten aber keinen Code, der als Programm ausgeführt wird. "BOOT/SYS" ist eine Tabelle, die geladen wird, wenn Level II BASIC erkennt, daß ein Expansion Interface angeschlossen ist. "DIR/SYS" ist das Directory und kann natürlich nicht "ausgeführt" werden. Sie belegen Platz auf der Diskette, wie andere Systemdateien auch, aber enthalten keinen ausführbaren Code. "SYSO/SYS" allerdings ist ein ausführbares Programm und MUSS in Spur 0, Sektor 5 liegen.

9.7 "ELECTRIC PENCIL" - Dateien

Die sind einfach! Es handelt sich um ganz gewöhnliche ASCII-Dateien mit einem Wagenrücklauf als "EOR"-Merker und einem "EOF"-Merker am Ende der Datei. Dieser "EOF"-Merker ist "00" und befindet sich in unserem Beispiel der Abb. 9.6 im relativen Byte "45". Das ist alles, Punkt!

9.8 MACRO-80 Dateien

Jetzt hatte ich geglaubt, mit den Dateiformaten schon fertig zu sein, da entdeckte ich noch eines. Der Text-Editor des MACRO-80 Editor-Assemblers von MICROSOFT erzeugt ein weiteres, unterschiedliches Format. Dieser Editor/Assembler ist unter den TRS-80 Anwendern nicht so sehr verbreitet, aber er ist trotzdem nicht schlecht. Er hat einige interessante Eigenschaften (eine andere Art zu sagen: "er ist OK, aber ich ziehe einen anderen vor"), aber er ist schwieriger anzuwenden und die Dokumentation ist nicht sehr informativ.

In Abb. 9.7 sehen Sie einen typischen Sektor mit Daten, die vom MACRO-80 Text-Editor erzeugt werden.

Es gibt nur einige kleine Abweichungen zu dem Format, das Apparats EDTASM und ELECTRIC PENCIL erzeugen. Wie ASCII- und ELECTRIC PENCIL-Dateien wird jede Zeile mit einem Wagenrücklauf abgeschlossen (OD). Wie bei einer ELECTRIC PENCIL-Datei wird das Datei-Ende mit "00" gekennzeichnet. Dieser Text-Editor schreibt aber im Gegensatz zu anderen

Programmen hinter das Datei-Ende keine unvorhersehbaren Füll-Bytes bis zum Ende des Sektors sondern LAUTER NULLEN (00).

Durch diese Eigenschaften sind die erzeugten Dateien kompatibel zu ELECTRIC PENCIL. Nachdem Sie eine solche Datei mit ELECTRIC PENCIL geladen haben, müssen Sie sie allerdings etwas modifizieren, damit Sie sie auf dem Drucker ausgeben können. Sie werden nach dem Laden bemerken, daß die Zeilennummern auf dem Bildschirm als Grafik-Symbole erscheinen.

Nun kommt eine raffinierte Sache: schieben Sie mit dem "Abwärtspfeil" den Cursor vom Anfang des Textes bis zum Text-Ende. Immer wenn der Cursor an einer Gruppe von Grafikzeichen vorbeifährt, werden sie in Zahlen umgewandelt. Was sagen Sie nun?

Hinter jeder Zahl steht allerdings ein kleiner "Rechtspfeil" auf dem Bildschirm (ASCII-Code "89" HEX). Dieses Symbol ist nicht in den zulässigen Zeichen des ELECTRIC PENCIL enthalten und muß jeweils entfernt werden, um den Text auf dem Drucker auszugeben.

Abb. 9.6

```
F00000 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 2A0D *****.
F00010 2020 2020 2054 4849 5320 4953 2041 4E0D .....THIS.IS.AN.
F00020 454C 4543 5452 4943 2050 454E 4349 4C0D ELECTRIC.PENCIL.
F00030 4649 4C45 0D2A 2A2A 2A2A 2A2A 2A2A 2A2A FILE.*****
F00040 2A2A 2A2A 0D00 E5E5 E5E5 E5E5 E5E5 E5E5 ****.....
F00050 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F00060 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F00070 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F00080 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F00090 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000A0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000B0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000C0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000D0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000E0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
F000F0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....
```

Wenn Sie den "Rechtspfeil" einmal entfernt haben, können Sie allerdings diese Datei nicht wieder mit dem Text-Editor laden, ohne daß er nicht noch einen Satz Zeilennummern hinzufügt. Es ist besser, Sie entfernen mit ELECTRIC PENCIL die Zeilennummern ganz und lassen MACRO-80 wieder vollständig neue Nummern ergänzen.

Sie werden bemerken, daß hier keine "Titelzeile" vorhanden ist, wie wir sie von "EDTASM" kennen. MACRO-80

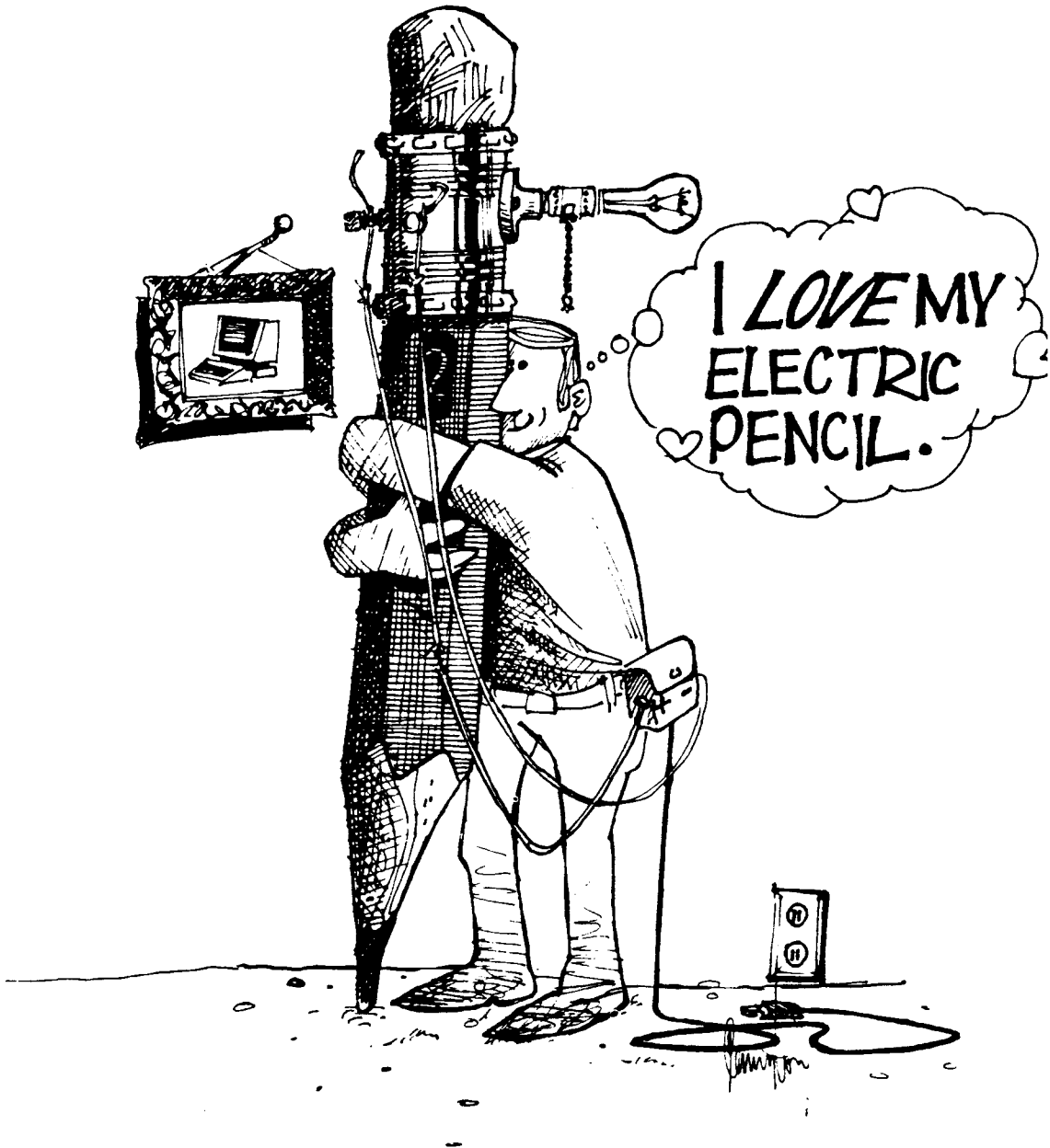
kann jede Art von ASCII-Datei laden und während des Ladevorganges seine eigenen Zeilennummern hinzufügen. Wenn Sie mit dem Editor eine Datei auf Diskette speichern, können Sie mit einem "Softwareschalter" verhindern, daß diese Zeilennummern mit übertragen werden. Es ist so möglich, mit MACRO-80 Dateien zu erzeugen, die ohne Änderung von ELECTRIC PENCIL gelesen werden können.

(Abb. 9.7)

```

F00000 B0B0 B1B0 B089 4245 4749 4E20 2020 4551 .....BEGIN...EQ
F00010 5520 2020 2020 3438 3030 300D B0B0 B2B0 U.....48000.....
F00020 B089 2020 2020 2020 2020 4F52 4720 2020 ..... .ORG...
F00030 2020 4245 4749 4E0D B0B0 B3B0 B089 2020 ..BEGIN.....
F00040 2020 2020 2020 4C44 2020 2020 2020 484C .....LD.....HL
F00050 2C30 4139 4448 0DB0 B0B4 B0B0 8920 2020 ,0A9DH.....
F00060 2020 2020 204C 4420 2020 202 2028 4646 .....LD.....(FF
F00070 4646 292C 484C 0DE0 B0B5 B0B0 8920 2020 FF),HL.....
F00080 2020 2020 2045 4E44 2020 2020 2042 4547 .....END.....BEG
F00090 494E 0D00 0000 0000 0000 0000 0000 0000 0000 IN.....
F000A0 0000 0000 0000 0000 0000 0000 0000 0000 .....
F000B0 0000 0000 0000 0000 0000 0000 0000 0000 .....
F000C0 0000 0000 0000 0000 0000 0000 0000 0000 .....
F000D0 0000 0000 0000 0000 0000 0000 0000 0000 .....
F000E0 0000 0000 0000 0000 0000 0000 0000 0000 .....
F000F0 0000 0000 0000 0000 0000 0000 0000 0000 .....

```



Ann. 2. Ü.: ÜBERSETZUNG ÜBERFLÜSSIG

10.0 Datenrettung

Wenn Sie so wie die meisten Leute dieses Kapitel zuerst lesen, dann kann ich Ihnen nur viel Glück wünschen.

Sie benötigen wirklich eine genaue Kenntnis über die Organisation einer Diskette und das Directory, bevor Sie die hier beschriebenen Dinge ausprobieren sollten. Also, egal wie mühsam es auch ist, springen Sie gefälligst an den Anfang des Buches und lesen Sie die vorangehenden Kapitel !

.....

Nun, nachdem Sie die ersten neun Kapitel gelesen haben (du liebe Güte, waren Sie aber schnell), können wir fortfahren...

10.1 Wiedergewinnung eines Hash-Codes für den "HIT"-Sektor

Da wir nicht wissen, wie der Hashcode berechnet wird, müssen wir uns etwas anderes einfallen lassen, um an ihn heranzukommen und das geht so:

1. Möglichkeit:

Speichern Sie ein einzeliliges BASIC- Programm auf einer anderen Diskette und verwenden Sie als Dateinamen den Namen, dessen Hashcode Sie wissen wollen. Zum Beispiel:

```
10 REM DAS IST EIN TEST
SAVE "(dateiname)"
```

Wir verwenden eine andere Diskette, damit DOS nicht zufällig den "FPDE" der Datei überschreibt, die Sie retten wollen. Wenn Sie allerdings diesen Teil des Directory in eine Pufferspur kopiert haben, kann Ihnen nichts passieren.

2. Möglichkeit:

Öffnen Sie per Direkteingabe in BASIC eine Datei und verwenden Sie als Dateinamen den entsprechenden Namen,

dessen Hashcode Sie herausfinden wollen. Zum Beispiel:

```
OPEN "0",1,"(Dateiname)" (ENTER)
```

Nachdem Sie so den Dateieintrag erzeugt haben, verwenden Sie SUPERZAP und suchen Sie den Hashcode aus der "HIT" heraus. Vergessen Sie nicht, ihn zu notieren! Anschließend löschen Sie die Test-Datei wieder (KILL).

10.2 Wiedergewinnen einer mit "KILL" gelöschten Datei

Wenn Sie eine Datei "KILL"en, passieren folgende Dinge:

1. Der Hashcode wird aus der "HIT" gelöscht.
2. Die "GAT" wird überarbeitet um die nun freigewordenen Granules richtig zu kennzeichnen.
3. Byte 0 im "FPDE" (und im "FXDE", falls existent) wird auf "00" geändert.

Alles andere bleibt wie es ist. Die Datei ist nach wie vor auf der Diskette vorhanden und auch die Eintragungen im "FPDE" und ggf. "FXDE" bleiben mit Ausnahme des ersten Bytes unverändert.

```
=====WARNUNG=====
Bei TRSDOS 2.2 werden alle Directory-
Eintragungen mit "00" überschrieben,
Wenn die Datei ge"KILL"t wird ! ! ! !
=====WARNUNG=====
```

Wie Sie der obigen Warnung entnehmen können, ist Radio Shack in seiner unendlichen Weisheit wieder etwas nettes eingefallen, wie sie uns Anwender ärgern können.

Sie können trotzdem Dateien auf einer TRS-DOS 2.2 Diskette retten, müssen aber auf der ganzen Diskette "herumsuchen", um alle Sektoren zu finden, die Teile der Datei enthalten.

10.2.1 Hier folgen die Schritte zur Wiedergewinnung einer "ge-KILLten" Datei:

Schritt 1:

Finden Sie den Hashcode des Dateinamens heraus (siehe 10.1).

Schritt 2:

"ZAP"pen Sie den Code an die richtige Stelle des "HIT"-Sektors.

Schritt 3:

"ZAP"pen Sie in Byte 0 des "FPDE" und ggf. "FXDE" jeweils den Wert "10"

10.2.2 Wenn es sich bei der Datei um ein BASIC-Programm handelt (Binär oder ASCII), dann.....

Schritt 1:

Laden Sie das Programm mit BASIC (LOAD).

Schritt 2:

Speichern Sie das Programm mit dem Dateinamen, unter dem Sie es nach 10.2.1 wiedergewonnen haben (SAVE). Damit wird die "GAT" korrigiert.

Schritt 3:

Führen Sie "DIRCHECK" aus, um sicherzugehen, daß keine Fehler im Directory vorliegen.

Anmerkung: Dateien lassen sich fehlerfrei laden, auch wenn Fehler in der "GAT" vorhanden sind, aber Vorsicht beim Speichern von Daten oder Programmen!

10.2.3 Wenn es sich um ASCII- oder Binär-Daten oder ein Assemblerprogramm handelt, dann....

Schritt 1:

Führen Sie "DIRCHECK" aus und notieren Sie sich alle "GAT"- und "HIT"-Fehler.

Schritt 2:

Unter Zuhilfenahme der "GAT"-Übersicht (Abb. 6.6) korrigieren Sie alle

"GAT"-Fehler, indem Sie die richtigen Werte in die "GAT" "ZAP"pen.

Schritt 3:

Wiederholen Sie Schritt 1 solange, bis keine Fehler mehr angezeigt werden.

10.3 Rettung einer Diskette, die keinen Systemstart ausführt oder deren Directory nicht lesbar ist.

So etwas ist schon eine große Gemeinsamkeit. Nun gibt es kaum einen Ausweg, es sei denn Sie hätten eine BACKUP-Diskette (Kopie), auf der wenigstens noch Teile der Directory lesbar sind. Andernfalls müssen Sie jetzt mit der Lieblingsbeschäftigung eines jeden Programmierers beginnen: SUCHEN SIE DIE DISKETTE AB und zwar von Anfang bis Ende, Punkt.

Bei Ihrer Suche, Sektor für Sektor sollten Sie sich viele Notizen machen, sonst können Sie sich bestimmt bald nicht mehr erinnern, auf welchen Sektoren welche Information steht.

Ich sehe, Bill Barden hat eine Frage, bitte Bill?

"Wie kommt es denn, daß ein Directory überhaupt aufgefressen werden kann?"

Hmmm, das ist eine gute Frage. Da gibt es ein Dutzend Gründe, die wichtigsten sind:

1. Sie haben eine geöffnete Datei ge"KILL"t, ohne vorher einen CLOSE-Befehl zu geben.

2. Sie haben das Diskettenlaufwerk mit einer eingesetzten Diskette eingeschaltet (meistens ist dann Spur 0 beschädigt, es kann aber auch das Directory sein).

3. Sie haben den Rechner oder das Expansion-Interface eingeschaltet und bereits eine Diskette im Laufwerk eingesetzt.

4. Sie haben versucht, auf den letzten Sektor einer fast vollen Diskette zu schreiben (bei TRS-DOS 2.1 oder 2.2)

5. DOS ist bei einer CLOSE-Operation "durcheinandergelassen" und hat einige Granules als frei gekennzeichnet. Einige weitere CLOSE- und SAVE-Befehle bringen dann in der Folge alles durcheinander, so daß schließlich sogar das Directory mit Daten überschrieben wird.

6. Ihr System enthält fehlerhafte Speicherbausteine und/oder der Dateikontrollblock enthält fehlerhafte Daten vor oder während einer "CLOSE"-Operation.

7. Fehlerhafte Logikschaltung im Diskettenlaufwerk.

8. Schreib/Lesekopf im Laufwerk falsch justiert.

9. Irgendjemand in Fort Worth mag Sie nicht.

10. Alle mögen Sie, Ihr System arbeitet einwandfrei, aber Sie wollen auch dieses Vergnügen der Datenrettung nicht verpassen.

10.3.1 Schritte, um ein total durcheinandergelassenes Directory zu retten.

1. Suchen Sie alle Teile einer Datei und notieren Sie, wo sie beginnen und die genaue Zahl der belegten Sektoren.

2. Formatieren Sie eine "Arbeitsdiskette".

3. Erzeugen Sie auf dieser Diskette einen "FPDE" mit der oben beschriebenen "OPEN"- "CLOSE"-Methode.

4. Übertragen Sie die anfangs gefundenen Sektoren von der Original- auf die Arbeitsdiskette, wenn möglich, in einen großen, zusammenhängenden

Block von Sektoren; damit wird "die Sache einfacher, da Sie nur einen EXTENT in den "FPDE" eintragen müssen.

5. Schreiben Sie einen EXTENT in den "FPDE", der auf den Datenblock zeigt.

6. Schreiben Sie in die "EOF"-Sektor Bytes (relative Bytes 14 und 15) die Anzahl der belegten Sektoren plus 1.

7. Schreiben Sie in das "EOF"-Byte (relatives Byte 3) die Lage des letzten Bytes der Datei im letzten Sektor. Wenn Sie das Ende nicht genau ausmachen können, dann machen Sie eine WWV (wilde wissenschaftliche Vermutung). Wenn Sie falsch liegen, können Sie es später immer noch ändern.

8. Falls es sich um ein BASIC-Programm handelt, laden Sie es und sehen Sie, wieviel tatsächlich geladen wird. Ihren Erfolg können Sie mit einem "LIST"-Befehl leicht feststellen. Wenn an einer Stelle Unsinn steht, notieren Sie sich die letzte fehlerfreie Programmzeile und machen Sie einen neuen Versuch, die schlechten Abschnitte zu ersetzen.

9. Wenn innerhalb eines Sektors nur einige Daten fehlerhaft sind, dann verwenden Sie die "COPY DISK DATA"-Funktion aus SUPERZAP um die richtigen Daten in den Sektor zu bringen.

10. Bei anderen als BASIC-Programmdateien, müssen Sie die Daten mit Befehlen wie LINEINPUT oder GET lesen und überprüfen.

11. Bei Maschinenprogrammen können Sie auch versuchen, die Datei zu überprüfen, indem Sie den DISASSEMBLER von Apparat benutzen, sofern Sie wissen, welche Befehle in dem Programm stehen müssen. Falls das nicht der Fall ist, können Sie das Programm nur laufen lassen und eventuelle Fehler mit "DEBUG" aufspüren.

10.3.2 Rettung einer Datei auf einer Diskette mit nicht lesbarem Directory.

Das ist nicht ganz so schlimm, ich will das Verfahren aber dennoch beschreiben. Wenn Sie feststellen, daß eine Systemdiskette keinen Systemstart ausführt (BOOT), oder wenn der DOS-Befehl "DIR" nicht arbeitet, dann liegt ein nicht lesbares Directory vor - nicht lesbar durch das System, aber meistens immer noch lesbar mit SUPERZAP. Hier die wichtigsten Gründe dafür:

1. Der Fehler liegt nicht am Directory, sondern tatsächlich im Sektor 0 in der Datei "BOOT/SYS".

2. Einer oder mehrere Directory-Sektoren haben einen Paritätsfehler.

3. Der Leseschutzstatus wurde durch irgendeinen Umstand von einem oder mehreren Directory-Sektoren entfernt.

Bevor wir uns mitten hineinstürzen, ist es eine gute Methode, die "BACK-UP"-Funktion von SUPERZAP zu probieren. In vielen Fällen wird die Diskette dadurch wieder "repariert". Falls dieser Schuß ins Dunkle danebengeht, versuchen Sie folgendes Verfahren:

Um den Fehler zu beheben, müssen Sie ihn zunächst genau identifizieren. Verwenden Sie dazu die "DD"-Funktion aus SUPERZAP.

1. Überprüfen Sie die Directory-Sektoren auf Paritätsfehler. Diese werden automatisch erkannt, wenn Sie einen solchen Sektor mit "DD" lesen wollen. Wenn nach einer Fehlermeldung der Inhalt des Sektors "gut" aussieht, kopieren Sie die Daten mit "SCOPY" in DENSELBEN Sektor zurück. Damit wird der Paritätsfehler selbstständig korrigiert.

2. Überprüfen Sie die Sektoren auf ihren "Leseschutz"-Status. Es muß in der letzten Zeile der SUPERZAP-Anzeige eine "6" links in der siebten

Spalte stehen, sonst ist der Sektor nicht lesegeschützt und muß korrigiert werden.

Dazu gehen Sie wie folgt vor:

a. Kopieren Sie den Sektor in einen "Puffersektor".

b. Kopieren Sie aus einem fehlerfreien Directory einen beliebigen Sektor in den Sektor mit fehlerhaftem Status.

c. Kopieren Sie die Originaldaten, die jetzt im Puffersektor stehen, wieder zurück, aber kopieren Sie NUR die ERSTEN 255 Bytes ("FE" HEX) zurück. Dadurch bleibt der Leseschutz-Status des Sektors erhalten.

d. Schreiben Sie per Hand mittels "MODFF" den Inhalt des 256. Byte wieder an die richtige Stelle.

3. Wenn bei Überprüfung des Directory kein Fehler vorliegt, dann kann nur "BOOT/SYS" falsch sein. Nehmen Sie einfach eine Diskette mit einem fehlerlosen "BOOT/SYS" und kopieren Sie von dieser Diskette den Sektor 0 aus Spur 0 auf die beschädigte Diskette.

10.4 Rettung einer elektrisch beschädigten Diskette

Wenn die Diskette mechanisch in Ordnung ist und auch kein Gerätefehler vorliegt, aber einige Sektoren oder Spuren nicht zu laden sind, dann ist es möglich, daß die Daten auf der Diskette durch statische Aufladungen verändert wurden. Auch kann es sein, daß Sie auf die Diskette geschrieben haben und sie nicht genau zentriert im Laufwerk steckte. Die meisten Ursachen, die schon unter 10.3 genannt wurden, können auch hier zutreffen. Dazu kommen noch einige Fehlermöglichkeiten, die nicht ganz so offensichtlich sind:

1. Nehmen Sie sich in Acht vor magnetisierten Büroklammern! Diese Dinger liegen überall herum und veran-

stalten ein Festessen mit Ihren Daten. Besonders begierig sind die Büroklammern, die in einem Magnethalter aufgehoben wurden, wie er in Büros gebräuchlich ist. Lassen Sie nie zu, daß Büroklammern auf Disketten gesteckt werden. Selbst wenn sie nicht magnetisch sind, können sie die Diskette mechanisch beschädigen.

2. Eine Diskette hat unter dem Telefon gelegen und ist damit zu einem Kandidaten für die Neuformatierung geworden. Es ist besonders günstig, wenn das Telefon läutet, Sie sparen sich ein Löschgerät!

3. Die Diskette hat neben einem Gerät mit einem Elektromotor oder einem Transformator gelegen. Damit kann sie hervorragend gelöscht werden.

Eine schlecht zentrierte Diskette ist ein häufiges Problem, besonders bei Laufwerken von Shugart, Pertec und Wangco. Das hängt mit dem relativ kurzen Zentrierkonus zusammen, den diese Laufwerke besitzen. Das Problem kann leicht dadurch vermieden werden, wenn Sie die Klappe erst dann schließen, nachdem der Antriebsmotor läuft. Allerdings können Sie mit TRSDOS 2.1 und 2.2 den berühmten "stillen Absturz" des Systems beobachten, wenn Sie die Klappe nicht geschlossen haben, bevor DOS auf die Diskette zugreifen will. NEWDOS und VTOS 3.0 warten auf Sie. VTOS 3.0 kann etwas leichter verwirrt werden, aber mit "SHIFT - BREAK" können Sie einen neuen Versuch starten.

Nun zur Rettung solcher Disketten:

1. Verwenden Sie die "VERIFY"-Funktion von SUPERZAP, um die Sektoren zu prüfen und notieren Sie sich alle als fehlerhaft erkannten Sektoren.

2. Formatieren Sie eine Diskette und duplizieren Sie Ihr beschädigtes Original darauf mit der "DISK BACK-UP"-Funktion von SUPERZAP und überspringen Sie alle Sektoren, die sich

auch nach einem nochmaligen Leseversuch nicht fehlerfrei lesen lassen.

3. Sehen Sie sich mit "DD" die Sektoren vor und hinter den als fehlerhaft erkannten Sektoren an, daraus kann man unter Umständen erkennen, ob sie überhaupt wichtige Daten enthalten haben.

4. Wenn Sie sich nicht sicher sind, ob ein bestimmter Sektor verwendet wird, sehen Sie sich die "GAT" an und stellen fest, ob der betreffende Sektor zugewiesen ist. Falls Sie einen Sektor als belegt erkennen, aber nicht wissen, von welcher Datei er belegt ist, dann sparen Sie sich das Absuchen des Directory. Entfernen Sie die Zuweisung in der "GAT" einfach und rufen Sie anschließend "DIRCHECK" auf. Sie bekommen dann in einer Fehlermeldung angezeigt, zu welcher Datei der betreffende Sektor gehört. Anschließend sollten Sie ggf. die "GAT" wieder korrigieren.

5. Versuchen Sie die fragliche Datei zu laden, die auf dem beschädigten Abschnitt des Directory war und verfahren Sie weiter nach den Schritten 8, 9 und 10 gemäß 10.3.1.

10.5 Rettung einer mechanisch beschädigten Diskette

Dieser unglückliche Umstand kann auf vielerlei Weise eintreten. Sie können die Diskette versehentlich zerkratzt haben, als Sie sie letztens als Schuhanzieher verwendet hatten. Vielleicht hat auch ein Angestellter des Radio Shack Centers sie mit einer Kreditkarte verwechselt und durch die Maschine laufen lassen. Unlängst besuchte mich ein Freund, der von einem Hähnchen-Wettessen kam und faßte eine Diskette mit seinen Fettfingern genau an der Stelle an, auf der normalerweise der Schreib/Lesekopf arbeitet.

In jedem Fall treffen die gleichen Lebensrettungsverfahren zu, wie im vorangehenden Abschnitt beschrieben.

ACHTUNG: bevor Sie die Diskette in das Laufwerk stecken, überzeugen Sie sich, daß keine losen Fremdkörper auf der Diskettenoberfläche sind, sonst verschmutzen oder beschädigen Sie den Schreib/Lesekopf Ihres Laufwerkes.

Stellen Sie fest, daß die Diskettenoberfläche durch fremde Substanzen wie Fingerabdrücke, Kaffee, Handlotion und so weiter verschmutzt ist, dann können Sie folgendes, halbbrutale Verfahren anwenden, das mir schon in einigen Fällen geholfen hat:

1. Öffnen Sie vorsichtig die Diskettentasche und entnehmen Sie ihr die Diskette. **FASSEN SIE DIE OBERFLÄCHE NICHT MIT DEN FINGERN AN!**

2. Waschen Sie die Diskette leicht in warmem Seifenwasser mit angefeuchteten **UND** eingeseiften Händen. Streichen Sie leicht über die Diskette, **NICHT REIBEN!**

3. Spülen Sie sie gründlich in warmem Wasser ab.

4. Wenn Wasser und Seife keinen Erfolg bringen, dann setzen Sie dem Wasser Alkohol zu (nicht trinken!) und versuchen Sie es erneut.

5. Wiederholen Sie Schritt 3.

6. Legen Sie die Diskette auf ein Blatt Zeitungspapier. Achtung, Papierhandtücher fusseln! Legen Sie ein anderes Blatt darüber und drücken Sie leicht auf die Diskette. Das wiederholen Sie solange, bis die Diskette trocken ist.

7. Reiben Sie die Diskette unter keinen Umständen ab !

8. Nun setzen Sie die Diskette wieder vorsichtig in eine **NEUE** Tasche ein und achten darauf, daß sie nicht an der Oberfläche berührt wird. (Hier haben Sie endlich einen Verwendungszweck für die Disketten, die andere Fehler haben, die Sie aber immer noch nicht weggeworfen haben.)

9. Fertigen Sie **SOFORT** eine Kopie dieser Diskette an.!

10.6 Beheben eines "BAD PARITY"-Fehlers

Diese Meldung ist sicher auch Ihnen bekannt. Ein Paritätsfehler in einem Sektor kann durch ein fehlerhaftes Bit verursacht werden, es können aber im schlimmsten Fall auch sämtliche Bits falsch sein. Manchmal sind auch die Daten in Ordnung, aber das Paritäts-Byte selbst ist falsch.

Wenn Sie feststellen, daß bei einem Laufwerk häufiger solche Fehler auftreten, als bei einem anderen, dann kann leicht das Laufwerk die Ursache sein. Es ist auch möglich, daß Paritätsfehler nur bei Leseoperationen auftreten, auch in diesem Fall sollten Sie das Laufwerk überprüfen.

Rettungsmethode 1:

1. Verwenden Sie die **SUPERZAP**-Funktion "**VERIFY DISK SECTORS**" zum Aufspüren von "schlechten" Sektoren. Wenn Sie nur wenige finden, dann....

2. Rufen Sie die "**DD**"-Funktion auf, um den Sektor zu lesen. Scheinen die Daten richtig zu sein und können sie dort keinen Fehler entdecken, dann geben Sie "**MOD00**" ein und sofort "**ENTER**". Dadurch wird der Sektor mit richtigem Paritätsbyte zurückgeschrieben, ohne daß an den Daten etwas geändert wurde.

Wenn Sie den Fehler damit nicht beheben können, dann versuchen Sie folgendes Verfahren:

1. Wenden Sie die "**BACKUP**"-Funktion aus **SUPERZAP** an. Versuchen Sie einen Sektor mit Paritätsfehler wiederholt zu lesen, indem Sie "**R**" eingeben. Wenn das nichts hilft, notieren Sie sich den Sektor und überspringen Sie ihn dann ("**S**").

2. Stellen Sie fest, ob die fehlerhaften Sektoren tatsächlich von einer

Datei benutzt werden, denn es hat wenig Sinn, einen Sektor zu retten, der garnicht verwendet wird. Die so erzeugte Kopie ist jetzt ohne Fehler auch mit der normalen BACKUP-Funktion des DOS zu kopieren.

3. Wenn von den fehlerhaften Sektoren tatsächlich einige von Dateien benutzt werden, dann können wir eine der beiden folgenden Methoden anwenden:

Methode 1:

a. Versuchen Sie, den Sektor mit "DD" zu lesen. Wenn der Versuch einigermaßen erfolgreich war, dann übertragen Sie den Sektor mit "SCOPY" in einen Puffersektor oder eine Puffer Spur.

b. Machen Sie weitere Versuche, den Sektor mit "DD" zu lesen und kopieren Sie auch einen nur teilweise gelesenen Sektor in einen Puffersektor, solange bis Sie sicher sind, daß keine weiteren gültigen Daten mehr aus dem Sektor gelesen werden können.

c. Drucken Sie die Inhalte der Puffersektoren mit "PD" aus.

d. Rekonstruieren Sie den Sektor sorgfältig durch quervergleichen der Ausdrücke, wenden Sie die "MODnn"-Funktion an und schreiben Sie in einen weiteren Puffersektor die so gewonnenen Daten Byte für Byte. Wenn die Puffersektoren längere Stücke sinnvoller Daten enthalten, können Sie auch die "COPY DISK DATA"-Funktion anwenden.

e. Nachdem der Sektor rekonstruiert ist, kopieren Sie ihn wieder auf seinen alten Platz zurück.

Methode 2:

a. Suchen Sie eine frühere Version des durcheinandergebrachten Sektors und kopieren Sie diese an seine Stelle.

b. Wenn die frühere Version unvollständig ist und Sie ohnehin die meisten Daten selbst ergänzen müssen, dann schieben Sie die, dem fehlerhaften Sektor folgenden Sektoren um eine Position vor (also über den fehlerhaften Sektor) und ändern im "FPDE" die Sektorzahl und das "EOF"-Byte um.

10.7 Beseitigung eines "DIRECT STATEMENT IN FILE"-Fehlers

Ich muß zugeben, die ersten hundert Fehlermeldungen dieser Art haben mich fast bis an den Rand des Wahnsinns getrieben. (Fast ?) Ich habe vor dem TRS-80 noch nie mit Computern zu tun, daher haben mich diese schleierhaften Fehlermeldungen ohne nähere Erläuterungen schon sehr verwundert.

Was es noch schlimmer machte war, daß weder das Level II-Handbuch noch das DOS-Handbuch eine Erklärung gaben, was es mit diesem "DIRECT STATEMENT IN FILE" (direkte Anweisung im Programm) auf sich hat.

Dieser Fehler kann unter folgenden Umständen passieren: er ist normalerweise die Auswirkung eines kleinen Fehlers im Level II-BASIC, der dann auftritt, wenn Sie ein Programm in ASCII-Code speichern (SAVE "filename",A), das Zeilen enthält, die länger als 240 Bytes sind, wenn der Programmtext zu seiner vollen Länge ausgedehnt wird (z.B. aus "?" wird "PRINT").

Im Level II-Handbuch heißt es klar und deutlich:

"Länge von Programmzeilen: bis zu 255 Zeichen."

Auf der anderen Seite kann aber BASIC nur Programmzeilen bis zu einer Länge von 240 Bytes auf einmal lesen (mit "LOAD")

Bei einem BASIC-Programm muß jeder Zeile eine Zeilennummer vorangestellt sein. Ein "DIRECT STATEMENT IN FILE"-Fehler tritt dann auf, wenn

BASIC die Zeilennummer und 240 Bytes von der Diskette liest und noch Daten übrigbleiben, denen keine Zeilennummer vorangeht. Nun weiß BASIC nicht, was es mit diesen Daten anfangen soll.

Was hat das aber mit dem ASCII-Format zu tun? Nun, Level II-BASIC verwendet tatsächlich Befehlskürzel (Tokens), um ein Programm im RAM zu speichern. Wenn Sie z.B. "PRINT" eingeben wird nicht dieser Text als ASCII-Zeichenfolge gespeichert, sondern stattdessen der Wert "B2" (HEX). Dadurch wird nur ein Byte anstelle von fünf Bytes für "PRINT" belegt.

Wenn Sie ein Programm eingeben, achtet das System darauf, wieviele Zeichen alle Befehle einer Zeile belegen, wenn sie voll ausgeschrieben werden, so daß Sie bei der Eingabe einer Programmzeile nie einen solchen Fehler erzeugen können (das System nimmt einfach keine weiteren Zeichen mehr an). Damit kann auch kein Fehler auftreten, wenn Sie dieses Programm in ASCII-Format speichern. Editieren Sie jedoch diese Zeile, so erlaubt Ihnen Level II, einige zusätzliche Zeichen einzufügen, gerade genug, um über die zulässige Grenze zu kommen. Da haben Sie es! Jetzt ist das Dilemma perfekt, wenn Sie nun dieses Programm auf Diskette speichern und es anschließend wieder laden möchten.

Wie können wir nun den Fehler beseitigen? Es ist eine recht einfache Sache, alles was wir machen müssen, ist eine Zeilennummer vor die widersprüchliche "direkte Anweisung" in der Datei schreiben.

10.7.1 ASCII-Datei mit "DIRECT STATEMENT"-Fehler

1. Finden Sie die letzte Programmzeile, die fehlerfrei geladen wurde (mit "LIST").

2. Finden Sie die letzten Zeichen, die fehlerfrei geladen wurden.

3. Finden Sie die Datei auf der Diskette nach dem unter 10.0 beschriebenen Verfahren.

4. Lesen Sie Sektor für Sektor, bis Sie den fehlerhaften gefunden haben. Das geht in einer ASCII-Datei leicht, da alles, einschließlich der Zeilennummer in lesbarer Form angezeigt wird.

5. Nun "ZAP"pen Sie eine Zeilennummer an beliebiger Stelle in die fehlerhafte Programmzeile. Diese Nummer muß größer sein als die vorangehende, jedoch kleiner als die nachfolgende (hoffentlich haben Sie das Programm nicht in Einerschritten nummeriert). Dadurch verlieren Sie zwar einige Zeichen, aber das ist nur ein kleiner Preis.

10.7.2 Eine Binär-Programmdatei mit einem "DIRECT STATEMENT"-Fehler

Das ist eine ganz seltene Sache und ist bei mir in einem Jahr vielleicht zweimal vorgekommen.

Auch dieser Fehler kann behoben werden, wenn auch ein wenig mühsamer. Jetzt sind alle Zeilennummern hexadezimal und die Befehle in Token-Form auf der Diskette gespeichert, aber Sie können Teile des Programms dennoch als lesbaren Text erkennen, denn Variablenbezeichnungen, Zeichenketten und Bemerkungen werden nicht verändert.

Es ist mir nicht bekannt, wodurch dieser Fehler entsteht und ich kann ihn aus diesem Grund auch nicht reproduzieren. Das folgende Beispiel ist daher erfunden, aber dennoch gültig.

Abb. 10.1 zeigt einen Abschnitt eines BASIC-Programms in Binär-Format. Sie werden erkennen, daß es sich wieder um ein Stück von SUPERZAP handelt und somit können Sie also auch damit experimentieren. Abb. 10.3 zeigt einen Ausdruck dieses Programmabschnittes

mit und ohne den simulierten Fehler, Sie sind damit in der Lage, den Programmtext mit dem zu vergleichen, was stattdessen auf der Diskette gespeichert ist.

In Abb. 10.1 sind die relativen Bytes "3D" und "3E" typische Zeilennummern, Sie finden dort die Werte "64" und "00". Um Sie zu entziffern, müssen Sie die Reihenfolge vertauschen. Wenn Sie Ihre Hausaufgaben gemacht hätten, anstelle Kaugummi zu kauen, wüßten Sie jetzt, daß hexadezimal "0064" gleich "100" dezimal ist.

Unser simulierter "DIRECT STATEMENT"-Fehler beginnt im relativen Byte "C6" und setzt sich in den folgenden fünf Bytes fort.

Diese Bytes lauten "95 3A 92 3A 94". Im Normalfall wissen Sie natürlich nicht genau, wo sich die fehlerhaften Bytes befinden. Alles, was Sie bemerken ist, daß sich ein Programm nur bis zu einer bestimmten Stelle in einer bestimmten Zeile laden läßt. Aber da liegt auch der Schlüssel, wie wir das verdammte Ding fassen können.

Da die genaue Position, in die wir eine neue Zeilennummer "ZAP"pen wol-

len, ziemlich unwichtig ist, nehme ich hier das relative Byte "C1" und beginne dort mit den Änderungen. Wir brauchen eine Zeilennummer, größer als 300, aber kleiner als die nächstfolgende, die hier 400 lautet. Was meinen Sie, 350 ist doch ein guter Vorschlag. Rechnen wir um: 350 dez. = 015E (HEX). Wie immer drehen wir die Sache herum und erhalten "5E 01". Zusätzlich müssen wir aber auch noch die Codes ergänzen, damit BASIC eine Zeile fehlerfrei lesen kann.

Diese Codes stehen in den drei Bytes, die einer Zeilennummer vorangehen und fangen stets mit "00" an. Da wir ja lediglich die Datei laden wollen (Fehler im Programmtext können wir dann später beheben), "leihen" wir uns einfach einen Code von einer anderen Zeilennummer ("00 80 69" ist der Code von Zeile 300) und nun haben wir alles, um die Operation zu beenden.

Wir beginnen mit unseren "ZAPS" ab dem relativen Byte "C1" und schreiben in dieses und die folgenden Bytes: "00 80 69 5E 01". Abb. 10.2 zeigt, wie der Sektor nach dieser Änderung aussieht.

Abb. 10.1

```

01A000 FFF4 6832 0093 3A20 4D41 494E 2F44 4953 ...2...:MAIN/DIS
01A010 4B20 4D45 4D4F 5259 2044 554D 502F 4D4F K.MEMORY.DUMP/MO
01A020 4449 4659 2052 4F55 5449 4E45 2E20 2056 DIFY.ROUTINE...V
01A030 4552 5349 4F4E 2032 2E30 0000 6964 008D ERSION.2.0.....
01A040 2031 3034 3030 0029 6996 0041 24D5 C93A .10400.)...A$.:.
01A050 208F 2041 24D5 2222 20CA 2031 3530 3A20 ...A$. " " ...150:..
01A060 203A 9520 4258 D5F6 2841 2429 3A20 9200 ...:BX..(A$):...
01A070 4E69 C800 8F20 4258 20D4 D534 3820 D220 N....BX...48...
01A080 4258 D6D5 3537 20CA 2042 58D5 4258 CE34 BX..57...BX.BX.4
01A090 383A 2092 0072 69FA 008F 2042 58D4 D536 8:.....BX..6
01A0A0 3520 D220 4258 D6D5 3730 20CA 2042 58D5 5...BX..70...BX.
01A0B0 4258 CE35 353A 2092 0080 692C 0142 58D5 BX.55:.....,BX.
01A0C0 CE42 583A 2092 953A 923A 9493 3A20 2A2A .BX:.....:..**
01A0D0 2A2A 2A2A 2A2A 2A20 5641 5249 4142 4C45 *****.VARIABLE
01A0E0 2041 4C4C 4F43 4154 494F 4E20 494E 4849 .ALLOCATION.INHI
01A0F0 4249 5445 4400 D569 9001 4432 2528 3129 BITED.....D2%(1)

```

Laden Sie nun das Programm, korrigieren Sie die Zeile, die wir gerade erzeugt haben und auch die vorangehende Zeile. Nun speichern Sie das Programm wieder zurück und alles ist in Ordnung.

10.8 Rettung von Daten-Files

Bisher haben wir uns mit der Rettung von Programm-Files befasst. Hier geht es jetzt um Dateien, die andere als Programminformationen enthalten. Es gibt nichts, was die Rettung solcher Dateien schwieriger oder leichter

macht, als bei den vorangehenden Typen. Prinzipiell haben wir es mit zwei Formaten zu tun: (1) ASCII-Dateien und (2) Dateien in komprimierter Binärform. Die "FPDE" und "FXDE" sind nicht anders zu verstehen, als bei anderen Dateien auch, so daß Sie diese Dateien nach den gleichen Verfahren auf der Diskette finden können, wie schon besprochen.

Abb. 10.2

```

01A000  FFF4 6832 0093 3A20 4D41 494E 2F44 4953 ...2...:MAIN/DIS
01A010  4B20 4D45 4D4F 5259 2044 554D 502F 4D4F K.MEMORY.DUMP/MO
01A020  4449 4659 2052 4F55 5449 4E45 2E20 2056 DIFY.ROUTINE...V
01A030  4552 5349 4F4E 2032 2E30 0000 6964 008D ERSION.2.0.....
01A040  2031 3034 3030 0029 6996 0041 24D5 C93A .10400.)...A$:..
01A050  208F 2041 24D5 2222 20CA 2031 3530 3A20 ...A$.""...150:..
01A060  203A 9520 4258 D5F6 2841 2429 3A20 9200 ...BX..(A$):...
01A070  4E69 C800 8F20 4258 20D4 D534 3820 D220 N.....BX...48...
01A080  4258 D6D5 3537 20CA 2042 58D5 4258 CE34 BX..57...BX.BX.4
01A090  383A 2092 0072 69FA 008F 2042 58D4 D536 8:.....BX..6
01A0A0  3520 D220 4258 D6D5 3730 20CA 2042 58D5 5...BX..70...BX.
01A0B0  4258 CE35 353A 2092 0080 692C 0142 58D5 BX.55:.....,BX.
01A0C0  CE00 8069 5E01 953A 923A 9493 3A20 2A2A .BX:.....:..**
01A0D0  2A2A 2A2A 2A2A 2A20 5641 5249 4142 4C45 *****.VARIABLE
01A0E0  2041 4C4C 4F43 4154 494F 4E20 494E 4849 .ALLOCATION.INHI
01A0F0  4249 5445 4400 D569 9001 4432 2528 3129 BITED.....D2%(1)

```

Normaler Programmausdruck von SUPERZAP

```

50 REM: MAIN/DISK MEMORY DUMP/MODIFY ROUTINE. VERSION 2.0
100 GOTO 10400
150 A$=INKEY$: IF A$="" THEN 150: ELSE BX=ASC(A$): RETURN
200 IF BX >=48 AND BX<=57 THEN BX=BX-48: RETURN
250 IF BX>=65 AND BX<=70 THEN BX=BX-55: RETURN
300 BX=-BX: RETURN
350 REM: ***** VARIABLE ALLOCATION INHIBITED
400 D2%(1)=VARPTR(D2%(5)): DEFUSR2 = VARPTR(D2%(0))
450 X=USR2(0): RETURN
500 ' ***** END OF VARIABLE ALLOCATION INHIBIT
550 GOSUB 150: GOTO 200

```

Programmausdruck von SUPERZAP mit berichtigtem simulierten Fehler
beachten Sie, daß Zeile 350 nun Unsinn enthält, aber das Programm wird geladen

```

50 REM: MAIN/DISK MEMORY DUMP/MODIFY ROUTINE. VERSION 2.0
100 GOTO 10400
150 A$=INKEY$: IF A$="" THEN 150: ELSE BX=ASC(A$): RETURN
200 IF BX >=48 AND BX<=57 THEN BX=BX-48: RETURN
250 IF BX>=65 AND BX<=70 THEN BX=BX-55: RETURN
300 BX=-
350 LSE:RETURN:STOPREM: ***** VARIABLE ALLOCATION INHIBIT
400 D2%(1)=VARPTR(D2%(5)): DEFUSR2 = VARPTR(D2%(0))
450 X=USR2(0): RETURN
500 ' ***** END OF VARIABLE ALLOCATION INHIBIT
550 GOSUB 150: GOTO 200

```

10.8.1 ASCII-Dateien

ASCII-Dateien sind am leichtesten zu lesen, da Sie den Inhalt im rechten Block der "DD"-Anzeige von SUPERZAP sofort erkennen können. Abb. 10.4 zeigt einen typischen Sektor mit solchen Daten.

Sie werden bemerken, daß jedes Datenelement einer ASCII-Datei durch ein "," von seinem Vorgänger getrennt ist. Das "EOF" der Datei wird durch ein leeres Element gekennzeichnet, d.h. zwei Kommata ohne Zwischenraum ",",. In unserem Fall finden Sie das z.B. in den relativen Bytes "E7" und "E8".

Um Daten zu reparieren, brauchen Sie nur die entsprechenden Bytes zu "ZAP"pen. Nehmen wir an, Sie wollten die "1000" in der ersten Zeile in eine "2000" ändern, so müssen Sie in das relative Byte "01" den Wert "32" schreiben. Die "32" ist ja der ASCII-Code für die Ziffer "2". (Aber benutzen Sie Ihr Wissen jetzt nicht, um in der Datei der Lohnbuchhaltung Ihr Gehalt zu erhöhen!)

10.8.2 Wahlfreie Dateien (Random Files)

Hier handelt es sich um Dateien, die z.B. aus BASIC durch "PUT"-Anweisungen entstanden sind. Bei diesen Dateien sind Zahlen in komprimierter Binärdarstellung gespeichert. Aus diesem Grund sind sie nicht so übersichtlich und leicht zu ändern.

Die Abb. 10.5 zeigt einen Sektor, der solche Daten enthält. Sie erkennen, daß es hier zwischen den einzelnen Datenelementen keine Abgrenzungen wie bei den ASCII-Dateien gibt. Um die einzelnen Elemente auseinanderzutrennen, setzen Sie in Ihrem Programm die "FIELD"-Anweisung ein.

Die numerischen Daten, wie "INTEGER", "SINGLE PRECISION" und "DOUBLE PRECISION" werden hier binär in komprimierter Form dargestellt. Damit Sie eine solche Datei wirksam bearbeiten können, sollten Sie sich unbedingt vorher eine Übersicht anfertigen, aus der hervorgeht, wie die einzelnen Daten angeordnet sind. Sie können eine der Übersichten aus Kapitel 6

Abb. 10.4

```

30E500 2031 3030 3020 2020 2020 2020 2020 2020 2020 .1000.....
30E510 3230 3734 4E2D 3736 2C47 454D 2020 2043 2074N-76,GER...C
30E520 4F52 504F 5241 5449 4F4E 2C32 3232 2057 ORPORATION,222.W
30E530 494C 5348 4952 4520 424C 5644 2E2C 4C4F ILSHIRE.BLVD.,LO
30E540 5320 414E 4745 4C45 532C 4341 4C49 464F S.ANGELES,CALIFO
30E550 524E 4941 2C39 3030 3137 2C48 4152 5259 RNIA,90017,HARRY
30E560 2048 2E20 4448 4F52 4520 2020 2020 2020 .H..DHORE.....
30E570 2041 2E46 2E47 2E41 2C20 4841 4E4B 2044 .A.F.G.A,.HANK.D
30E580 2E20 504F 4E42 414D 2020 472E 472E 2020 ..PONBAM..C.G...
30E590 412E 462E 472E 412C 5245 5345 5256 4544 A.F.G.A,RESERVED
30E5A0 2C49 2044 4944 4E27 5420 5448 494E 4B20 ,I.DIDN'T.THINK.
30E5B0 594F 5520 574F 554C 4420 4649 4E44 2054 YOU.WOULD.FIND.T
30E5C0 4849 532C 0D46 5241 4E4B 2C0D 322E 3020 HIS,.FRANK,.2.0.
30E5D0 2030 362F 3031 2F37 390D 2020 2E2E 2E20 .06/01/79.....
30E5E0 482E 502E 2020 0D2C 2C5E E5E5 E5E5 E5E5 H.P.,,.....
30E5F0 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 .....

```

als Anhalt nehmen. Wenn Sie diese Übersicht fertig haben, ist es leicht, die einzelnen Datenelemente in einem Sektor durch Abzählen herauszufinden und ggf. zu ändern.

Wenn Sie sich die Abb. 10.5 genau ansehen, werden Sie erkennen, daß hier zwei identische Subrecords verwendet werden. Nehmen Sie Ihr Disk-Handbuch von Radio Shack zur Hand und sehen sich dort einmal das Kapitel über Random-Files (wahlfreie Dateien) an. Es werden da "logische" und "physikalische" Records erwähnt. Offensichtlich entspricht ein Sektor einem "physikalischen" Record. Wenn Sie diesen in weitere Abschnitte unterteilen, so nennt man diese Abschnitte "logische" Records.

Es gibt einen kleinen Unterschied bei der Behandlung solcher Dateien durch die verschiedenen Betriebssysteme. TRSDOS 2.1 und NEWDOS 2.1 erlauben pro physikalischen Record nur 255 Bytes. Ändern Sie nicht das relative Byte "FF", wenn die Datei unter einem dieser Betriebssysteme verwendet werden soll! TRSDOS 2.2, VTOS 3.0 und das neue SUPERDOS von APPARAT erlauben die Verwendung aller 256 Bytes.

Lassen Sie uns die Abb. 10.5 etwas genauer ansehen, bevor wir zum nächsten Kapitel übergehen. Die ersten 40 Bytes enthalten einen Namen, und die Anschrift. ("Aber Sie sagten doch, daß Zahlen in komprimierter Binärform dargestellt werden und nicht in ASCII!") Ja, das habe ich, aber Sie können auch ASCII verwenden, wenn Sie in Ihrem Programm eine Zahl als Zeichenkette verarbeiten.

In den nächsten vier Bytes finden Sie die Postleitzahl als "SINGLE PRECISION"-Zahl binär verschlüsselt. Im ersten Subrecord lautet die Postleitzahl "90266". Um die Verwirrung komplett zu machen: "90266" (dez.) = "1609A" (HEX). In der Datei wird diese Zahl aber als "004D3091" (HEX) dargestellt. Nun frage ich Sie: gibt das einen Sinn? Ja, zufällig doch.

Dazu muß man erst einmal verstehen, wie Zahlen intern im Rechner verarbeitet werden. Die Komprimierungsverfahren von BASIC erfordern, daß außer der eigentlichen Ziffernfolge auch das Vorzeichen, der Exponent und die Lage des Dezimalpunktes gespeichert wird. (Das ist übrigens das

(Abb. 10.5)

```

F01700  5E45 4143 4850 4954 2047 494E 4745 5220 PEACHPIT.GINGER.
F01710  3131 3530 2054 454E 4E59 534F 4E20 2332 1150.TENNYSON.#2
F01720  3420 2020 2020 2020 004D 3091 5543 4A54 4.....M0.UCJT
F01730  0000 0000 2D86 E000 3086 8000 2020 2020 .....-...0.....
F01740  2020 2020 2020 2020 2020 2020 2020 2020 .....
F01750  2020 2020 2020 2020 2020 2020 2020 2020 .....
F01760  0000 0000 2045 86F0 0020 4B86 8800 2020 .....E....K.....
F01770  2020 2020 2020 0000 0000 2020 2020 2050 .....P
F01780  4F4C 4543 4154 2052 5554 4820 3131 3636 OLECAT.RUTH.1166
F01790  3120 4B49 4F57 4120 4156 452E 2023 3420 1.KIOWA.AVE..#4.
F017A0  2020 2020 2020 2080 E02F 9155 4344 4300 ...../..UCDC.
F017B0  0000 002D 86E0 0030 8680 0020 2020 2020 ...-...0.....
F017C0  2020 2020 2020 2020 2020 2020 2020 2020 .....
F017D0  2020 2020 2020 2020 2020 2020 2020 2000 .....
F017E0  0000 0020 4586 F000 204B 8688 0020 2020 ....E....K.....
F017F0  2020 2020 2000 0000 0020 2020 2020 0000 .....

```

Thema eines weiteren Buches, Arbeitstitel: BASIC, kommentiert, ausgedruckt und erzählt.)

In der Zwischenzeit ist alles, was Sie wirklich wissen müssen, wie Ihre Zahlen aussehen. Hier zeige ich Ihnen den Weg, den Sie einschlagen sollten, um diese hexadezimale Darstellung mit einem kleinen BASIC-Programm zu decodieren. Geben Sie das folgende Programm ein und starten Sie es:

```

100 A = 50266          :' WERT VON A
                        SETZEN
110 A$ = MKI$ (A)      :' UMWANDLUNG
                        IN ZEICHEN-
                        KETTE
120 PRINT A$          :' A$ ANZEIGEN
130 PRINT LEN (A$)    :' LAENGE AN-
                        ZEIGEN
140 FOR X=1 TO LEN (A$)
150 PRINT
    ASC(MID$(A$,X,1)) :' DEZIMAL-
                        WERTE AN-
                        ZEIGEN
160 NEXT              :' SCHLEIFE

```

PROGAMM
ANZEIGE BEDEUTUNG

MO	ANZEIGE VON A\$
4	LAENGE VON A\$
0	1. ASCII ZEICHEN = 00 HEX
77	2. ASCII ZEICHEN = 4D HEX
48	3. ASCII ZEICHEN = 30 HEX
145	4. ASCII ZEICHEN = 91 HEX

Um zu sehen, wie andere Variablentypen dargestellt werden, ersetzen Sie im Programm den Befehl "MKI\$" mit einem der folgenden Befehle

MKI\$ - Umwandlung ganzer Zahlen (INTEGER)
 MKS\$ - Umwandlung einfach genauer Zahlen (SINGLE PRECISION)
 MKD\$ - Umwandlung doppelt genauer Zahlen (DOUBLE PRECISION)

Durch Lesen des Level II-Handbuches und des DISK-Handbuches erfahren Sie noch mehr über diese Zahlentypen.

Nun nehme ich an, daß Sie das BASIC-Programm ausprobiert haben und ein wenig mehr über die Zahlendarstellung wissen, sich eine Übersicht über Ihre Datei angelegt haben, so daß Sie nun alles "ZAP"ppen können, was Sie möchten.

VIEL GLÜCK !

sich die Bytes im "FPDE" danach nochmals ansehen, stellen Sie fest, daß sie nun automatisch auf den richtigen Wert gesetzt wurden. In Abb. 11.2 finden Sie die notwendigen "ZAPs" zusammengestellt, die Sie zur Berichtigung des Fehlers ausführen müssen.

11.2 Verlorengegangene ELECTRIC PENCIL Datei

Dafür gibt es zwei unglückliche Umstände:

UNGLÜCKLICHER UMSTAND NR. 1:

Sie haben mit dieser Datei vor einigen Tagen gearbeitet. Alles schien in Ordnung und nachdem Sie Ihren Text eingegeben hatten, haben Sie die Datei auf die Diskette geschrieben. Sie nahmen die Disketten aus den Laufwerken und haben ausgeschaltet.

Einige Tage oder Stunden später setzen Sie sich wieder an den Computer und wollen mit dieser Datei weiterarbeiten. Sie starten ELECTRIC PENCIL und laden die Datei. Was ??? Nicht mehr da?! Auf dem Bildschirm erscheinen nur drei Zeilenschaltungen und sonst nichts. Nachdem das Blut langsam wieder in Ihren Kopf steigt, und Sie schließlich doch Ihren Augen

trauen, erkennen Sie den Grund: Sie haben die Datei sicher auf einer anderen Diskette gespeichert.

Siebenundvierzig Disketten später geben Sie auf und sagen sich "...verdammte, ich bin sicher, ich habe das auf Diskette geschrieben. Wie kann das nur passiert sein? Die Maschine muß es aufgeessen haben." So beschließen Sie, daß es Geheimnisse über der menschlichen Vorstellungskraft gibt und gehen fernsehen.

Die Wahrheit aber ist, daß niemand irgendetwas "aufgefressen" hat. Alles arbeitete fehlerfrei. Sie haben nur nach der langen Sitzung vor dem Bildschirm Ihren Text weggespeichert mit dem Cursor am ANFANG des Textes anstelle in an das ENDE zu setzen.

UNGLÜCKLICHER UMSTAND NR. 2:

Sie haben versehentlich die Datei mit einem falschen Dateinamen ge"KILL"t. Ich kann zwar nicht sagen, warum Sie das gemacht haben, aber manchmal passiert es eben.

Die Behebung des UNGLÜCKLICHEN UMSTANDES NR. 2 ist eine Sache des Kapitels 9.2, sehen Sie dort nach.

Den UNGLÜCKLICHEN UMSTAND NR. 1 können Sie leicht wie folgt beheben:

Abb. 11.2

Stellen Sie sicher, daß dieser Wert größer ist, als die tatsächliche Länge der Datei. "FF" wird in den meisten Fällen der richtige Wert sein.

```
111340 1000 00FF 0045 5252 3232 2020 2050 434C .....ERR22...PCL
111350 9642 9642 FF00 2401 FFFF FFFF FFFF FFFF .B.B..$......
```

Überschreiben Sie diesen Platz auch mit "FF".

1. Finden Sie das "EOF"-Byte in der Datei.
2. Ändern Sie es in ein beliebiges ASCII-Zeichen um ("20" oder "0D" arbeiten ausgezeichnet).
3. "ZAP"pen Sie in das "EOF"-Byte der Directory ein "FF".
4. "ZAP"pen Sie in die Bytes 14 und 15 des "FPDE" einen Wert, der größer ist, als die tatsächliche Sektorzahl ("FF" funktioniert hier in den meisten Fällen auch).

Da der alte "EOF"-Merker auch noch in der Datei enthalten ist, brauchen Sie sich über seine Lage keine Gedanken zu machen. Rufen Sie einfach "PENCIL" auf und laden Sie die Datei. Wenn ein "DOS ERROR 22" auftritt, dann haben Sie einen zu kleinen Wert für die Sektorzahl in den "FPDE" geschrieben.

11.3 Rettung einer verlorengegangenen ELECTRIC PENCIL Datei im RAM

Das hat zwar nichts mit Disketten zu tun, aber bevor Sie etwas auf Diskette retten können, müssen Sie es erst einmal im RAM unter Kontrolle haben. Folgendes kann Ihnen passieren:

Sie geben einen Text ein und plötzlich macht die Maschine einen neuen Systemstart oder Sie haben eine sehr lange Zeile eingegeben, drücken "ENTER" und plötzlich verschwindet der Text vom Bildschirm und lauter merkwürdige Symbole erscheinen. So etwa könnte der Bildschirm bei ANDROID NIM aussehen, nachdem einer der Androiden eine Handgranate verschluckt hat.

Hier die Beschreibung der "Rettungsaktion":

1. Hören Sie auf zu jammern, weder Fort Worth noch Michael Shroyer können Sie hören.

2. Drücken Sie "CONTROL O" und springen nach DOS (wenn Sie nicht schon da sind).

3. Geben Sie "DEBUG - ENTER" ein.

4. Drücken Sie die "BREAK"-Taste, um DEBUG zu aktivieren.

5. Geben Sie "G5C61 - ENTER" ein.

6. DA IST IHR TEXT WIEDER !

7. Retten Sie den Text sofort auf Diskette, drücken Sie nicht die "BREAK"-Taste, um aus dem Funktionsmenue herauszukommen, sonst landen Sie wieder in DEBUG, drücken Sie stattdessen den "Rechtspfeil".

8. Wenn der Bildschirm "komisch" aussah, bevor Sie den Text ganz verloren hatten, setzen Sie "COMMAND V" ein, um die Fehler zu berichtigen.

10. Ersetzen Sie die Zeile an der Sie gearbeitet haben, bevor es "komisch" wurde mit einem kürzeren Text. Wenn Sie nicht mehr wissen, welche Zeile das war, verlassen Sie PENCIL und reparieren Sie die Datei mit "SUPER-ZAP", indem Sie ein Leerzeichen oder einen Wagenrücklauf (20 HEX oder OD HEX) in die zu lange Zeile einfügen.

11.4 Eine Datei ist zu lang und läßt sich nicht laden

Das ist kein Fehler, sondern Tatsache. Wie kann es aber sein, daß Sie einen Text eingegeben haben, der im RAM Platz hatte, aber der nun, nachdem Sie ihn wieder von Diskette lesen wollen, plötzlich nicht mehr hineinpasst und die Meldung "FILE AREA FULL" erscheint?

PENCIL erlaubt tatsächlich, daß sie einen Text eingeben, der länger ist, als ein Text, den Sie von Diskette laden können. (Ist das nicht nett, Ollie?) Ja, ja, wieder eine feine Sache, um das Leben angenehmer zu machen. Aber noch ist nicht alles

verloren - es ist garnichts verloren!
Wir müssen lediglich die zu lange
Datei in zwei kürzere Dateien zerle-
gen und alles ist wieder gut.

Erzeugen Sie einen "FPDE", indem Sie
eine "blinde" Datei aus PENCIL
speichern. Verwenden Sie ruhig einen
endgültigen Namen für diese Datei.

Nun aktivieren Sie SUPERZAP und gehen
zu einem der letzten Sektoren der zu
langen Datei und kopieren Sie diese
letzten Sektoren in den Bereich, auf
den der EXTENT ihrer "blinden" Datei
zeigt. "ZAP"pen Sie in den "FPDE"
dieser Datei "FF" in Byte 3 und 14,
wie beim "DOS ERROR 22". Nun kann
dieser Teil geladen werden. Laden Sie
ihn mit PENCIL und speichern Sie ihn
sofort wieder, um den "FPDE" zu
korrigieren.

Als nächstes müssen wir jetzt noch
die Original-Datei berichtigen, damit
PENCIL nicht versucht, die gesamte
Datei zu laden. Gehen Sie zum ersten
Sektor, den wir in die andere Datei
kopiert haben. (Achso, Sie haben
vergessen, welcher Sektor das war,
Sie sollen doch Notizen machen, ver-
flixt nochmal!) "ZAP"pen Sie an be-
liebiger Stelle in diesen Sektor eine
"00". Nun können Sie auch diese Datei
laden, die Zeilen gegen Ende des
Textes löschen, die schon in der an-
deren Datei stehen und dann auch
diesen Textblock wieder auf Diskette
speichern und alles ist gut.

11.5 Einige gute Eigenschaften von ELECTRIC PENCIL

Hier sind einige Dinge, die Ihr Leben
leichter machen können:

Um ELECTRIC PENCIL auch unter NEWDOS
2.1 lauffähig zu machen, müssen Sie
nur drei Bytes im relativen Sektor
"0" von "PENCIL" auf "00 00 00" än-
dern. Suchen Sie den relativen Sektor
"0" und finden Sie in der Nähe des
relativen Bytes "AE" den Code

"F332 9B46 C36F"

Ändern Sie die Daten um, so daß sie
lauten:

"F300 0000 C36F".

Wenn sie den Cursor von PENCIL bei
der Wiederholfunktion etwas be-
schleunigen wollen, so brauchen Sie
nur ein Byte abzuändern. Suchen Sie
im relativen Sektor 10 (HEX) in der
Nähe des relativen Bytes "7B" den
Code

"0600 10FE 1116".

Ändern Sie ihn ab, so daß er lautet:

"0664 10FE 1116"

Statt "64" können Sie einen anderen
Wert wählen, je kleiner um so
schneller läuft der Cursor ("00" be-
deutet hier 256 dez.). Ich habe 50
eingegeben, aber damit wird er schon
zu einem kleinen Wiesel.

Hier noch einige nette Dinge, die mit
ELECTRIC PENCIL möglich sind:

Schreiben Sie BASIC-Programme mit
PENCIL. Es ist sicher fein, wenn man
BASIC-Programme mit den vieler guten
Editierfunktionen von PENCIL schrei-
ben könnte. Das ist möglich, denn ich
wende es dauernd an. Im Anhang finden
Sie das Programm "SEARCH", das ich
auch so erstellt habe.

Es gibt kein Geheimnis dabei, keinen
Trick, nur anwenden müssen Sie es.
Schreiben Sie Ihre Programme ganz
normal mit ELECTRIC PENCIL. Wenn Sie
fertig sind, speichern Sie den Text
auf Diskette, verlassen PENCIL und
rufen BASIC auf. Laden Sie dann das
Programm und starten Sie es. Es sind
nur zwei Dinge zu beachten: (1) der
Dateiname hat den Anhang "/PCL" und
(2) schließen Sie das Ende einer
Programmzeile NUR mir einem Wagen-
rücklauf (ENTER) ab. Probieren Sie
aus, wie einfach das ist!

Laden eines Programms, das mit BASIC
erstellt wurde in ELECTRIC PENCIL, um

es dort zu editieren. Auch das ist möglich. Stellen Sie sich vor, Sie möchten alle PRINT-Anweisungen durch LPRINT ersetzen. Das ist mit PENCIL kein Problem. Hier ist alles, was Sie dazu tun müssen: (1) Stellen Sie sicher, daß keine Programmzeile ohne Leerzeichen länger als 30 Zeichen ist. Wenn Sie die Befehle so dicht aneinanderpacken, müssen Sie ab und zu ein Leerzeichen einstreuen. (2) Speichern Sie das Programm mit BASIC in ASCII-Format (SAVE "filename",A). Verwenden Sie im Dateinamen den Anhang "/PCL". "ZAP"pen Sie anschließend in das letzte Byte des Programms den Wert "00" und Sie sind bereit, die Datei mit PENCIL zu laden.

Schreiben von ASSEMBLER- oder FORTRAN-Programmen mit PENCIL. Schreiben Sie das Programm einfach mit PENCIL, mit oder ohne Zeilennummern. Speichern Sie den Text wie gewohnt und verlassen Sie PENCIL. Rufen Sie den Editor (EDIT) von MACRO-80 auf und laden Sie den Text. EDIT fügt eigene Zeilennummern an den Text an und bietet Ihnen die Möglichkeit, den Text wieder mit oder ohne Zeilennummern zu speichern. Assemblieren oder compilieren Sie den Text wie gewohnt.



12.0 Fehlerbeseitigung in den "GAT"- und "HIT"-Sektoren

"GAT"-Fehler können teilweise entsetzliche Folgen haben. TRSDOS 2.1 gibt manchmal Granules frei, die jedoch in Wirklichkeit belegt sind. Wenn sich dieser Fehler einmal eingeschlichen hat, so besitzt er die böswillige Eigenschaft, sich immer weiter auszubreiten, bis endlich die gesamte Diskette durcheinandergebracht ist und das System "abstürzt".

Solange nur einige Fehler in der "GAT" bestehen, gibt es noch Hoffnung und es entsteht nur wenig Durcheinander auf der Diskette. Dieser Fehler ist auch die Ursache dafür, daß eine Datei plötzlich Daten einer anderen Datei enthält, weil einer ihrer Sektoren von der anderen Datei überschrieben wurde. DOS schreibt ohne weiteres auf Sektoren, die schon Daten enthalten, wenn der Sektor in der "GAT" als unbenutzt gekennzeichnet ist.

12.1 Die Reparatur der "GAT"

1. Mit DIRCHECK überprüfen Sie das Directory der fraglichen Diskette und notieren sich alle Fehler in der "GAT" und "HIT", die angezeigt werden.

2. Wenn Sie nicht über NEWDOS verfügen, dann müssen Sie jeden Eintrag in den "FPDE" und "FXDE" einzeln durchsehen, sich jeweils die Anzahl der Granules und die Adresse des EXTENT notieren und anschließend diese Werte mit den einzelnen Granules Spur für Spur vergleichen. (Leider gibt es so keinen schnelleren Weg.)

3. "ZAP"pen Sie jeden widersprüchlichen Code in der "GAT" mit dem korrekten Wert.

=====ACHTUNG=====

Löschen Sie unbedingt die Dateien, die Sektoren "guter" Dateien benutzen, sonst treten später weitere Fehler auf!

=====ACHTUNG=====

12.2 Die Reparatur von "HIT"-Fehlern

Prinzipiell gilt die gleiche Vorgehensweise wie oben beschrieben. Das Suchen der fehlerhaften "HIT"-Bytes ist etwas einfacher.

In der "HIT" müssen genau soviele Hash Codes vorhanden sein, wie aktive Dateien. Jede Datei, die durch "DIR" angezeigt wird UND sich laden läßt, bzw. mit "INPUT" oder "GET" gelesen werden kann, hat einen gültigen Hash Code. Wenn EINE dieser beiden Funktionen, also Anzeige im Directory oder Lesen versagt, dann ist es das Anzeichen für einen Diskettenfehler.

Wie man einen Hash Code entschlüsselt, ist in Kapitel 10.1 gezeigt. Mit diesem Wissen können Sie dann auch die Codes auf einer Diskette auf ihre Richtigkeit überprüfen.

13.0 Was Sie sonst noch können....

Es ist immer gut, wenn man neue Anregungen erhalten kann. Einige Denkanstöße möchte ich Ihnen hier geben. Ansonsten bleibt es Ihrer Phantasie überlassen, was Sie alles mit Ihrem TRS-80 anstellen wollen.

13.1 Erzeugen von ELECTRIC PENCIL Dateien mit BASIC

Das ist so einfach, daß Sie sich wundern, warum das nicht schon früher

irgendwo veröffentlicht wurde. Sie wissen, daß PENCIL Dateien fast ganz gewöhnliche ASCII Dateien sind. Mit einigen Experimenten und mit der Hilfe von SUPERZAP habe ich alles herausgefunden, was ich wissen mußte. Versuchen Sie folgendes Experiment:

Laden Sie BASIC und geben Sie das folgende Programm ein:

```
100 CLEAR : CLS
110 OPEN "0",1,"TEST/PCL"
120 AS="DAS IST EIN TEST FUER
    EINE PENCIL DATEI"
130 PRINT #1,AS
140 PRINT #1,CHR$(0)
150 CLOSE
```

Natürlich ist dieses Programm sehr einfach und es kann sicher wesentlich eleganter und ausgeklügelter geschrieben werden, aber es ist leicht zu übersehen und man "sieht", wie es funktioniert. Nun rufen Sie PENCIL auf und laden "TEST"

13.2 Laden eines BASIC-Programms oder einer ASCII-Datei in ELECTRIC PENCIL

Hier gibt es nur dann Probleme, wenn Sie Ihr BASIC Programm dicht "gepackt" haben, ohne einige Leerzeichen einzufügen

Eine der guten Möglichkeiten ist das allgemeine Suchen und Austauschen mit PENCIL. So tauschen Sie z.B. innerhalb weniger Sekunden alle "PRINT"-Anweisungen gegen "LPRINT"-Befehle aus. Sie können auch ein BASIC-Programm von einem "Dialekt" in einen anderen umschreiben. Tippen Sie mit PENCIL einfach ein Programm aus einer Zeitschrift ab, ohne sich schon Gedanken zu machen, ob der eine oder andere Befehl so vom TRS-80 verstanden wird. Nachdem Sie das Programm komplett eingegeben haben, ändern Sie die unzutreffenden Befehle um oder

ergänzen am Ende des Programmes Unterrouتين, die die entsprechende Funktion eines Befehls ausführen, den der TRS-80 nicht kennt. Danach suchen Sie diese Befehle und tauschen sie gegen eine "GOSUB"-Anweisung aus. In einigen Minuten können Sie so ein Programm anpassen, wofür Sie sonst unter Umständen Tage benötigen würden.

Beachten Sie aber, daß PENCIL MINDESTENS ALLE 30 Zeichen ein Leerzeichen braucht, um die Bildschirmanzeige ordnungsgemäß auszuführen.

Lassen Sie uns einen Versuch starten: Geben Sie das obige Programm ein und speichern Sie es mit

```
SAVE "FILETEST/PCL",A
```

Beachten Sie, daß PENCIL nur Dateien mit dem Anhang "/PCL" annimmt. Das "A" am Ende des Befehls bewirkt die Speicherung des Programms als ASCII-Datei.

Nun suchen Sie mit SUPERZAP das Ende der Datei und überschreiben das letzte "OD" mit "OO". Dann starten Sie PENCIL und laden "FILETEST".

Wenn Sie NEWDOS verwenden, brauchen Sie nicht unbedingt SUPERZAP, um das "EOF"-Byte auf "OO" zu setzen. Nach Speichern des Programms laden Sie das folgende kurze Programm und führen es aus:

```
100 OPEN "E",1,"TEST/PCL"
110 PRINT #1,CHR$(0)+" "
120 CLOSE
```

Damit wird die Programmdatei "FILETEST/PCL" zum Anfügen weiterer Daten geöffnet und die "EOF"-Marke "OO" an das Ende des Programms "geklebt". Mit den noch folgenden Leerzeichen wird erreicht, daß der berühmte "DOS ERROR 22" nicht auftreten kann.

Mit TRSDOS können Sie etwas Ähnliches machen, Sie müssen allerdings mit

einem BASIC-Programm Zeile für Zeile lesen, diese in eine andere Datei schreiben und wenn alle Zeilen übertragen sind, "OO" und die Leerzeichen anfügen. Aber vielleicht legen Sie sich doch NEWDOS zu....

Die oben beschriebenen Verfahren funktionieren bei Daten-Files ebenso.

13.3 Sperren der "LIST"-Funktion bei BASIC-Programmen

Es gibt keinen totalen Programmschutz. Das nun gezeigte Verfahren macht ein BASIC-Programm "unLISTbar", solange der Anwender dieses Programms niemals dieses Buch in die Hand bekommt oder den Trick selbst herausfindet.

Speichern Sie ein BASIC-Programm, das eine "blinde" Zeichenkette enthält, wie z.B.:

```
DU $="*****"
```

Mit SUPERZAP suchen Sie anschließend diese Zeichen auf der Diskette (der Code für "*" ist "2A" HEX). Überschriften Sie diese "2A"s mit "1212 1212 1212 u.s.w." Nun laden Sie das Programm und versuchen es zu "LIST"en.

Wie ist es mit LLIST? --- Ziemlich viel Papier, nicht wahr?

13.7 Ergänzen von neuen Funktionen in SUPERZAP

SUPERZAP ist ein sehr gut geschriebenes BASIC-Programm und daher leicht mit weiteren Funktionen zu erweitern.

Wenn Sie Disketten reparieren, müssen Sie sehr oft zwischen SUPERZAP und DIRCHECK hin- und herspringen. Mit NEWDOS können Sie so vorgehen:

In SUPERZAP drücken Sie "BREAK" und geben dann CMD "DIRCHECK" ein. Nachdem DIRCHECK seine Aufgabe erfüllt hat, meldet sich BASIC wieder

mit "READY". Geben Sie nun "CONT ENTER" und "X" ein und das SUPERZAP-Menue ist wieder da.

Mein SUPERZAP hat allerdings in "DD" eine "Aufwärtspfeil-Funktion" zum Aufruf von DIRCHECK. Laden Sie SUPERZAP und geben Sie die untenstehenden Programmzeilen ein. Dann speichern Sie das geänderte SUPERZAP mit SAVE"SUPERZAP".

Probieren Sie anschließend die "DD"-Funktion aus und drücken Sie den "Aufwärtspfeil". Phantastisch --- oder ?

```
4310 IF A$="↑" THEN 60000
      .
      .
      .
60000 CLS : PRINT@345,"DIRCHECK"
60010 CMD"DIRCHECK" : A$=R : GOTO 4400
```

13.8 Lesen der Directory mit BASIC

Unter NEWDOS versuchen Sie einmal eine Datei "DIR/SYS" zu öffnen und lesen Sie den ersten Record mit GET in einen Puffer, den Sie mit FIELD 1, 255 AS A\$ definieren.

Sie werden eine Fehlermeldung erhalten, aber versuchen Sie dennoch, die gelesene Zeichenkette anzuzeigen (geben Sie "PRINT A\$ " ein). Was? Es hat ja trotzdem funktioniert. Nun können Sie die Fehlermeldung einfach mit einer "ON ERROR GOTO"-Anweisung unterdrücken und das Programm fortsetzen. Mit diesem kleinen Trick läßt sich nun das Directory einer Diskette auch aus einem BASIC-Programm lesen.



Anhang A - 1. Glossar

ACCESS - (Zugriff) - Vorgang des Suchens, Lesens oder Schreibens von Daten auf einem Speichermedium (in diesem Fall auf der Diskette).

ACCESS TIME - (Zugriffszeit) - Die Zeit, die vergeht, zwischen einer Anweisung, die einen Zugriff auslöst und der Verfügbarkeit der geforderten Daten.

ACTIVE RECORD TABLE - (Tabelle aktiver Datensätze) - Eine Tabelle mit binären Werten, in der die Position eines einzelnen Wertes den Zustand eines Datensatzes in der gleichen relativen Position angibt, d.h. die n-te Binärzahl gibt den Zustand des n-ten Datensatzes an. Beispiel: wenn die achte Zahl in der Tabelle eine Null ist, dann ist der achte Datensatz inaktiv, und umgekehrt, wenn die achte Zahl in der Tabelle den Wert eins hat, dann ist der achte Datensatz aktiv oder belegt.

ADDRESS - (Adresse) - Eine Kennung (Zahl, Name oder Symbol) für einen Speicherplatz, in dem Daten gespeichert werden.

ALGORITHM - (Algorithmus) - Ein Rechenverfahren.

ALPHANUMERIC (CHARACTERS) - (Alphanumerische Zeichen) - Zusammenfassende Bezeichnung für Ziffern, Buchstaben, Satzzeichen und Sonderzeichen.

ALPHANUMERIC STRING - (Alphanumerische Zeichenkette) - Eine Folge von Zeichen, die Ziffern, Buchstaben, Satzzeichen und Sonderzeichen und auch Leerzeichen enthalten kann. (Anm.: ein Leerzeichen bedeutet für den Computer ein Zeichen, wie jedes andere auch.)

ASCII - Abkürzung für American Standard Code for Information Interchange (Amerikanischer Standard Code für den Datenaustausch). Hier handelt es sich

um ein genormtes Verfahren, wie alphanumerische Zeichen und besondere Steuerzeichen in digital codierter Form dargestellt werden.

ASSEMBLY LANGUAGE - (Assembler-sprache) - Eine Sprache auf der Maschinenebene, deren Befehle (Mnemoniks) von einem Assemblerprogramm in einen maschinenlesbaren Code umgesetzt werden.

BASE - (Basis) - Eine Zahl, die angibt, wieviele unterschiedliche Zahlzeichen in einem Zahlensystem vorhanden sind.

BASE 2 - (Basis 2) - Das binäre Zahlensystem.

BASE 10 - (Basis 10) - Das Dezimalsystem.

BASE 16 - (Basis 16) - Das Hexadezimal- oder Sedezimalsystem.

BINARY - (Binär) - Siehe BASE 2

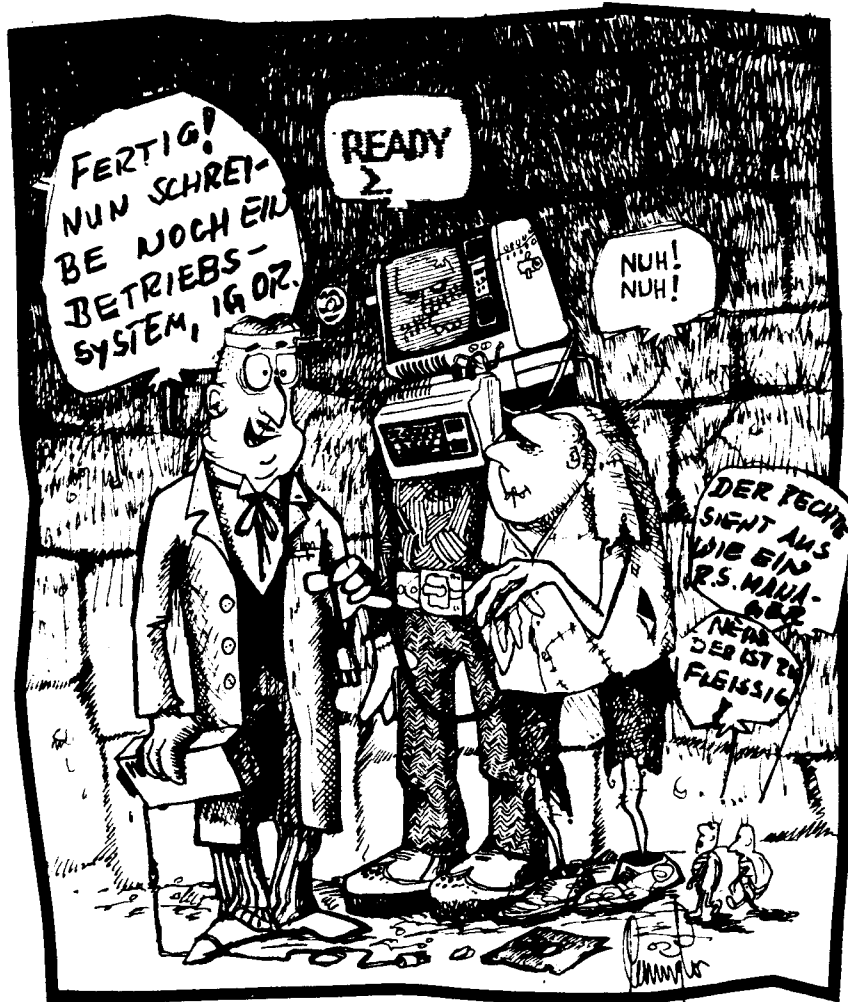
BIT - (Bit) - Eine einzelne Binärstelle, deren Wert "0" oder "1" betragen kann.

BOOLEAN - (Boolesche Algebra) - Eine Form der Algebra, angewendet auf Binärzahlen, die der gewöhnlichen Algebra ähnelt. Sie ist besonders nützlich bei der logischen Analyse von Binärzahlen, wie sie in Computern verwendet werden.

BOOT - BOOTSTRAP - (Urladen - Urlader) - Ein Maschinenprogramm, mit dem bei jeder Inbetriebnahme des Systems die erforderlichen Programmteile des Betriebsprogramms geladen werden. (bootstrap - Schuhanzieher).

BUFFER - (Puffer) - Ein kleiner Speicherbereich, der zur Zwischenspeicherung von in Bearbeitung befindlichen Daten dient.

BUFFER TRACK - (Pufferspur) - Eine Spur auf einer Diskette, auf der Daten zwischengespeichert werden.



BUG - ("Käfer") - Ein Programmfehler, der zu einer Fehlfunktion des Programms führt, kann sich jedoch auch auf einen technischen Fehler im Gerät beziehen.

BYTE - (Byte) - Acht "Bits". Ein Byte kann numerische Werte zwischen 0 und 255 darstellen.

COMMAND FILE - (Befehlsdatei) - Eine Datei mit Befehlen oder Anweisungen, die als Folge automatisch ausgeführt werden.

CRC - CYCLIC REDUNDANCY CHECK - Ein besonderes Verfahren, um aufgrund von Redundanzen eine Fehlerprüfung durchzuführen, hauptsächlich eingesetzt bei der Datenspeicherung auf magnetischen Trägern.

DATA ELEMENT - (Datenelement) gleichbedeutend mit DATA ITEM (Datenelement) oder FIELD (Datenfeld) - ein RECORD (Datensatz) ist aus einem oder mehreren Datenelementen aufgebaut - ein oder mehrere Datensätze bilden eine FILE (Datei).

DATA TYPE - (Datentyp) - Die Form, in der Daten gespeichert und verarbeitet werden, z.B. INTEGER (Ganzzahl), SINGLE PRECISION (einfache Genauigkeit), DOUBLE PRECISION (doppelte Genauigkeit) oder STRING (alphanumerische Zeichenkette).

DIRECT ACCESS - (Direkter Zugriff) - Zugriff auf ein Datenelement durch direkte Angabe seiner Position auf der Diskette, ohne daß ein Bezug zu einem vorher gelesenen Datenelement bestehen muß.

DIRECT STATEMENT IN FILE - (Programmdatei enthält eine direkte Anweisung) - Eine Anweisung, die in einer Programmdatei existiert, ohne daß ihr eine Zeilennummer vorausgeht.

DIRECTORY - (urprüngl.: Adressbuch) - Eine Tabelle, aus der die Position von Daten gelesen werden kann.

DISPLACEMENT - (Versatz) - Wird bei der relativen Adressierung verwendet, um die Position von Daten in Bezug auf einen definierten Punkt anzugeben.

DISTRIBUTED FREE SPACE - (Verteilte Zwischenräume) - In bestimmten Intervallen eingefügter freier Speicherplatz in einer Datenstruktur, um das Einfügen weiterer Daten zu ermöglichen.

DOUBLE PRECISION - (Doppelte Genauigkeit) - Ein positiver oder negativer numerischer Wert mit 16 signifikanten Stellen, z.B.: 99999999999999.99 (diese Definition gilt für den TRS-80, sie kann bei anderen Rechnern abweichen).

DUMP - Transferieren von Daten aus einem Speicherbereich (Quelle) in einen anderen (Ziel). Quelle und Ziel können im Arbeitsspeicher oder auf einem Massenspeicher liegen, das Ziel kann außerdem z.B. auch ein Sichtgerät oder Drucker sein.

DYNAMIC STORAGE ALLOCATION - (Dynamische Speicherplatzzuweisung) - Die Zuweisung von Speicherplatz erfolgt durch eine Prozedur entsprechend dem jeweils erforderlichen Speicherplatzbedarf, anstelle in festen Längen.

EOF - END OF FILE - (Datei-Ende) - Man kann z.B. den letzten Sektor angeben, der von einer Datei belegt wird und ggf. zusätzlich auch die Lage des letzten Bytes dieser Datei innerhalb des letzten Sektors.

EXTENT - Ein zusammenhängender Block von Daten.

FILE - (Datei) - Eine Sammlung von zueinandergehörigen Datensätzen, die als eine Einheit betrachtet werden.

FILE PARAMETERS - (Dateiparameter) - Die Daten, die die Struktur einer Datei beschreiben oder definieren.

FILESPEC - (Dateiname) - Beim TRS-80 kann der Dateiname folgende Elemente enthalten: den eigentlichen Namen (z.B.: "SUPERZAP"), einen Anhang (z.B.: "/BAS"), ein Schutzwort (z.B.: ".PASSWORD") und eine Laufwerksangabe (z.B.: ":2"). Nur der eigentliche Name ist zwingend erforderlich.

FIELD - (Datenfeld) - Siehe "DATA ELEMENT"

FILE AREA - (Datei-Speicherbereich) - Der physikalische Platz auf der Diskette, auf dem die Datei gespeichert ist.

FPDE - FILE PRIMARY DIRECTORY ENTRY - (Datei-Haupteintrag) - Datenblock im Directory mit Informationen über eine Datei, deren Namen, Länge und Position auf der Diskette.

FXDE - FILE EXTENDED DIRECTORY ENTRY - (Datei-Ergänzungseintrag) - Entspricht in seinem Aufbau dem "FPDE". Wird dann benötigt, wenn die Kapazität des "FPDE" nicht ausreicht, um die gesamte Information über die Lage der Datei auf der Diskette zu speichern.

GAT - GRANULE ALLOCATION TABLE - (Tabelle der zugewiesenen Granules oder Halbspuren) - Eine Tabelle, aus der die belegten und freien Granules einer Diskette ersichtlich sind. Die "GAT" hat die Form einer ACTIVE RECORD TABLE.

GRANULE - (Halbspur) - Eine Einheit von fünf Sektoren, also einer halben Spur. Der kleinste Speicherbereich, der durch das Betriebssystem einer Datei zugewiesen werden kann.

HASH CODE - (Hash Code) - Eine Zahl, die nach einem bestimmten Algorithmus aus einem Schlüsselwort berechnet und zur direkten Adressierung von Speicherplätzen oder -bereichen verwendet wird.

HEADER RECORD - (Titel, Präambel) - Ein Datensatz, der für eine Gruppe folgender Datensätze allgemeine In-

formationen enthält.

HEXADECIMAL - (Hexadezimal, Sedezimal) - Siehe BASE 16.

HIT - HASH INDEX TABLE - (Index Tabelle mit Hash Codes) - Ein Adressierungsverfahren, bei dem der Bezug zu einer Datei über eine Codezahl in einer Tabelle hergestellt wird. Die Position des Codes in der Tabelle entspricht der Position des entsprechenden Eintrages im Directory.

INDEX - (Index) - Ein Wert, der die Lage eines Datensatzes oder allgemein die Lage von Daten innerhalb eines Speichers symbolisiert.

INDIRECT ADDRESSING - (Indirekte Adressierung) - Eine Methode zur Angabe eines Speicherplatzes oder -bereiches, wobei der angegebene Suchschlüssel selbst NICHT die Adresse darstellt, sondern einen Speicherplatz bestimmt, in dem die eigentliche Adresse zu finden ist.

INSTRING - INSTRING SEARCH - (Ab-suchen einer Zeichenkette) - Eine Zeichenkette wird darauf überprüft, ob in ihr ein bestimmtes Zeichen oder eine bestimmte andere Zeichenkette enthalten ist und an bzw. ab welcher Stelle innerhalb der Zeichenkette sich dieses Zeichen bzw. die Zeichenkette befindet.

INTEGER - (Ganzzahl) - Eine natürliche ganze Zahl. Beim TRS-80 kann sie Werte zwischen +32767 und -32768 annehmen.

IPL - INITIAL PROGRAM LOADER - (Ur-lader) - Ein Programm, das normalerweise beim Einschalten oder Rücksetzen des Systems aktiviert wird, um das Betriebsprogramm von Diskette in den Speicher zu laden.

KEY - (Schlüssel, Schlüsselwort) - Ein Datenelement, um die Lage oder Bezeichnung eines Datensatzes oder einer anderen Gruppierung von Daten anzugeben.

LABEL - (Marke) - Ein Vorrat von Symbolen, um ein Datenelement, einen Datensatz oder eine Meldung zu bezeichnen oder zu beschreiben. Kann manchmal gleichzeitig die Adresse darstellen.

LEAST SIGNIFICANT BIT/BYTE - (Stelle/Bit oder Byte mit der niedrigsten Wertigkeit)

LIST - (Liste) - Eine geordnete Menge von Datenelementen, auch Kette genannt.

LOAD MODULE - Ein Programm, das dazu bestimmt ist, in einen Speicherbereich geladen und dort ausgeführt zu werden.

LOCK OUT (TRACKS) - (Sperrern von Spuren) - Unbrauchbare Spuren auf einer Diskette, die beschädigt sind, werden beim Formatieren der Diskette für den Zugriff durch das Betriebssystem gesperrt.

LOGICAL FILE - (Logische Dateistruktur) - Die Aufteilung einer Datei, wie sie von einem Anwenderprogramm vorgenommen wird. Die tatsächliche physikalische Aufteilung oder Anordnung auf der Diskette kann völlig anders sein.

LOGICAL OPERATOR - (Logischer Operator) - Ein mathematisches Symbol, das die Verknüpfung von Operanden nach den Regeln der Booleschen Algebra vorschreibt. (UND, ODER, NICHT u.s.w.)

LOGICAL RECORD - (Logischer Datensatz) - Ein Datensatz, wie er von einem Anwenderprogramm gebildet wird. Er kann völlig anders konstruiert sein als der physikalische Datensatz mit dem die Daten auf der Diskette gespeichert werden.

LSB - Siehe LEAST SIGNIFICANT BIT

MACHINE LANGUAGE - (Maschinensprache) - Programm in maschinenlesbarer Form.

MONITOR - (Monitorprogramm) - Ein Programm, das die Bearbeitung anderer Programme steuert oder ein Programm zum Test und zur Fehlersuche von bzw. in anderen Programmen.

MOST SIGNIFICANT BIT/BYTE - (Stelle/Bit oder Byte mit der höchsten Wertigkeit)

MSB - Siehe MOST SIGNIFICANT BIT

NYBBLE - Die rechte oder linke Vierergruppe in einem Byte.

OPERATING SYSTEM - (Betriebssystem) - Programm zur Durchführung aller Verwaltungsaufgaben, die die Speicherung von Daten auf Diskette betreffen. Weiterhin werden vom Anwender über das Betriebssystem die verschiedensten Aktivitäten eingegeben und veranlasst.

PARITY CHECK - (Paritätsprüfung) - Verfahren zur Datensicherung gegen technische und Übertragungsfehler. Zu einer bestimmten Menge von Daten werden nach festgelegten Algorithmen Kontrollzahlen oder Zeichen gebildet und mit übertragen bzw. gespeichert, die es einem Empfänger oder bei späterem Lesen der Daten ermöglichen, eine Fehlerprüfung und ggf. -korrektur vorzunehmen.

PHYSICAL - (Adj.: physikalisch) - Im Gegensatz zu "logisch" bezieht sich dieser Begriff darauf, wie Daten tatsächlich gespeichert werden oder wie ein System in der Realität aufgebaut ist.

PHYSICAL RECORD - (Physikalischer Datensatz) - Eine Menge von Daten, wie sie tatsächlich auf einer Diskette gespeichert werden.

POINTER - (Zeiger) - Eine Adresse oder ein Datenelement in einem Datensatz, in dem eine Information über einen vorangehenden oder folgenden Datensatz enthalten ist.

RANDOM ACCESS - (Wahlfreier Zugriff)

- Der direkte Zugriff auf Daten in einem Speicher, unabhängig von Ihrer Lage ist möglich, ohne daß vorher auf andere Daten zugegriffen werden muß, auch "direkter Zugriff" genannt.

RECORD - (Datensatz) - Eine Gruppe von Datenelementen, die als eine Einheit behandelt und verarbeitet werden.

RELATIVE ADDRESS - (Relative Adresse) - Eine Adresse, die sich auf einen Ursprungs- oder Ausgangspunkt bezieht. Die r.A. gibt den Abstand oder Versatz zwischen diesem Punkt und dem adressierten Element an.

SECTOR - (Sektor) - Die kleinste adressierbare Speichereinheit auf einer Diskette, beim TRS-80 aus 256 Bytes bestehend.

SEQUENTIAL ACCESS - (Sequentieller Zugriff) - Zugriffsmethode, bei der ein Datensatz nach dem anderen gelesen oder geschrieben werden muß.

SINGLE PRECISION - (Einfache Genauigkeit) - Eine positive oder negative Zahl mit 6 signifikanten Stellen, z.B.: 99999.9 (diese Festlegung gilt für den TRS-80 und kann bei anderen Systemen unterschiedlich sein).

SORT - (Sortierlauf) - Verfahren, um die Elemente einer Datei in eine andere Reihenfolge nach einem bestimmten Sortierkriterium zu bringen, z.B.: der Größe nach in aufsteigender Reihenfolge.

SOURCE CODE - (Quellencode) - Der Text, aus dem ein maschinenlesbarer Code abgeleitet werden kann.

SYSTEM FILE - (System Datei) - Eine Datei oder ein Programm, das durch das Betriebssystem des Rechners verwendet wird.

TABLE - (Tabelle) - Eine Anordnung von Daten, geeignet für schnellen Zugriff oder schnelles Absuchen, wobei jedes Element eindeutig durch

seine Lage oder eine Marke gekennzeichnet ist.

TOKEN - (Kurzform) - Ein Ein-Byte-Code, der einem längeren Wort aus zwei oder mehr Zeichen entspricht.

TRACK - (Spur) - Die kreisförmig auf einer Diskette angeordneten Aufzeichnungsflächen, die vom Schreib-Lesekopf des Laufwerkes überstrichen werden.

VERIFY - (Prüflesen, Verifizieren) - Eine Prüfung von geschriebenen Daten auf Fehlerfreiheit durch sofortiges Lesen und Vergleichen mit den Ausgangsdaten.

WORKING STORAGE - (Arbeitsspeicher) - Ein Speicherbereich, üblicherweise im Hauptspeicher eines Computers, der zur Speicherung von Zwischenergebnissen reserviert ist.



Anhang A 2

Level II BASIC-Tokens

Programmbefehle in Level II BASIC und DISK-BASIC werden nicht so gespeichert, wie sie eingegeben und auf dem Bildschirm angezeigt werden, sondern als Ein-Byte-Codezeichen. Die folgende Liste zeigt die Tokens in HEX und dezimal sowie die zugehörigen BASIC-Befehle.

HEX - dez. Befehl	HEX - dez. Befehl	HEX - dez. Befehl
80-128	END	AA-170 KILL
81-129	FOR	AB-171 LSET
82-130	RESET	AC-172 RSET
83-131	SET	AD-173 SAVE
84-132	CLS	AE-174 SYSTEM
85-133	CMD	AF-175 LPRINT
86-134	RANDOM	B0-176 DEF
87-135	NEXT	B1-177 POKE
88-136	DATA	B2-178 PRINT
89-137	INPUT	B3-179 CONT
8A-138	DIM	B4-180 LIST
8B-139	READ	B5-181 LLIST
8C-140	LET	B6-182 DELETE
8D-141	GOTO	B7-183 AUTO
8E-142	RUN	B8-184 CLEAR
8F-143	IF	B9-185 CLOAD
90-144	RESTORE	BA-186 CSAVE
91-145	GOSUB	BB-187 NEW
92-146	RETURN	BC-188 TAB
93-147	REM	BD-189 TO
94-148	STOP	BE-190 FN
95-149	ELSE	BF-191 USING
96-150	TRON	C0-192 VARPTR
97-151	TROFF	C1-193 USR
98-152	DEFSTR	C2-194 ERL
99-153	DEFINT	C3-195 ERR
9A-154	DEFSNG	C4-196 STRING\$
9B-155	DEFDBL	C5-197 INSTR
9C-156	LINE	C6-198 POINT
9D-157	EDIT	C7-199 TIME\$
9E-158	ERROR	C8-200 MEM
9F-159	RESUME	C9-201 INKEY\$
A0-160	OUT	CA-202 THEN
A1-161	ON	CB-203 NOT
A2-162	OPEN	CC-204 STEP
A3-163	FIELD	CD-205 +
A4-164	GET	CE-206 -
A5-165	PUT	CF-207 *
A6-166	CLOSE	F0-208 /
A7-167	LOAD	D1-209 (UP ARROW)
A8-168	MERGE	D2-210 AND
A9-169	NAME **	D3-211 OR
		D4-212 >
		D5-213 =
		D6-214 <
		D7-215 SGN
		D8-216 INT
		D9-217 ABS
		DA-218 FRE
		DB-219 INP
		DC-220 POS
		DD-221 SQR
		DE-222 RND
		DF-223 LOG
		E0-224 EXP
		E1-225 COS
		E2-226 SIN
		E3-227 TAN
		E4-228 ATN
		E5-229 PEEK
		E6-230 CVI
		E7-231 CVS
		E8-232 CVD
		E9-233 EOF
		EA-234 LOC
		EB-235 LOF
		EC-236 MKI\$
		ED-237 MKS\$
		EE-238 MKD\$
		EF-239 CINT
		F0-240 CSNG
		F1-241 CDBL
		F2-242 FIX
		F3-243 LEN
		F4-244 STR\$
		F5-245 VAL
		F6-246 ASC
		F7-247 CHR\$
		F8-248 LEFT\$
		F9-249 RIGHT\$
		FA-250 MID\$
		FB-251 **
		FC-252 **
		FD-253 **
		FE-254 **
		FF-255 ISA **

** = vom System nicht benutzt

Anhang B

Dieser Text ist zuerst im "OCTUG Newsletter" erschienen. "OCTUG" ist die "Orange County TRS-80 Users Group". Es ist einer der besseren Anwenderclubs und ihre Zeitschrift ist hervorragend. Wenn Sie Mitglied werden wollen, schreiben Sie an:

OCTUG
2531 E. Commonwealth Ave.
Fullerton, CA 92631
USA

Wartung der TRS-80 Diskettenlaufwerke

Solange Sie nicht die Möglichkeit haben, in einem relativ staubfreien, trockenen Raum zu arbeiten, ist es besser, Sie lassen das Laufwerk, wie es ist. Die Werkzeuge, die Sie benötigen, sind ein Kreuzschlitz- und ein normaler Schraubendreher, eine Spraydose mit Reinigungsmittel (Freon o.ä. - das Mittel muß ausdrücklich auch für Kunststoffe geeignet sein!), etwas Isopropylalkohol, einige fusselfreie Tücher und ein leichtes Silicon-Schmiermittel. Wenn es Probleme mit dem Überhitzen der Stromversorgung gibt, benötigen Sie außerdem noch einen Lötkolben (ca. 40 Watt), Lötzinn und Wärmeleitpaste. Entfernen Sie die Gehäuseabdeckung nach Herausdrehen der vier Kreuzschlitzschrauben. Nachdem Sie auch die drei Schrauben entfernt haben, die das Laufwerk halten, können Sie es nach vorne herausziehen. Dabei lösen Sie auch den Anschlußstecker für die Stromversorgung. Das Laufwerk ist nun vom übrigen Teil des Gehäuses getrennt.

Bei der Handhabung sollten Sie unbedingt die Verunreinigung des Antriebsriemens und der Riemenscheiben durch Berühren mit den Händen und durch Öl und Fett vermeiden. Durch

Lösen der beiden kleineren Halteschrauben kann die Hauptplatine entfernt werden. Lösen Sie aber vorher den Platinenstecker und den Anschlußstecker für den Schreib-/Lesekopf. Die Laufwerk-Mechanik ist nun zugänglich.

Untersuchen Sie die Mechanik auf Spuren von Schmutz und Staub und sonstige Fremdkörper. Der Schlitten mit dem Schreib/Lesekopf kann hin- und hergeschoben und so auf Leichtigbarkeit geprüft werden. Sollten Sie Fremdkörper entdecken, so entfernen Sie sie durch Absprühen der Mechanik mit Freon-Spray. Der Schreib/Lesekopf, die Filz-Andruckplatte und die Oberflächen der LED und des Fototransistors können Sie mit Isopropylalkohol und einem Tuch abwischen. Drücken Sie nicht zu kräftig auf diese Teile, damit Sie nicht aus ihrer Justierung geraten.

Die Verwendung von Metallteilen bei der Reinigung muß unterbleiben, um keine empfindlichen Oberflächen zu zerkratzen. Geben Sie einen dünnen Film Siliconfett auf die beiden runden Führungen des Kopfschlittens und einige Tropfen Fett in die Spirale der Antriebsscheibe. Der Antriebsriemen und die beiden Riemenscheiben werden mit Freon und einem Tuch gereinigt. Ein Tupfer Fett in die beiden Schlitze der Frontplattenverriegelung beendet die Wartungsarbeiten am Laufwerk.

Wenn Betriebsstörungen auftreten, nachdem das Laufwerk einige Zeit eingeschaltet ist, kann Überhitzung die Ursache sein. Prüfen Sie in diesem Fall die beiden Festspannungsregler-ICs mit den drei Anschlüssen, die auf der Gehäuserückseite montiert sind. Die Metalloberfläche der ICs muß guten Kontakt mit dem Gehäuse haben und Sie sollten eine Glimmerscheibe mit beidseits aufgetragener Wärmeleitpaste erkennen können.

Bei manchen Laufwerken wurden zur Befestigung der Regler Kunststoffschrauben verwendet, die leicht abreißen. Dadurch hat das IC keinen ausreichenden Wärmekontakt zum Gehäuse mehr.

In diesem Fall löten Sie die Anschlüsse der ICs ab, bauen sie aus und bringen auf beiden Seiten der Glimmerscheiben neue Wärmeleitpaste auf. Danach montieren Sie die ICs wieder unter Verwendung neuer Schrauben. Richten Sie die Anschlüsse so aus, daß keine Kurzschlüsse entstehen können und löten Sie sie wieder an.

Der Zusammenbau der Einheit erfolgt in umgekehrter Reihenfolge wie die Demontage. Alle Steckverbinder haben Führungsschlitze, so daß sie nicht falsch aufgesteckt werden können.

Anhang B.1

Murphy und seine verflixten Gesetze:
...alles was daneben gehen kann, wird daneben gehen.

Hier sind die neuesten Erkenntnisse und Gesetze der konstanten Gemeinheit für den Bereich der Computerprogrammierung:

Jedes Programmiervorhaben, das gut beginnt, endet schlecht.

Jeder Programmiervorhaben, das schlecht beginnt, endet fürchterlich.

Wenn eine Programmieraufgabe leicht aussieht, so ist sie schwer.

Wenn eine Programmieraufgabe schwer aussieht, so ist sie unmöglich.

Jedes gegebene, lauffähige Programm wird nicht mehr benötigt.

Jedes gegebene Programm kostet mehr und läuft länger als erwartet.

Wenn ein Programm nützlich ist, dann muß es geändert werden.

Wenn ein Programm unnützlich ist, dann muß es dokumentiert werden.

Jedes gegebene Programm dehnt sich aus, bis es allen verfügbaren Speicherplatz belegt.

Der Wert eines Programmes ist umgekehrt proportional zum Gewicht seines Ausdruckes.

Die Komplexität eines Programms steigert sich solange, bis es die geistigen Fähigkeiten des Programmierers übersteigt, der es warten muß.

Frühestens sechs Monate nach Veröffentlichung eines Programms wird der schlimmste Fehler erkannt.

Maschinenunabhängiger Code ist nicht unabhängig.

Zusätzliche Programmierer, die bei einem eiligen Programmiervorhaben eingesetzt werden, verzögern es.

Es gibt immer noch einen Programmfehler.

Wenn ein Ausdruck einen Anfang hat, dann hat er auch ein Ende.

Die letzten vier Seiten eines wichtigen Ausdruckes gehen stets verloren.

Die Wahrscheinlichkeit, daß ein Programm die in es gesetzten Erwartungen erfüllt, sind umgekehrt proportional zum Vertrauen des Programmierers in seine Fähigkeit die Aufgabe zu erfüllen.

